Restricted Delegation
Seamlessly spanning administrative domains

Jon Howell
David Kotz

Dartmouth College

December 2, 1999

1 The logic of restricted delegation

Historically and currently, access control and authentication is managed through ACLs. Examples include:

- the list of users in `/etc/passwd`, the NIS `passwd` map, or an NT domain
- permissions on Unix files or ACLs on NT objects
- a list of known hosts in `.ssh/known_hosts`
- a list of IP addresses in `.rhosts` (for rsh) or `.htaccess` (http)

The limitations of ACLs always cause problems when spanning administrative domains (and often even inside administrative domains). The best example is the inability to express transitive sharing. Alice shares read access to object X with Bob (but not access to X's ACL), and Bob wants to share some of it with Charlie. Bob can share all of it by giving up his identity. He can share part of it by copying it or acting as an "access oracle" to X. All three mechanisms, however, undermine the underlying security model.

Restricted delegation is not based on ACLs. Instead, every object is completely controlled by some principal (in the diagram, the system administrator Merlin). That principal can act as an administrator of the resource, sharing it (or restricted access to it) with a user principal (Alice). Alice can treat her restricted access to the resource as a complete (but "smaller") resource, and turn around and similarly act as the administrator in another sharing relationship with a third principal (Bob). In each sharing relationship, the administrator (the principal sharing the resource) can restrict how much access the recipient receives.

With this simple tool, many difficult problems dissolve into simple expressions of restricted permission. ACLs can be managed by distributed parties. Group membership no longer requires separate security-sensitive code to implement, only a separate user interface. Authentication and access control, even between principals separated by indirection such as proxies, firewalls, or protocol translators, are handled in a consistent and simple fashion. Time limits on delegations are easily expressed in the same framework. The result is a trusted computing base that is not only smaller and easier to verify, but more flexible.

2 Why use a formal logic and semantics?

With a logic, we manipulate only a few, generic entities; in our case, these are principals, names, and statements. Once we grasp the meaning of these three simple concepts of the logic, we can map sophisticated features of our implementation onto those concepts and verify that our security model holds. If we use a decision procedure that is justified by a logic, then we can understand the ramifications of any proposed extension to the implementation by mapping it into new sorts of statements in the logic.

To give a logic meaning, it is backed by a formal semantics that provides a mathematical model. The formal semantics provides intuitive justification for the logic, and the logic (or a decision procedure derived from it) is used to build a tractable implementation. The semantics helps us specify precisely what the instruments of a protocol, such as statements and certificates, mean. It helps us understand what the decision procedure is telling us. A semantics enhances our ability to communicate, and hence understand, what promises a security model can make.

3 Advantages of our extension to the calculus of Lampson, *et al.*

Our extension preserves the original simplicity, expressiveness, and formality of the calculus for access control. For example, our extension retains conjunct and quoting principals.

Conjuncts (principals that represent two or more other principals in agreement) are first-class principals in that they may be used anywhere any other principal may be. In the upper-right diagram, the DNS server can only be modified by agreement of the CIO and the system administrator: the conjunct principal that represents their agreement speaks for the DNS server. The webmaster has obtained one restricted delegation to speak for the CIO, and another to speak for the system administrator. Together, those delegations give the webmaster control over the DNS server with respect to the intersection of the restrictions.

A quoting principal is a principal that represents one principal claiming to speak on behalf of another. In the lower-right diagram, both Alice and Bob have logged on to a host computer, each by giving the host computer permission to speak for them only when explicitly quoting them (upper arrows). Without quoting, the host would speak directly for both Alice and Bob. Then if Alice asked the host to read Bob's files, the host would have to check that permission itself (rather than relying on the Home Directories server), lest the server conclude "yes, host speaks for Bob, so it is okay to read the file." Therefore quoting makes it easier to build correctly multiplexed servers (such as the host) by keeping them out of the trusted computing base.

4 Summary

Restricted delegation allows transitive sharing, a requirement for sharing that spans administrative boundaries. Each principal in a chain of delegation can restrict what is shared with the next, so that each principal can act both as a user (recipient) and administrator (giver) of any resource. A logic and formal semantics helps us intuitively grasp the consequences of a security model. They justify the decision procedure that ends up in the implementation, and they can confirm or reject the soundness of proposed extensions. Our logic and semantics adds restricted delegation to the calculus for access control while preserving first-class conjunct and quoting principals.

A technical report is available at:
`http://www.cs.dartmouth.edu/~jonh/research/delegation/`