Distributed Policy Management for Java 2

ISOC NDSS’99
4-5 February 1999, San Diego

Jonna Partanen
Helsinki University of Technology
Jonna.Partanen@hut.fi
Overview

- Introduction
- Java 2 security model
- Authorization certificates, SPKI
- Using SPKI certificates to improve Java 2 security policy management
- Implementation
- Conclusions
Introduction

• We are considering a very large, distributed Java environment
  – Computers
  – Cellular phones
  – PDAs
• The users want to run software from many different sources without compromising security
The Problem

• How to manage the security policy?
  – In a scalable way?
  – With minimum dependency of external security mechanisms?
  – In a way transparent to the applications?
ProtectionDomains

Security Policy

SecureClassLoader

ProtectionDomain

JVM
Java 2 Access Control

- When the class tries to access a protected resource, the AccessController checks the permissions in the class’ protection domain
  - The class cannot add permissions to its protection domain
  - The class cannot change its protection domain
The Current Solution has Limitations

- Access rights are defined in local configuration files
  - Changing the policy requires editing the files
  - The files can get very complex
- Access rights are practically static
- How can the administrator know what access rights a certain class needs?
Authorization Certificates

- Identity certificates bind a name to a key
  - Usually ACLs are then used to define what the name is allowed to do
- Authorization certificates bind access rights directly to a key
  - Close to the concept of capability
  - Can provide anonymity
SPKI Certificates

- Simple Public Key Infrastructure
- Being published as Experimental RFC
- SPKI certificates are signed five-tuples
  - Issuer
  - Subject
  - Delegation
  - Tag (i.e. authorization)
  - Validity
Certificate Loops

- When authorization is delegated, the certificates form chains.
- When used, the chain is closed into a loop:

  - **Policyadm**
  - **A**
    - **Self, A, propagate, access X, forever**
  - **X**
    - **Self**
    - Proves possession of C key
  - **Client**
  - **C**
    - **B**
      - **B, C, limited access to X, forever**
  - **Policyadm**
  - **B**
    - **A, B, propagate, access X, limited time**
SPKI Certificates for Java

- Issuer, subject, delegation, validity etc. expressed according to the SPKI specs
- Tag definition is more focused: tags express Java permission objects
  
  `(tag (java-permission
       (type java.io.FilePermission)
       (target /tmp/myfile) (action read)))`

- Tags may also express a set or “any” permissions
Authorizing Java Classes

Administrator

Self, A, propagate, all permissions, forever

A, D, propagate, permission to read files, this month

SW developer

D, h(MyClass), propagate, permission to read file X, forever

ClassLoader checks the hash when MyClass is loaded

Java code

Self key

MyClass

Self key

ClassLoader checks the hash when MyClass is loaded

JVM/Policy

Self key
Prototype

- Public interfaces for SPKI certificates
- A Provider that implements the SPKI certificate functionality
- A Policy that uses dynamic protection domains and SPKI certificates to grant permissions
- A simple certificate repository
  - Is being replaced with DNS
Distributed Protection Domains

- If the protection domains could have temporary keys, they could delegate their permissions to other domains
  - The JVM must provide the keys
  - The JVM must help bind the temporary key to the object
- For example, a client could authorize an agent on a server to perform tasks on its behalf
Conclusions

• SPKI certificates can be used to make Java security policy management
  – Secure
  – Distributed
  – Scalable
  – Dynamic