Agent Based Architecture for Supporting Application Layer Security

Carlisle House
Introduction

• Traditional mechanisms are static
  – Difficult to change or adapt new or updated security policies and mechanisms

• Current Distributed Security Environments
  – Applications and users have varying security requirements
  – Current static systems
    • Limit users and application security choices
    • Must rely on underlying infrastructure
      – Delegation and security management constrained by static security policies and mechanisms
Needed Distributed Environment Traits

• Capable of supporting numerous policies and mechanisms
• Add, replace, or revoke policies and mechanisms
• Allow applications to choose what type of security guarantees they want
• Dynamically enforce customized policies
• Restrict use of policy to applications and systems on a need-to-know basis
Proposed Distributed Security Environment

- Uses mobile agents (ACs) to provide functionality
  - Signed code fragments that specify security policies and mechanisms
- Framework required to evaluate and enforce policies specified by ACs
  - Also provide run-time revocation, update and enforcement of policies and mechanisms
Architecture Design

• Components
  – Active Capabilities (ACs)
  – AC Management
    • Distributed network of AC Administrators
    • Software Framework Repositories
    • AC Servers
  – Software Framework
  – Evaluation/Enforcement Engine
Active Capabilities

• Use software framework for context
  – Framework is componentized and arranged to be downloaded from software framework repository via secure channel
Active Capabilities

• AC Format
  – First field contains information about type of active capability
  – Second field contains arbitrary piece of code which specifies security policies or mechanisms
  – Third field contains a digital signature
    • Created by AC Admin or trusted entity
      – Ensures integrity of capability
AC Management

• AC Administrators
• Software Framework Repositories
• AC Servers
AC Administrator

- Responsible for verifying, validating, and certifying code inside ACs and signing it
- Additionally manage distribution of ACs via secure channels
  - Applications can contact AC Server and obtain ACs to embed in own code
  - Trusted applications can create own ACs
AC Administrator

- Each protection domain has at least one AC Admin
- Uses PKI with one key-pair for AC Management
  - Admin uses private key to sign message digest of AC code
  - Distributes public key for each protection domain
AC Server and Software Framework Repository

• AC Server is front-end for Software Framework Component Repository
  – Allows for extraction of components

• Software Framework Repository
  – Houses Active Capabilities available for download
Software Framework

- Provides context for Active Capabilities
  - May use hierarchical structure of object-oriented classes of standard security policies
Evaluation/Enforcement Engine

- **AC Cache**
  - Used to cache ACs that don’t change often

- **Run-time resolvable references to AC evaluation sandboxes**
  - Imposes static and dynamic constraints on code

- **Run-time resolvable references to componentized software framework**
Evaluation/Enforcement Engine

- Runs with superuser privilege
- Networked by cryptographically secured channel to communicate with AC management infrastructure
  - Used to download ACs, framework and sandbox components
Evaluation/Enforcement Engine

• Operation Scenario
  – Application makes call to access specified resource
  – Request is encapsulated and passed to evaluation engine
  – Engine builds context and evaluates AC associated with requested policy or mechanism
  – Depending on result of evaluation, application is granted or denied access
  – If granted, enforcement must export interface to redirect their requests through granted policy or mechanism
Application Scenario Example

• Consider network configuration to the right
• Traditional idea of detection uses pattern matching and analysis i.e. data mining
• This can detect abnormal or suspicious behavior, however, cannot prepare in advance the countermeasures of attack in progress

Diagram:
- Dynamic Firewall Node
- Compromised Node
- Regular Node
Scenario Example cont'd

- Can use dynamic policies customized to particular attacks
  - Update all ACs on nodes infected or near infection
  - Nodes near infected node can update their AC to block all access from infected node
  - Basically building a secure network node model simulating quarantine
Conclusion

- Development of Agent-based Security leads to dynamic security system that installs and updates varying security policies and mechanisms that are customizable to application or user
References

• An Agent Based Architecture for Supporting Application Level Security
  – Zhaoyu Liu, Prasas Naldurg, Seung Yi, Tin Qian, Ron H. Campbell, M. Dennis Mickunas
  – Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign
Questions