SMASH: A Secure Modular Mobile Agent System

Adam Pridgen and Christine Julien
The University of Texas at Austin
Excellence in Distributed and Global Environments
Mobile and Pervasive Computing Group
May 5, 2006
Overview

• Motivating Application
• SMASH
  • Overview
  • Agent Architecture
  • Platform Architecture
  • Agent and Platform Interaction
• Implementation
• Conclusion
• Questions
Motivating Application

- Distributed, Collaborative data processing
  - Not all parties have equal data access rights
    - Commercial sensitivity of data varies
    - Partners may share unequal proprietary rights
  - Data is processed by many different platforms
    - Risk of data leaks or loss arise
    - Agents and platforms have varying levels of access to available resources
Motivating Application (2)

• Requirements for a solution on this application
  • Flexibility for processing cross domain data
  • Granular access controls on cross domain data
  • Ability to dynamically govern policies and resources
    • Adjust access controls based on the task properties
    • Adjust access controls based on the platform constraints
  • Agent and platform security
    • Component integrity
    • Non-repudiation between components
SMASH Overview

• Open platform system
  • Permit anonymous access
  • Allow anonymous coordination
  • Adapt privileges to current state

• Security by design
  • Utilize hardware and software security techniques
  • Require mutual authentication between the agent and platform
  • Validate all software participating in the system
  • Ensure security fault tolerance of system

• Modular agent design
  • Build the agent to suit the application
  • Flexibility for security needs
SMASH Agent Architecture

- Signature
- Main Module
  - Agenda
  - Credentials
  - Creators Key
  - Code
  - Application Data
  - Itinerary
  - Time To Live
- Storage Module
  - Data
  - Digest
  - Digest Public Key
  - Data and Execution State
SMASH Agent Architecture (2)

- Agent Agenda Goals
  - Task
    - High-level description
    - Special parameters
    - System libraries needed
  - Resources
    - Resources required
  - Authorization Requirements
    - Authorization recommendations
    - Special privileges needed
    - Access rights needed
SMASH Agent Architecture (3)

- Agent Credentials
  - Details authentication methods
    - Allows for flexible authentication procedures
    - System can be built around multiple authentication methods
  - Provides required authentication data
    - Tokens
    - Hashed Session Keys
    - Public Keys

```xml
<AuthenticationMethods>
  <attribute> NumberOfMethods = 3 </attribute>
  <RSA>
    <attribute> PKSize = 2048 </attribute>
    <attribute> PublicKey = X </attribute>
  </RSA>
  <ECC>
    <attribute> PKSize = 1048 </attribute>
    <attribute> PublicKey = Y </attribute>
  </ECC>
  <Kerberos>
    <attribute> TicketServer = foo </attribute>
    <attribute> ServiceRequested = bar </attribute>
    <attribute> KToken = Z </attribute>
  </Kerberos>
</AuthenticationMethods>
```
SMASH Agent Architecture (4)

- Itinerary
  - Map of the platforms the agent may visit
    - Not ordered by visitation
  - Allows the agent to verify the identity of the platform
  - Permits referral to another platform
  - Extensible for unknown needs

```xml
<Itinerary>
  <attribute> NumberOfPlatforms = 3 </attribute>
  <Platform Xray>
    <attribute> AuthenticationMethod = RSA </attribute>
    <attribute> EPKSize = 1024 </attribute>
    <attribute> ExpectedPublicKey = XPKH </attribute>
    <attribute> ESDKSize = 512 </attribute>
    <attribute> ExpectedSoftwareDigest = XSH </attribute>
    <attribute> Options = None </attribute>
  </Platform Xray>
  <Platform Yankee>
    <attribute> AuthenticationMethod = ECC </attribute>
    <attribute> EPKSize = 1024 </attribute>
    <attribute> ExpectedPublicKey = XPKH </attribute>
    <attribute> ESDKSize = 512 </attribute>
    <attribute> ExpectedSoftwareDigest = XSH </attribute>
    <attribute> Options = 2 </attribute>
    <Options>
      <attribute> AllowRedirect = True </attribute>
      <attribute> RedirectHost = Any </attribute>
    </Options>
  </Platform Yankee>
  <Platform Zulu>
    <attribute> AuthenticationMethod = Kerberos </attribute>
    <attribute> EPKSize = 1024 </attribute>
    <attribute> ExpectedPublicKey = XPKH </attribute>
    <attribute> ESDKSize = 512 </attribute>
    <attribute> ExpectedSoftwareDigest = XSH </attribute>
    <attribute> Options = 1 </attribute>
    <Options>
      <attribute> ServicesRequest = ProcTime </attribute>
      <attribute> AllowRedirected = True </attribute>
      <attribute> RedirectHost = HighlyTrusted </attribute>
    </Options>
  </Platform Zulu>
</Itinerary>
```
SMASH Agent Architecture (5)

• Module Integrity Protection
  • Signature
    • Function of Creator Key and other agent components
    • Created using asymmetric cryptography algorithms
      • ECC
      • RSA
  • Creator Key
    • Platform or user’s public key
    • Identifies the originating entity
  • Storage Module Integrity Protection
    • Similar to Main Module
    • Digest Public Key
      • Formed using the current Platform’s PK

[Diagram of SMASH Agent Architecture]
SMASH Platform Architecture

- Hardware Based Security and Identification
  - Relies on a Trusted Platform Module
  - Supports Asymmetric and Symmetric Cryptography
- Compartmentalized Approach Promotes Security
  - Segregate Anonymous and Authenticated Agents
- Platform Management
  - Manage agents and resources
  - Track capable tasks
  - Maintain active resources
  - Can treat agents as resources
- Blackboard
  - Permits coordination among all system components
SMASH Platform Architecture (2)

- **Untrusted Layer**
  - Provides limited execution
  - Contains Unauthenticated Transient Agent Activity

- **Agent Manager**
  - Track agents through the platform

- **Authentication and Authorization**
  - Mutual Authentication
  - Authorization
SMASH Platform Architecture (3)

- Trusted Containment
  - Holds Authenticated Agents
- Security Manager
  - Obtains and manages agents required resources
  - Manages task that can be performed
  - Prepares the agent for execution
- Blackboard
  - Public media for coordination
SMASH Agent and Platform Interaction

- Agent Arrives to Platform
  - Platform verifies Agent Signature
  - Agent may compare itinerary entry to the platform TPM
- Agent registers with Agent Manager
SMASH Agent and Platform Interaction (2)

- Agent capabilities in the Untrusted Zone
  - Can access Blackboard
  - Can leave the platform
  - Can authenticate
SMASH Agent and Platform Interaction (3)

- Agent indicates need to authenticate
  - Platform verifies the Agent
  - Agent verifies the Platform
- Agent Manager updates Agent's status
SMASH Agent and Platform Interaction (4)

- Agent moves to the Trusted Zone
  - Platform ensures it can meet the Agent's demands
  - Platform acquires resources
SMASH Agent and Platform Interaction (5)

- Platform obtains all resources
  - Loads Agent Resources
  - Sets up Agent Environment
  - Loads Agent Bootstrap code
SMASH Agent and Platform Interaction (6)

- Agent Executes
  - Utilizes minimal privilege for each Task
  - May or may not complete all Tasks
SMASH Agent and Platform Interaction (7)

- Agent preparation for Dispatch
  - Saves all data and state
  - Passes module to Platform
  - Platform signs the module
  - Agent Manager updates Agents Status for Dispatch
SMASH Agent and Platform Interaction (8)

- Agent moves to Untrusted Zone
  - Could choose to perform non-privileged actions
- Agent leaves the Platform
- Agent Manager remove the Agent
Implementation

• Platform Implementation
  • Security Management with SE Linux
    • Governs platform with granular role based access controls
    • Assists with security fault tolerance
  • Integrity from the Trusted Platform Module (TPM)
    • System Software can be checked periodically
    • Identification engrained in the hardware
  • Agent Control with Custom Security Manager
    • Implemented C++
    • Compartmentalized Design

• Agent Implementation
  • Uses Python Programming Language
    • Provides Object Oriented and Module Based Agent Programming
Conclusions

- SMASH addresses complex problem of security and resources
  - Extending resource availability beyond simple permissions
  - Enhancing system security through hardware and software
  - Providing agent modularity for more extensible applications
- Embraces security and flexibility
  - Provide support flexible security requirements
  - Permits agents to move about without being authenticated
  - Allows all components to coordinate
  - Affords fault tolerance and compartmentalization
- Utilizes Modular Agents
  - Enables agent portability
  - Reuse common modules
  - Accommodates simplicity of agents
Questions?

More research at:
http://mpc.ece.utexas.edu