Enforcing “Sticky” Security Policies throughout a Distributed Application

David Chadwick
University of Kent
Acknowledgements

• The work being presented here is current research supported by the EC FP7 TAS3 project

• Thanks to all in the project team who contributed to this

• We are only 10 months into a 4 year project so this work will evolve and develop in the coming years
  – and you can contribute to it now 😊
Contents

• Policy Enforcement in Existing Systems
• Sticky policies
• Sticking policies to data
• Enforcing sticky policies
• Transferring sticky policies throughout a distributed system
Policy Enforcement in Existing State of the Art Systems

- Still many issues about obligation enforcement
  - Before, With, or After resource access
  - Two phase commit
  - Obligation language
  - Application independent or dependent
Sticky Policies

- Policies that control how data is to be accessed and used, and that accompany data throughout an entire distributed system
- Policies may be access control policies, privacy policies, obligation policies etc
- Policies may be written in a variety of policy languages
- Add a new level of complexity
Envisaged Data Structure

- When multiple policies give different values, what is the overall access decision?
- Policy Precedence, set by Authoritative Source, will determine the policy combining rule to use e.g. trust policy over-rides, Govt policy over-rides, data subject’s policy decides etc.
- Active ongoing research
Other Issues

• How are data items with sticky polices to be aggregated. How is a combined sticky policy created?

• How are data items with sticky policies to be dis-aggregated? How is a fractional sticky policy created?
Structure of Application
Independent Policy Enforcement

Existing Systems

AppDep PEP

App Indep PDP

Proposed System

AppDep PEP

App Indep PEP

Master PDP

Policy PDP

Policy PDP

Policy PDP

Will Enforce Application Independent Obligations

Will Enforce Policy Precedence

Will evaluate each policy with language dependent PDP
When are Sticky Policies Enforced?

- Is the requestor allowed to make this outgoing call?
- Is this requestor allowed to make this incoming request?
- Is the sender allowed to make this outgoing call?
- Is the sender allowed to send this data?
- Is the requestor allowed to receive all this data?
- Is the requestor allowed to receive it?
How are Sticky Policies to be Transferred?

• Mont et al\(^1\) propose to stick policies to application data by using Identity Based Encryption
  – Use public ID of recipient and encrypt data and policy for it.
• But what about existing (legacy) applications?
• What is the trust/security model?
  – If recipient is trusted to enforce policy then don’t need to make sure that only it can receive the data
  – If recipient is not trusted to enforce policy then don’t send the data to it
  – If confidentiality in transfer is required then can use SSL (IBE allows TTP to read it as well)
• So we are proposing more conventional means of transfer that can cater for legacy applications as well as new ones

Three Models are Proposed

- Encapsulating Security Layer for new applications that can store data and sticky policies
- Application Protocol Enhancement for existing applications that have flexibility in their protocol to carry sticky policies
- Back Channel for legacy applications that cannot store or transfer policies
The Encapsulating Security Layer Model for New Applications

Stores Data+Sticky Policies together

Service

PEP

App Indep PEP

Data+Sticky Policies

Master PDP

Policy PDP

Data+Sticky Policies

Internet

Data+Sticky Policies

PEP

App Indep PEP

Data+Sticky Policies

Master PDP

Policy PDP

Stores Data+Sticky Policies together

Service
The Application Protocol Enhancement Model

Stores Data + Sticky Policies perhaps separately

Stores Data + Sticky Policies perhaps separately
The Back Channel Model

Stores Data only

Service

PEP

Sticky Policies

App Indep PEP

App Indep PDP

Policy PDP

Policy PDP

Internet

PEP

Sticky Policies

App Indep PEP

App Indep PDP

Policy PDP

Policy PDP

Service

Stores Data only
One Legacy Implementation Approach

- **Legacy Service**
  - Legacy Protocol
  - Stores Sticky Policies + pointers to Appln Data

- **Gateway PEP**
  - Enhanced Application Protocol Data + Sticky Policies

- **App Indep PEP**
  - Enhanced Application Protocol Data + Sticky Policies

- **App Indep PDP**
  - Policy PDP

- **Legacy Service**
  - Legacy Protocol
  - Stores Sticky Policies + pointers to Appln Data

- **Gateway PEP**
  - Enhanced Application Protocol Data + Sticky Policies

- **App Indep PEP**
  - Enhanced Application Protocol Data + Sticky Policies

- **App Indep PDP**
  - Policy PDP
## Comparison of 3 Models

<table>
<thead>
<tr>
<th>Feature</th>
<th>Appln prot enhancement</th>
<th>Encapsulat’g security layer</th>
<th>Back channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires modification to the application layer protocol</td>
<td>Yes, since extra policy information needs to be carried</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Requires trust in the PEP</td>
<td>Yes, to parse the message into security blocks and to carry the sticky policy between systems</td>
<td>Yes, to parse the message into security blocks</td>
<td>Yes, to parse the message into security blocks</td>
</tr>
<tr>
<td>Requires changes to the PEP interface</td>
<td>Yes somewhat, since policies are being transferred across the interface</td>
<td>Yes significantly, since the AIPEP will be transporting the application messages</td>
<td>No</td>
</tr>
<tr>
<td>Has control over distributed system security</td>
<td>Yes somewhat, but has to rely on application more</td>
<td>Yes complete control</td>
<td>Yes, significant control</td>
</tr>
<tr>
<td>Implementation effort for application</td>
<td>Easier</td>
<td>Most complex</td>
<td>Easiest (no changes needed)</td>
</tr>
<tr>
<td>Implementation effort for AIPEP</td>
<td>Easiest</td>
<td>Most complex</td>
<td>Medium difficulty</td>
</tr>
<tr>
<td>Flexible use of security in application data transfer between systems</td>
<td>No, application must act independently of sticky policy</td>
<td>Yes, can understand sticky policy and act accordingly</td>
<td>No, applicat’n must act independently of sticky policy</td>
</tr>
<tr>
<td>Supports integrated trust ne</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Trust Negotiation

• How does the sending system know if the receiving system is willing or able to enforce the sticky policies

• Trust Negotiation is one mechanism in which this can be determined

• Trust negotiation is the term used to refer to the gradual release of credentials by either party in order to mutually establish and build up trust between them

• May be carried out by a trust negotiation service on behalf of the application
Difference Between Sticky Policies and DRM

• DRM assumes the recipient is a bad guy who wants to rip off the content owner
  – Recipient is restricted in what they can do with the data

• Sticky policies (esp. privacy ones) assume recipient wants to enforce the sticky policies (e.g. user’s consent) but lacks mechanisms to automate this
  – Applications that want to allow back door access to data and/or remove sticky policies from data are not constrained in this respect
Any Questions?