OMOS: A Framework for Secure Communication in Mashup Applications

Saman Zarandioon, Danfeng (Daphne) Yao, Vinod Ganapathy
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
{samanz,danfeng,vinodg}@cs.rutgers.edu

December 2008
What is a **Mashup** application?
What is a **Mashup** application?

- Seamlessly combine contents from multiple heterogeneous data sources.
- Overall goal: more integrated and convenient end-user experience.
- Becoming very popular - Web 2.0
Mashups

- What is a **Mashup** application?
- My favorite mashup website *Zillow!*
Mashups

Introduction

Mashups

Architecture

Security in client-side services

OMOS

Experiments

OpenMashupOS.com
Mashups

Introduction

Mashups

Architecture

Security in client-side services

OMOS

Experiments
Mashups

- What is a **Mashup** application?
- My favorite mashup website *Zillow*!
- **Web desktop** (webtop) (e.g., eyeOS, DesktopTwo, G.ho.st, Netvibes, and Online OS).
Mashups

Introduction

Mashups

Architecture

Security in client-side services

OMOS

Experiments

OpenSourcePlatform.org

AC SAC 2008 – 2 / 11
Ways that service providers can expose their services:

- **Server-side services**
Ways that service providers can expose their services:

- **Server-side services**
Ways that service providers can expose their services:

- **Server-side services**
- **Client-side services**
Ways that service providers can expose their services:

- **Server-side services**
- **Client-side services**

User is involved; AJAX-oriented; More responsive/efficient
Service providers use *ad-hoc non-secure* methods.
Security in client-side services

- Service providers use **ad-hoc non-secure** methods.
- Consumers need to *trust* service providers: Not suitable when dealing with sensitive personal data.
Security in client-side services

- Service providers use ad-hoc non-secure methods.
- Consumers need to trust service providers: Not suitable when dealing with sensitive personal data.
- HTML, JavaScript and browsers are not designed to support client-side communication.
Service providers use ad-hoc non-secure methods.

Consumers need to trust service providers: Not suitable when dealing with sensitive personal data.

HTML, JavaScript and browsers are not designed to support client-side communication.

Trade-Off Between Usability and Security: All or Nothing, Complete isolation vs. complete exposure.
Security in client-side services

- Service providers use **ad-hoc non-secure** methods.
- Consumers need to *trust* service providers: Not suitable when dealing with sensitive personal data.
- HTML, JavaScript and browsers are not designed to support client-side communication.
- Trade-Off Between Usability and Security: All or Nothing, Complete isolation vs. complete exposure.
OpenMashupOS (OMOS) is a mashup framework that is designed to support secure client-side services.
OpenMashupOS (OMOS) is a mashup framework that is designed to support secure client-side services.

Design Goals:
■ **OpenMashupOS (OMOS)** is a mashup framework that is designed to support secure client-side services.

■ **Design Goals:**
  
  - To be compatible with all major browsers without *any* change or extension to the browsers.
OpenMashupOS (OMOS) is a mashup framework that is designed to support secure client-side services.

**Design Goals:**

- To be compatible with all major browsers without *any* change or extension to the browsers.
- To provide a *powerful* abstraction that is *flexible* and *easy* to *understand* and *use* by mashup developers.
Overview

■ **OpenMashupOS (OMOS)** is a mashup framework that is designed to support secure client-side services.

■ **Design Goals:**

- To be compatible with all major browsers without *any* change or extension to the browsers.
- To provide a *powerful* abstraction that is *flexible* and easy to *understand* and *use* by mashup developers.
- To guarantee *mutual authentication*, *data confidentiality*, and *message integrity* for communication between service provider and consumer.
OpenMashupOS (OMOS) is a mashup framework that is designed to support secure client-side services.

**Design Goals:**

- To be compatible with all major browsers without *any* change or extension to the browsers.
- To provide a *powerful* abstraction that is *flexible* and easy to *understand* and *use* by mashup developers.
- To guarantee *mutual authentication*, *data confidentiality*, and *message integrity* for communication between service provider and consumer.
Mashlet is a client side component that runs in the browser under the privilege of the principal that is defined by the domain name of the server that hosts the mashlet.
Mashlet

- **Mashlet** is a client side component that runs in the browser under the privilege of the principal that is defined by the domain name of the server that hosts the mashlet.

- Mashlets should be able to communicate securely on the client side, meaning that the communication protocol guarantees:
Mashlet is a client side component that runs in the browser under the privilege of the principal that is defined by the domain name of the server that hosts the mashlet.

Mashlets should be able to communicate securely on the client side, meaning that the communication protocol guarantees:

- Mutual Authentication
- Confidentiality
- Message Integrity
**Mashlet** is a client side component that runs in the browser under the privilege of the principal that is defined by the domain name of the server that hosts the mashlet.

Mashlets should be able to communicate securely on the client side, meaning that the communication protocol guarantees:

- Mutual Authentication
- Confidentiality
- Message Integrity
Mashlet

- **Mashlet** is a client side component that runs in the browser under the privilege of the principal that is defined by the domain name of the server that hosts the mashlet.

- Mashlets should be able to communicate securely on the client side, meaning that the communication protocol guarantees:
  - Mutual Authentication
  - Confidentiality
  - Message Integrity
Using OMOS API, mashlets can communicate with their siblings and parents.
Secure Frame-to-frame Communication

Security of OMOS communication protocol relies on *Same Origin Policy (SOP)*:

- Protects confidentiality of domains against each other. (DOM elements, events, cookies, ...)
- URL property of an iframe is write-only.
- Partial change of URL is not allowed.
Secure Frame-to-frame Communication

<table>
<thead>
<tr>
<th>Destination Frame Address</th>
<th>Source Frame Address</th>
<th>Endpoint ID</th>
<th>Secret Key</th>
<th>Sequence Number</th>
<th>List of data fragments</th>
</tr>
</thead>
</table>

Experiments

iframe[1]
a.com

iframe[2]
b.com

DLPacket
Secure Frame-to-frame Communication

<table>
<thead>
<tr>
<th>Destination Frame Address</th>
<th>Source Frame Address</th>
<th>End point ID</th>
<th>Secret Key</th>
<th>Sequence Number</th>
<th>List of data fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td>iframe[1] a.com</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iframe[2] b.com</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://b.com/proxy#DLPacket
Secure Frame-to-frame Communication

<table>
<thead>
<tr>
<th>Destination Frame Address</th>
<th>Source Frame Address</th>
<th>Endpoint ID</th>
<th>Secret Key</th>
<th>Sequence Number</th>
<th>List of data fragments</th>
</tr>
</thead>
</table>

Experiments

OpenMashupOS.com
Secure Frame-to-frame Communication

<table>
<thead>
<tr>
<th>Destination Frame Address</th>
<th>Source Frame Address</th>
<th>End point ID</th>
<th>Secret Key</th>
<th>Sequence Number</th>
<th>List of data fragments</th>
</tr>
</thead>
</table>

- **Introduction**
- **OMOS**
- **Overview**
- **Mashlet**
- **Secure Frame-to-frame Communication**
- **Communication Stack**
- **MDP Layer**
- **MHTTP Layer**
- **Experiments**
## Secure Frame-to-frame Communication

### Introduction

#### OMOS Overview

**Mashlet**

**Secure Frame-to-frame Communication**

**Communication Stack**

**MDP Layer**

**MHTTP Layer**

### Experiments

<table>
<thead>
<tr>
<th>Destination Frame Address</th>
<th>Source Frame Address</th>
<th>Endpoint ID</th>
<th>Secret Key</th>
<th>Sequence Number</th>
<th>List of data fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Diagram**

- b.com/proxy
- iframe[1] a.com
- iframe[2] b.com
- DLPacket
Secure Frame-to-frame Communication

<table>
<thead>
<tr>
<th>Destination Frame Address</th>
<th>Source Frame Address</th>
<th>End point ID</th>
<th>Secret Key</th>
<th>Sequence Number</th>
<th>List of data fragments</th>
</tr>
</thead>
</table>

- iframe[1]
a.com

- iframe[2]b.com

DLPacket
Key exchange protocol:

A

frame[1]
a.com
<<Datalink>>

Generate random nonce A

frame[2]
b.com
<<Datalink>>
Secure Frame-to-frame Communication

Key exchange protocol:

A

\[
\text{frame[1]} \\
\text{a.com} \\
\langle\langle\text{Datalink}\rangle\rangle
\]

\[
\text{frame[2]}, \text{frame[1]}, \text{EID1}, (A), 0, \text{MDP packet(s)}
\]

\[
\text{frame[2]} \\
\text{b.com} \\
\langle\langle\text{Datalink}\rangle\rangle
\]
Secure Frame-to-frame Communication

Key exchange protocol:

A

frame[1]
a.com
<<Datalink>>

frame[2]
b.com
<<Datalink>>

Send MDP packet(s) to MDP layer
Secure Frame-to-frame Communication

Key exchange protocol:

A

frame[1]

a.com

<<Datalink>>

B

frame[2]

b.com

<<Datalink>>

Generate random nonce B
Secure Frame-to-frame Communication

Key exchange protocol:

A

frame[1]
a.com
<<Datalink>>

frame[1],frame[2],EID1,(A,B),0,MDP packet(s)

B

frame[2]
b.com
<<Datalink>>
Key exchange protocol:

Verifies that the first element of key pair is equal to A
Send MDP packets to MDP layer
Secure Frame-to-frame Communication

Key exchange protocol:

![Diagram of secure frame-to-frame communication](image)
Each layer hides complex implementation details of communication in lower layers.
3-way Handshake
3-way Handshake

ServerSocket

Client Socket

Create

Socket
3-way Handshake

Client Socket

ServerSocket

Socket

OnConnected

Acknowledgement
3-way Handshake

Client Socket

Server Socket

Acknowledgement + Data

OnConnected

OnDataReceived
MDP Layer

Introduction

OMOS Overview

Mashlet

Secure Frame-to-frame Communication

Communication Stack

Experiments

Client Socket

OnDataReceived

Data

OnDisconnected

Disconnection Request

Disconnected

OnDisconnected

OnDisconnected

Socket
Versatile asyncRequest: mashlet-to-mashlet, same-domain & cross-domain mashlet-to-server communication.
Results

Throughput (postMessage/Opera)

Transfer rate (bytes/msec)

Message size (bytes)

- Opera