Mashup 应用安全研究*

陈 丰

(重庆理工大学 计算机学院,重庆 400050)

摘 要:提出了一种新的安全解决方案,主要采用 frame 片段标识符技术在浏览器端实现跨域数据通信,在此基础上主要关注 3 个安全因素:数据保密、身份验证和数据完整。该方案不需要对当前的浏览器环境进行任何修改就可实现异源内容相互间安全的通信。

关键词:聚合;安全;同源策略;跨域通讯

中图分类号:TP393.08 **文献标识码**:A **文章编号**:1674-7720(2010)13-0079-04

Research on security of Mashup application

CHEN Feng

(School of Computer, Chongqing University of Technology, Chongqing 400050, China)

Abstract: This paper proposes a security architecture that employs frame.src URL's fragment identifier to implement inter-domain communication in browser-side. At the same time it provides high assurance on the mutual authentication, data confidentiality, and message integrity of Mashup application. The technology discussed in this paper provides a secure inter-domain communication mechanism without any modifications to current browsers.

Key words: Mashup; security; same-origin policy; inter-domain communication

在 Web2.0 时代,一种新型的基于 Web 的数据集成应用程序——Mashup,逐渐在 Internet 上变得越来越流行。在 IBM、微软等企业的推动下,今后数年内 Mashup 将成为主流应用。即使商业用户的 IT 技术水平并不高,同样也可创建符合自身需求的 Mashup,并将这些 Mashup 加以组合,IT 产业的安全性也将随之大幅提高[1]。

目前,Mashup 应用仍然处于萌芽阶段,但 Mashup 和其他类型的 Web 应用一样,也面临着各种各样的安全问题。因此,本文主要分析了 Mashup 的安全问题,并提出一种新的安全解决方案以提高 Mashup 的安全性。

1 Mashup(聚合)的基本概念

Mashup 是一类 Web 应用(或 Web 站点),集成了来自多个源站点的内容和代码并将其集成到一个页面中,其主要目标是为用户提供一种更加集成和方便的使用体验[2]。例如,www.housingmaps.com 将 Google Maps 上的地图与 Craigslist(美国网上广告公司)的房屋信息集成在一起,因此,用户可以从谷歌地图上直观地获得不同地点的房屋价格信息并能据此进行房屋租赁或销售等业务。这种概念和呈现方式非常简单,房屋价格信息和地图数据集成之后提供的可视化功能非常强大。

通常将上例中的 www.housingmaps.com 称为集成者,它集成了由第三方(如 Google 和 Craigslist,也称为内容提供者)提供的内容和代码,并为用户提供了所需要的一站式的服务。而第三方提供的资源,在 Mashup 中被称为组件(由 HTML 与 JavaScript 组成,旨在被集成到集成者的网页中,它通常是客户端的一些 Web 服务)。

Mashup 应用通常主要有两种框架结构:服务器端框架和客户端框架,如图 1 所示。

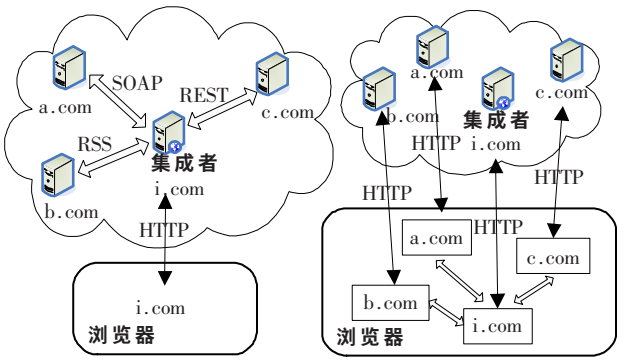
服务器端框架是一种传统的聚合方式,首先在服务器端集成异源的内容和服务,然后将集成后的页面返回到浏览器。图 1 所示为框架中的集成者(i.com)实际上起到代理的作用,因此,存在一些弊端:

(1)对第三方资源的请求/响应都需经过 i.com,降低了性能;

(2)大量用户可能导致过度的服务器负载,i.com 易成为瓶颈点,可扩展性差;

(3)i.com 易成为黑客的攻击点。

* 基金项目:重庆市科委自然科学基金项目(CSTC2009AC2068)



(a)服务器端框架 (b)客户端框架

图 1 两种主要的 Mashup 应用框架

近年来,随着 Ajax 技术及相关技术的发展,Mashup 应用出现了客户端框架,即在浏览器端集成异源的内容和服务,例如 www.housingmaps.com 就采用此种框架。同时,内容提供者(即第三方)也更愿意提供客户端的服务,如 Google 公司提供的 Maps API 就是一种非常流行的用于客户端的组件,集成者可以方便地将该组件集成到自己的 Mashup 应用中。由图 1 可知,集成者(i.com)将来自异源的内容和服务集成到一个网页中,集成者和组件、组件与组件之间可以在浏览器端进行通信,组件(如 a.com)可以使用 Ajax 技术异步访问服务器来实现网页局部刷新的效果,给用户带来更好的使用体验。基于此,本文主要研究了客户端框架下的安全应用问题。

2 浏览器的安全模型及 Mashup 的应用安全

2.1 同源策略与"全有或全无"模型

目前由浏览器所实现的安全模型是同源策略,最早出自 Netscape Navigator2.0。这里的同源是指同协议、同域名和同端口[3]。如果 2 个网页具有相同的协议、域名和端口,则认为它们是同源的。所谓同源策略,是指由某一来源的脚本不允许读取或设置来自另一不同来源的文档的属性。该机制将异源的 Web 应用程序分离开,即如果多个浏览器窗口、<frame>或<iframe>元素(下文统一用<frame>表示)中的文档是从不同的服务器下载的,则它们无法相互访问数据和脚本,浏览器的脚本只被允许访问来自同源的资源(如 DOM 和 Cookie 等资源),并能创建 XMLHttpRequest 对象异步访问文档来源所指服务器的资源。例如,<frame>A(来自 http://a.com)不能访问<frame>B(来自 http://b.com)中的任何 DOM 元素,反之亦然。来自 a.com 的脚本也只能创建 XMLHttpRequest 对象异步访问 a.com,而不能异步访问 b.com。

但上文所提的同源策略有一个例外,文档中包含的脚本资源文件和图像文件被视为文档的组成部分,因此认为与文档是同源的,即使它们实际上存在于不同的网域中。例如,a.com/service.html 中包含<script src="http://b.com/lib.js">,虽然 lib.js 来自 b.com,但被认为是 a.com/service.html 的组成部分,即 lib.js 与 service.html 是同源的,都来自 a.com,因此 lib.js(虽然来自 b.com)中的脚本也就能够访问 a.com/service.html 的 DOM 等资源,并能创建 XMLHttpRequest 对象异步访问 a.com。

同源策略比较适用于 Internet 发展的早期。由于这时很少网站将重要的应用逻辑放在浏览器端,即使有,这些网站也只限于自己的服务器而不同第三方打交道,因此,同源策略能有效避免恶意的攻击。但 Internet 发展到今天,情况已有了很大的不同。许多集成者更愿意集成多个来源的内容到他们的站点中,为用户提供定制化和更有附加值的服务。由前面的分析可知,Mashup 的本质就是要集成多个来源的内容和服务,而同源策略却是不允许使用来自不同源的内容。这导致在开发 Mashup 应用时只能采用"全有或全无"模型,即站点 a.com 或者完全不信任来自站点 b.com 的内容,在集成时将来自 b.com 的内容隔离在<frame>元素中;或者完全信任来自站点 b.com 的脚本,在集成时将来自站点 b.com 的脚本放到<script>标记中,而这些脚本能完全访问来自 a.com 的资源。因此,需要实现一种方法使得浏览器能够支持合法的跨域数据访问,但是无需牺牲终端用户的安全和对自身数据的控制。

2.2 Mashup 应用安全

Mashup 拥有两个最本质的特性:互通信能力和安全性。互通信能力是指集成者与组件或组件间相互通信的能力。在简单的 Mashup 中,可能不需要这种互通信能力,此时可将组件包含在<frame>中,利用同源策略来隔离异源的资源以达到保护的目的。但在越来越趋于复杂的 Mashup 应用中,集成者与组件间确实需要相互通信的能力。安全性要求来自某源的组件不能存取来自另一源的组件的保密信息,如 DOM、cookies 等信息。根据同源策略及"全有或全无"模型,当不同来源的组件集成到集成者的同一网页时,相互之间不能通信,除非采取一些技术手段来绕过同源策略的限制。结果通常造成开发人员被迫必须在安全性和功能之间进行权衡,导致为了满足功能而牺牲应用的安全性。为此,已经存在的 Mashup 应用采用了许多绕开同源策略的措施:

(1)Ajax 代理,Mashup 服务器端框架经常使用的一种方法[4]。例如,由于同源策略的限制,a.com/service.html 中的 JavaScript 代码是不能访问位于 b.com 中的某一 Web 服务 ServiceB 的,但可以在 a.com 中建立新的 Web 服务 ServiceProxy,则 a.com/service.html 可通过 Ajax 访问 a.com/ServiceProxy,而 ServiceProxy 的功能就是将请求转发给 b.com/ServiceB,由 ServiceB 产生的响应结果再按原路径返回给 a.com/service.html。

(2)<frame>片段标识符技术,Mashup 客户端框架经常使用的方法。通过 frame.src 属性的片段标识符(URL

中#符号后的部分)，使用了页面脚本和隐藏的<frame>标记之间进行通信以绕过同源策略的限制。

当前开发 Mashup 时，集成者通常将第三方提供的组件封装到<frame>标记中，利用同源策略所提供的安全机制，以实现安全的目的。但集成者与组件间通常更需要协作，笔者认为在浏览器端利用<frame>片段标识符技术进行通信时应当满足以下 3 个主要的安全因素：

(1)数据保密。在 Mashup 应用中，经常有这样的场景：如用户可能在某一网页中输入内容，并希望此内容将只提供给特定的组件，而不被其他第三方截获。或者，集成者、组件提供者和用户可能共享一个秘密信息，而不希望此信息被无关的第三方截获。因此，在浏览器端实现消息传递机制时，应能保证消息只能被相关的参与方获取，而其他无关的第三方不能截获到此消息。

(2)身份验证。参与通信的双方能够相互进行身份验证是浏览器端消息传递机制另一个很重要的安全要求。在 Mashup 应用中进行身份验证主要是指通信双方能验证彼此的域名。

(3)数据完整。数据完整是指集成者和组件提供者提供给用户的内容是没有被攻击者篡改过的，也即意味着被攻击者篡改过的消息能被检测出来。

3 <frame>片段标识符安全通信技术

3.1 基于<frame>片段标识符的跨域数据访问

<frame>标记能够在一个 HTML 文件中封装和显示另一个完整的 HTML 文件。通常称<frame>所在的网页为父页面，而<frame>所包含的网页(通过向 frame.src 属性指定一个 URL 来确定)称为子页面。

目前开发 Mashup 应用时，集成者通常将组件封装到<frame>标记中。当<frame>的源 URL 与其父页面同源时，根据同源策略，父页面中的 JavaScript 可通过<frame>的 DOM 访问它的所有内容。同样，<frame>也可以通过其父页面的 DOM 访问父页面的所有内容。然而，当异源时，父页面不能访问<frame>的内容，<frame>也无法访问父页面的内容。此时，可利用 frame.src 属性的片段标识符(如 http://a.com/proxy.html#data_here，# 符号后的 data_here 就是要传递的数据)来实现跨域数据访问。<frame>的外部代码(如主页面中的 Javascript 代码)不能读取 frame.src，但可以重写 frame.src，此即 frame.src 属性的只写特性。这样，<frame>的外部代码就可以根据已知文档的 URL，构造“URL+#+要传递的数据”字符串，指定给 frame.src 属性，并将触发子页面的 onLoad 事件，于是在子页面的 onLoad 事件处理程序中可接受父页面传过来的数据并决定下一步的操作。如图 2 所示，集成者 i.com/main.html 由于同源策略的限制，在浏览器端不能访问 a.com 的资源。因此，可通过以下步骤实现 i.com/main.html 将数据 data_here 传送给 a.com。

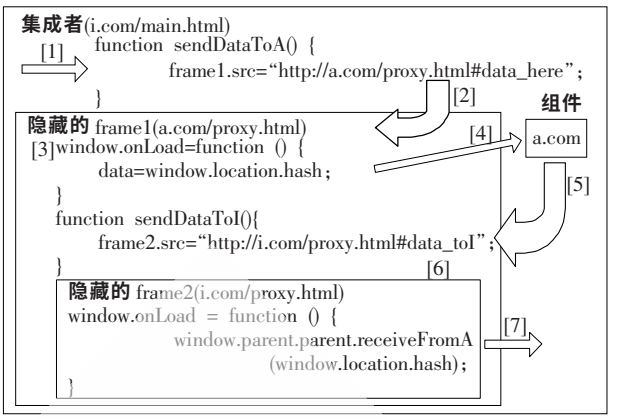


图 2 基于<frame>片段标识符的跨域数据访问

(1)i.com/main.html(即父页面，包含 frame1 元素)设置 frame1.src="http://a.com/proxy.html#data_here"。

(2)frame1 下载页面 a.com/proxy.html。通过 frame.src 来传递数据可以使用#将数据加到 frame 的 URL 结尾来实现。

(3)子页面(a.com/proxy.html)的 onLoad 事件被激活。此时，可通过 window.location.hash 取出 i.com/main.html 传递过来的数据 data_here。

(4)由于 frame1 和组件 a.com 同源，故能相互访问彼此的资源，即组件 a.com 也就能访问 i.com/main.html 传过来的数据 data_here。

当组件 a.com 处理完数据 data_here 后，需要将数据 data_toI 传递给父页面 i.com/main.html，也可以通过 frame1 来传递吗？答案是否定的，frame1 不能给它的父页面传递任何消息，因为它们位于不同的源。但是 frame1(a.com)可以包含另一个 frame2，图 2 中，frame1 通过设置 frame2.src 为父页面的 URL(即 i.com)。这时，i.com/main.html 包括 frame1(域为 a.com)，而 frame1 又包含 frame2(域为 i.com)。同时，可以发现 frame2.parent.parent 正是 i.com/main.html，且 frame2 和 main.html 同源，因此在 frame2 中可以通过 frame2.parent.parent 访问 main.html 的任何内容，调用 main.html 中的 JavaScript 函数。实现步骤如下：

(1)组件 a.com 通过在 frame1(a.com/proxy.html)中设置 frame2.src="http://i.com/proxy.html#data_toI"；

(2)frame2 下载页面 i.com/proxy.html。之后子页面的 onLoad 事件被激活，在 onLoad 事件中可通过 window.location.hash 取得组件 a.com 传过来的数据 data_toI。

(3)由于 frame2(i.com/proxy.html)和 i.com/main.html 同源，故在 frame2 中可调用 main.html 的 receiveFromA 函数将数据 data_toI 传递给 main.html。

但上述使用<frame>片段标识符的跨域数据访问存在一个安全问题：不确定数据的来源。当 main.html 通过 frame1.src 向组件 a.com 传送数据 data_here 时，由于同源

策略的安全机制，虽然可以保证数据能被组件 a.com 获得。但是，当组件 a.com 得到数据 data_here 时，没有直接的方法得知数据的来源。而在大多数情况下，不确定来源的数据是不安全的。因此，必须要有能实现对通信双方的身份进行验证的方法克服此安全漏洞。受 TCP 三次握手的启发，本文提出一种利用共享密钥技术在浏览器端实现跨域通信双方身份验证的方法，如图 3 所示。父页面 i.com/main.html 首先产生密钥 SK1，并使用<frame>片段标识符技术将 SK1 传递给子页面 frame1(a.com/proxy.html)。接收方 frame1 可通过将密钥 SK1 作为数据包的一部分传回 main.html 以证明接收方 frame1 确实来源为 a.com。但是 frame1 还需要证明发送方是来自于 i.com，因此，frame1 又产生一个新的密钥 SK2，并将 SK1 和 SK2 一起传送给 main.html，若 main.html 能将 SK2 作为密钥再传给 frame1 就能证明发送方 main.html 确实来自 i.com。至此，main.html 和 frame1 都拥有了密钥 SK2，故可将 SK2 作为共享密钥，并使用它来完成接下来的通信。图 3 所示 main.html 将“SK2+要传的数据”作为数据包传给 frame1，frame1 将先验证密钥是否等于 SK2，如果是，则接受传过来的数据。因此，对于恶意的攻击可以很容易被识别和去除。

图 3 利用密钥来验证身份

3.2 安全分析

从数据保密、身份验证和数据完整 3 个方面来分析上述基于<frame>片段标识符的跨域数据访问方案的安全性。

(1)数据保密。利用浏览器提供的同源策略限制和 frame.src 的只写特性来实现数据保密性。在父页面和<frame>之间传送的数据不会被父页面上其他的 DOM 元素看到，因为<frame>与父页面处在不同的源。而且数据不出现在浏览器缓存中，数据也不会跨过网络，因此可以认为数据包只能被接收的<frame>或来自 a.com 的其它页面观察到。

(2)身份验证。图 3 中利用共享密钥的技术能保证浏览器客户端跨域通信双方身份的验证。frame1 只有当其将父页面传过来的密钥 SK1 再传回父页面时，才能让它的身份得到验证；类似的，父页面也只有将密钥 SK2 传回 frame1 时，才能验证它的身份。之后，通过利用父页面和 frame1 的共享密钥 SK2 来保证双方通信的机密性。

(3)数据完整。攻击者篡改过的数据能被检测出来。要篡改通信的数据，攻击者需要知道共享密钥 SK2，否则在目的地，传过来的数据包将被拒绝并被丢弃。因此，没有经过身份验证的组件是不能篡改数据的。

当前所有浏览器端所实现的安全模型是同源策略，其导致了异源内容间只能是“完全信任”或“完全不信任”的结果。随之带来的问题是：Mashup 不能在实现功能性需求的同时也能很好地解决安全问题。因此，需要浏览器实现一种新的安全模型，使得浏览器能够支持合法的跨域数据访问，而无需牺牲终端用户的安全。但是，主流浏览器要实现新的安全模型并在业内普遍使用需要几年的时间，同时一些研究者提出的解决方案中或者要求改进当前的浏览器环境；或者不能提供一个灵活和安全的点到点的域间通信机制[5-6]。本文提出了一种新的域间通信机制，它不需要对当前的浏览器环境进行任何修改就可适用，并且主要考虑了 3 个安全因素：数据保密、身份验证和数据完整。在不远的将来，通过不断改善浏览器技术，及增加新的 Web 标准使 Mashup 更加安全，Mashup 方式也将会得到更广泛的应用。

参考文献

[1] 佚名.Garter：IT 业界未来四年的十大新技术[EB/OL]. http://networking.ctocio.com.cn/NetInformation/105/8147605.shtml，2008-06-03.

[2] Jyrki Hakkola. Mashup Security[EB/OL]. http://www.tml.tkk.fi/Opinnot/T-111.5550/2008/seminaari_hakkola_jyrki.pdf，2008-11-04.

[3] Aaron Bohannon. Building Secure Web Mashups [M/OL]. www.cis.upenn.edu/~bohannon/mashups.pdf，2008-7-16.

[4] 陈腊梅，李为.AJAx 跨域访问的研究与应用[J].计算机工程与设计，2008(11):5680-5684.

[5] Frederik De Keukelaere, Sumeer Bhola. SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers[EB/OL]. www2008.org/papers/pdf/p535-dekeukelaereA.pdf，2008-7.

[6] JACKSON C, WANG H J. Subspace: Secure CrossDomain Communicationfor Web Mashups [EB/OL]. www.collinjackson.com/research/papers/fp801-jackson.pdf.2007-5.

(收稿日期：2009-10-11)

作者简介：

陈丰，男，1971 年生，讲师，硕士，主要研究方向：信息系统开发、企业资源计划、数据仓库与数据挖掘和嵌入式系统。