

<<AJAX安全技术>>

图书基本信息

书名：<<AJAX安全技术>>

13位ISBN编号：9787121079306

10位ISBN编号：7121079305

出版时间：2009-1

出版时间：电子工业出版社

作者：霍夫曼

页数：403

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<AJAX安全技术>>

前言

在AJAX的面前，火、车轮、电的意义都显得淡然无光。从AJAX诞生的那一刻起，人们终于实现了梦寐以求的理想——在Web应用程序中刷新局部页面。相信那一天JamesGarrett站在浴室里时，一定是上帝给了他灵感才想到了这个词——AJAX。

但是像毁灭阿兹特克（Aztecs）的西班牙殖民者科尔蒂斯（Cortés），或者像《星球大战（StarWar）》前传中，最初被认为是救世主、最终却带来毁灭的达斯·维达一样，很多起初认为是美好的事物结局却不一定很好。

同样，对于AJAX来说，其安全上的巨大漏洞已经成为自身的最大威胁，如果任其发展，最终也会同前面的二者一样，产生混乱并最终毁于一旦。

在这巨大的恐惧面前，为了保护无辜的人们、战胜邪恶并重新恢复宇宙中的和平，终于有两个人站了出来，他们便是Billy和Bryan。

衷心感谢你阅读这本书。

<<AJAX安全技术>>

内容概要

一本防范AJAX安全漏洞的实用指南。

如今，越来越多的网站都被改写成AJAX应用程序，甚至传统的桌面软件也通过AJAX，迅速转向了Web领域。

但是在这个过程中，人们通常都没有考虑到安全的问题。

如果不恰当地设计、编写了AJAX应用程序，那么它们会比传统桌面程序存在更多的安全漏洞。

AJAX开发人员无时无刻都希望有一本指南，能够指导他们如何来保护自己的应用程序——他们终于等到了这一天。

《AJAX安全技术》一书，系统地分析了当今最危险的AJAX漏洞用现实中的代码阐述了大量关键性的安全理念，并对实际中的案例，例如MySpace的Samy蠕虫病毒，进行了详尽分析。

更重要的是，不管你使用何种主流的Web编程语言和环境，例如 .NET、Java或PHP，本书都给出了许多具体、前沿的建议。

通过本书你将了解到以下几点： 如何刚氏AJAX特有的安全风险，包括过度细分的Web服务、应用程序控制流程篡改以及对程序逻辑的操控。

如何预防针对AJAX的攻击手段，包括JavaScript劫持、持久化存储窃取以及对mashup程序的渗透。

如何避免基于XSS和SQL注入的攻击，包括由AJAX衍生出来的SQL注入攻击（只需要两次请求就可以暴露整个后台数据库）。

如何使用Google Gears和Dojo开发安全的离线AJAX应用程序。

如何发现Prototype、DWR及ASRNET AJAX等AJAX框架中的安全问题以及我们自己仍需实现哪些功能。

如何更安全地编写AJAX代码，如何确定并修改已有代码中的安全缺陷。

不管是编写或者维护AJAX应用程序的开发人员、架构师，还是打算或正在设计新AJAX程序的项目经理，以及包括QA和渗透测试人员在内的所有软件安全人士，《AJAX安全技术》一书都是必不可少的。

<<AJAX安全技术>>

作者简介

Jesse James Garrett——Adaptive PATH公司主席及创始人。

Billy Hoffman是惠普安全实验室的首席安全研究员，专注于如何自动发掘Web应用程序中的漏洞。他经常在Black Hat、RSA、Toorcon、Shmoocon、Infosec及AJAXWorld等会议上发表演讲，也曾受到过FBI的邀请进行演讲。

<<AJAX安全技术>>

书籍目录

第1章 AJAX安全介绍11.1 AJAX基础知识21.1.1 什么是AJAX21.1.2 动态HTML (DHTML) 101.2 AJAX架构 (Architecture) 的转变过程111.2.1 胖客户端架构111.2.2 瘦客户端架构121.2.3 AJAX : 最适合的架构141.2.4 从安全角度看胖客户端应用程序151.2.5 从安全角度看瘦客户端应用程序151.2.6 从安全角度看AJAX架构171.3 一场完美的攻击风暴171.3.1 不断增加的复杂度、透明度及代码量181.3.2 社会学问题201.3.3 AJAX应用程序 : 富有吸引力的、战略上的目标211.4 本章小结22第2章 劫持232.1 攻击HighTechVactions.net242.1.1 攻击票务系统242.1.2 攻击客户端数据绑定302.1.3 攻击AJAXAPI342.2 黑夜中的盗窃39第3章 Web攻击413.1 基本攻击分类413.1.1 资源枚举413.1.2 参数操纵453.2 其他攻击663.2.1 跨站请求伪造攻击663.2.2 钓鱼攻击683.2.3 拒绝服务 (Denial-of-Service , DoS) 683.3 保护Web应用程序免受资源枚举和参数操作的攻击693.4 本章小结70第4章 AJAX攻击层面714.1 什么是攻击层面714.2 传统Web应用程序的攻击层面724.2.1 表单输入734.2.2 cookie744.2.3 报头754.2.4 隐藏的表单输入754.2.5 请求参数764.2.6 上传文件784.3 传统的Web应用程序攻击 : 一份成绩单794.4 Web服务的攻击层面814.4.1 Web服务的方法814.4.2 Web服务的定义824.5 AJAX应用程序的攻击层面834.5.1 AJAX应用程序攻击层面的来源844.5.2 黑客的最爱864.6 正确的输入验证864.6.1 有关黑名单及其他补丁的问题874.6.2 治标不治本904.6.3 白名单输入验证934.6.4 正则表达式964.6.5 关于输入验证的其他想法964.7 验证富客户端的用户输入984.7.1 验证标记语言984.7.2 验证二进制文件1004.7.3 验证JavaScript源代码1004.7.4 验证序列化数据1064.8 关于由用户提供的内容1094.9 本章小结110第5章 AJAX代码的复杂性115.1 多种语言和架构115.1.1 数组索引115.1.2 字符串操作1135.1.3 代码注释115.1.4 事不关己, 高高挂起115.2 JavaScript的怪异之处1175.2.1 解释, 而不是编译1175.2.2 弱类型1185.3 异步性1205.3.1 竞争条件1205.3.2 死锁及哲学家用餐问题1245.3.3 客户端同步化1275.3.4 留意你所采纳的建议1285.4 本章小结129第6章 AJAX应用程序的透明度1316.1 黑盒对白盒1316.1.1 示例 : mylocalweatherforecast.com1336.1.2 示例 : 用AJAX实现的mylocalweatherforecast.com1356.1.3 对比结果1396.2 像API一样的Web应用程序1406.3 一些特殊的安全错误1416.3.1 不恰当的身份认证1416.3.2 过度细化服务端API1436.3.3 在JavaScript中存储会话状态1466.3.4 与用户相关的敏感数据1476.3.5 包含在客户端的注释及文档1486.3.6 在客户端进行的数据转换1496.4 通过隐藏来保证安全1526.5 本章小结154第7章 劫持AJAX应用程序1557.1 劫持AJAX框架1557.1.1 意外的方法冲突1567.1.2 人为的方法冲突1587.2 劫持“即时”的AJAX1637.3 劫持JSONAPI1677.3.1 劫持对象定义1727.3.2 JSON劫持的根源1737.3.3 如何防范JSON劫持1737.4 本章小结176第8章 攻击客户端存储1798.1 客户端存储系统概述1798.2 HTTPcookies1818.2.1 cookie访问控制规则1838.2.2 HTTPcookie的存储能力1888.2.3 cookie的生命期1918.2.4 cookie存储的其他安全问题1928.2.5 cookie存储总结1938.3 Flash本地共享对象1948.4 DOM存储2018.4.1 会话存储2028.4.2 全局存储2048.4.3 DOM存储的细节讨论2058.4.4 DOM存储安全2078.4.5 DOM存储总结2088.5 InternetExplorersUserData2098.6 一般客户端存储的攻击和防范方法2148.6.1 跨域攻击2148.6.2 跨目录攻击2158.6.3 跨端口攻击2168.7 本章小结216第9章 离线AJAX应用程序2199.1 离线AJAX应用程序2199.2 GoogleGears2209.2.1 GoogleGears内置的安全特性及其缺点2219.2.2 探索工作者池2249.2.3 泄露并篡改本地服务器 (LocalServer) 中的数据2269.2.4 直接访问GoogleGears数据库2299.2.5 SQL注入和GoogleGears2309.2.6 客户端SQL注入有多危险2349.3 Dojo.Offline2369.3.1 保证密钥安全2379.3.2 保证数据安全2389.3.3 可作为密钥的良好密码2399.4 再论客户端输入验证2409.5 创建离线应用程序的其他方式2419.6 本章小结242第10章 请求来源问题24310.1 Robots、Spiders、Browsers及其他网络爬虫24310.2 请求来源不确定性和JavaScript24510.2.1 从Web服务器的角度看AJAX请求24610.2.2 是你自己, 还是貌似你的某人24910.2.3 使用JavaScript发送HTTP请求25110.2.4 在AJAX出现之前的JavaScriptHTTP攻击25210.2.5 通过XMLHttpRequest窃取其他内容25410.2.6 实战结合XSS/XHR进行攻击25810.3 防范措施26010.4 本章小结261第11章 WebMashup和聚合程序26311.1 互联网上计算机可以使用的的数据26311.1.1 20世纪90年代早期 : 人类Web的黎明26311.1.2 20世纪90年代中期 : 机器Web的诞生26411.1.3 2000年左右 : 机器Web逐渐成熟26611.1.4 可公用的Web服务26611.2 Mashup : Web中的弗兰肯斯坦26811.2.1 ChicagoCrime.org26911.2.2 HousingMaps.com27011.2.3 其他的Mashup应用程序27011.3 创建Mashup应用程序27111.4 桥接、代理及网关27411.5 攻击AJAX代理27511.6 Mashup程序中的输入验证27911.7 聚合网站28211.8 安全性和可信度的降低28711.9 本章小结290第12章 攻击表现层29112.1 从内容

<<AJAX安全技术>>

信息中分离表现信息29112.2 攻击表现层29412.3 对级联样式表的数据挖掘29512.4 外观篡改29712.5 嵌入程序逻辑30512.6 目标级联样式表30612.7 防范表现层攻击31112.8 本章小结312第13章 JavaScript蠕虫31313.1 JavaScript蠕虫概述31313.1.1 传统的计算机病毒31413.1.2 JavaScript蠕虫31613.2 创建JavaScript蠕虫31813.2.1 JavaScript的局限性31913.2.2 传播JavaScript蠕虫32013.2.3 JavaScript蠕虫携带的恶意代码32013.2.4 JavaScript中的信息窃取32113.3 关于内网32213.3.1 窃取浏览器历史记录32613.3.2 窃取搜索引擎的查询结果32713.3.3 总结32813.4 案例学习：Samy蠕虫32913.4.1 工作原理33013.4.2 病毒携带的程序33313.4.3 关于Samy蠕虫的结论33413.5 案例学习：Yamanner蠕虫（JS/Yamanner-A）33613.5.1 工作原理33713.5.2 病毒携带的程序33913.5.3 关于Yamanner蠕虫的结论34013.6 从实际JavaScript蠕虫中能学到的经验34213.7 本章小结343第14章 测试AJAX应用程序34514.1 黑魔法34514.2 并不是所有人都使用浏览器来查看网页34914.3 两手都要抓，两手都要硬35114.4 安全测试工具35214.4.1 生成网站目录35314.4.2 漏洞检测35414.4.3 分析工具：Sprajax35614.4.4 分析工具：ParosProxy35714.4.5 分析工具：LAPSE（Eclipse中轻量级的程序安全分析工具）35914.4.6 分析工具：WebInspectTM36014.5 其他一些关于安全测试的想法361第15章 AJAX框架分析36315.1 ASP.NET36315.1.1 ASP.NETAJAX（以前被称为Atlas）36315.1.2 ScriptService36715.1.3 安全缺点：UpdatePanel对ScriptService36815.1.4 ASP.NET和WSDL36915.1.5 ValidateRequest37315.1.6 ViewStateUserKey37415.1.7 ASP.NET配置和调试37515.2 PHP37615.2.1 Sajax37615.2.2 Sajax和跨站请求伪造37815.3 JavaEE38015.4 JavaScript框架38215.4.1 对客户端代码的一个警告38315.4.2 Prototype38315.5 本章小结385附录A Samy蠕虫源代码387附录B Yamanner蠕虫源代码397

章节摘录

第1章 AJAX安全介绍 1.1 AJAX基础知识 1.1.1 什么是AJAX 通常来说，浏览器为了动态显示Web页面，会把整个页面通过一个请求发送给服务器。服务端应用程序接收到响应后，会创建该页面的HTML代码，并返回给浏览器。浏览器会丢弃掉当前的页面，并显示新的HTML页面。这样，用户才能够在浏览器中看见新的页面，并与其进行交互。

虽然这个过程十分简单，但是却非常浪费资源。通常，服务端为客户端生成的新页面几乎与被其丢弃的当前页面一样。在网络中传输的整个页面占据了大量的带宽，而这根本就是毫无必要的。同时，在请求处理的过程中，用户也无法再使用应用程序，他们不得不等着服务端返回响应信息。当服务端最终将响应信息返回给浏览器时，浏览器在重新加载页面的同时，还不得不闪烁一下。

如果Web客户端只需要请求页面中的一小部分，而不是向服务端发送整个页面的请求，那么对各个方面都会带来益处。

服务端可以更快速地处理请求，并且发送响应信息时会占用更少的带宽。同时，客户端由于请求的整体时间变短，不仅可以为用户提供交互性更好的界面，也可以消除由重新加载整个页面而导致的闪烁。

应该说，AJAX是为了解决该问题而产生的众多技术的融合，它使得Web应用程序的客户端可以不断地从Web服务端更新部分页面。

而用户也不必再提交表单，或者离开当前的页面。

客户端的脚本代码（通常都是JavaScript）可以向页面的部分片段（Fragment）发起异步的，或者非阻塞（Non-blocking）的请求。

这些片段可以是一些原始的数据，在客户端再被转换成HTML代码；也可以本身就是HTML代码，直接被插入到浏览器的文档（Document）对象中。

不管怎样，在服务端完成对请求的处理，并将响应片段返回给客户端浏览器之后，客户端的脚本代码都会使用这些数据来修改页面中的文档对象模型（Document Object Model，DOM）。

<<AJAX安全技术>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>