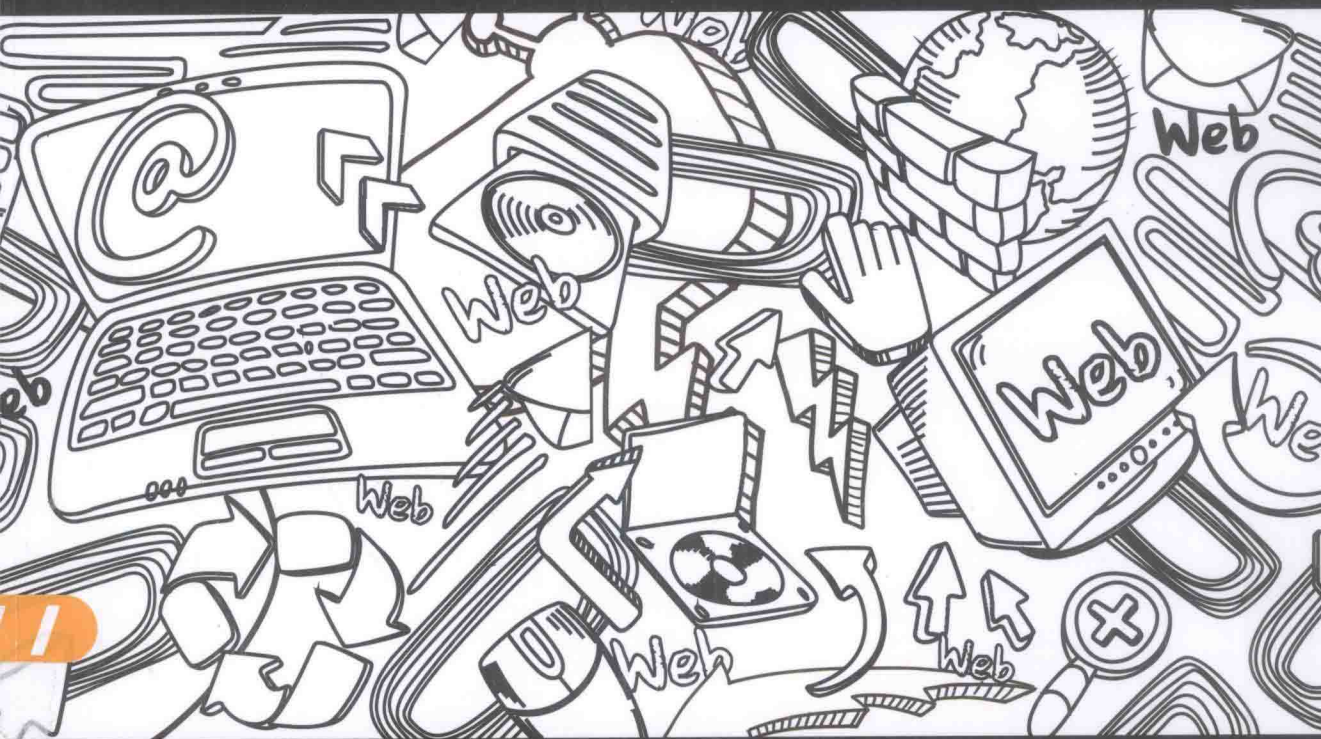


Web安全 深度剖析



张炳帅 编著

Web安全深度剖析

本书根据作者多年经验积累写成，作者从一个渗透测试者的角度，深入浅出地剖析了一个网站或一家企业会遇到的安全问题。最终本书解答了一个很多人会感兴趣的问题：服务器到底是怎么被黑掉的？相信这对所有开展互联网业务的公司，以及所有想从事安全行业的初学者来说，都是一本很好的指南手册。

——吴翰清 阿里巴巴集团研究员，《白帽子讲 Web 安全》作者

《Web 安全深度剖析》书如其名，是一本极具实用性、贴近实战的 Web 安全指导书籍，覆盖了 Web 渗透测试中实际遇到的各类安全漏洞、利用技巧和攻击场景，能够帮助安全从业者和爱好者快速了解并建立 Web 渗透测试所需的知识与技能，值得深入研读。

——诸葛建伟 清华大学副研究员，蓝莲花战队领队



博文视点Broadview



新浪微博
weibo.com

@博文视点Broadview

上架建议：计算机/网络安全

ISBN 978-7-121-25581-6



定价：59.00元

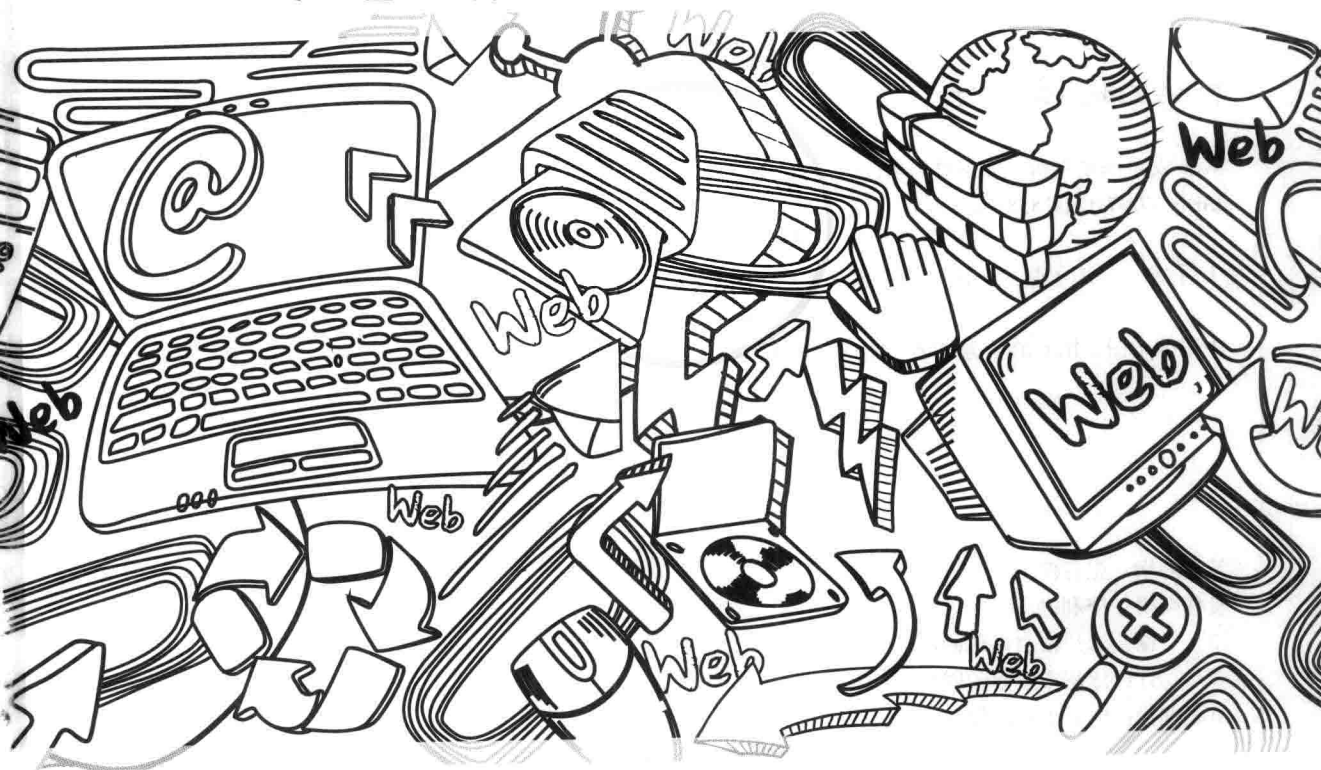


策划编辑：张月萍
责任编辑：李利
封面设计：吴海燕

需要整本电子书的，联系QQ：2667271557

需要整本电子书的，联系QQ：2667271557

Web安全 深度剖析



张炳帅 编著

电子工业出版社
Publishing House of Electronics Industry
北京·BEIJING

需要整本电子书的，联系QQ：2667271557

需要整本电子书的，联系QQ：2667271557

内 容 简 介

本书总结了当前流行的高危漏洞的形成原因、攻击手段及解决方案，并通过大量的示例代码复现漏洞原型，制作模拟环境，更好地帮助读者深入了解 Web 应用程序中存在的漏洞，防患于未然。

本书从攻到防，从原理到实战，由浅入深、循序渐进地介绍了 Web 安全体系。全书分 4 篇共 16 章，除介绍 Web 安全的基础知识外，还介绍了 Web 应用程序中最常见的安全漏洞、开源程序的攻击流程与防御，并着重分析了“拖库”事件时黑客所使用的攻击手段。此外，还介绍了渗透测试工程师其他的一些检测方式。

本书最适合渗透测试人员、Web 开发人员、安全咨询顾问、测试人员、架构师、项目经理、设计等人员阅读，也可以作为信息安全等相关专业的教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Web 安全深度剖析 / 张炳帅编著. —北京：电子工业出版社，2015.4
ISBN 978-7-121-25581-6

I. ①W... II. ①张... III. ①互联网络—安全技术 IV. ①TP393.408

中国版本图书馆 CIP 数据核字（2015）第 036493 号

策划编辑：张月萍

责任编辑：李利健

印 刷：涿州市京南印刷厂

装 订：涿州市京南印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×1092 1/16 印张：22.5 字数：590 千字

版 次：2015 年 4 月第 1 版

印 次：2015 年 4 月第 1 次印刷

印 数：3500 册 定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zltts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

需要整本电子书的，联系QQ：2667271557

推荐序

纵观国内网络安全方面的书籍，大多数都是只介绍结果，从未更多地考虑过程。而本书恰是从实用角度出发，本着务实的精神，先讲原理，再讲过程，最后讲结果，是每个从事信息安全的从业人员不可多得的一本实用大全。尤其是一些在企业从事信息安全的工作人员，可以很好地依据书中的实际案例进行学习，同时，在校学生也可以依据本书的案例进行深入学习，有效地贴近企业，更好地有的放矢。

——陈亮 OWASP 中国北京主负责人

我有幸见证了《Web 安全深度剖析》诞生的全过程，作者认真严谨的写作风格，深入求证的研究态度，深厚的程序员功底，丰富的网络和现场教育培训经验，使本书成为适合 Web 渗透测试的必选作品。本书内容丰富，知识点全面，适合网络安全爱好者和从业者学习研究。

——一剑西来 红黑联盟站长，暗影团队管理员

我收到《Web 安全深度剖析》样章后，一口气通读下来，感觉酣畅淋漓。作者用深入浅出的手法，贴近实战，基本涵盖了 Web 安全技术中实际遇到的方方面面。本书适合 Web 安全从业人员研读，也推荐有志在 Web 安全方向发展的人学习。

——lake2 腾讯安全平台部副总监

与其说这是一本 Web 安全的书籍，不如说是一本渗透实战教程，该书总结了不少常见的 Web 渗透思路和奇技淫巧，非常适合初学者和有一些基础的人阅读。安全圈有一句老话：未知攻，焉知防。这本书可以帮助大家找到学习安全知识的兴趣，也可以找到学习安全知识的方法。

——林伟（网名：陆羽）

360 网络攻防实验室负责人，国内知名安全社区 T00ls.net 创始人之一

本书根据作者多年工作经验积累写成，作者从一个渗透测试者的角度，深入浅出地剖析了一个网站或一家企业会遇到的安全问题。最终本书解答了一个很多人感兴趣的问题：服务器到底是怎么被黑掉的？相信这对所有开展互联网业务的公司，以及所有想从事安全行业的初学者来说，都是一本很好的学习指南。

——吴翰清 阿里巴巴集团研究员，《白帽子讲 Web 安全》作者

终于等到张炳帅这本书了，他拥有非常丰富的实战经验，这本书里的“干货”绝对值得细细品味，我期待已久！

——余弦 知道创宇技术副总裁

《Web 安全深度剖析》书如其名，是一本极具实用性、贴近实战的 Web 安全指导书籍，覆盖了 Web 渗透测试中实际遇到的各类安全漏洞、利用技巧和攻击场景，能够帮助安全从业者和爱好者快速了解并建立 Web 渗透测试所需的知识与技能，值得深入研读。

——诸葛建伟 清华大学副研究员，蓝莲花战队领队

前言

本书总结了当前流行的高危漏洞的形成原因、攻击手段及解决方案，并通过大量的示例代码复现漏洞原型，制作模拟环境，更好地帮助读者深入了解 Web 应用程序中存在的漏洞，防患于未然。

本书抛开一些研究性、纯理论性的内容，也就是外表看似很高端，但实用性不大的课题，所总结的漏洞知识可以说是刀刀见血、剑剑穿心。漏洞直接危害到企业的安全。

本书也是笔者多年来的工作总结，几乎每个场景都是最常见的，如果你从事与 Web 渗透测试相关的工作，就会遇到本书中的场景。

本书结构

本书从攻到防，从原理到实战，由浅入深、循序渐进地介绍了 Web 安全体系。全书分 4 篇共 16 章，这是一个庞大的体系，几乎可以囊括目前常见的一切 Web 安全类技术。

本书目录结构就非常像渗透测试人员的一次检测流程，从信息探测到漏洞扫描、漏洞利用、提权等。

基础篇

第 1 章到第 4 章为基础篇，是整个 Web 安全中最基础的技术。

第 1 章描述了服务器是如何被黑客入侵的，并从中引出 Web 安全的概念，同时也告诉读者如何更快、更好地学习 Web 安全。

第 2 章详细讲述了 Web 安全的一个核心知识点：HTTP 协议。如果是零基础的读者，建议一定要多看 HTTP 协议，因为后续章节中的许多内容都会涉及 HTTP 协议。

第 3 章介绍了信息探测的知识点。渗透测试人员工作时，一般都是从信息探测入手的，也就是常说的踩点。信息探测是渗透测试的基本功，是必须学习的内容。本章介绍了 Google Hack、Nmap、DirBuster、指纹识别等技术。

第 4 章讲解了渗透测试人员常用的安全测试工具，包括：BurpSuite、AWVS、APPSCAN 等工具。

原理篇

第 5 章到第 10 章为原理篇，阅读本篇内容需要读者具备一定的代码功底。在这些章节中讲述了 Web 应用程序中最常见的安全漏洞。笔者将这些常见的高危漏洞提取出来，每个漏洞作为单独的一个章节来讲解，从原理到利用。

第 5 章是 SQL 注入章节，讨论了 MySQL、SQL Server、Oracle 数据库的注入方式、注入技巧和不同数据库的注入差异。

攻击者对数据库注入的目的有：数据窃取、文件读写、命令执行。掌握其核心思想后，对 SQL 注入的学习就比较容易了。

在讲解 SQL 注入原理后，介绍了 SQLMap、Havij 等注入工具，同时也介绍了绕过部分 WAF 的思路。

第 6 章介绍了 XSS 攻击，其中讲解了 XSS 的形成原理、三种 XSS 类型、会话劫持、蠕虫等前端技术，最后提出了 XSS 有效的解决方案。

第 7 章讲解了上传漏洞和 Web 容器的漏洞。有时候程序是没有问题的，但如果与 Web 容器漏洞相结合，可能就会造成上传漏洞。

第 8 章描述了命令执行漏洞的形成原因和利用方式，同时也介绍了 Struts2 命令执行漏洞及命令执行漏洞的修复方案。

第 9 章讲解了 PHP 包含漏洞的原理和利用方式，同时也介绍了包含漏洞的修复方案。

第 10 章讨论的知识点比较广泛，比如 CSRF、逻辑漏洞、远程部署漏洞、代码注入等高危漏洞。

实战篇

第 11 章讲述了开源程序的攻击流程与防御，并着重分析了“拖库”事件时黑客所使用的攻击手段。

综合篇

如果仅仅掌握 Web 安全漏洞，而对其他漏洞、攻击手法一窍不通，是无法全面找出漏洞的。本书在综合篇里介绍了渗透测试工程师的一些其他检测方式。

第 12 章详细讲述了暴力破解的测试方式，分别使用 Hydra、Burp Suite、Medusa 等工具对 MSSQL、MySQL、Web 应用程序进行破解，最后讲述了验证码的安全性及防止暴力破解的解决方案。

第 13 章讲述了旁注攻击。当目标 Web 应用程序无法寻找到漏洞时，攻击者常常会使用旁注攻击来入侵目标。本章剖析了旁注攻击的几个关键点，包括 IP 逆向查询、SQL 跨库查询、绕

过 CDN 等技术。

第 14 章讲述了提权。服务器提权可以更好地解释服务器的脆弱性，本章对 Linux、Windows 提权均做了分析。比如 Windows 下的三种提权方式：本地溢出提权、第三方组件提权和系统关键点利用。另外，也剖析了一部分提权时的采用手段，比如 DLL 劫持、端口转发、服务器添加后门等技术。

第 15 章讲述了 ARP 攻击与防御。安全是一个整体，并不是 Web 应用程序找不到漏洞时，黑客就没办法了，黑客使用 ARP 欺骗技术可以轻松劫持到你的密码。本章从 ARP 协议开始讲解，接着深入讲解 ARP 欺骗的原理，其中介绍了 Cain、Ettercap、NetFuke 等嗅探工具。

第 16 章讲述了社会工程学。社会工程学可以说是 APT 攻击中的关键一环，也被称为没有“技术”却比“技术”更强大的渗透方式。

需要的工具

本书的核心是从原理到实战案例的剖析，很多时候，工具只是起辅助作用。读者要注意一点：在实际的渗透中，更多地靠经验、思路，工具反而是其次，不要被众多的“神器”所迷惑。工具仅仅是让我们更方便、高效一些，工具是“死”的，目前的软件开发水平还完全达不到智能化，工具只能按照程序员的思维流程来执行。所以，我们完全依赖的还是自己的大脑。

本书所使用的工具可以在 <http://www.secbug.org/tools/index.html> 中下载。

本书是写给谁的

本书最适合渗透测试人员、Web 开发人员、安全咨询顾问、测试人员、架构师、项目经理、设计等人员阅读，也可以作为 Web 安全、渗透测试的教材。这是一本实用的 Web 安全教材。

- 渗透测试人员：渗透测试人员要求具备的技术在大学并没有课程设置，也没有正规、专业的技术培训。可以说，做渗透测试的人员都要靠自学，付出比其他人更多的努力才能胜任这个工作。笔者希望读者从本书中学到知识，进一步提高自己的渗透测试水平。
- Web 开发人员：程序员不一定是黑客，但是有一定水平的黑客、白帽子一定是程序员。因此，一个合格的程序员学习安全知识是非常快的。本书介绍了大量的示例代码，并分析了其中的漏洞，从开发人员的角度讲述如何避免和修复漏洞，希望开发人员能够通过本书的学习提高自己防御安全的水平，站在新的高度去看待程序。
- 信息安全相关专业的学生：本书也适合信息安全等相关专业的学生阅读，书中所有的知识点几乎都是从零开始的，你们可以循序渐进地学习。同时，笔者也希望能给大学老师带来一些灵感，然后培育出更多的网络安全人才。

在学习时，笔者常把原理性的知识比喻为内功，而具体的实操、技术点比喻为招式，只有招式而没有内功是根本无法变成高手的，有了内功和招式才可能成为高手。

安全是把双刃剑。剑在手中，至于是用其来做好事还是做坏事，只在于一念之差。笔者强烈要求各位读者仅在法律的许可范围内使用本书所提供的信息。

致谢

感谢 EvilShad0w 团队的每一位成员，你们在一起交流技术、讨论心得时从来都是无私地分享，你们都有一颗对技术狂热的心。在我眼里，你们都是“技术帝”。

感谢破晓团队的每一位成员，感谢你们相信我，愿意跟我一起闯，你们的存在是支撑我继续下去的力量。

感谢联合实验室的成员，是你们在百忙之中细细品味这本书，并指出不足之处。

感谢袁海君、杜萌萌、LiuKer、7z1、天蓝蓝、小 K、小歪、岩少、晴天小铸、GBM 的支持。有你们的支持，我才能完成这本书的写作，也感谢你们对这本书做出的贡献，我将谨记于心。这里要特别感谢小杜，你为我审阅稿子，找出书中的许多错误。

感谢红黑联盟站长一剑西来、天云祥科技有限公司 CEO 杨奎，你们给了我许多机会，也教会了我如何去思考。

感谢我的领导沈局、邬江、邓小刚，你们对待我就像对待自己的学生一样，给了我许多教导。

最后，感谢我的父母，感谢你们将我抚育成人，为我付出一切。这份爱时刻提醒着我，要努力、要上进！

路虽艰，行则必达。事虽难，做则必成。

破晓-SecBug.Org

目 录

第 1 篇 基础篇

- 第 1 章 Web 安全简介 2
 - 1.1 服务器是如何被入侵的 2
 - 1.2 如何更好地学习 Web 安全 4
- 第 2 章 深入 HTTP 请求流程 6
 - 2.1 HTTP 协议解析 6
 - 2.1.1 发起 HTTP 请求 6
 - 2.1.2 HTTP 协议详解 7
 - 2.1.3 模拟 HTTP 请求 13
 - 2.1.4 HTTP 协议与 HTTPS 协议的区别 14
 - 2.2 截取 HTTP 请求 15
 - 2.2.1 Burp Suite Proxy 初体验 15
 - 2.2.2 Fiddler 19
 - 2.2.3 WinSock Expert 24
 - 2.3 HTTP 应用：黑帽 SEO 之搜索引擎劫持 24
 - 2.4 小结 25
- 第 3 章 信息探测 26
 - 3.1 Google Hack 26
 - 3.1.1 搜集子域名 26
 - 3.1.2 搜集 Web 信息 27
 - 3.2 Nmap 初体验 29
 - 3.2.1 安装 Nmap 29
 - 3.2.2 探测主机信息 30
 - 3.2.3 Nmap 脚本引擎 32

3.3	DirBuster	33
3.4	指纹识别	35
3.5	小结	38
第 4 章	漏洞扫描	39
4.1	Burp Suite	39
4.1.1	Target	39
4.1.2	Spider	40
4.1.3	Scanner	42
4.1.4	Intruder	43
4.1.5	辅助模块	46
4.2	AWVS	49
4.2.1	WVS 向导扫描	50
4.2.2	Web 扫描服务	52
4.2.3	WVS 小工具	53
4.3	AppScan	54
4.3.1	使用 AppScan 扫描	55
4.3.2	处理结果	58
4.3.3	AppScan 辅助工具	58
4.4	小结	61

第 2 篇 原理篇

第 5 章	SQL 注入漏洞	64
5.1	SQL 注入原理	64
5.2	注入漏洞分类	66
5.2.1	数字型注入	66
5.2.2	字符型注入	67
5.2.3	SQL 注入分类	68
5.3	常见数据库注入	69
5.3.1	SQL Server	69
5.3.2	MySQL	75
5.3.3	Oracle	84
5.4	注入工具	89
5.4.1	SQLMap	89
5.4.2	Pangolin	95
5.4.3	Havij	98
5.5	防止 SQL 注入	99
5.5.1	严格的数据类型	100

5.5.2	特殊字符转义	101
5.5.3	使用预编译语句	102
5.5.4	框架技术	103
5.5.5	存储过程	104
5.6	小结	105
第 6 章	上传漏洞	106
6.1	解析漏洞	106
6.1.1	IIS 解析漏洞	106
6.1.2	Apache 解析漏洞	109
6.1.3	PHP CGI 解析漏洞	110
6.2	绕过上传漏洞	110
6.2.1	客户端检测	112
6.2.2	服务器端检测	115
6.3	文本编辑器上传漏洞	123
6.4	修复上传漏洞	127
6.5	小结	128
第 7 章	XSS 跨站脚本漏洞	129
7.1	XSS 原理解析	129
7.2	XSS 类型	130
7.2.1	反射型 XSS	130
7.2.2	存储型 XSS	131
7.2.3	DOM XSS	132
7.3	检测 XSS	133
7.3.1	手工检测 XSS	134
7.3.2	全自动检测 XSS	134
7.4	XSS 高级利用	134
7.4.1	XSS 会话劫持	135
7.4.2	XSS Framework	141
7.4.3	XSS GetShell	144
7.4.4	XSS 蠕虫	149
7.5	修复 XSS 跨站漏洞	151
7.5.1	输入与输出	151
7.5.2	HttpOnly	158
7.6	小结	160
第 8 章	命令执行漏洞	161
8.1	OS 命令执行漏洞示例	161
8.2	命令执行模型	162

8.2.1	PHP 命令执行	163
8.2.2	Java 命令执行	165
8.3	框架执行漏洞	166
8.3.1	Struts2 代码执行漏洞	166
8.3.2	ThinkPHP 命令执行漏洞	169
8.4	防范命令执行漏洞	169
第 9 章	文件包含漏洞	171
9.1	包含漏洞原理解析	171
9.1.1	PHP 包含	171
9.1.2	JSP 包含	180
9.2	安全编写包含	184
9.3	小结	184
第 10 章	其他漏洞	185
10.1	CSRF	185
10.1.1	CSRF 攻击原理	185
10.1.2	CSRF 攻击场景（GET）	186
10.1.3	CSRF 攻击场景（POST）	188
10.1.4	浏览器 Cookie 机制	190
10.1.5	检测 CSRF 漏洞	193
10.1.6	预防跨站请求伪造	197
10.2	逻辑错误漏洞	199
10.2.1	挖掘逻辑漏洞	199
10.2.2	绕过授权验证	200
10.2.3	密码找回逻辑漏洞	204
10.2.4	支付逻辑漏洞	205
10.2.5	指定账户恶意攻击	209
10.3	代码注入	210
10.3.1	XML 注入	211
10.3.2	XPath 注入	212
10.3.3	JSON 注入	215
10.3.4	HTTP Parameter Pollution	216
10.4	URL 跳转与钓鱼	218
10.4.1	URL 跳转	218
10.4.2	钓鱼	220
10.5	WebServer 远程部署	224
10.5.1	Tomcat	224
10.5.2	JBoss	226

10.5.3 WebLogic 229

10.6 小结 233

第 3 篇 实战篇

第 11 章 实战入侵与防范 236

11.1 开源程序安全剖析 236

11.1.1 0day 攻击 236

11.1.2 网站后台安全 238

11.1.3 MD5 还安全吗 243

11.2 拖库 248

11.2.1 支持外连接 248

11.2.2 不支持外连接 253

11.3 小结 262

第 4 篇 综合篇

第 12 章 暴力破解测试 264

12.1 C/S 架构破解 265

12.2 B/S 架构破解 272

12.3 暴力破解案例 275

12.4 防止暴力破解 277

12.5 小结 278

第 13 章 旁注攻击 279

13.1 服务器端 Web 架构 279

13.2 IP 逆向查询 280

13.3 SQL 跨库查询 282

13.4 目录越权 283

13.5 构造注入点 284

13.6 CDN 286

13.7 小结 288

第 14 章 提权 290

14.1 溢出提权 290

14.2 第三方组件提权 294

14.2.1 信息搜集 294

14.2.2 数据库提权 296

14.2.3 FTP 提权 302

14.2.4 PcAnywhere 提权 312

14.3 虚拟主机提权 314

14.4	提权辅助.....	315
14.4.1	3389 端口.....	315
14.4.2	端口转发.....	318
14.4.3	启动项提权.....	320
14.4.4	DLL 劫持.....	321
14.4.5	添加后门.....	322
14.5	服务器防提权措施.....	324
14.6	小结.....	325
第 15 章	ARP 欺骗攻击.....	326
15.1	ARP 协议简介.....	326
15.1.1	ARP 缓存表.....	326
15.1.2	局域网主机通信.....	327
15.1.3	ARP 欺骗原理.....	328
15.2	ARP 攻击.....	329
15.2.1	Cain.....	329
15.2.2	Ettercap.....	332
15.2.3	NetFuke.....	336
15.3	防御 ARP 攻击.....	339
15.4	小结.....	340
第 16 章	社会工程学.....	341
16.1	信息搜集.....	341
16.2	沟通.....	343
16.3	伪造.....	344
16.4	小结.....	345
严正声明	346

第 1 篇

基础篇

- 第 1 章 Web 安全简介
- 第 2 章 深入 HTTP 请求流程
- 第 3 章 信息探测
- 第 4 章 漏洞扫描

第 1 章

Web 安全简介

1.1 服务器是如何被入侵的

在介绍 Web 安全的内容之前，我们先了解一下一台在互联网中的服务器是如何被攻击者入侵的。

攻击者想要对计算机进行渗透，有一个条件是必需的：就是攻击者的计算机与服务器必须能够正常通信。服务器提供各种服务供客户端使用，那么此时服务器是如何与客户端通信的？依靠的就是端口。攻击者入侵也是靠端口，或者说是计算机提供的服务。当然不排除一些“物理黑客”，直接进入服务器所在的机房对服务器动手。

过去的黑客攻击方式大多数都是直接针对目标进行攻击，比如端口扫描、一些服务的密码爆破（如：FTP、数据库）、缓冲区溢出攻击等方式直接获取目标权限，在 2000 年至 2008 年，使用溢出软件扫描主机，在 100 台计算机中可能会有 20 台计算机中招，可见服务器有多么脆弱。如今，这种直接对服务器进行溢出攻击的方式越来越少，因为系统的溢出漏洞太难挖掘了，新的战场已转移到 Web 之上。

早期的互联网是非常单调的，一般只有静态的文档，随着技术的发展，互联网慢慢变得多姿多态，每个人都可以在互联网中遨游，向网友“诉说”。小学时教科书上所说的“地球村”也真正实现了。

如今的 Web 应该称之为 Web 应用程序，与早期的 Web 有天壤之别，如今的 Web 功能非常强大，网上购物、办公、游戏、社交等活动都不在话下，而使用者（客户端）需要做的仅仅是拥有一个浏览器，就可做到这么多任务。

是什么让 Web 如此强大？它离不开四个要点：数据库、编程语言、Web 容器和优秀的 Web 应用程序的设计者，这四个缺一不可。

优秀的设计人员设计个性化的程序，编程语言将这些设计变为真实的存在，且悄悄地与数据库连接，让数据库存储好这些数据，而 Web 容器负责的则是作为终端解析用户请求和脚本语言等。当用户通过统一资源定位符（URL）访问 Web 时，最终看到的是 Web 容器处理后的内

容，即 HTML 文档。

Web 默认运行在服务器的 80 端口之上，也是服务器所提供的服务器之一，Web 攻击的方式非常多，同时 Web 也是脆弱的，在 2005 年，搜狐的主站就存在 SQL 注入漏洞，由此可以想象当初国内的 Web 安全水平。如今，Web 安全依然是一个热门的话题，并没有随着时间的推移而被冲淡。为什么？原因是多方面的。

首先是程序开发人员，很多开发人员并没有安全意识，总以为黑客的存在很神秘，自己根本接触不到；其次，开发者并不知道哪里的代码存在“Bug”，这时的 Bug 并非是代码的某些功能不完善，而是代码出现的漏洞。

那么有经验的程序员呢？有经验的程序员可能会考虑到安全问题，但毕竟不是专业的安全人员，且一个项目组并非每个人都是“大牛”。另外，当项目上线之后的服务器环境可能会有变化，本来没有问题的代码可能就变得有问题了。再如，管理员密码泄露、一些配置性错误等都会存在安全问题。所以原因是多方面的，不要说自己的网站是安全、没有问题的，可能是你还没有发现它而已。

说了那么多，那么到底攻击者是如何攻陷服务器的呢？到底存在哪些漏洞？图 1-1 即是一张服务器的风险点，攻击者入侵服务器可能就是从这些点下手的，同时也是攻击者所掌握的部分技能图。

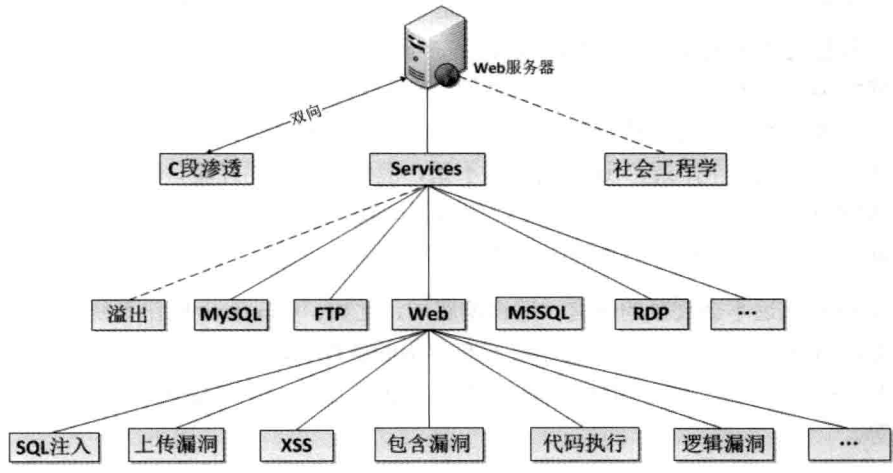


图 1-1 风险点

攻击者在渗透服务器时，直接对目标下手一般有三种手段，当我们了解了攻击者的手段之后，防御也就变得简单了。

- C 段渗透：攻击者通过渗透同一网段内的一台主机对目标主机进行 ARP 等手段的渗透。
- 社会工程学：社会工程学是高端攻击者必须掌握的一个技能，渗透服务器有时不仅仅只靠技术。详细内容请参照第 16 章“社会工程学”。
- Services：很多传统的攻击方式是直接针对服务进行溢出的，至今一些软件仍然存在溢出漏洞。像之前的 MySQL 就出现过缓冲区溢出漏洞。当然，对这类服务还有其他入侵方式，这些方式也经常用于内网的渗透中，在后面的章节中都会一一讲述。

然而 Web 也是服务之一，Web 服务相对于其他服务而言，渗透的方式增加了许多，图 1-1 中也只是列出了部分风险。看似比较简单的一张图，但其中包含了太多的内容，本书都是围绕该图而写。

1.2 如何更好地学习 Web 安全

作为一名 Web 渗透测试人员，我们应该如何更好地学习这些“漏洞”呢？想起了一句老话：黑客都是程序员，但程序员不一定是黑客。从某个角度来说，渗透测试人员其实就是一名黑客，这是不能否认的事实。但黑客也有好坏之分，俗话说：黑客是好的，骇客才是真正的攻击者。但貌似很多人对“骇客”一词并不熟悉，特别是一些安全研究者，反而用白帽子和黑帽子来区分。黑帽子即利用黑客技术实施攻击，进行网络犯罪和牟利的人群，而白帽子则是利用手中的技术进行反黑客的人群。

渗透测试人员与攻击者的性质不一样，攻击者只需要找到程序的一个突破口，拿到权限即可，而渗透测试人员或“白帽子”则不一样，渗透测试人员必须要找到系统所有的漏洞，才能保证系统的安全。然而真正想做到对这些漏洞知其所以然，必须学习编程技术。虽然现在各种利用工具层出不穷，但想要在这条道路上走得更远，必须有编程基础，甚至要比程序员更强的功底。

本节的内容与笔者的职业有关，笔者曾经是一名培训师，所以习惯尽可能地把内容写得更明了。笔者接触过太多刚踏入安全领域或者在安全领域有一段工作经验的人，但事实上并不是每个人都懂得编程技术，很多人都会问：到底学习哪门语言好？这一直是初学者学习编程的第一个问题。很多人都认为先学精一门，然后学其他的，当你学习完一门语言之后，其他语言就变得简单了。“一门通，门门通”这句话固然不错，但根据笔者的经验，笔者认为能用到的语言就是你应该学习的语言，每门语言所擅长的领域不一样，你的领域适合什么语言，就选择哪门语言。下面列举一些笔者认为不错的语言及其应用领域。

① C/C++：永远不会衰败的语言，适合偏底层，比如，Windows 操作系统 80%以上都是由 C/C++完成的，C/C++也经常用于写应用层 C/S 架构的软件。如果想研究缓冲区溢出，或者针对底层协议写一些软件，那么非 C/C++莫属。例如：NC、LCX、DNSSniffer、Hydra、溢出程序、远程控制等。

② Java：真正跨平台的语言，“一次编译，到处运行。”即是 Java 的口号。Java 适合应用层的开发，无论是 C/S 架构还是 B/S 架构，Java 都能做到，但在国内使用 Java(JSP)做 B/S 架构居多，很多大型企业都采用了 Java 作为 Web 开发的首选。例如：Burp Suite、reDuh、Paros proxy、WebScarab、Owasp Zap 等。

③ C#：与 Java 有 70%的雷同，同样适用于开发应用层程序，无论是 C/S 架构还是架构 B/S，C#都可以做到，拥有强大的 .NET Framework 支持，但是不能跨平台。例如：Pangolin、Jsky、微软官网等。

④ PHP：跨平台的语言，脚本语言，无须编译，但 PHP 的能力仅限于 Web，速度较慢，也不支持多线程。作为一名 Web 安全研究者，几乎所有的人都会学习它。

⑤ Python：号称“大蟒蛇”，跨平台，脚本语言，无须编译，适用于一些 Shell 操作，最近 Python 也在 Web 领域取得了一些成就，开发较快，运行速度较慢（相对于 C/C++来说），不过很多安全研究者都比较喜欢 Python。例如：SQLMap、W3af、Python 编写的安全工具太多了，在渗透测试平台“Backtrack”下到处都可以看到 Python 的身影。

⑥ HTML：属于前端语言之一，是渗透测试人员必备的语言。

⑦ JavaScript：属于前端语言之一，掌握 JavaScript 后，可以帮助渗透测试人员更好地理解 XSS 跨站脚本攻击。

⑧ 数据库：数据库分为很多种，有 Oracle、MySQL、SQL Server、DB2 等，操作数据库的语言即 SQL 语句，掌握一门 SQL 语言是必需的，因为几乎没有网站不使用数据库。

读者可根据实际需求选择一门适合自己的语言，这里并不是让安全研究人员去写代码，一般要达到能通读代码的要求即可，当然能写是最好的。如果不去做代码审计工作，一般一门语言就足够了，但本着学习 Web 的目标去学习 HTML、JavaScript、SQL，以及任意一门 Web 语言，这样的搭配是比较合适的。

虽然说渗透测试时，没有代码基础也能出色地完成任务，但相对来说，掌握语言的基础是非常有帮助的，因为在渗透过程中有时无法避免有针对性地编写一些代码，代码功底是“菜鸟”和“大牛”一个明显的分水岭。

下面让我们一起在 Web 安全的海洋里遨游吧。

第 2 章

深入 HTTP 请求流程

随着 Web 2.0 时代的到来，互联网从传统的 C/S 架构转变为更加方便快捷的 B/S 架构。B/S 即浏览器/服务器结构，就像我们访问过的所有网站，客户机上只需要一个浏览器即可上网冲浪。

当客户端与 Web 服务器进行交互时，就存在 Web 请求，这种请求都基于统一的应用层协议（HTTP 协议）交互数据。

2.1 HTTP 协议解析

HTTP（HyperText Transfer Protocol）即超文本传输协议，是一种详细规定了浏览器和万维网服务器之间互相通信的规则，它是万维网交换信息的基础，它允许将 HTML（超文本标记语言）文档从 Web 服务器传送到 Web 浏览器。

2.1.1 发起 HTTP 请求

如何发起一个 HTTP 请求？这个问题似乎很简单，当在浏览器地址栏中输入一个 URL，并按回车键后就发起了这个 HTTP 请求，很快就会看到这个请求的返回结果。

URL（统一资源定位符）也被称为网页地址，是互联网标准的地址。URL 的标准格式如下：

协议://服务器 IP [:端口]/路径/[?查询]

例如，`http://www.xxser.com/post/httpxieyi.html` 就是一个标准的 URL。

借助浏览器可以快速发起一次 HTTP 请求，如果不借助浏览器应该怎样发起 HTTP 请求呢？其实可以借助很多工具来发起 HTTP 请求，例如，在 Linux 系统中的 `curl` 命令。严格地说，浏览器也属于 HTTP 工具的一种。

在 Windows 中，也可以用 `curl.exe` 工具来发起请求，通过 `curl + URL` 命令就可以简单地发起一个 HTTP 请求，非常方便。但 Windows 没有自带 `curl.exe`，用户必须进行下载才可以使用。

例如，`curl http://www.baidu.com` 可以返回这个页面的 HTML 数据，如图 2-1 所示。也可以查看访问 URL 后服务器返回的 HTTP 响应头，加上 `-I` 选项即可，如图 2-2 所示。



图 2-1 HTTP 请求返回的 HTML 数据

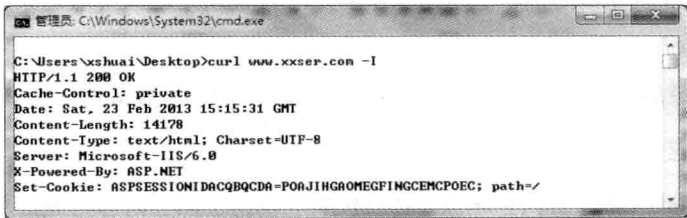


图 2-2 HTTP 响应头

此时脱离了浏览器来获取服务器响应和 HTML 数据，你可以发现，就某些方面而言，浏览器在 HTTP 协议方面只不过多了 HTML 渲染的功能，让用户看到更直观的界面。

2.1.2 HTTP 协议详解

HTTP 协议目前最新版的版本是 1.1，HTTP 是一种无状态的协议。无状态是指 Web 浏览器与 Web 服务器之间不需要建立持久的连接，这意味着当一个客户端向服务器端发出请求，然后 Web 服务器返回响应（Response），连接就被关闭了，在服务器端不保留连接的有关信息。也就是说，HTTP 请求只能由客户端发起，而服务器不能主动向客户端发送数据。

HTTP 遵循请求（Request）/应答（Response）模型，Web 浏览器向 Web 服务器发送请求时，Web 服务器处理请求并返回适当的应答，如图 2-3 所示。

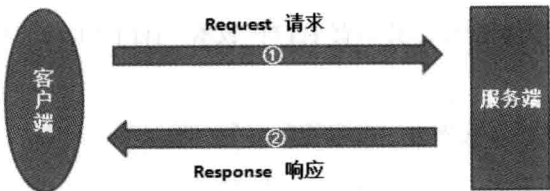


图 2-3 HTTP 请求与响应

下面通过实例来观察 HTTP 的请求与响应。

1. HTTP 请求与响应

(1) HTTP 请求

HTTP 请求包括三部分，分别是请求行（请求方法）、请求头（消息报头）和请求正文。下面是 HTTP 请求的一个例子。

```
POST /login.php HTTP/1.1           //请求行
HOST: www.xxser.com                //请求头
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:15.0) Gecko/20100101 Firefox/15.0
                                     //空白行，代表请求头结束
Username=admin&password=admin      //请求正文
```

HTTP 请求行的第一行即为请求行，请求行由三部分组成，该行的第一部分说明了该请求是 POST 请求；该行的第二部分是一个斜杠 (/login.php)，用来说明请求的是该域名根目录下的 login.php；该行的最后一部分说明使用的是 HTTP 1.1 版本（另一个可选项是 1.0）。

第二行至空白行为 HTTP 中的请求头（也被称为消息头）。其中，HOST 代表请求的主机地址，User-Agent 代表浏览器的标识。请求头由客户端自行设定。关于消息头的内容，在后面章节中将会详细介绍。

HTTP 请求的最后一行为请求正文，请求正文是可选的，它最常出现在 POST 请求方法中。

(2) HTTP 响应

与 HTTP 请求对应的是 HTTP 响应，HTTP 响应也由三部分内容组成，分别是响应行、响应头（消息报头）和响应正文（消息主题）。下面是一个经典的 HTTP 响应。

```
HTTP/1.1 200 OK                                //响应行
Date: Thu, 28 Feb 2013 07:36:47 GMT             //响应头
Server: BWS/1.0
Content-Length: 4199
Content-Type: text/html;charset=utf-8
Cache-Control: private
Expires: Thu, 28 Feb 2013 07:36:47 GMT
Content-Encoding: gzip
Set-Cookie: H_PS_PSSID=2022_1438_1944_1788; path=/; domain=.xxser.com
Connection: Keep-Alive
//空白行，代表响应头结束
<html>                                           //响应正文或者叫消息主题
  <head><title> Index.html </title></head>
  .....
```

HTTP 响应的第一行为响应行，其中有 HTTP 版本（HTTP/1.1）、状态码（200）以及消息“OK”。

第二行至末尾的空白行为响应头，由服务器向客户端发送。

消息报头之后是响应正文，是服务器向客户端发送的 HTML 数据。

2. HTTP 请求方法

HTTP 请求的方法非常多，其中 GET、POST 最常见。下面是 HTTP 请求方法的详细介绍。

(1) GET

GET 方法用于获取请求页面的指定信息（以实体的格式）。如果请求资源为动态脚本（非 HTML），那么返回文本是 Web 容器解析后的 HTML 源代码，而不是源文件。例如请求 index.jsp，返回的不是 index.jsp 的源文件，而是经过解析后的 HTML 代码。

如下 HTTP 请求：

```
GET /index.php?id=1 HTTP/1.1
HOST: www.xxser.com
```

使用 GET 请求 index.php，并且 id 参数为 1，在服务器端脚本语言中可以选择性地接收这些参

数，比如 `id=1&name=admin`，一般都是由开发者内定好的参数项目才会接收，比如开发者只接收 `id` 参数项目，若加了其他参数项，如：

```
Index.php?id=1&username=admin //多个参数项以“&”分隔
```

服务器端脚本不会理会你加入的内容，依然只会接收 `id` 参数，并且去查询数据，最终向服务器端发送解析过的 HTML 数据，不会因为你的干扰而乱套。

(2) HEAD

HEAD 方法除了服务器不能在响应里返回消息主体外，其他都与 GET 方法相同。此方法经常被用来测试超文本链接的有效性、可访问性和最近的改变。攻击者编写扫描工具时，就常用此方法，因为只测试资源是否存在，而不用返回消息主题，所以速度一定是最快的。一个经典的 HTTP HEAD 请求如下：

```
HEAD /index.php HTTP/1.1
HOST: www.xxser.com
```

(3) POST

POST 方法也与 GET 方法相似，但最大的区别在于，GET 方法没有请求内容，而 POST 是有请求内容的。POST 请求最多用于向服务器发送大量的数据。GET 虽然也能发送数据，但是有大小（长度）的限制，并且 GET 请求会将发送的数据显示在浏览器端，而 POST 则不会，所以安全性相对来说高一点。

例如，上传文件、提交留言等，只要是向服务器传输大量的数据，通常都会使用 POST 请求。一个经典的 HTTP POST 请求如下：

```
POST /login.php HTTP/1.1
Host: www.xxser.com
Content-Length: 26
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Origin: http://home.2cto.com
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.17 (KHTML, like Gecko)
Chrome/24.0.1312.57 Safari/537.17 SE 2.X MetaSr 1.0
Content-Type: application/x-www-form-urlencoded
Accept-Language: zh-CN,zh;q=0.8
Accept-Charset: GBK,utf-8;q=0.7,*;q=0.3
```

```
user=admins&pw=123456789
```

用 POST 方法向服务器请求 `login.php`，并且传递参数 `user=admins&pw=123456789`。

(4) PUT

PUT 方法用于请求服务器把请求中的实体存储在请求资源下，如果请求资源已经在服务器中存在，那么将会用此请求中的数据替换原先的数据，作为指定资源的最新修改版。如果请求指定的资源不存在，将会创建这个资源，且数据位请求正文，请求如下：

```
PUT /input.txt
HOST: www.xxser.com
Content-Length: 6
```

123456

这段 HTTP PUT 请求将会在主机根目录下创建 input.txt，内容为 123456。通常情况下，服务器都会关闭 PUT 方法，因为它会为服务器建立文件，属于危险的方法之一。

(5) DELETE

DELETE 方法用于请求源服务器删除请求的指定资源。服务器一般都会关闭此方法，因为客户端可以进行删除文件操作，属于危险方法之一。

(6) TRACE

TRACE 方法被用于激发一个远程的应用层的请求消息回路，也就是说，回显服务器收到的请求。TRACE 方法允许客户端去了解数据被请求链的另一端接收的情况，并且利用那些数据信息去测试或诊断。但此方法非常少见。

(7) CONNECT

HTTP 1.1 协议规范保留了 CONNECT 方法，此方法是为了用于能动态切换到隧道的代理。

(8) OPTIONS

OPTIONS 方法是用于请求获得由 URI 标识的资源在请求/响应的通信过程中可以使用的功能选项。通过这个方法，客户端可以在采取具体资源请求之前，决定对该资源采取何种必要措施，或者了解服务器的性能。HTTP OPTIONS 请求如下：

```
OPTIONS / HTTP/1.1
HOST: www.xxser.com

HTTP/1.1 200 OK
Allow: OPTIONS, TRACE, GET, HEAD, POST
Server: Microsoft-IIS/7.5
Public: OPTIONS, TRACE, GET, HEAD, POST
X-Powered-By: ASP.NET
Date: Sun, 14 Jul 2013 15:50:58 GMT
Content-Length: 0
```

以上为 HTTP/1.1 标准方法，详情请参照“<http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>”，但 HTTP 中的请求方法还不止这些，例如 WebDAV。WebDAV（Web-based Distributed Authoring and Versioning）是一种基于 HTTP/1.1 协议的通信协议，它扩展了 HTTP 1.1，在 GET、POST、HEAD 等几个 HTTP 标准方法以外添加了一些新的方法，使应用程序可直接对 Web Server 进行读写，并支持写文件锁定（Locking）和解锁（Unlock）、文件复制（Copy）、文件移动（Move）。另外，还可以支持文件的版本控制。

3. HTTP 状态码

当客户端发出 HTTP 请求，服务器端接收后，会向客户端发送响应信息，其中，HTTP 响应中的第一行中，最重要的一点就是 HTTP 的状态码，内容如下：

```
HTTP/1.1 200 OK
```

此的状态码为 200，在 HTTP 协议中表示请求成功。HTTP 协议中的状态码由三位数字组成，

第一位数字定义了响应的类别，且只有以下 5 种。

- 1xx：信息提示，表示请求已被成功接收，继续处理。其范围为 100~101。
- 2xx：成功，服务器成功地处理了请求。其范围为 200~206。
- 3xx：重定向，重定向状态码用于告诉浏览器客户端，它们访问的资源已被移动，并告诉客户端新的资源地址位置。这时，浏览器将重新对新资源发起请求。其范围为 300~305。
- 4xx：客户端错误状态码，有时客户端会发送一些服务器无法处理的东西，比如格式错误的请求，或者最常见的是，请求一个不存在的 URL。其范围为 400~415。
- 5xx：有时候客户端发送了一条有效请求，但 Web 服务器自身却出错了，可能是 Web 服务器运行出错了，或者网站都挂了。5XX 就是用来描述服务器内部错误的，其范围为 500~505。

常见的状态码描述如下。

200：客户端请求成功，是最常见的状态。

302：重定向。

404：请求资源不存在，是最常见的状态。

400：客户端请求有语法错误，不能被服务器所理解。

401：请求未经授权。

403：服务器收到请求，但是拒绝提供服务。

500：服务器内部错误，是最常见的状态。

503：服务器当前不能处理客户端的请求，一段时间后可能恢复正常。

4. HTTP 消息

HTTP 消息又称为 HTTP 头（HTTP header），由四部分组成，分别是请求头、响应头、普通头和实体头。从名称上看，我们就可以知道它们所处的位置。

（1）请求头

请求头只出现在 HTTP 请求中，请求报头允许客户端向服务器端传递请求的附加信息和客户端自身的信息。常用的 HTTP 请求头如下。

① Host

Host 请求报头域主要用于指定被请求资源的 Internet 主机和端口号，例如：HOST: www.xxser.com:801。

② User-Agent

User-Agent 请求报头域允许客户端将它的操作系统、浏览器和其他属性告诉服务器。登录一些网站时，很多时候都可以见到显示我们的浏览器、系统信息，这些都是此头的作用，如：

User-Agent: My privacy

③ Referer

Referer 包含一个 URL，代表当前访问 URL 的上一个 URL，也就是说，用户是从什么地方来到本页面。如：Referer: www.xxser.com/login.php，代表用户从 login.php 来到当前页面。

④ Cookie

Cookie 是非常重要的请求头，它是一段文本，常用来表示请求者身份等。在后面将会详细讲述 Cookie。

⑤ Range

Range 可以请求实体的部分内容，多线程下载一定会用到此请求头。例如：

表示头 500 字节：bytes=0~499

表示第二个 500 字节：bytes=500~999

表示最后 500 字节：bytes=-500

表示 500 字节以后的范围：bytes=500-

⑥ x-forward-for

x-forward-for 即 XXF 头，它代表请求端的 IP，可以有多个，中间以逗号隔开。

⑦ Accept

Accept 请求报头域用于指定客户端接收哪些 MIME 类型的信息，如 Accept: text/html，表明客户端希望接收 HTML 文本。

⑧ Accept-Charset

Accept-Charset 请求报头域用于指定客户端接收的字符集。例如：Accept-Charset:iso-8859-1,gb2312。如果在请求消息中没有设置这个域，默认是任何字符集都可以接收。

(2) 响应头

响应头是服务器根据请求向客户端发送的 HTTP 头。常见的 HTTP 响应头如下。

① Server

服务器所使用的 Web 服务器名称，如 Server:Apache/1.3.6(Unix)，攻击者通过查看此头，可以探测 Web 服务器名称。所以，建议在服务器端进行修改此头的信息。

② Set-Cookie

向客户端设置 Cookie，通过查看此头，可以清楚地看到服务器向客户端发送的 Cookie 信息。

③ Last-Modified

服务器通过这个头告诉浏览器，资源的最后修改时间。

④ Location

服务器通过这个头告诉浏览器去访问哪个页面，浏览器接收到这个请求之后，通常会立刻访问 Location 头所指向的页面。这个头通常配合 302 状态码使用。

⑤ Refresh

服务器通过 Refresh 头告诉浏览器定时刷新浏览器。

(3) 普通头

在普通报头中，有少数报头域用于所有的请求和响应消息，但并不用于被传输的实体，只用于传输的消息。例如：Date，表示消息产生的日期和时间。Connection，允许发送指定连接的选项。例如，指定连接是连续的，或者指定“close”选项，通知服务器，在响应完成后，关闭连接。Cache-Control，用于指定缓存指令，缓存指令是单向的，且是独立的。

注意：普通报头作为了解即可。

(4) 实体头

请求和响应消息都可以传送一个实体头。实体头定义了关于实体正文和请求所标识的资源的元信息。元信息也就是实体内容的属性，包括实体信息类型、长度、压缩方法、最后一次修改时间等。常见的实体头如下。

① Content-Type

Content-Type 实体头用于向接收方指示实体的介质类型。

② Content-Encoding

Content-Encoding 头被用作媒体类型的修饰符，它的值指示了已经被应用到实体正文的附加内容的编码，因而要获得 Content-Type 报头域中所引用的媒体类型，必须采用相应的解码机制。

③ Content-Length

Content-Length 实体报头用于指明实体正文的长度，以字节方式存储的十进制数字来表示。

④ Last-Modified

Last-Modified 实体报头用于指示资源的最后修改日期和时间。

在本节介绍了 HTTP 请求和响应，读者对这些常见的请求一定要熟练掌握。特别是 HTTP 的状态码，更需要牢记。

2.1.3 模拟 HTTP 请求

在了解了 HTTP 协议之后，本节通过实际操作来学习 HTTP 协议，下面使用 Telnet 模拟 HTTP 请求来访问 www.baidu.com。

第一步：打开 CMD 运行框，输入 Telnet www.baidu.com 80 后按回车键（此时是黑屏状态），然后利用快捷键“Ctrl+]”来打开 Telnet 回显（Telnet 默认不回显），如图 2-4 所示。

第二步：按回车键后，进入编辑状态，如图 2-5 所示。



图 2-4 Telnet 界面

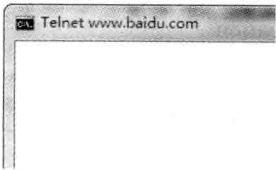


图 2-5 可编辑的 Telnet

第三步：输入 GET /index.html HTTP/1.1，按回车键，接着输入 Host:www.baidu.com，再连续两次按回车键（两次回车代表提交请求）。输入速度一定要快，否则将会连接失败，或者将代码写入记事本，使用时可以直接复制，如图 2-6 所示。



图 2-6 HTTP 请求

第四步：接收服务器返回数据，这一步不需要任何操作，只需等待几秒，就可以接收到服务器返回的数据，如图 2-7 所示。

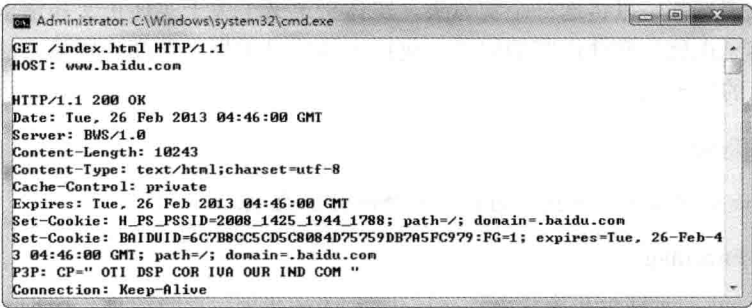


图 2-7 接收到的响应和数据

本节用手工方式进行了一次百度的访问，在后面的章节中，我们将会通过抓包的方式再深入了解 HTTP 协议。

2.1.4 HTTP 协议与 HTTPS 协议的区别

HTTPS 协议的全称为 Hypertext Transfer Protocol over Secure Socket Layer，它是以安全为目的的 HTTP 通道，其实就是 HTTP 的“升级”版本，只是它比单纯的 HTTP 协议更加安全。

HTTPS 的安全基础是 SSL，即在 HTTP 下加入 SSL 层。也就是 HTTPS 通过安全传输机制进行传送数据，这种机制可保护网络传送的所有数据的隐秘性与完整性，可以降低非侵入性拦截攻击的可能性。

既然是在 HTTP 的基础上进行构建的 HTTPS 协议，所以，无论怎么样，HTTP 请求与响应都是以相同的方式进行工作的。

HTTP 协议与 HTTPS 协议的主要区别如下。

- HTTP 是超文本传输协议，信息是明文传输，HTTPS 则是具有安全性的 SSL 加密传输协议。

- HTTP 与 HTTPS 协议使用的是完全不同的连接方式，HTTP 采用 80 端口连接，而 HTTPS 则是 443 端口。
- HTTPS 协议需要到 CA 申请证书，一般免费证书很少，需要交费，也有些 Web 容器提供，如 Tomcat。而 HTTP 协议却不需要。
- HTTP 连接相对简单，是无状态的，而 HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，相对来说，它要比 HTTP 协议更安全。

2.2 截取 HTTP 请求

很多网站为了减少服务器端的压力，在后台方面减少验证，而只在 Web 前端使用 JavaScript 进行验证，殊不知这样大大增加了安全隐患。在渗透测试中，经常会进行 HTTP 请求的截取来发现一些隐秘的漏洞。例如：绕过 JavaScript 验证、发现隐藏标签内容等。

2.2.1 Burp Suite Proxy 初体验

Burp Suite 是用于 Web 应用安全测试工具的集成平台，它包含许多工具，并为这些工具设计了许多接口，以促进加快测试应用程序的过程。所有的工具都共享一个能处理并显示 HTTP 消息、持久性、认证、代理、日志、警报的一个强大的可扩展的框架。

Burp Suite 工具箱及说明如表 2-1 所示。

表 2-1 Burp suite 功能预览

工 具	说 明
Proxy	一个拦截 HTTP/S 的代理服务器，作为一个在浏览器和目标应用程序之间的中间人，允许拦截、查看、修改在两个方向上的原始数据包
Spider	一个应用智能感应的网络爬虫，它能完整地枚举应用程序的内容和功能
Scanner	是一个高级工具，执行后，它能自动发现 Web 应用程序的安全漏洞
Intruder	是一个定制的高度可配置的工具，对 Web 应用程序进行自动化攻击，如：枚举标识符、表单破解和信息搜集
Repeater	是一个靠手动操作来补发单独的 HTTP 请求，并分析应用程序响应的工具
Sequencer	是一个用来分析那些不可预知的应用程序会话令牌和重要数据项的随机性的工具
Decoder	是一个极为方便的解码/编码工具
Comparer	是一个实用的工具，通常是通过一些相关的请求和响应得到两项数据的一个可视化的“差异”

本节主要使用 Proxy 模块来进行一次实际的绕过 JavaScript 验证。为了更好地说明问题，本书用 Java 编写了一个名为 ProxyTest 的 Web 应用测试程序进行测试。

这个 Web 应用程序的功能很简单，如果用户在输入框中输入“<”、“>”、“script”等敏感字符，将会弹窗提示“存在敏感字符”，如果不存在敏感字符，将会提交到 PrintStr 页面，并且显示提交后的字符串，关键代码如下。

```
<script type="text/javascript">
function check(f){
    var str = f.username.value;
    var c = new Array('script','<','>','input','img');
```

```
for(var i=0; i<c.length; i++){
    if(str.indexOf(c[i])!=-1){    //循环判断是否存在敏感字
        alert('你输入的数据存在敏感字符 :' + c[i] );
        return false ;
    }
}
return true ;
}
</script>
```

首先提交敏感字符“script”进行测试，可以发现已经被拦截，如图 2-8 所示。



图 2-8 拦截到的敏感字符

接下来使用 Burp Suite 绕过这段 JavaScript 验证。

第一步：配置网络代理。

打开 Burp Suite，选择“Proxy”选项卡，然后选择“Options”选项卡，在“Proxy Listeners”（代理监听）模块中可以发现有三个按钮，分别是 Add、Edit 和 Remove。单击“Add”按钮，在“Bind to port”（绑定端口）框中输入端口号 6666，此时注意输入的端口必须是未开启的状态。

在“Bind to address”单选框中选择“Loopback only”，然后单击“OK”按钮，即可完成 Burp Suite 端口监听的配置（Burp Suite 会默认启用新添加的端口），如图 2-9 所示。

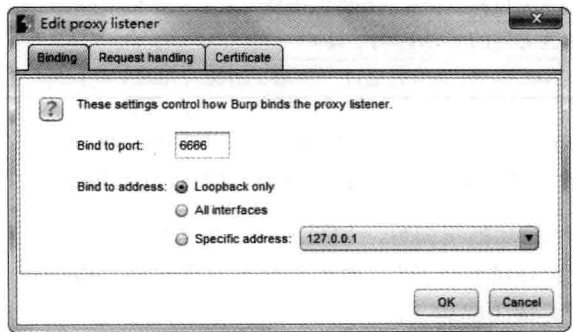


图 2-9 Burp Suite 端口监听配置

接下来继续配置浏览器的代理设置，在此采用火狐（Firefox）浏览器。打开 Firefox，在工具栏中选择：“选项”→“高级”→“网络”→“设置”→“手动配置代理”。在 HTTP 代理框

中输入 127.0.0.1，此时的端口号为在“Burp Bind to port”输入框中所输入的端口号。此次输入端口为 6666，其他则不需要输入，单击“确定”按钮，就完成了 Burp Suite 和 Firefox 的配置，如图 2-10 所示。

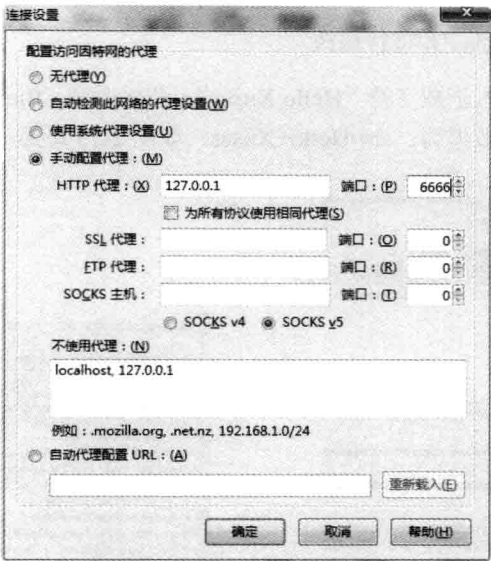


图 2-10 火狐代理配置

第二步：查看拦截信息。

在火狐浏览器地址栏中输入网址：<http://192.168.1.101:8080/ProxyTest/index.html>，按回车键，会发现服务器久久不能回应信息，原因是 Burp 已经把 HTTP 请求拦截了，此时的浏览器会一直处于阻塞状态，如图 2-11 所示。

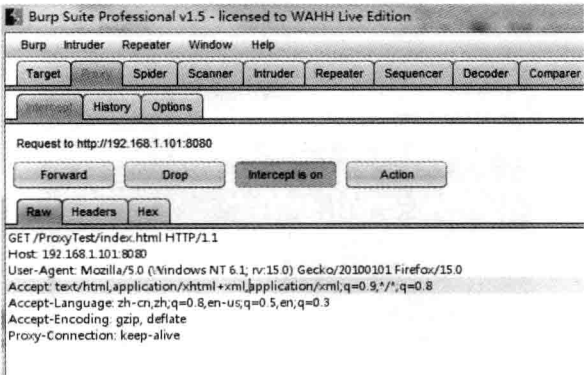


图 2-11 拦截后的请求

在 Intercept 模块中有 4 个按钮，分别是 Forward（跳转到下一步）、Drop（放弃本次请求）、Intercept is on（拦截开关）和 Action（动作选项）。

在 Raw 信息框中，可以清楚地看到拦截后的 HTTP 请求，Headers 和 Hex 信息框是以不同的方式来显示 HTTP 请求的。

单击“Forward”按钮进行跳转，服务器才能接收到浏览器发送的请求。在不用拦截的时候，

单击“Intercept is on”按钮即可，Burp 会关掉拦截器。

在 History 模块中，可以显示拦截的历史记录，包括 Request 和 Response 信息，如图 2-12 所示。

第三步：拦截输入信息，并进行修改。

在网页地址栏中，输入正规字符“Hello Xxser”，进行提交。Burp 已经拦截到请求，可以发现，发送为 POST 请求，数据为：str=Hello+Xxser，如图 2-13 所示。



图 2-12 Burp 拦截后的历史记录

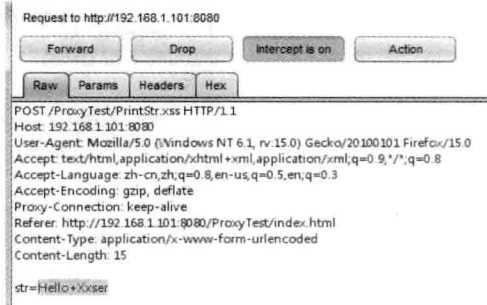


图 2-13 拦截后的 POST 请求

接下来，将 str=Hello+Xxser 修改为 str=<script>alert(/xss/)</script>，然后单击“Forward”按钮，向服务器发送请求。

通过修改 HTTP 请求，绕过了前端 JavaScript 验证，并成功地向服务器提交了敏感数据，造成了 XSS 跨站漏洞，如图 2-14 所示。

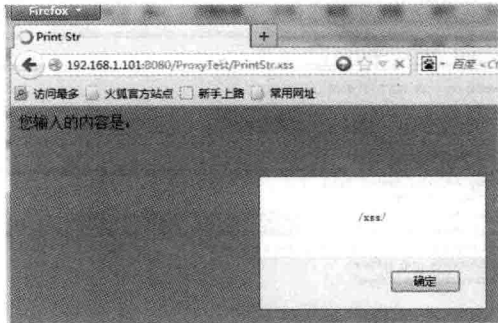


图 2-14 绕过 JavaScript 验证后触发跨站脚本漏洞

JavaScript 属于前端验证，在浏览器未提交数据时进行验证，而我們是在通过验证，并拦截 HTTP 请求后修改数据，JavaScript 的验证根本起不了任何作用。由此可见，前端验证是不可靠的。

作为一名 Web 开发人员，一定要牢记，前端 JavaScript 验证是为了防止用户输入错误，服务器端验证是为了防止恶意攻击。

在渗透测试中，往往会拦截 HTTP 请求并加以分析，因为在进行漏洞扫描时，一些隐藏较

深的安全问题是扫描不出来的。所以，在必要的情况下，需要拦截 HTTP 请求和响应，并进行分析。

2.2.2 Fiddler

Fiddler 是一款优秀的 Web 调试工具，它可以记录所有的浏览器与服务器之间的通信信息（HTTP 和 HTTPS），并且允许你设置断点，修改输入/输出数据。无论是在 Web 开发中还是在渗透测试中，Fiddler 对我们来说都有很大的作用。

Fiddler 可以在其官方网址（<http://fiddler2.com/>）进行下载。

1. 拦截 HTTP(S)请求

在安装 Fiddler 后，Fiddler 会自动为 IE 浏览器、火狐浏览器以及 Chrome 浏览器安装启动插件，并且默认监听 IE 浏览器的数据。Fiddler 会自动为 IE 浏览器配置代理信息，用户无须进行其他配置操作。如果是其他浏览器，想要进行数据拦截，就必须配置代理服务器，其配置过程与 Burp Suite 相似。

在此以 IE 8 浏览器为例进行说明，可以在工具栏中选择“工具”→“Fiddler”启动。Fiddler 默认只记录 HTTP 请求，但不会记录 HTTPS，需要进行配置才可以记录。选择“Tools”→“Fiddler Options”→“HTTPS”，勾选“Decrypt HTTPS traffic”复选框，如图 2-15 所示。

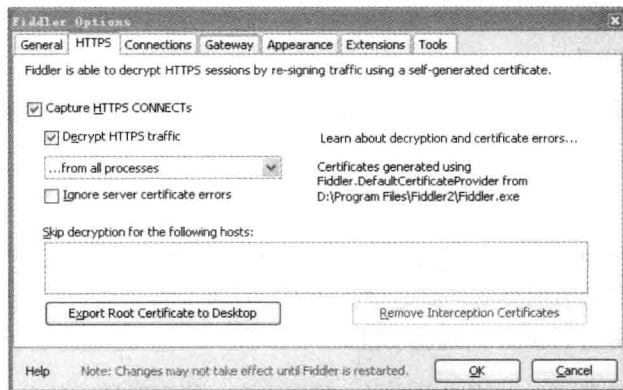


图 2-15 开启 HTTPS 协议截取功能

在勾选“Decrypt HTTPS traffic”后弹出的对话框中，单击“OK”按钮即可。现在访问 <http://www.xxser.com>，即可拦截所有的 Request 和 Response 信息。

2. Fiddler 功能简介

Fiddler 工具的主界面如图 2-16 所示。

监控开关是用于控制是否监听数据包的快捷按钮，如果监控关闭，可以发现 IE 的代理服务已经关闭，因此无法继续监听 IE 浏览器的访问记录。

Fiddler 监听进程的类型主要可以分为：所有类型、Web 浏览器和非浏览器。用户可根据选择类型对该类型进行监控，也可以选择“Hide All”隐藏所有。如果想要对指定的进程进行监控，可以通过任务栏“Any Process”选择指定的进程，其操作非常方便。

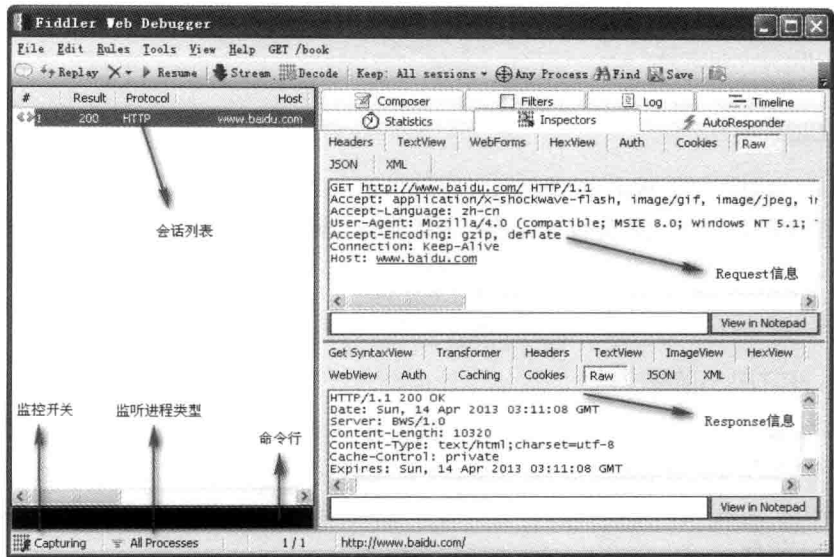


图 2-16 Fiddler 主界面

Fiddler 的命令行工具叫作 QuickExec，它允许用户直接输入命令进行操作。例如：

- cls，清除会话列表。
- select，选择会话。
- bup，截取 Request。

更多的命令可以通过输入 help 或访问 Fiddler 官方网站的命令帮助页面获取，此页面包含了所有的命令。

3. 过滤器

使用 Fiddler 拦截请求某个网站请求时，会拦截图片、CSS、JS 等文件信息，导致我们浏览一个页面时，会产生非常多的会话。而这些会话中只有极个别的会话是我们需要的，寻找起来非常费劲，这时可以用到 Fiddler 的过滤功能。

Fiddler 提供的过滤器可以过滤请求消息、响应消息、状态码等。对于一些不需要关注的 JS 文件、CSS 文件、Flash 文件，以及一些图片文件，我们只需要选择相关的复选框，即可进行过滤。

在 Fiddler 左侧区域“Filters”→“Response Type and Size”模块中，可选择“Show only HTML”去除 JS、CSS、IMAGE 等无用的会话，如图 2-17 所示。

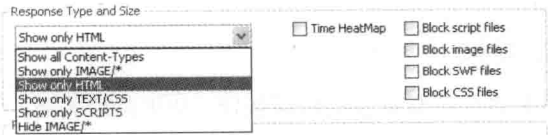


图 2-17 Response 显示设置

4. 设置断点

设置断点是 Fiddler 最强大的功能之一，在设置好断点后，Fiddler 会捕捉所有经过的消息，我们可以任意修改 HTTP 请求信息，包括 Host、Cookie 或表单的数据。设置断点通常用两种方法。

方法 1：通过选择“Rules”→“Automatic Breakpoints”菜单，选择断点的插入点，有三个选项，分别是：Before Request（请求之前）、After Response（响应之后）和 Disabled（不拦截）。

方法 2：通过命令进行断点设置。例如，需要对 www.xxser.com 请求进行拦截，可以执行命令“bpu www.xxser.com”。所有发往 www.xxser.com 的请求都将会被拦截，而访问其他网站则不会被拦截。设置拦截响应信息则可以使用“bpafter”命令。

请求一旦被拦截，此时网站就处于阻塞状态。在会话列表中选择被拦截的网站，在左侧会自动跳转到“Inspectors”模块中，如图 2-18 所示。

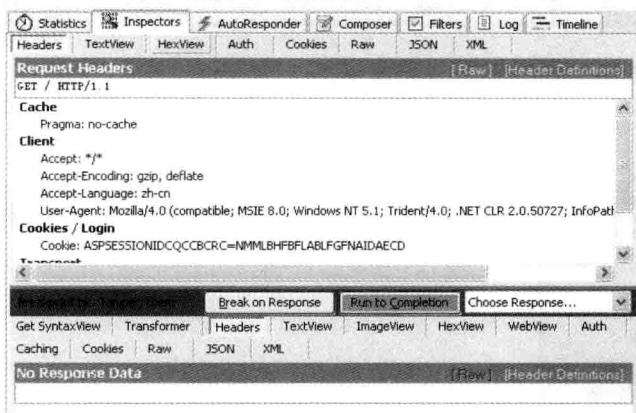


图 2-18 拦截后的 Request 信息

在看到拦截后的 Request 信息后，Fiddler 提供了方便的查看方式，其中包括 Hex View、Cookies、Raw、WebForms 等。如果想要修改 Header 信息，可以在“Headers”模块中用鼠标右键请求行单击，在弹出的快捷菜单中选择“Edit Header”修改头信息。如果是表单信息，则可以直接选择“WebForms”模块，对其 Name 或者 Value 进行修改，非常方便，WebForms 界面如图 2-19 所示。

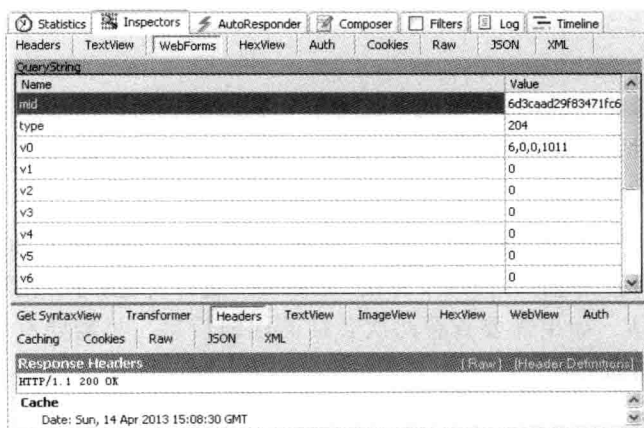


图 2-19 表单信息

对请求头查看或修改完成后，接下来可以通过选择“Break on Response”来中断消息，也可以理解为发送请求。但是这种模式会在返回消息（Response）时继续进行拦截，依然可以修改 Response 信息。例如，把对 www.xxser.com 的响应修改为“Fiddler Test”，那么浏览器的页面

将会变为修改后的信息。需要注意的是，在修改 Response 时一定要设置好过滤信息，否则，寻找响应信息是非常麻烦的一件事情。

如果不希望再次进行拦截，可以选择“Run to Completion”来结束此会话。

5. 编码器

Fiddler 对编码和解码提供了良好的支持，这对于渗透测试人员来说可谓是方便至极。单击菜单栏中的“TextWizard”，可启动编码器，如图 2-20 所示。

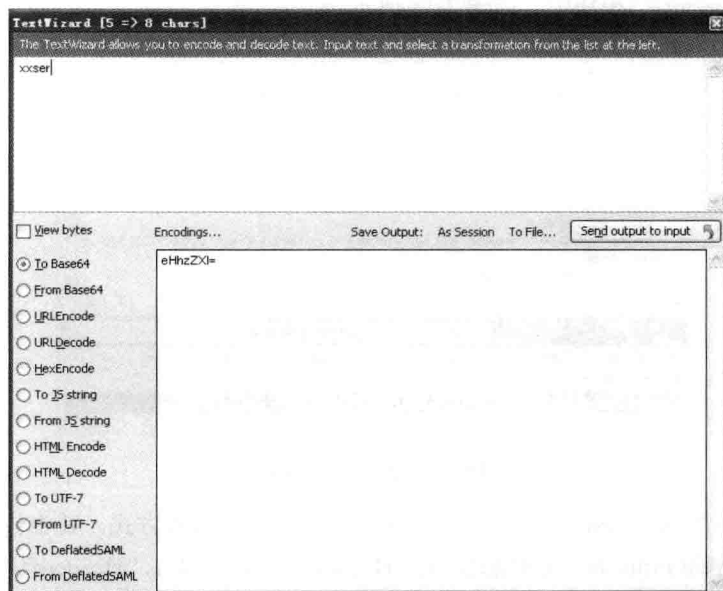


图 2-20 Fiddler 编码器

目前 Fiddler 支持的编码和解码种类有：Base64、URL、JS、HTML、UTF-7，以及默认的 SAML 编码。

6. 会话编辑器

“Composer”是编辑器的意思，它可以针对单个 URL 的会话进行分析，在左侧会话栏目中选中指定的 URL 会话，并将其拖至“Composer”模块内，编辑器会自动分析请求，并且填写到输入框中，如图 2-21 所示。

在请求编辑器中，可以方便地进行调试，例如，想要进行 XSS 或者 SQL 注入测试，可以在某个字段中插入语句。在输入内容后，单击“Execute”进行发送请求，在发送请求后，Fiddler 会继续记录本次会话。如果需要查看此次会话的详细信息，只需要双击会话，即可进入“Inspectors”模块进行查看。

7. 插件支持

Fiddler 不仅是独立的程序，还可以安装相应的插件，从而使 Fiddler 变得更强大。访问 <http://fiddler2.com/fiddler2/extensions.asp>，在此页面可以获得所有关于插件的介绍，以及官方提供的插件下载。

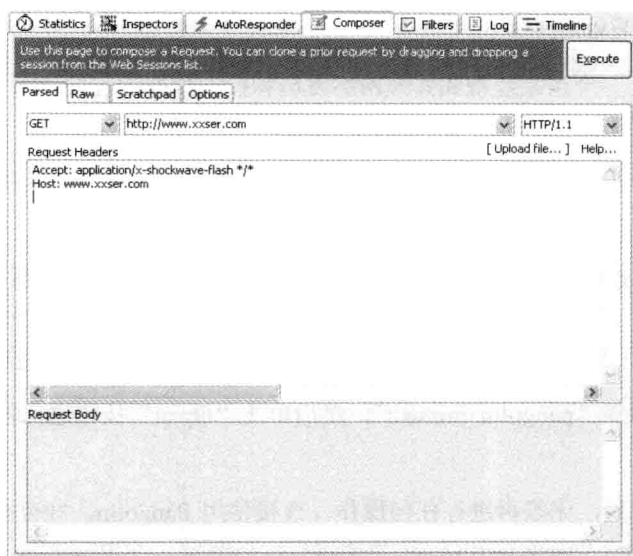


图 2-21 请求编辑器

Fiddler 也提供了一些对外接口，这些接口允许任何人开发自己喜欢的插件。下面介绍测试人员常用的一些第三方插件。

(1) Intruder21

Intruder21 是一款针对 Web 应用程序的 Fuzzing 工具。它与 Burp 的 Intruder 模块非常类似。目前版本为 0.01，暂时还做不到像 Burp Suite 的功能那样强大。Intruder21 的详细说明请参见官方网站：

<http://yamagata.int21h.jp/tool/intruder21/>

(2) x5s

x5s 能够帮助渗透测试人员快速发现跨站脚本漏洞，它的主要目标是帮助渗透测试人员找出最可能出现跨站脚本的地点。

- 针对用户输入安全编码不适用的情况。
- Unicode 字符转换可能绕过安全过滤系统的情况。
- 非最短的 UTF-8 编码可能绕过安全过滤系统的情况。

更多详细的描述信息请参看官方网站：

<http://xss.codeplex.com/>

(3) Ammonite

Ammonite 是一款 Web 应用程序的安全扫描插件，它可以有效地检测出 SQL 注入、OS 命令行注入、本地文件包含、缓冲区溢出和 XSS 漏洞。这些功能给渗透测试人员带来了极大的便利。


更多详细的介绍请参看官方网站：

<http://ammonite.ryscc.com/features.html>

2.2.3 WinSock Expert

WinSock Expert 一个用来监视和修改网络发送和接收数据的程序，可以用来帮助渗透测试人员调试网络应用程序。

WinSock Expert 非常小，仅有 200KB，但是它的功能却十分出色，可以随意抓取指定进程的数据包。

打开 WinSock Expert，单击  按钮，WinSock Expert 将会列出系统所有的进程，可以选择指定的进程进行监听。但有些进程是无法进行监听的，例如，QQ、HTTPS 协议。下面以监听 Pangolin（穿山甲，一款 SQL 注入工具）为例，来观察 Pangolin 是如何进行 SQL 注入判断的。

在进程列表中选择“pangolin_pro.exe”，然后单击“Open”按钮进入监听状态，如图 2-22 所示。

在选择好进程之后，无须再进行任何操作，直接使用 Pangolin，即可抓取 Pangolin 的网络通信信息。这里以 <http://demo.testfire.net> 为例进行注入测试。在 Pangolin 进行注入测试时，可以发现 Winsock 已经截取了数据包，如图 2-23 所示。



图 2-22 选择监听进程

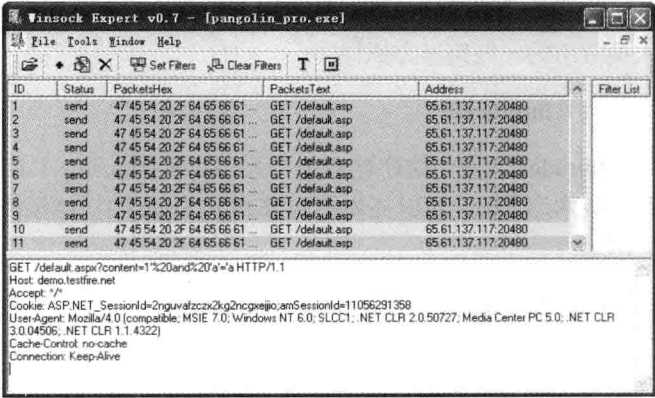


图 2-23 截取 Pangolin 注入测试语句

通过此次抓包，可以清楚地看到 Pangolin 的每一次请求，每一次发送的 SQL 语句，这对我们的学习是非常有帮助的。在使用一些软件的时候，想要知道这些软件在网络通信方面到底做了什么，可以通过此方法进行截取数据包的操作。

现在的抓包工具非常多，常见的有 WireShark、MiniSniffer、Iptool 和 Sniffer。WinSock 相对于这类的大型网络数据监听工具则显得比较袖珍。但是对于一些工作量比较小的软件而言，使用 WinSock 完全可以对付。

2.3 HTTP 应用：黑帽 SEO 之搜索引擎劫持

SEO（Search Engine Optimization）即为搜索引擎优化。简单地说，就是让网站的排名更高，比如，搜索“网络安全”这个关键词，那么排名第一的网站就可能做了 SEO 优化，排名越高，网站的流量就越多，利益也就越大。

那么黑帽 SEO 就是指通过作弊的手段欺骗搜索引擎，获取非正常的排名，让网站更靠前，

流量更大。

大家可能会问：在介绍 HTTP 协议，怎么又提到黑帽 SEO 呢？在 Web 安全领域，可以说与很多行业息息相关，有利益，就有攻击的可能性。而 SEO 行业就是一个典型的例子。

黑帽 SEO 中一个提升排名的手段就是友情链接，与较大的网站（以百度权重高、PR 高为基准）做友情链接，那么对自身的网站排名就越有优势，但是比较大的网站站长怎么会无缘无故与你做友情链接呢？于是黑帽 SEO 一般都会对网站进行入侵，然后偷偷地挂友情链接（黑链），从而获得更好的排名，这样的例子并不少。

搜索引擎劫持也是黑帽 SEO 手段的一种。笔者记得有一次，客户来找我们的时候说自己的网站被入侵，需要修复，症状就是直接输入域名可以进入自己的网站，而使用百度、谷歌等搜索引擎搜索关键字看到自己的网站后，再打开却跳转到其他的网站，客户说自己已经用 Web 杀毒扫描过，并没有发现木马等病毒。

其实这是黑帽 SEO 利用 HTTP 协议搞的鬼。

在 HTTP 中有个请求头叫作 **Referer**，还有一个头叫作 **User-agent**，黑帽 SEO 就是利用这两个头来欺骗搜索引擎的。**Referer** 头用于告诉 Web 服务器，用户是从哪个页面找过来的，而 **User-agent** 头则用于告诉 Web 服务器用户使用的浏览器和操作系统信息。当用户通过搜索引擎打开此网站时，一般会引出源页面（**Referer** 头），如：

Referer: http://www.baidu.com/s?tn=baiduhome_pg&ie=utf-8

Referer: http://www.goole.com.hk/ search?newwindow=1&safe=strict

利用这点，黑帽 SEO 就可以用任何 Web 语言进行针对搜索引擎的流量劫持，一般步骤如下。

① 建立劫持搜索引擎库，如：以 Baidu、Google 等域名为关键字。

② 获取 HTTP **Referer** 头。

③ 遍历搜索引擎库，并与 **Referer** 的内容相比较，如果两者相同或者存在搜索引擎关键字，那么页面将会发生跳转，也就是域名劫持。

这就是针对搜索引擎的劫持，那么 **User-agent** 又能做什么呢？**User-agent** 主要用来劫持搜索引擎的蜘蛛，与劫持流量类似，但是具体的作用却不一样。

2.4 小结

掌握 HTTP 协议是一个合格的渗透测试人员的基本功，在后续的内容中基本上都是针对 HTTP 协议的分析，所以本章是重中之重，读者一定要熟练掌握。

第 3 章

信息探测

在进行安全测试之前，最重要的一步就是信息探测，也就是我们常说的“踩点”。那么信息探测到底有什么用呢？这就好比两个人同时在竞争，如果事先了解竞争对手的优势、核心竞争力、缺点等信息，那么竞争的成功率可能就大一些。

在渗透测试中，搜集目标资料是渗透测试人员的必备技能。这可谓知己知彼，百战不殆。

在搜集目标资料时应该搜集哪些资料呢？其实最主要的就是服务器的配置信息和网站的信息，其中包括网站注册人、目标网站系统、目标服务器系统、目标网站相关子域名、目标服务器所开放的端口和服务器存放网站等。可以说，只要是与目标网站相关联的信息，我们都应该尽量去搜集。

3.1 Google Hack

3.1.1 搜集子域名

毫无疑问，Google 是当今世界上最强大的搜索引擎。然而，在黑客手中，它也是一个秘密武器，它能搜索到一些你意想不到的信息。

利用 Google 搜集网站子域名是一件非常简单也非常复杂的事情。简单是指只要用 Google 搜索一下即可；复杂是指要从海量的信息中寻找子域名。

下面以 baidu.com 为例，进行百度的子域名查询。

打开 Google，在搜索设置中设置每页搜索结果数为 100 条，这样方便查看（注：Google 默认每页结果为 10 条）。

如图 3-1 所示，在搜索栏中输入：site:baidu.com。搜索后找到 12.9 亿条结果，这是一个非常庞大的数字，想要在这海量的信息中（包括重复数据）搜集到百度的所有子域名是不现实的。所以，一般会选取前 10~20 页作为信息搜集，包括现在流行的子域名查询工具，对于大型网站进行子域名搜集，基本是不可靠（不全面）的。一方面是因为任务量太大，另一方面则是收录问题，有些新建立的网站可能搜索引擎还没有收录。所以目前没有一种比较好的办法完全搜集

子域名。



图 3-1 查询指定域名

Google Hack 并没有太多的技术含量，其实只是根据 Google 提供的语法来进行信息查询。下面将会详细讲解如何更好地利用 Google 提供的语法。

3.1.2 搜集 Web 信息

在 3.1.1 节中，使用了 site 关键字进行指定网站的查询，本节将详细介绍 Google Hack 常用的语法，以进行敏感信息探测。Google 常用语法请参照见表 3-1。

表 3-1 Google 常用语法

关 键 字	说 明
site	指定域名
intext	正文中存在关键字的网页
intitle	标题中存在关键字的网页
info	一些基本信息
inurl	URL 存在关键字的网页
filetype	搜索指定文件类型

案例一：搜索存在敏感信息的网站

输入“intitle:管理登录 filetype:php”，这句话的意思为查询网页标题含有“管理登录”，并且为 php 类型的网站，Google 可以轻松地搜索到很多该类型的网站，如图 3-2 所示。

只需要一个关键字，你就可以利用 Google 找到存在某些特征的网站，以达到快速找到漏洞主机的目的。

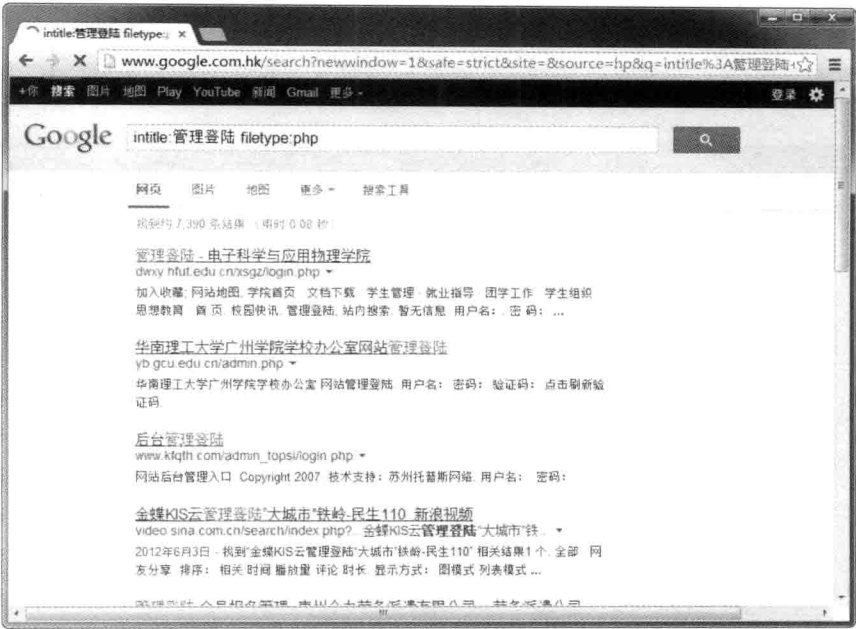


图 3-2 大量敏感信息泄露

案例二：搜集 Discuz 论坛主机

语法为：intext:Powered by Discuz

如果攻击者拥有 Discuz 漏洞，并且配合 Google 来寻找 Discuz 主机，那么后果是相当严重的。

Google 也不是万能的，它只能查询出蜘蛛爬行过的网页，而蜘蛛遵照网站根目录 robots.txt 的约定进行爬行，如果一些敏感目录不希望蜘蛛进行爬行，则可以写在网站根目录 robots.txt 中。虽然这样不会被蜘蛛爬行到，但是攻击者可以直接访问 robots.txt，如图 3-3 所示。

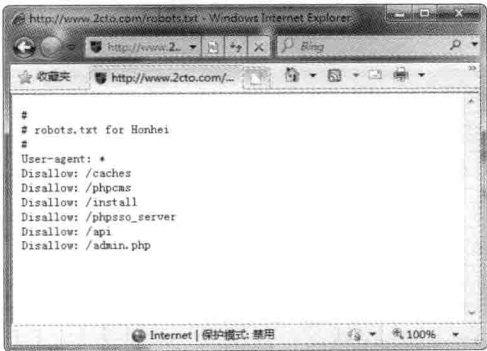


图 3-3 robots.txt 的访问

需要注意的是，不仅是 Google 能探测敏感信息，百度、雅虎等搜索引擎都是可以的，其方法与 Google Hack 很相似，这里不再一一赘述。

小知识：网络蜘蛛

蜘蛛在爬行时，会爬行整个网站，网站内所有的链接都会被一一提交到 Google 的数据库中。

那么如何完全隐藏网站的敏感信息呢？

从开发人员的角度来讲，一定要保证敏感信息不被外部引用。如果在一些对外暴露的页面中引用一些敏感目录，那么 Google 的蜘蛛就会顺藤摸瓜找到地址。同时，也要保证敏感信息的名子很复杂，否则很有可能被攻击者扫描出地址。

3.2 Nmap 初体验

3.2.1 安装 Nmap

Nmap 是一个开源的网络连接端扫描软件，用来扫描计算机开放的网络连接端，确定哪些服务运行在哪些连接端，并且推断计算机运行哪个操作系统。另外，它也用于评估网络系统安全。它是网络管理员必用的软件之一。

如今，Nmap 更是增加了许多实用的插件，可以用来检测 SQL 注射、网页爬行、数据库密码检测等，号称“扫描之王”。欲了解更多的信息，请登录官方网站（简称官网）：<http://www.namp.com>。

1. 获取 Nmap

Nmap 可以被快速安装到 Windows、UNIX、Mac 等操作系统中。本次安装系统为 Windows，Nmap 安装包可以在<http://nmap.org/download.html#windows>中下载，本次安装版本为 Nmap-6.25。

2. 安装 Nmap

安装 Nmap 是比较简单的，按照提示进行安装即可。

3. 安装 Nmap 后的配置

Nmap 安装完毕后，为了使用更方便，还需要进行一些环境变量的设置：用鼠标右键单击“我的电脑”，在弹出的菜单中选择“属性”→“高级系统设置”→“高级”→“环境变量”。在系统栏目里找到 Path，对 Path 进行编辑。

输入 Nmap 安装目录，本次安装为 D:\Program Files\Nmap\，如图 3-4 所示。

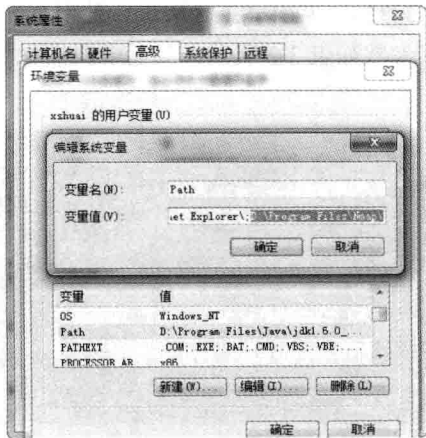


图 3-4 编辑环境变量

4. 使用 Nmap

打开 CMD 命令行，输入 Nmap，可启动 Nmap.exe。这里输入 Zenmap，来启动 Zenmap 的图形化界面，如图 3-5 所示。

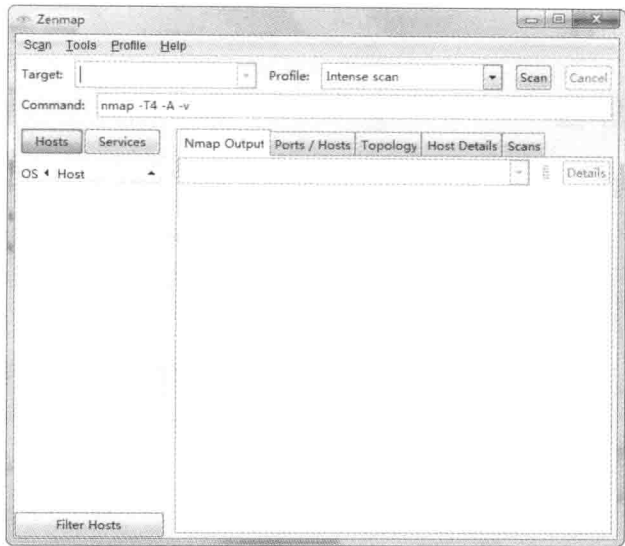


图 3-5 Zenmap 图形化界面

注：Zenmap 是 Nmap 的图形化界面，可以在安装 Nmap 目录下寻找到 Zenmap.exe。

小知识：

任何程序想要在 CMD 命令行下进行快捷访问，都必须配置环境变量。

3.2.2 探测主机信息

Nmap 支持多种扫描方式，包括 TCP Syn、TCP Connect、TCP ACK、TCP FIN/Xmas/NULL、UDP 等。Nmap 扫描的用法较为简单，并且提供丰富的参数来指定扫描方式。

案例一：扫描指定 IP 所开放的端口

输入命令：`nmap -sS -p 1-65535 -v 192.168.1.106`，表示使用半开扫描，指定端口为 1 到 65535，并且显示扫描过程，常用扫描参数如表 3-2 所示，如需多扫描参数，请参照 `Nmap -help` 命令，扫描结果如图 3-6 所示。

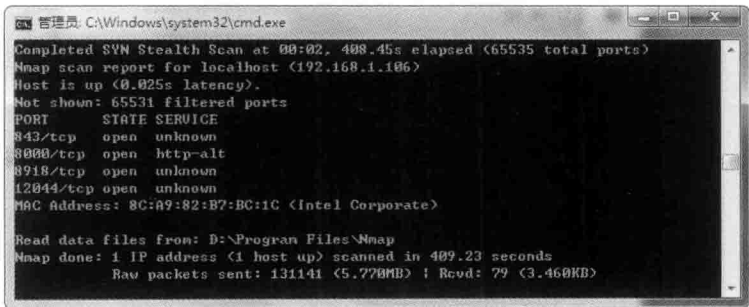


图 3-6 Nmap 扫描结果