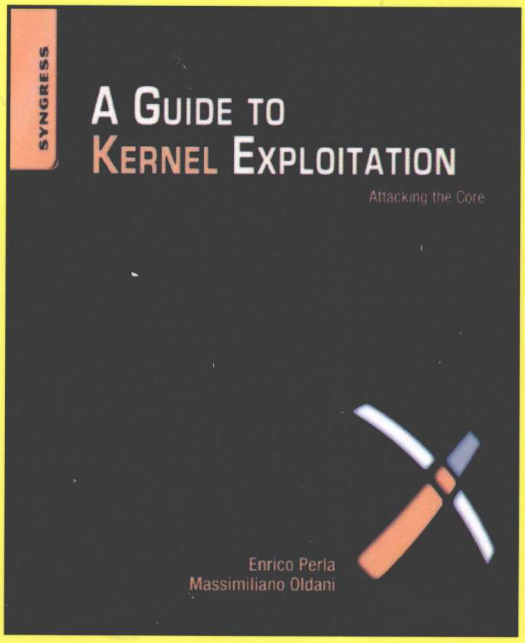
HZ BOOKS
华章科技

全面剖析内核漏洞利用的概念、方法和步骤，深入探讨UNIX、Mac OS X和Windows等操作系统内核的利用代码编写方法，并详细阐述了内核漏洞的防御方法及措施

安全技术大系
SECURITY

ELSEVIER



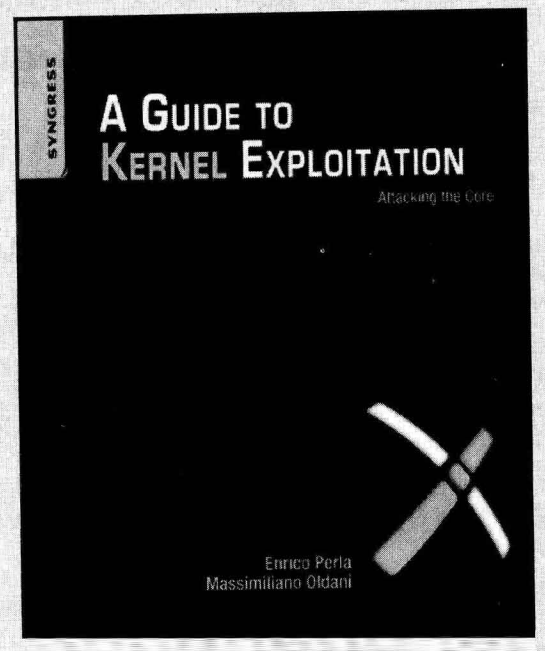

# 内核漏洞的利用与防范

A Guide to Kernel Exploitation: Attacking the Core

（美）Enrico Perla
Massimiliano Oldani 著

吴世忠　郁莲　郭涛　董国伟　译

机械工业出版社
China Machine Press




# 内核漏洞的利用与防范

A Guide to Kernel Exploitation: Attacking the Core

（美）Enrico Perla
Massimiliano Oldani 著

吴世忠 郁莲 郭涛 董国伟 译

机械工业出版社
China Machine Press

本书系统地讲解内核级别漏洞利用所需的理论技术和方法，并将其应用于主流操作系统——UNIX 家族、Mac OS X 和 Windows。本书分 4 个部分：第一部分介绍漏洞利用的目标、内核以及理论基础；第二部分深入介绍了目前主流操作系统的细节，并针对不同错误类别分别编写了漏洞利用程序。第三部分将关注点从本地场景转移到远程利用的情景；第四部分介绍未来内核的攻防模式。本书不仅从软件安全研究人员的角度谈论如何发现软件漏洞，也从软件开发者的角度给出了防止软件出现漏洞的方法，以帮助软件编程人员开发出安全的软件系统。

本书内容详实，实例丰富，可操作性强，涉及主流操作系统内核漏洞利用的各个方面，适合软件开发人员、测试人员、安全工程师等阅读。

A Guide to Kernel Exploitation: Attacking the Core
Enrico Perla and Massimiliano Oldani
ISBN: 978-1-59749-486-1

ISBN: 978-981-272-926-2


**本书版权登记号：图字：01-2011-2875**

**图书在版编目（CIP）数据**

内核漏洞的利用与防范 /（美）佩拉（Perla，E.）等著；吴世忠等译．—北京：机械工业出版社，2012.3
（信息安全技术丛书）
书名原文：A Guide to Kernel Exploitation:Attacking the Core

ISBN 978-7-111-37429-9

I. 内…　II. ①佩…　②吴…　III. 操作系统－程序设计　IV. TP316.2

中国版本图书馆 CIP 数据核字（2012）第 020500 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：秦　健
北京京师印务有限公司印刷
2012 年 3 月第 1 版第 1 次印刷
186mm×240mm • 23 印张
标准书号：ISBN 978-7-111-37429-9
定价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

# 译 者 序

随着信息技术的飞速发展，互联网日益成为人们生活中不可缺少的一部分，社交网络、微博、移动互联网、云计算、物联网等各种新技术、新应用层出不穷。但不管是 Facebook、twitter 等新兴互联网公司的迅速崛起，还是 Android 日益成为智能手机市场的主流操作系统，信息安全一直都是永恒的话题。“震网病毒”事件凸显网络武器的实战破坏能力，关键信息基础设施和政府网络保护已成为网络空间防御的新重点；“维基泄密”事件彰显网络空间攻防双方的不对称性，百密难免一疏成为保密防范永远的痛；这些信息安全事件存在一个共同点，那就是信息系统或软件存在的漏洞是问题的根源。因而，漏洞分析日益成为信息安全领域理论研究和实践工作的焦点，越来越引起世界各国的关注与重视。

随着操作系统安全性逐步提高，安全防护软件日臻完善，攻击者利用用户态漏洞实施攻击变得越来越困难，因此，内核漏洞日益成为攻击的焦点。内核漏洞的利用将给系统的安全带来巨大的威胁，甚至会彻底瓦解安全防护措施。因此，内核漏洞的挖掘与利用已成为研究热点。

为推动国内的漏洞分析和风险评估工作，提高国家信息安全保障能力和防御水平，中国信息安全测评中心长期跟踪和关注相关领域的理论进展和技术进步，有针对性地精选一些优秀书籍译成中文，供国内参考借鉴。

本书内容涵盖了开发内核级漏洞利用手段所需的主要理论和方法，包括：内核和内核漏洞利用的基本概念、内核漏洞的分类方法及主要类别、成功利用内核漏洞所需经历的三个阶段等，并在研究 UNIX 家族、Mac OS X 和 Windows 等不同类型操作系统自身特性的基础上，深入探讨了针对它们的利用代码编写方法，并建立了一套面向操作系统内核漏洞利用开发的行之有效的方法论，从攻防双方不同的视角，介绍了内核漏洞远程利用的方法以及防御措施。书中大量的实际漏洞利用技术及案例，不但能够帮助读者更好地理解那些深奥的理论，还可以帮助安全研究人员更加深入地了解内核攻击的方式和方法，为防御系统级的攻击，降低安全隐患提供了重要的理论支撑和技术保障。

在本书翻译过程中，译者得到中国信息安全测评中心的张普含、王嘉捷、柳本金、王欣等同志，以及北京大学软件与微电子学院的何建杉、余天天、王斌、万成铖、沈阳、张开元、张任伟、李金诺、彭磊、汤云杰等师生的支持和帮助，在此深表感谢。

本书得到中国信息安全测评中心“漏洞分析与风险评估”专项工程的支持。

# 序　言

最初邀请我写本书序言时，我拒绝了，因为我觉得我不应该出现在他们的光芒中，这些光芒是他们辛勤工作应得的。然而，在阅读了这本书的一些章节之后，我认识到，如果错过这个机会将会非常遗憾，能为世界上最著名的两位内核漏洞利用者的书写序，我感到无比荣幸。

我很少阅读关于漏洞利用技术的书籍，因为很多书只提供了很少或是过时的知识，或者只是简单地枚举出其他人完成的攻击。另外，书籍并不能提供动手进行漏洞利用开发的学习，或是带来在数天努力工作之后一个“#”命令行提示的乐趣，特别是一个内核漏洞被利用后所带来的快感。是时候将这种感觉写在纸上了，它将节省其他开发人员的时间，减少令人崩溃和头痛的处境。

除了需要漏洞利用重要的技巧和艺术之外，编写漏洞利用特别是内核漏洞利用需要对操作系统原理具有深层次理解。这本书绝对有助于实现上述目的，并且还能丰富我书架上内核和驱动编程方面的书籍。

我当然知道这本书适合的读者，希望大量的内核和驱动开发人员也来读这本书。我的下个内核代码审查的工作就要开始了，希望这本书在那之前就可以出版。

Sebastian Krahmer

系统程序员与漏洞利用工程师

# 前　言

## 本书概览

目前针对用户态漏洞利用的安全措施较以往有所增加，同时，内核态的漏洞利用也变得越来越流行。本书覆盖了开发可靠和有效的内核态攻击所需的理论技术和方法，并将它们应用于不同的操作系统——UNIX 家族、Mac OS X 和 Windows。

内核利用既是艺术也是科学。每个操作系统都有其自身的特点，所以必须建立攻击模型以便充分分析其目标。本书讨论了最流行的操作系统并介绍如何控制它们。

本书介绍了主流操作系统的概念和安全策略，可以帮助信息安全研究人员更加深入了解内核攻击的方式和方法，为抵制系统级内核攻击，降低安全隐患提供了重要的理论支撑和技术保障。

## 本书的组织结构

本书分 4 个部分，共 9 章。

**第一部分　内核态** 介绍了本书的目标——内核，并讨论了本书后续内容所依赖的理论基础。这一部分包括：

- **第 1 章　从用户态利用到内核态利用**　这一章介绍了漏洞利用的世界，并分析了致使安全研究人员和攻击者将漏洞利用的焦点从用户态应用程序转到系统内核上来的原因。
- **第 2 章　内核漏洞分类**　这一章讨论了不同类型漏洞（错误类别）的分类，并阐述了它们的共同特性和利用方法。越是能够描述多种错误类型的模型，我们越是能够针对其设计出可靠有效的利用技术。这个分类也便于我们从另一方面看待问题：防御。越多地理解错误类型，越是能够创造出更好的保护和防御手段。
- **第 3 章　成功内核利用进阶**　这一章剖析了内核利用的各个阶段，并描述了针对第 2 章中每个错误类型的最佳利用方法和技术。鉴于操作系统实现子系统的方法不一样，本章致力于提供能够应用于不同内核和体系结构的利用方法。

**第二部分　UNIX 家族、Mac OS X 和 Windows**　在这一部分中我们将深入探讨不同操作系统的自身特性，并编写针对不同错误类型的利用代码。对于每种操作系统，我们介绍了相应的调试工具和方法，这在编写漏洞利用代码时非常实用。在本部分中，我们介绍了对真实漏洞的利用，而不是人为设计的例子。这一部分包括：

- **第 4 章　UNIX 家族**　这一章分析了 UNIX 类系统，主要讨论了 Linux 并简单讨论了（开源）Solaris 操作系统。部分章节也涉及这些操作系统提供的主要工具（如动态追踪、内核调试器等）的调试技术。

- **第 5 章　Mac OS X**　这一章覆盖了目前非常流行的 Mac OS X 操作系统，主要讨论了其 Leopard 版本。同时分析了主要的错误类型（如栈和堆的利用），在查找漏洞时，我们介绍了如何利用闭源内核部分进行逆向工程分析。
- **第 6 章　Windows**　这一章覆盖了当今最流行的操作系统——微软 Windows。和之前的内容不同，在这一章中，我们没有涉及内核的源代码；相反，我们对内部（以及漏洞 / 攻击方法）的理解来自对不同内核部分的逆向工程。本章着重介绍了逆向分析工具的调试方法，这是非常重要的。

**第三部分　远程内核漏洞利用**　这一部分将我们的关注点从本地场景（对于内核攻击来说这是常见的场景）转移到远程情况。实际上，我们进入了更加复杂的境地，我们学到的在本地攻击中使用的许多技术已经不适用了。虽然错误类型仍然一样，但是我们需要为我们的“武器库”增加新的“武器”。第三部分共两章，重温了本书之前的部分（第一部分偏向理论，第二部分偏向实践）。这一部分的主要内容包括：

- **第 7 章　远程内核漏洞利用面临的挑战**　这一章从理论开始介绍，分析了为什么以及在多大程度上远程情境会影响我们的方法，并给出新的技术来解决远程利用问题。虽然这一章更偏向理论，但是也给出了一些实际例子，特别关注于 Windows 操作系统，而 UNIX（Linux）的情景将在第 8 章阐述。
- **第 8 章　一个 Linux 案例**　这一章依次分析了针对一个真实漏洞的可靠利用、一次性利用和远程利用的开发——在 Linux 内核中发现的一个影响 SCTP 子系统（http://cve.mitre.org/cgi-bi/cvename.cgi?name=CVE-2009-0065）的 bug。

**第四部分　展望**　这一部分总结全书，结束我们对内核安全的分析，由单独的一章组成：

- **第 9 章　内核演变：未来内核攻防模式**　在这一章中，基于我们所学的内核利用知识对未来进行展望。为了深入探讨攻击和防御技术，在这一章中我们讨论了计算机安全的基础：信息流控制。然后，我们将其作为标准来检查和理解一些 bug 和漏洞利用的基本特点，这样我们才能够更好地展望未来。

本书中介绍的利用实例和工具的代码在网站 www.attackingthecoe.com 上可以找到，它也是报告错误的主要参考；您也可以联系我们获得更多的资料。

## 小结

编写一本书是一个繁琐但很有趣的过程。它给了作者一个将大脑中浮现的感兴趣的概念表达出来的机会。编写这本书对我们来说是一个挑战。我们力争用准确而清晰的陈述来传递那种发现新的防御方法和提供有价值的攻防信息时的激情和乐趣。我们希望读者能够喜欢我们的成果，如同我们乐于将它奉献出来那样。

# 致　谢

本书献给所有那些相信，就安全而论，代码编辑器（和 shell）的功能比邮件客户端的功能更重要的人。

本书得到许多人的帮助、支持和耐心指点。我们要特别感谢：

- Matthew Cater、Rachel Roumeliotis、Graham Speake、Audrey Doyle 和 Julie Ochs，感谢他们为我们提供的编排。
- Nemo，感谢他为第 5 章提供的宝贵资料，以及持续的反馈。
- Ruggiero Piazzolla，感谢他帮助建设网站，尤其是，让它更易于使用。
- Marco Desiati 和 Michele Mastrosimone，感谢他们的艺术设计。我们之前的尝试和他们设计后的结果相比简直是小孩子的涂鸦。
- Abh，感谢他不知疲倦地花费大量时间修改本书的内容和代码示例。
- Sebastian Krahmer，感谢他为本书写了序言，并审读了多个章节，感谢他参与技术和观点的探讨。
- （排名不分先后）Andrea Lelli、Scott Rotondo、xorl（精彩的博客，顺便提一下）、Brad Spengler、Window Snyder、Julien Vanegue、Josh Hall、Ryan Austin、Bas Albert、Igor Falcomata'、clint、Reina Alessandro、Giorgio Fedon、Matteo Meucci、Stefano Di Paola、Antonio Parata、Francesco Perna、Alfredo Pesoli、Gilad Bakas、David Jacoby 和 Ceresoni Andrea，感谢他们对本书的反馈和意见，并帮助提升了本书的总体质量。我们知道肯定还有人在这里没有提及（再也没有比“你知道你是谁”这句话更合适的了）……对此感到抱歉。

最后，但同样重要的是，还有一些需要特别感谢的人没有在这里列出。

Enrico 要感谢 Mike Pogue 和 Jan Setje-Eilers 以及作为一家人的 Lalla、Franco 和 Michela 做的所有事。上午 9 点和晚上 10 点半的电话，使得离家千里之外的人感到家的温暖。

Massimiliano 要感谢：

- halfdead（笔名——译者注），让我看到了美妙的安全世界仍然有很多乐趣。
- 我的家人：Noemi、Manuela、Giuseppe、Stefano(Bruce)，以及 Irene，在我编写本书的数月之中，他们放弃了许多周末来支持我；我真的很爱他们。

# 作者简介

Enrico Perla 目前是 Oracle 公司的内核开发人员。他于 2007 年获得 Torino 大学学士学位，并于 2008 年获得 Trinity Dublin 大学计算机科学硕士学位。他的兴趣范围包括底层系统编程、底层系统攻击、漏洞利用和漏洞利用对策等。

Massimiliano Oldani 目前担任 Emaze 网络的安全顾问。他主要的研究课题包括操作系统安全和内核漏洞。

## 技术编辑简介

Graham Speake 是 Yokogawa 电气公司首席系统架构师，该公司是一个工业自动化供应商。他目前为内核开发人员，并为多个国家的客户提供安全咨询和解决方案。他擅长工业自动化和过程控制安全、渗透测试、网络安全和网络设计。Graham 经常在安全会议上发表演讲，并经常给世界各地的客户进行安全培训。Graham 的背景包括：BP 和 ATOS/Origin 的安全顾问，以及福特汽车公司的工程师。

Graham 拥有位于威尔士的 Swanseada 大学学士学位，并且是 ISA 的成员之一。

# 目　录

## 第三部分 远程内核漏洞利用

## 第四部分　展望

# 第一部分

# 内 核 态

我们对于内核漏洞利用世界的探索之旅就从这儿开始。在本书的第一部分中，我们将讲述内核是什么，为什么安全社区对它如此关注，以及内核级的漏洞是什么样的，如何成功地利用它们。我们并不会直接讲述具体的操作系统的细节以及漏洞利用技术，而是首先帮助您理解底层的内核概念和内核漏洞利用方法。这不仅能够使您更容易地研究本书中介绍的不同操作系统（特别是在第二部分中），从而抽丝剥茧地了解其内部细节，而且也能使您在面对因不断升级而变得极其复杂的内核时游刃有余。

# 第 1 章

# 从用户态利用到内核态利用

**本章主要内容**

- 内核和内核漏洞利用的世界
- 为什么用户态漏洞利用不再有效
- 一个漏洞利用者的内核观
- 开源操作系统和闭源操作系统

## 引言

本章介绍了本书的研究对象——内核。我们将首先介绍内核的知识，然后分析为什么漏洞利用者把关注点从用户态的漏洞攻击转移到内核上，以及用户态漏洞利用和内核态漏洞利用的不同之处。然后会把关注点放在不同内核的区别上，例如 Windows 内核和 UNIX 内核的不同，与此同时，我们也会探索计算机体系架构的变化对内核漏洞利用所产生的重要影响，例如，同一段代码在 32 位的系统中漏洞利用可能是可行的，而在 64 位系统中却是不可行的，或者在 x86 机器上是可行的，而在 SPARC 机器上却是不可行的。最后简单地讨论了在开源操作系统和闭源操作系统上进行内核漏洞利用的差异。

## 内核和内核漏洞利用的世界

在踏上通向内核漏洞利用世界的征途之前，有一个需要首先明确的任务：什么是内核，以及漏洞利用意味着什么。当您看到一台电脑的时候，可能会想到一系列互相连接的物理设备（如 CPU、主板、内存、硬盘、键盘等），可以使用这些物理设备方便快捷地写邮件、看电影以及网上冲浪等。在硬件和应用软件之间存在着一层软件，它负责使硬件高效地工作，并建立基础设施，使应用软件运行在基础设施之上。这层软件就是操作系统，它的核心就是内核。

在现代的操作系统中，内核负责那些常规化的事情，如管理虚拟内存、硬件驱动访问、输入输出处理等。内核通常比一般的用户程序大，它是复杂而有趣的一段代码，经常是由汇编语

言（低级机器语言）和C语言混合编写而成。此外，内核通过一些底层的体系架构机制把自己和其他运行的程序隔离开。事实上，大部分的指令集体系架构（Instruction Set Architecture，ISA）至少提供两种执行模式：权限模式和非权限模式，在权限模式下，所有的机器级指令都是可用的，而在非权限模式下，只有一部分指令是可用的。此外，内核通过在软件级别上实现与用户程序的隔离来保护自己。当建立虚拟内存子系统时，内核会确保其能够访问任何进程的地址空间（即虚拟内存的地址范围），而用户进程只能访问用户态内存但不能访问内核态的内存。我们称只让内核可见的内存为内核态内存，而用户进程可见的内存为用户态内存。内核态中的代码执行拥有所有权限，并可以访问系统中任何有效的内存地址，而用户态中的代码执行对内存的访问则会受到如前所述的所有限制。这种基于硬件和软件的隔离是必须的，可以避免内核受到来自用户态程序的意外破坏、不良行为的篡改或者恶意代码的破坏。

对于一个安全和稳定的系统来说，保护内核免受其他运行程序的干扰是第一步，但这还远远不够：还应该在不同的用户态程序中设置不同程度的保护。考虑一个典型的多用户系统环境。每一个用户都希望拥有一个私有的文件系统区域，这样它们就可以在私有的文件系统里存储自己的数据、运行自己的程序（例如邮件读取软件），而不用担心这些数据和程序被其他用户关闭、修改和窃取。同样，为了使一个系统可用，必须有某种方法来识别、添加和删除用户或是限制用户对共享数据的影响。例如，一个恶意的用户不应该占用文件系统的所有可用空间或者系统互联网连接的所有带宽。因此，内核在软件级别上提供了为不同用户分配不同权限的功能。

用户由唯一的值来识别，通常是一个数字，称为用户标识（userid）。在所有用户标识中，有一个用户标识是用来标识特殊用户的，这个特殊用户拥有比普通用户更高的权限，负责管理其他用户、设置使用权限、配置系统之类的管理任务。在 Windows 世界中，这个特殊用户叫做“Administrator”；而在 UNIX 世界中，这个特殊用户习惯上称为 root，它的 uid（userid）通常赋为 0。在本书后面的内容中，将用一个通用的术语“超级用户”来指代这个特殊用户。

超级用户被授予了修改内核的权力。原因很明显：和其他软件一样，内核也需要升级，例如要修改潜在的漏洞、支持新设备等。如果得到了超级用户的控制权，就能完全控制这台计算机。所以，获取超级用户的控制权成为系统攻击者的目标。

**注意**　超级用户通过传统的权限隔离架构与其他没有权限的用户区别开来。所以权限用户和非权限用户们是非此即彼的关系：如果一个用户需要运行某种权限操作 X，它就必须是权限用户，这样，它就拥有了执行 X 之外的权限操作的权力。您将会看到，可以从安全的角度出发通过以下方法改进这个模型，可以把所有权限按类型拆分，然后只赋予用户所需要的权限以便完成某项特殊任务。这样的话，成为超级用户并不意味着获取控制整个系统的能力，因为真正控制一个特殊用户态程序可以做什么或不可以做什么是赋予它的权限。

## 漏洞利用的艺术

“我希望我能设法证明缓冲区溢出漏洞利用应该是一种艺术。”[1]

——Solar 设计者

在攻击者能够获取超级用户权限的各种方法中，漏洞利用的成功实施通常是一件最令人兴奋的事。初学者经常把漏洞利用看成很神奇的过程，然而奇迹是不存在的，所谓的“奇迹”只不过是创造力、智慧和许多刻苦努力的结果。换句话说，这是一门艺术。其实漏洞利用的原理非常简单：软件肯定有 bug，而这些 bug 会让软件运行不正常，或者错误地执行它本应正常处理的任务。利用 bug 意味着把这种不正常的行为作为攻击者实施攻击的有利手段。并不是所有的 bug 都是可利用的，只有那些称为漏洞（vulnerability）的 bug 才能利用。分析一个应用程序来确定漏洞的过程称为审计（auditing）。这个过程包含以下几步：

- 读入应用程序源代码（如果有源代码的话）。
- 对应用程序的二进制代码实施逆向工程，即读入编译过代码的反汇编码。
- 模糊测试应用程序接口，就是将随机或者基于模式自动生成的数据作为输入来测试程序。

审计过程既可以人工执行，也可以用静态或动态的分析工具来辅助进行。介绍审计的过程已经超出了本书的范围，如果您感兴趣，请参考本章最后“相关阅读”一节，那里介绍了关于审计方面的书。

漏洞通常分为若干不同类别。如果您曾接触过关于安全方面的邮件、博客或者电子杂志，应该听说过缓冲区（栈或堆）溢出、整数溢出、格式字符串和竞态条件。

**注意** 针对上述的漏洞类别，我们将在第 2 章更详细地描述。

本书对大多数术语并没有单独去解释，我们认为您应该已经明确它们的含义，而且在本书中准确理解它们的含义也不是很重要的。重要的是，所有属于某种类别的漏洞拥有一些相似的模式和漏洞利用方法。学习这些漏洞模式和漏洞利用方法（通常称为漏洞利用技术）对于漏洞攻击是非常有帮助的。发现漏洞模式和漏洞利用方法可能会极为简单，也可能会非常具有挑战性，这实质上就是漏洞利用者的创造力将漏洞利用流程转换为一种艺术形式。首先，一个漏洞利用方法必须足够可靠，可以应用到足够广泛的目标上。如果一个漏洞利用方法只对一种特定的场景有效或者仅仅是让程序崩溃，那么这个漏洞利用方法的价值并不大。除了要可靠外，一个漏洞利用方法必须高效。换句话说，漏洞利用者应该尽量减少使用蛮力，尤其是当这样做可能引起被攻击机器的警报时。

漏洞利用可以锁定本地或远程的服务：

- 本地漏洞利用（local exploit）要求攻击者已经获得访问目标机器的权限，其目的是提升攻击者的用户权限，直至获得对整个系统的控制权。
- 远程漏洞利用（remote exploit）首先要锁定攻击者没有访问权限的机器，这样他可以通过网络连接到将要攻击的机器。相比本地攻击，远程攻击是更有挑战性（并且，从某种程度来说权力更强大）的漏洞利用类型。通过本书您将发现，要想成功地进行漏洞利用，尽量多地搜集目标机器的信息是必要的，当攻击者可以访问目标机器的时候，就更容易

搜集目标机器的信息。所以，远程漏洞利用的目标就是使攻击者有权限访问远程机器。如果被攻击的应用程序运行在高权限级别状态下，那么攻击者还会获得提升用户权限的额外收获。

如果仔细分析一个常规的漏洞利用，会发现漏洞利用主要分三个部分：

- **准备阶段**（preparatory phase）：搜集目标机器的信息，创建有利的攻击环境。
- **编写** shellcode：这是一段机器语言指令，当执行时，通常会提升攻击者的权限级别或是执行命令（如 shell 的一个新实例）。在接下来的代码段里可以看到，机器指令序列可以编码成很容易被攻击代码操纵的十六进制代码，并且放置在被攻击机器的内存中。
- **触发阶段**（triggering phase）：将 shellcode 编写好放入目标进程（例如通过输入注入）后，会触发漏洞并使目标程序执行控制重定向到 shellcode。

```
char kernel_stub[] =
"\xbe\xe8\x03\x00\x00"                  // mov    $0x3e8,%esi
"x65\x48\x8b\x04\x25\x00\x00\x00\x00"   // mov    %gs:0x0,%rax
"\x31\xc9"                              // xor    %ecx, %ecx  (15
"\x81\xf9\x2c\x01\x00\x00"              // cmp    $0x12c,%ecx
"\x74\x1c"                              // je     400af0
<stub64bit+0x38>
"\x8b\x10"                              // mov    (%rax),%edx
"\x39\xf2"                              // cmp    %esi,%edx
"\x75\x0e"                              // jne    400ae8
<stub64bit+0x30>
"\x8b\x50\x04"                          // mov    0x4 (%rax),%edx
"\x39\xf2"                              // cmp    %esi,%edx
"\x75\x07"                              // jne    400ae8
<stub64bit+0x30>
"\x31\xd2"                              // xor    %edx,%edx
"\x89\x50\x04"                          // mov    %edx, 0x4(%rax)
"\xeb\x08"                              // jmp    400af0
<stub64bit+0x38>
"\x48\x83\xc0\x04"                      // add    $0x4,%rax
"\xff\xc1"                              // inc    %ecx
"\xeb\xdc"                              // jmp    400acc
<stub64bit+0x14>
"\x0f\x01\xf8"                          // swapgs (54
"\x48\xc7\x44\x24\x20\x2b\x00\x00\x00"  // movq   $0x2b, 0x20(%rsp)
"\x48\xc7\x44\x24\x18\x11\x11\x11\x11"  // movq   $0x11111111, 0x18(%rsp)
"\x48\xc7\x44\x24\x10\x46\x02\x00\x00"  // movq   $0x246,0x10(%rsp)
"\x48\xc7\x44\x24\x08\x23\x00\x00\x00"  // movq   $0x23, 0x8 (%rsp)/* 23
32-bit , 33 64-bit cs */
"\x48\xc7\x04\x24\x22\x22\x22\x22"      // movq   $0x22222222,(%rsp)
"\x48\xcf";                             // iretq
```

攻击者的一个目标就是尽可能成功地把代码执行控制重定向到 shellcode 开始的内存地址。一种简单却低效的方法是尝试所有可能的内存地址：攻击者一个接一个地尝试，如果所尝试的地址错误，用户程序将崩溃，然后攻击者再尝试下一个地址；在某一点上攻击者最终触发 shellcode。这种方法称为蛮力法（brute forcing），蛮力法通常对时间和资源的要求较高，尤其