第5章　系统安全

系统安全是网络空间安全的核心，需要用系统化的思维方式，采用系统工程方法建设可信安全系统。网络空间由于存在许多安全问题，包括各种各样的攻击，使得人们处于安全威胁包围之中。保障网络空间的安全性，需要面对网络空间的现实系统安全威胁。以实现安全策略为基础，权衡计算环境各组成组件的功能、威胁、代价等因素，引入可信计算才有可能实现系统安全。为此，本章针对网络空间组成要素，讨论网络空间中单元计算系统的安全及其实现，包括系统安全基础理论、操作系统安全、系统安全硬件基础，旨在为实现系统安全提供一定技术支持。

5.1　系统安全基础

网络空间是一个多维复杂的计算环境，主要由各式各样的计算机通过网络按照一定的体系结构连接而成。计算机是网络空间的承载主体，计算机设备在硬件和软件系统的协同工作下实现“计算”功能。因此，通常从硬件安全和软件安全两个方面讨论系统安全问题。系统硬件安全包括芯片等硬件的安全和物理环境的安全。系统安全建立在硬件系统和软件系统安全基础之上。

5.1.1　系统与系统安全

系统是指由若干相互联系、相互作用的要素所构成的有特定功能与目的的有机整体。此处的系统是由网络设备及其相关和配套的设备、基础设施等组件构成的，按照一定的应用目标和规则对信息进行采集、加工、存储、传输、检索等处理的人机交互系统。简言之，系统是由组件连接起来构成的整体。这个定义说明，一个系统是一个统一的整体，同时系统由多种组件构成。系统不仅是组件的集合和连接的集合，更是一个完整的计算单元，但组件与组件之间的关系又内外有别，即属于同一个系统的组件之间的关系与该系统外其他组件之间的关系不同，系统是存在边界的。边界把系统包围起来，以区分内部组件和外部组件。位于系统边界内部的属于系统的组成组件，位于系统边界外部的属于系统的环境。系统的边界有时是明显的，容易确定；有时是模糊的，难以确定。例如，一部手机的边界可以说是外壳，看得见摸得着，而一个操作系统的边界却很难严格划分。有时，系统的边界也不是唯一的、一成不变的，可能会随着观察角度的不同发生变化，但无论如何，系统存在边界。

系统安全是在系统生命周期内，运用系统安全工程和系统安全管理方法，辨识系统中的隐患，并采取有效的控制措施使其危险性最小，从而使系统在规定的性能、时间和成本范围

内达到最佳的安全程度。系统安全的基本原则是在一个新系统的构思阶段就必须考虑其安全性的问题，制定并执行安全工作规划（系统安全活动），而不仅限于事后分析并积累事故处理经验，系统安全活动贯穿于系统的整个生命周期。

网络空间的系统安全聚焦于系统的安全性。系统安全包含两层含义：一是以系统思维应对网络空间安全问题；二是如何应对系统所面临的安全问题。两者相辅相成，深度融合。系统安全的基本思想是，在系统思维的指导下，从系统建设、使用和废弃的整个生命周期应对系统所面临的安全问题，正视系统的体系结构对系统安全的影响，以生态系统的视角全面审视网络空间安全性。上述两层含义说明，研究系统安全需要正确的方法论，要利用系统化思维方式，通过系统安全措施，建立和维护系统的安全性，从系统的全生命周期权衡系统的安全性。

按照系统化思维方式，网络空间中系统的安全性是系统的宏观属性，不能简单地依靠系统的微观组成部件建立起来。系统的安全性在很大程度上依赖微观组成部件的相互作用，但是难以把控。例如，用经典的观点可以把网络空间中系统的安全性描述为机密性、完整性和可用性。以操作系统为例，操作系统由进程管理、内存管理、外设管理、文件管理及处理器管理等子系统组成，即便各子系统都能保证不泄露机密信息，操作系统也无法保证不泄露机密信息。隐蔽信道泄露机密信息就是其中一个典型的例子。也就是说，操作系统的机密性无法还原到子系统之中，隐蔽信道泄露是由于多个子系统的相互作用而引起的。又如，研究网络购物系统的安全性，仅研究构成该网络购物系统中的计算机、软件或网络等安全性是不够的，必须把整个网络购物系统看作一个整体，才有可能找到解决其安全问题的具体措施。

系统的体系结构对系统的安全性至关重要。系统的体系结构可划分为微观体系结构和宏观体系结构两个层面。从计算技术的角度看，微观体系结构的系统主要是机器系统，由计算机软件、硬件组成；宏观体系结构的系统是系统的生态。其中，机器系统又可细分为硬件、操作系统和应用系统等层次。本章主要从操作系统安全、系统安全硬件基础角度讨论系统的安全问题。

5.1.2 系统安全原理

系统安全是人们为解决复杂系统的安全性问题而研究、开发出来的安全理论、方法体系，是系统工程与安全工程结合的完美体现。系统安全活动贯穿于整个系统生命周期，直到系统废弃为止。网络空间的系统安全在遵从系统安全一般原理的基础上，应从网络系统建设者的角度，将安全理论贯彻落实到网络系统建设之中。

1. 基本原则

基本原则是在一个新系统的构思阶段就必须考虑安全性。在网络空间中，系统设计与实现在其生命周期中是两个重要的阶段。在长期的工程设计实践中已经形成了许多对系统安全具有重要影响的原则，其基本原则主要为限制性原则、简单性原则和方法性原则。

1）限制性原则

限制性原则主要用于制定并执行安全工作规划（系统安全活动），属于事前分析和预防。

（1）最小特权原则。

所谓最小特权，指“在完成某种操作时所赋予网络中每个主体（用户或进程）必不可少的特权”。最小特权原则是应限定网络中每个主体所必需的最小特权，确保可能的事故、错误、

网络部件的篡改等原因造成的损失最小。也就是说，系统中执行任务的实体（程序或用户）应该只拥有完成该项任务所需特权的最小集合。如果只要拥有 N 项特权就足以完成所承担的任务，就不应该拥有 $N+1$ 项或更多的特权。

（2）失败、保险默认安全原则。

安全机制对访问请求的决定应采取默认拒绝方案，不采取默认允许方案。只要没有明确的授权信息，就不允许访问，而不是只要没有明确的否定信息，就允许访问。例如，当登录失败过多，就锁定账户。

（3）完全仲裁原则。

安全机制实施的授权检查必须能够覆盖系统中的任何一个访问操作，避免出现能逃过检查的访问操作。该原则强调访问控制的系统全局观，除了涉及常规的控制操作，还涉及初始化、恢复、关停和维护等操作。全面落实完全仲裁原则是发挥安全机制作用的基础。

（4）特权分离原则。

对资源访问请求进行授权或执行其他安全相关行动，不能仅凭单一条件就做决定，应该增加分离的条件因素。例如，给一把密码锁设置两个不同的钥匙，分别让两人各自保管，必须两人同时拿出钥匙才可以开锁。

（5）信任最小化原则。

系统应该建立在尽量少的信任假设的基础上，减少对不明对象的信任。对于与安全相关的所有行为涉及的所有输入和产生的结果，都应该进行检查。

2）简单性原则

简单性原则包括机制经济性原则、公共机制最小化原则和最小惊讶原则。

（1）机制经济性原则。

任何系统设计与实现都不可能保证绝对没有缺陷，所以应该把安全机制设计得尽可能简单扼要。为了排查此类缺陷，检测安全漏洞，有必要对系统代码进行检查。简单扼要的机制比较容易处理，复杂、庞大的机制则比较难以处理。

（2）公共机制最小化原则。

如果系统中存在可以由两个以上用户共用的机制，应该把共用机制数量减到最少。每个可共用的机制，特别是涉及共享变量的机制，都代表着一条信息传递的潜在通道，设计机制时要格外小心，以防止破坏系统的安全性，造成信息泄露。

（3）最小惊讶原则。

系统的安全特性和安全机制的设计应该尽可能符合逻辑且简单，与用户的经验、预期和想象相吻合，尽可能少给用户带来意外或惊讶，以便用户自觉自愿、习以为常地接受和正确使用，并且在使用中少出差错。

3）方法性原则

方法性原则包括公开设计原则、层次化原则、抽象化原则、模块化原则、完全关联原则和设计迭代原则。

（1）公开设计原则。

不能把系统安全性的希望寄托在保守安全机制设计秘密的基础上，应该在公开安全机制设计方案的前提下，借助容易保护的特定元素，如密钥、口令或其他特征信息等，增强系统的安全性。公开设计思想有助于使安全机制接受广泛的审查，进而提高安全机制的鲁棒性。

(2) 层次化原则。

采用分层的方法设计和实现系统,以便某层的模块只与其紧邻的上层和下层模块进行交互,通过自顶向下或自底向上的技术对系统进行测试,每次可以只测试一层。

(3) 抽象化原则。

在分层的基础上,屏蔽每一层的内部细节,只公布该层的对外接口,以便每一层内部执行任务的具体方法可以灵活确定、及时变更,不会对其他层次的系统组件产生影响。

(4) 模块化原则。

把系统设计成相互协作的组件集合,用模块实现组件功能,用相互协作的模块集合实现系统,使得每个模块的接口就是一种抽象。

(5) 完全关联原则。

把系统的安全设计与实现和该系统的安全规格说明紧密联系起来。

(6) 设计迭代原则。

进行规划设计时,考虑必要时可以改变设计。如果系统的规格说明与系统的使用环境不匹配,则需要改变设计,以便减弱对安全性的影响。

2. 威胁建模

威胁建模是分析应用程序安全性的一种方法。所谓威胁建模,就是标识潜在安全威胁并审视风险缓解途径的过程。威胁建模的目的是在明确了系统的本质特征、潜在攻击者的基本情况、最有可能被攻击的角度、攻击者最想得到的利益等情况后,为防御者提供应采用的控制或防御措施的机会。

威胁建模是一种结构化的方法,能够帮助识别、量化和解决与应用程序相关的安全风险。威胁建模不是代码审查方法,却是对安全代码审查过程的补充。在软件开发生命周期(SDLC,Software Development Life Cycle)中包含威胁建模可以确保从一开始就以内置的安全性开发应用程序,与作为威胁建模过程一部分的文档相结合,可以使审阅者更好地理解系统,并且可以看到应用程序的入口点及每个入口点的相关威胁。威胁建模的概念并不新鲜,但近年来有了明显的思维转变。现代威胁建模需要从潜在攻击者的角度来看待系统,而不是防御者的角度。

威胁建模可以在软件设计和在线运行时进行,遵循“需求→设计→开发→测试→部署→运行→结束”的软件开发生命周期。在新系统或新功能开发设计阶段,增加安全需求说明,可以通过威胁建模满足软件安全设计需求。如果系统已经上线运行,可以通过威胁建模发现新的风险,作为渗透测试的辅助工作,尽可能地发现所有的安全漏洞。

3. 访问控制

对系统进行安全保护比较理想的方法是提前做好防护准备,防止安全事件的发生。访问控制的目标就是防止系统中出现不按规矩对资源访问的事件。

访问控制包含以下三方面的含义。

(1) 机密性控制,保证数据资源不被非法读取。

(2) 完整性控制,保证数据资源不被非法增加、改写、删除和生成。

(3) 有效性控制,保证数据资源不被非法访问主体使用和破坏。

访问控制是系统机密性、完整性、可用性及合法使用的基础。访问控制的一般模型如图 5.1 所示,其核心部分由访问控制仲裁和安全(控制)策略组成。

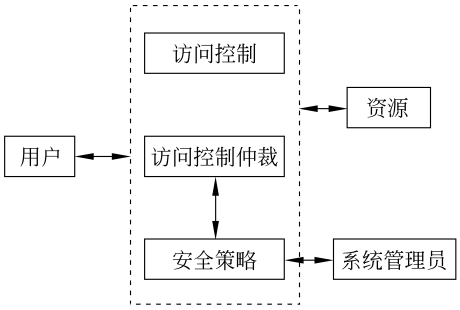

图 5.1　访问控制的一般模型

访问控制模型不仅是理论上的设计，重要的是能够在实际信息系统中实现，以确保信息系统中用户使用的权限与所拥有的权限相对应，防范用户进行非授权的访问操作。既要实现访问控制保证授权用户使用的权限与其所拥有的权限对应，又能拒绝非授权用户的非授权行为。目前，网络信息系统实现访问控制的模型主要有目录表、访问控制矩阵、访问控制列表、访问控制能力表、访问控制安全标签列表及权限位等。

4. 安全检测

网络空间中的系统及环境存在大量不确定因素，时刻处于变化中，安全事件也不可能完全避免，也不可能根除。虽然安全事件不可避免，但是应能感知安全事件的发生，以便采取措施，增强事后补救能力。系统的安全监测就是要提供一种安全机制，从开机引导到运行各个环节进行功能性监测，以发现系统中某些重要组成部分是否存在被攻击的迹象。

（1）入侵检测。

入侵检测是对入侵行为的发觉，是在安全监测中被广泛采用的一种重要形式。入侵检测通过对信息网络系统中的若干关键点收集信息并对其进行分析，从中发现网络或系统中是否有违反安全策略的行为和被攻击的迹象，一旦发现不良情况就及时报告或发出警报。

入侵检测机制具有较大的伸缩性，其检测范围小到单台设备，大到一个大型网络。

（2）渗透测试。

渗透测试是一种通过模拟黑客攻击的技术和方法，挫败目标系统的安全控制措施并最终取得访问控制权限的安全测试方式。渗透测试的意义在于通过安全测试发现应用存在的未知问题，解决问题，并提高开发者的安全意识。

（3）运维检测。

运维检测是在系统运行期间，不断发现安全问题并解决问题，优化安全策略，完善防护、检测和恢复安全机制。通过运维检测可以找到系统中存在的安全漏洞，并且评估目标系统和网络环境是否存在可能被攻击者利用的漏洞，以及由此引起的风险大小，为进一步完善制定相应的安全措施与解决方案提供依据。

（4）应急响应。

随着信息化社会的加速发展，网络和信息系统已经成为重要的基础设施，各种潜在的网络信息危险因素也与日俱增，应用安全问题越来越复杂，安全威胁问题正在飞速增多。尤其是混合威胁的风险，如黑客攻击、蠕虫病毒和勒索病毒等。因此，完善的网络安全体系在保护体系之外必须建立相应的应急响应体系。

5. 安全管理

一般意义上的安全管理是把一个组织的资产标识出来，并制定、说明和实施保护这些资产的策略和流程，其中，资产包括人员、建筑物、机器、系统和信息资产。安全管理的目的是使一个组织的资产得到保护。由资产的范围可知，安全管理涵盖了对系统和信息的保护。

安全管理的一项重要内容是安全风险管理，是把风险管理原则应用到安全威胁管理之中，主要包括标识威胁、评估现有威胁控制措施的有效性、确定风险的后果、基于可能性和影响的评级确定风险等级，划分风险类型并选择适当的风险策略或风险响应。

系统安全领域的安全管理是一般性安全管理的一个子域，聚焦网络系统的日常管理问题，将安全理念贯彻到系统管理工作中，帮助系统管理人员明确和落实系统管理工作中的安全责任，从系统管理员的角度提升系统的安全性。

5.2 操作系统安全

操作系统是连接计算机硬件与上层应用及用户的桥梁。它不仅管理着系统的核心资源，如进程调度、内存分配释放、磁盘 I/O(Input/Output)处理等，同时还向上层应用提供必需的系统资源及各种服务。目前针对操作系统的攻击手段越来越多，越来越复杂。例如，利用操作系统自身的漏洞进行恶意破坏，导致资源配置被篡改，恶意程序被植入执行；利用缓冲区溢出攻击非法接管超级权限等。又如，著名的莫里斯蠕虫(Morris Worm)病毒仅用 99 行代码就感染了 6000 余台 UNIX 系统主机。该病毒就是利用操作系统的缓冲区栈溢出漏洞而实现的。

操作系统安全一直备受关注，目前已有数以万计的操作系统漏洞被发现或被黑客利用。因此，操作系统的安全性是整个计算机系统乃至网络空间的安全基础。操作系统不够安全，就不可能真正解决网络安全和其他应用软件的安全问题。

操作系统安全是操作系统对计算机系统的硬件和软件进行有效的控制，能够为所管理的资源提供相应的安全保护，如存储保护、运行保护、标识与鉴别、安全审计等。

5.2.1 操作系统安全威胁

操作系统安全威胁是对于一定的输入，经过系统处理，产生了危害系统安全的输出。随着外界环境复杂程度的增加和与外界交互程度的提高，操作系统的安全性显得越来越重要，安全问题也日益突出。

1. 操作系统面临的安全威胁

通过操作系统，人们可以更方便地使用计算机，而不必考虑计算机底层各种不同硬件产品之间的差异，大大提高了工作效率。操作系统实现了对计算机硬件的抽象和对计算机各种资源的统一管理，对用户和开发人员隐藏了硬件操作的细节，使用户能更方便地使用计算机，使开发人员不必关心各种硬件设备的实现细节和差异，只需要关心操作系统提供的接口功能即可。但正因为操作系统的这些强大功能，也使操作系统遭受着各种各样的安全威胁，大多威胁是通过利用操作系统和应用服务程序的弱点或缺陷实现的。

按照形成安全威胁的途径划分，操作系统安全威胁可以分为如下 6 种类型。

(1) 不合理的授权机制。

为了完成某项任务依照最小特权原则分配给用户必要的权限,如果分配了不必要的过多权限,那么额外的权限就可能会被用来进行一些意外操作,对系统造成危害,即授权机制违反了最小特权原则。有时授权机制还要符合责任分离原则,将安全相关的权限分散到数个用户,避免集中在一个人手中造成权力的滥用。

(2) 不恰当的代码执行使缓冲区溢出。

所谓缓冲区溢出,是向固定长度的缓冲区写入超出预先分配长度的内容,造成缓冲区数据溢出,从而覆盖了缓冲区相邻的内存空间。例如,在 C 语言实现的系统中普遍存在的缓冲区溢出问题,以及移动代码的安全性问题等。缓冲区溢出攻击是以某种方式破坏进程的内存空间,进而控制程序执行流程。缓冲区溢出攻击的最终目标是从漏洞中收集任何有用的信息以控制进程的执行。

(3) 不恰当的主体行为控制。

如对动态创建、删除、挂起、恢复主体的行为控制不够恰当。

(4) 不安全的进程间通信。

进程间通信的安全对于基于消息传递的微内核系统十分重要,因为系统中很多系统服务都是采用进程形式提供的。系统进程需要处理大量外部正当的或恶意的请求。对于共享内存的进程间通信,还存在数据存储的安全问题。

(5) 网络协议的安全漏洞。

在目前网络大规模普及的情况下,很多攻击性的安全威胁都是通过网络协议自身固有的安全缺陷在线入侵造成的。

(6) 服务的不当配置。

对于一个已经实现的安全操作系统来说,能够在多大程度上发挥其安全设施的作用,取决于系统的安全配置。按照安全威胁的表现形式划分,操作系统面临的安全威胁分为 5 种:计算机病毒、逻辑炸弹、木马、后门和隐秘通道。

按照安全威胁的行为方式划分,操作系统面临的安全威胁通常为 4 种:切断、截取、篡改和伪造。

2. 操作系统安全威胁案例分析

黑客攻击操作系统主要有两个目的:一是窃取用户的私密数据;二是对操作系统进行恶意破坏,使其无法正常运行。黑客常用的攻击手段就是利用缓冲区溢出漏洞植入恶意程序,并非法取得系统的超级权限,进而窃取用户的私密数据或者对操作系统进行恶意破坏。

1) 操作系统安全威胁模型

假设黑客位于操作系统外部,仅能通过正常输入/输出的方式与操作系统下的受害进程进行交互,因此,黑客需要以合法用户的身份通过 I/O 子系统将其构造的恶意代码输入系统中,并接受系统的合法性校验。经过校验的数据将在操作系统内存中暂存,进而被受害进程处理。该模型可以这样进一步理解,假设受害系统是一个 Web 服务器,攻击者通过构造并发送恶意数据包的方式,将恶意信息通过操作系统内核的网络协议栈传送,然后等待进程通过套接字读取内存缓冲区中的恶意数据,最后在 Web 服务进程执行时产生攻击者所期望的恶意行为。

基于此类威胁模型,操作系统外部攻击者的根本目标是使操作系统环境下运行的进程

产生偏离正常行为的异常或恶意行为。攻击者的攻击目标可能是如下几种情况。

(1) 进程直接崩溃。

例如,攻击者希望使系统下的超文本传输协议(HTTP,Hypertext Transfer Protocol)服务进程崩溃,使其无法对合法用户提供 Web 服务。同理也可以使内核崩溃造成宕机。

(2) 任意内存位置读。

攻击者读取操作系统环境下受害进程的任意内存位置时,泄露了操作系统用户的机密信息。例如,窃取 SSH(Secure Shell)服务进程内存中用户的私钥等。

(3) 任意内存位置写。

攻击者对受害进程或内核的任意内存位置写入既定的数据,进而改变进程行为,甚至直接控制目标系统。例如,缓冲区堆栈溢出攻击就是利用任意内存位置写操作实现的。

(4) 权限提升。

攻击者获得正常服务之外的一系列权限,变化执行任意代码的权限,或者获得系统管理员账号的权限。

2) 缓冲区溢出攻击分析

操作系统安全威胁的一个典型案例是缓冲区溢出攻击。缓冲区溢出是一种非常普遍、非常危险的漏洞,普遍存在各种操作系统、应用软件中。缓冲区溢出攻击是利用缓冲区溢出漏洞进行的攻击。利用缓冲区溢出攻击可以执行非授权指令,甚至可以取得系统特权,进而进行各种非法操作,导致程序出现运行失败、系统关机、重新启动等后果。

在缓冲区溢出中,比较危险的是堆栈溢出,入侵者可以利用堆栈溢出,在函数返回时改变返回程序的地址,让其跳转到任意地址,其危害是使程序崩溃导致拒绝服务,或者跳转并且执行一段恶意代码。

缓冲区溢出攻击的目的在于扰乱具有某些特权运行的程序功能,取得程序的控制权。如果该程序具有足够的权限,那么整个主机就被控制了。一般而言,攻击者攻击 root 程序,然后执行类似“exec(sh)”的执行代码来获得 root 权限的 shell。为了达到目的,攻击者必须实现两个目标:一是在程序的地址空间里安排适当的代码;二是通过适当的初始化寄存器和内存,让程序跳转到攻击者安排的地址空间。

在程序的地址空间安排适当代码的方法有以下两种。

(1) 植入法。

攻击者向被攻击的程序输入一个字符串,用被攻击程序的缓冲区存放攻击代码。缓冲区可以设在堆栈(stack)、堆(heap)等地方。

(2) 利用已经存在的代码。

例如,攻击代码要求执行“exec(bin/sh)”,而在 libc 库中的代码执行“exec(arg)”,其中,arg 是一个指向一个字符串的指针参数,攻击者只要把传入的参数指针改为指向“/bin/sh”即可。

控制程序跳转就是改变程序的执行流程,使之跳转到攻击代码。最基本的方法是溢出一个没有边界检查或者其他弱点的缓冲区,来扰乱程序的正常执行顺序。实际中,许多缓冲区溢出是用暴力方法改变程序指针的,例如,利用活动纪录实现堆栈溢出攻击;在函数指针附近寻找一个能够溢出的缓冲区,然后溢出这个缓冲区来改变函数指针;长跳转缓冲区。

简单且常见的缓冲区溢出攻击在一个字符串中综合了代码植入和活动记录技术。攻击

者定位一个可供溢出的自动变量，然后向程序传递一个很大的字符串，再引发缓冲区溢出，改变活动记录的同时植入恶意代码。

由操作系统安全威胁案例分析可知，利用操作系统漏洞实施攻击是多个步骤顺序进行的，大致过程包括：定位潜在的受害系统；构造、输入恶意代码以利用操作系统漏洞；借助漏洞实施攻击，如窃取用户个人信息，或以受害主机为跳板侵害其他主机。

3. 操作系统安全威胁的起因

操作系统安全问题之所以层出不穷，类型繁多，原因在于操作系统是一个规模庞大的软件系统，而且是计算机系统的重要组成部分。计算机系统包括计算机硬件系统和软件系统。计算机硬件系统由控制器、运算器、存储器、输入设备和输出设备 5 部分组成，但软件系统却发生了翻天覆地的变化。一个软件系统不仅仅由众多子系统组成，而且各系统模块之间相互依赖、关系复杂。值得注意的是，现代操作系统主要功能的设计与实现，例如，对计算机硬件的抽象、为用户提供的操作接口、对计算机系统各种资源的管理（处理器管理、内存管理和文件管理）等都是以实现功能并确保性能最优为目标，再考虑安全性问题。当出现安全威胁后通常采用打补丁的方法予以补救。

纵观操作系统安全威胁的一些典型案例可知，对操作系统最基本的攻击方法一般是以内存为直接对象，利用操作系统内存管理机制设计上的漏洞而实现的，而且针对内存堆区和栈区的攻击案例特别多。对于栈区溢出攻击，主要是利用不安全的输入覆盖栈帧中的返回地址来实现进程控制流劫持的。对于堆区溢出攻击，主要是通过恶意篡改堆区管理数据结构造成程序崩溃、堆块覆盖、甚至任意位置内存读写的。例如，利用释放后重用（UAF，Use After Free）漏洞、Double-Free 漏洞、堆区越界读和堆喷等实现的一些堆区攻击。现代操作系统已应用了很多相应的防御方案。

5.2.2　操作系统安全经典模型

技术高明的攻击者大都对系统的实现机理非常清楚，作为防御者或者开发者也应该如此，即便是一名普通的用户，也应该有所了解。讨论系统安全需要从安全机制的角度进行，但由于安全机制依赖安全模型，因此需要先介绍相关的安全模型。操作系统安全的经典模型主要以贝尔-拉普拉（BLP，Bell-LaPadula）模型、毕巴（Biba）模型及克拉克-威尔逊（Clark-Wilson）模型为代表。

1. 贝尔-拉普拉模型

贝尔-拉普拉模型（BLP 模型）是 Bell 和 LaPadula 于 20 世纪 70 年代提出的防止信息泄露的一个安全系统的数学模型。BLP 模型主要解决的是信息的保密性问题，是一种多级安全模型，其核心思想是在自主访问控制上增加强制访问控制，以实施相应的安全策略。

BLP 模型依据系统的用户（主体）和数据（客体）的敏感性建立访问控制方法。一般主体地位与客体敏感性统一用安全级别来描述，对主体和客体做出相应的安全标记，给每个主体和客体都赋予一定的安全等级，因此也被称为多级安全系统。主体的安全等级称为安全许可（Security Clearance），客体的安全等级称为安全等级（Security Classification）。BLP 模型是以自动机理论作为形式基础的安全模型，是一个状态机，定义的系统包含一个初始状态 Z_0 和由（Req，Dec，Sta）形式的三元组组成的序列。其中，Req 表示请求，Dec 表示判定，Sta 表示状态。三元组序列中相邻状态之间满足某种关系 W。如果一个系统的初始状态是安

全的，且三元组序列中的所有状态都是安全的，则该系统就是一个安全系统。

BLP 模型定义的状态用(b,$\boldsymbol{M}$,L,H)四元组定义。其中，$\boldsymbol{M}$ 是访问控制矩阵；L 是安全级别函数，用于确定任意主体与客体的安全级别；H 是客体间的层次关系，典型情况是客体在文件系统中的树状结构关系；b 是当前访问的集合，当前访问是当前状态下允许的访问，由三元组(Sub，Obj，Acc)表示，其中，Sub 表示主体，Obj 表示客体，Acc 表示访问方式。BLP 模型将主体对客体的访问方式 Acc 划分为读(r)、读写(w)、只写(a)、执行(e)4 种。当主体和客体位于不同的安全等级时，主体对客体的访问就必须按照一定的访问规则进行。主体和客体安全等级分别记为 $L(s)$ 和 $L(o)$，则 BLP 模型的 2 个特征可表示如下。

(1) 不上读(NRU，No Reads Up)规则。主体 Sub 能写客体 Obj：

当且仅当 $L(o)\leqslant L(s)$ 并且主体对客体具有自主访问控制读权限时，才允许主体读取客体内容。

(2) 不下写(NWD，No Writes Down)规则。主体 Sub 能写客体 Obj：

当且仅当 $L(s)\leqslant L(o)$ 并且主体对客体具有自主访问控制写权限时，才允许主体向客体写入内容。

在 BLP 模型中，用一个自主访问矩阵 $\boldsymbol{M}$ 实施自主访问控制，主体 Sub 只能按照访问矩阵允许的权限对客体 Obj 进行相应的访问。每个客体 Obj 还有一个拥有者(Owner，属主，一般是客体的创建者)。拥有者是唯一有权修改客体访问控制表的主体，拥有者对其客体具有全部控制权。如表 5-1 所示，User2 对 Object2 具有读写权限，在其访问控制表的交叉单元格中，即表明了其访问权限为 w。User1 为 Object3 的拥有者，拥有所有权限。

表 5-1　主体对客体的读写

	Object1	Object2	Object3
User1	w	e	owner
User2	r	w	r
User3	-	r	w

BLP 的核心内容由简单安全特性(ss-特性)、星号安全特性(*-特性)、自主安全特性(ds-特性)和一个基本安全定理构成。

简单安全特性与星号安全特性构成强制访问安全策略。其中，星号安全特性又可分为自由星号特性和严格星号特性。这些访问规则可采用如下公式表示，其中 λ 表示主体或客体的安全标签。

简单安全特性：主体能读客体，一定有 $\lambda(s)\geqslant\lambda(o)$。

自由星号特性：主体能写客体，一定有 $\lambda(s)\leqslant\lambda(o)$。

严格星号特性：主体能写客体，一定有 $\lambda(s)=\lambda(o)$。

读操作时，信息从客体流向主体，因此需要 $\lambda(s)\geqslant\lambda(o)$，等价于 $\lambda(o)\rightarrow\lambda(s)$。相反，写操作时，信息从主体流向客体，因此需要 $\lambda(s)\leqslant\lambda(o)$，等价于 $\lambda(s)\rightarrow\lambda(o)$。强制访问控制中的条件是“必须有”，表明该条件是必要条件，也可以增加其他的必要条件的控制，例如，要求访问必须同时满足自主访问特性。

但是，星号安全特性是以主体的当前安全级别进行访问控制判定的。