第3章

软件安全分析与设计

3.1 软件安全需求分析

3.1.1 安全需求分析的方法

用户对软件的需求是开发软件产品的依据。需求分析的基本任务是获取并表达目标系统必须实现的功能、性能和安全性等要求，确定系统必须“做什么”。软件需求分析主要包括以下任务。

(1) 确定对系统的综合要求。包括功能需求、安全性需求、性能需求(例如响应时间和存储容量等)、可靠性和可用性需求、出错处理需求、接口需求(例如系统与环境的通信格式)、逆向需求(主要描述容易产生误解的需求，描述“系统不应该做什么”)，以及设计或实现系统时应遵守的限制条件等。

(2) 确定对系统的数据要求。可利用图形工具更直观地表示数据结构。

(3) 导出系统的逻辑模型。例如数据流图、实体-联系图和状态转换图等。

(4) 修正系统开发计划。根据在分析过程中所获得的对系统更深入的了解，可修订系统的成本和进度，修正以前制订的开发计划。

软件安全需求分析是从攻击者的角度分析系统的安全漏洞，定义了系统“不应当”以某种方式处理某功能。软件的安全性需求是软件需求的重要组成部分，与业务功能需求处于同一需求水平，并对业务功能需求具有约束力，可保证需求的完整性和一致性。软件安全开发应当从需求分析阶段开始就“自下而上”地考虑安全问题。一个缺少安全性需求分析的软件开发项目难以保证系统信息的保密性、完整性和可用性。下面介绍软件安全需求分析的主要任务。

1. 确定安全要求

开发安全可信的软件系统必须考虑软件的安全和隐私问题。在软件项目的初始计划阶段就应当确定软件的信任度要求，从而帮助开发团队确定关键里程碑和预期交付成果，并使集成安全和隐私的过程不影响业务功能的开发安排。软件项目初期的安全需求分析工作包括：为在预期环境中运行的应用程序确定最低的安全和隐私要求，确立并部署软件安全漏洞的跟踪系统。

在此期间主要进行系统级的安全分析工作，其分析结果有助于决定系统体

系架构的可行性。安全需求分析帮助开发团队确定系统的关键安全要求，从而帮助团队选择与体系架构相关的安全策略，包括选用哪种安全架构、由系统内部哪个部件完成哪个安全需求项、选用硬件还是软件才能实现高性价比的安全、选用哪个操作系统、选择哪种额外的安全验证机制、采用新技术有什么安全风险等。

软件需求分析人员需要构建尽可能完整和准确的安全需求规格说明，并在整个安全开发过程中帮助开发人员理解安全需求，以减少后期维护带来的巨额花销。需求分析人员还需要与用户密切沟通，以避免双方在交流信息的过程中出现误解或遗漏，以及必须严格审查、验证需求分析的结果。需求分析人员首先确定目标系统的业务运行环境、规则和技术环境，然后根据业务特点设置不同安全级别的基础设施。为医疗云、政务云和金融云等私有云环境提供不同安全级别的深度防御措施（例如防火墙、反病毒软件和应用监控等），以增加攻击者的攻击成本。需求分析人员也可根据业务选用不同的技术框架，其目标是用最小代价实现必要的安全控制，确保技术成本和安全收益的平衡。例如，若应用系统的用户是各门店营业员，则通过专线技术可避免服务对互联网开放，使用专门的定制设备（例如柜员机等）可避免不安全的使用环境。

在项目开始初期，要求建立正式的书面安全计划，确定各项工作的责任方和配合方，根据团队中各方的能力水平，制定合理的安全措施，并为各部门的安全分析会议建立沟通机制，以便于各方交流工作的内容和进程。对于一个软件开发厂商来说，需要指定一个人或者一个团队来领导或负责软件安全开发过程，其工作内容包括：领导安全需求分析小组总结安全标准的技术要求、开发团队以往进行安全开发的成功经验和失败教训，从而建立一个安全需求分析的经验库；在新软件的项目实施中，将经验库中的安全需求分析经验应用于新软件产品的架构和软硬件等需求分析；通过把分散的安全需求集中管理，建立起开发团队对于安全分析的统一认识，逐步升级为更高层次的企业安全开发标准；推动团队不同小组间的沟通，及时更新安全与隐私策略，定期以电子邮件等方式与开发小组沟通在安全和隐私方面的软件缺陷问题。可见，在项目开发初期阶段进行的安全需求介入，涉及安全处理策略和人员沟通问题，可将与软件安全相关的要求和约束加入软件开发过程。

2. 确定质量门（缺陷等级）

质量门和缺陷等级标准用于确立安全和隐私质量的最低可接受程度。在项目开发初期定义这些标准可加强对安全风险的理解，有助于团队在开发过程中发现和修复安全缺陷。项目团队必须协商确定每个开发阶段的质量门（例如，在 check in 代码之前必须会审并修复所有编译器警告缺陷），随后将质量门交由安全顾问审批。安全顾问可根据需要添加特定于项目的说明以及更严格的安全要求。另外，项目团队须阐明其对安全门的遵从性，以便完成最终的安全评析。为了便于缺陷的定位、跟踪和修改，需要将所发现的缺陷按照严重程度或优先级别等进行划分。

缺陷的严重程度是指软件缺陷对软件质量的破坏程度，即此缺陷的存在将对软件的功能和性能产生怎样的影响。应该从软件最终用户的视角对缺陷的严重程度做出判断，关注对用户使用所造成后果的严重性。缺陷的严重程度可从高到低划分如下。

（1）致命。系统不能执行正常功能或重要功能，甚至危及人身安全。

（2）严重。严重地影响系统基本功能和要求的实现，产生系统崩溃或资源严重不足，且没有办法更正。

（3）较重。严重地影响系统基本功能和要求的实现，但存在合理的更正办法（重新安装或重新启动该软件不属于更正办法）；操作界面缺陷；打印内容或格式缺陷。

（4）一般。操作界面不规范；辅助说明描述不清楚；长时间操作但不给用户提示，类似死机；提示窗口文字未采用行业术语；可输入区域和只读区域没有明显的区分标志；操作者感到不方便，但不影响执行工作功能。

（5）轻微。其他缺陷或者建议修正的缺陷。

缺陷的优先级是表示处理软件缺陷的先后顺序的指标，即哪些缺陷需要优先修正，哪些缺陷可以稍后修正。缺陷优先级别是给管理者做决策使用的，因为缺陷的修正顺序是个复杂的过程，不仅仅涉及纯粹的技术问题。项目经理通常依据缺陷优先级来安排开发任务的顺序，以降低项目的风险和成本。缺陷的优先级可从高到低划分如下。

（1）紧急。缺陷必须被立即解决。

（2）正常。缺陷需要正常排队以等待修复或列入软件发布清单。

（3）不急。缺陷可以在方便时被纠正。

3. 评估安全和隐私风险

安全风险评估用于判断系统中易受攻击部分的风险级别。安全评估主要是对安全需求分析、设计和实现结果的正确性、完整性和安全性进行评估，对需求分析的评估主要以文档评审和人工分析为主，对设计可从软件和硬件两方面进行评估，对于实现的评估可以采用渗透测试等方法。安全评估方可以是不属于项目团队且双方认可的小组，负责进行安全检测、设立风险评估调查问卷，收集开发人员在安全开发过程中遇到的安全漏洞并加以解决。

隐私影响评级，包括 P1、P2 和 P3 这三个级别，是从隐私的角度评估软件待处理数据的敏感性。在开发一个高隐私风险软件之前必须进行隐私影响评级，以帮助用户根据隐私风险和成本等因素来判断产品是否具有商业价值。以下介绍隐私影响评级的这三个级别的含义。

（1）P1（高隐私风险）。产品、服务、特征或者错误报告等通过实时传输匿名数据来监视用户，更改设置或文件类型关联，或者安装软件。

（2）P2（中隐私风险）。由用户发起的一次性行为影响产品、服务、特征中的隐私信息；匿名数据传输。

（3）P3（低隐私风险）。特征、产品或服务中没有涉及隐私信息的行为，没有匿名或个人数据传输，没有代替用户更改设置，且没有安装软件。

3.1.2 攻击用例

一般的软件需求分析，以用户在理想环境下正确使用软件为前提描述系统的行为，这就意味着对系统的功能理解是建立在系统不会被有意滥用的假设之上的。这种需求分析无法获得与软件安全性相关的需求。软件安全需求分析可采用基于攻击模式的分析方

法。攻击模式所针对的问题即为软件攻击者的目标对象，所描述的是攻击者用于破坏软件产品的技术。安全需求分析人员应当以攻击者的角度看待系统，进行威胁分析，从而帮助修复安全漏洞、提高软件的安全性。

用例建模对软件或系统的预期行为进行描述，是软件需求分析的常用技术。用例描述的是用户预期的行为，包括为满足业务需求而采取的操作序列和事件序列。通过用例建模可以明确描述特定行为所发生的时间和条件，能有效地减少不明确的业务需求。但是，由于用例收集困难且运行用例的具体情境受限，用例建模通常只能针对最重要的（而不是所有的）系统行为进行建模，因此用例模型不能成为需求规范文档的替代品。用例建模包括确定参与者（Actor，访问主体）、预期的系统行为（Use Case，用例）、参与者和用例之间的关系，以及（行为或事件的）执行序列。其中参与者可能是一个角色或非人类物，例如一个人、管理员或后台批处理过程。

在正常用例的基础上可以开发误用用例（MisuseCase/AbuseCase，滥用用例）。误用用例通过对负面场景的建模来帮助识别安全需求，其中负面场景是指系统的非预期行为，用户不希望在正常用例的场景中发生这种行为。误用用例用于描述系统或软件可能受到的安全威胁，所提供的是恶意用户的使用视角。误用用例的建模过程与正常用例的类似，主要区别在于其建模的内容是参与者与系统的非预期行为。误用用例描述的是人为故意的或意外出现的场景，用于获取和描述安全需求而不是业务功能需求。

安全需求分析人员创建误用用例的手段包括头脑风暴、阻止正常用例场景中的部分操作等。下面的例子对某在线电子商务商店进行用例建模：根据系统对用户行为的预期，用户必须先创建一个账户并登录，然后才能在电子商务商店下订单。在用户使用产品搜索功能时，系统不需要对用户进行身份验证。但是在用户使用下订单功能时，系统必须要求用户登录并对用户身份进行验证。系统在监测用户登录并验证用户身份的过程中，可确定用户用例的范围并记录用例对系统所做的任何假设。安全需求分析人员应当创建对应攻击场景的误用用例，例如攻击者可能盗用合法客户的用户名或暴力破解用户密码，也可能利用窃取的客户信用卡信息下订单。安全需求分析人员在创建这类误用用例时，不仅要考虑来自系统外部的攻击，还要考虑来自系统内部的攻击，例如潜在的内部攻击者包括能直接访问未受保护敏感数据的数据库管理员。安全需求分析人员在处理内部攻击的误用用例时，应当使用安全审计等安全控制机制，以降低来自内部误用的威胁。

误用用例描述了系统可能会遭受哪些攻击以及如何被攻击。误用用例可帮助分析者深入理解系统的假设以及攻击者如何利用和破坏这些假设，也能帮助团队进行系统体系结构的安全分析和代码的安全性测试。安全需求分析人员在构建误用用例时，应当与功能需求分析人员一起组成安全需求小组，从一组需求、一组标准用例和一些攻击模式入手，用文档记录存在的安全威胁。安全需求分析通常包括以下步骤。

（1）确定正向的安全需求。定义软件应该完成的安全功能和可接受的行为。

（2）创建反向的安全需求。定义软件不满足安全需求时的后果。

（3）考虑所有能获得系统访问权的用户。

（4）构建误用用例。以反向安全需求为基础，描述恶意用户和攻击者等可能如何滥用系统，并确定在某安全功能缺失或失效时会导致的后果；攻击模式可以为构建误用用例

提供指导。

3.2 软件安全设计

软件设计阶段的任务是依据软件需求规格设计软件结构和数据结构。据统计，软件产品中的很多安全问题都是来自软件安全性设计的缺陷，即在设计时对安全性考虑不足；随着软件开发进度的推进，安全设计缺陷的修复成本不断增加。因此，必须在软件设计阶段就重视软件的安全性问题，以降低软件开发成本、提高软件产品的安全性。软件设计所依据的软件需求规格通常定义了软件的功能和特征、数据形式以及与外部的接口和交互方式，此外还应包括安全需求的内容，如安全规范和安全风险评估等。

3.2.1 安全设计的内容与原则

软件安全设计以软件的安全性和功能等需求为依据，提出符合安全性要求的软件设计规格。软件安全设计可分为软件安全架构设计和软件安全详细设计两个阶段。在软件安全架构设计（又称总体设计、概要设计）阶段，首先根据软件的功能和安全性需求构建软件架构，然后对架构进行安全性分析和完善。在软件安全详细设计阶段，对软件功能模块和数据结构进行详细设计并考虑安全性问题。

软件安全架构设计通常包括以下两个步骤。

（1）软件架构师根据软件需求进行初步的软件架构设计，包括划分软件的功能模块、确定每个模块的功能和接口、建立模块之间的交互关系以及建立数据模型等。

（2）软件安全分析师在架构师的帮助下对软件架构进行安全建模，然后检查软件架构是否满足安全需求或安全标准。如果发现不满足，安全分析师将根据分析结果，指导架构师对软件架构进行修改，并在得到修改后的架构设计方案之后重复步骤（2），直至设计的软件架构符合安全需求和标准。

安全专家从大量软件设计的实践中总结出了很多安全设计的经验，但尚未形成公认的标准。下面介绍软件安全设计的一些主要原则。

1. 最小化攻击面原则

软件的攻击面（Attack Surface）是指用户、潜在攻击者和其他程序所能访问的所有软件功能与代码的总和。一个软件的攻击面越大，其安全风险就越大。最小化攻击面就是要关闭无须对外开放的软件端口，从而防范一些潜在的安全问题。在默认情况下，软件应该关闭那些不常用的功能；分离软件的内核代码与用户任务处理代码，分离软件的功能代码与界面代码，目的是将开放给用户的功能代码降到最少，减少攻击者可利用的漏洞。

为实现最小化攻击面，安全人员需要对软件进行攻击面分析，包括分析所有的库访问、接口、界面以及可执行代码等。攻击面分析旨在发现各种攻击面实例，并尽量降低其攻击面，将高攻击面实例转化为低攻击面实例。表 3-1 列出了几种常见的高攻击面实例及其对应的低攻击面实例。最小化攻击面的措施通常包括减少可默认执行的代码量、限制可访问代码的人员范围、限定可访问代码的人员身份，以及降低执行代码所需的权限。

例如，微软公司在系统软件和应用软件的设计过程中，采用了如表 3-2 所示的最小化攻击面的多种具体措施。

表 3-1　高攻击面和低攻击面实例对比

高 攻 击 面	低 攻 击 面
权限默认打开	权限默认关闭
打开 socket	关闭 socket
UDP(网络协议)	TCP
匿名访问	认证用户访问
时不时打开端口	只在需要时打开端口
可以访问因特网	只能访问本地子网

表 3-2　微软公司最小化软件攻击面的措施示例

软 件 产 品	最小化攻击面的措施
Windows	RPC 需要认证，默认打开防火墙
IIS 6.0/7.0	默认关闭 NETWORK SERVICE 权限
SQL Server 2005/2008	默认关闭 Xp_cmdshell 存储过程，默认不开放远程链接
VS 2005/2008	默认仅本地访问 Web Server

2. 最小授权原则

最小授权原则是指只授予每个用户或程序在执行操作时所必需的最小权限。普通用户在没有管理员权限时难以执行攻击操作，而一个普通管理员在没有 root 权限时通常也不能对系统造成毁灭性的破坏。因此遵从最小授权原则，可防范使用者利用过高的权限进行非法操作，限制事故、误操作或攻击带来的危害。

应用最小授权原则的常见措施包括：只为程序中需要特权的部分授予特权，只授予必需的那部分权限，将特权的有效时间限制至最短。例如，微软公司在其系统软件和应用软件的设计中曾采用如下的最小授权方案。

(1) 对 Windows 系统软件，将网络进程、本地服务和用户进程分别设在不同的权限组，即 NETWORK SERVICE 权限组、LOCAL SERVICE 权限组和 USER 权限组，只有核心进程组才能使用系统(SYSTEM)权限。

(2) 对新版本的 Office 应用软件，在打开不可信来源文档时，将文档默认设置为不可编辑，同时默认不可执行代码。因此，即使 Office 软件被发现有缓冲区溢出漏洞，攻击者也无法利用漏洞来执行 shellcode 等恶意代码。

3. 权限分离原则

权限分离原则是指在分配权限时，不能将所有权限全部分配给单个用户使之能独立

操纵系统，而应该将权限分配给多个不同的用户以协同操作系统。如果使用一个 root 账号使之能在所有系统通用，虽然便于系统维护，但是当攻击者攻破其中任意一个系统就可以拿到最高权限从而操纵所有系统，具有很高的危险性。实现权限分离的常见措施包括：禁止 root 用户远程登录；为不同身份的管理员分配不同的权限；除高级管理员之外的普通管理员不能单独拥有所有权限；对于一些敏感操作，需要至少两个不同权限的管理员来协同执行等。

4. 纵深防御原则

纵深防御(Defense-in-Depth)原则是指在软件设计中采用多重安全机制(而不是依赖于单一安全机制)的防御技术。单一防御技术即使再强也无法保证 100%安全，而且当单一防御方法失效时，整个系统就会处于无防御状态，极易被攻击。在软件业务功能的设计中，可采用如下的多重防御手段：对于表单中的字段，不仅在页面进行校验，而且在后台也有专门的校验机制；在代码设计中，不仅要考虑代码功能实现的安全性，而且要加入具有安全防护功能的代码，以防范内存溢出、代码注入和跨站攻击等；在信任区、非信任区之间二次部署防火墙等。

下面介绍在系统设计中常用的两种纵深防御措施。

1) 边界防御

(1) 防火墙(Firewall)。借助硬件或软件，在内部和外部网络环境之间建立的一种保护屏障，用于阻断威胁计算机安全的因素。防火墙会实时拦截不安全的访问请求，只有通过防火墙检测的访问请求才能进入系统的计算机内。

(2) 入侵检测系统(Intrusion Detection System，IDS)。通常在已受到防火墙保护的网络中使用，入侵检测系统不是阻止攻击，而是监控系统并标识任何看起来可疑或非法的行为、状态和事件等。

2) 监控

监控(Monitor)为系统中发生的任何操作和事件生成日志，并对日志进行记录和分析，以尽可能地识别攻击事件并做出快速反应。监控日志不仅可以帮助识别攻击行为，而且能在攻击发生之后，帮助分析攻击的细节和过程，进而防范未来的攻击。

5. 默认安全配置原则

默认安全配置是指在客户熟悉软件配置选项之前使用默认的安全配置选项，不仅能帮助客户理解和学习安全配置的用法，而且能保证应用程序的初始配置状态是一个比较安全的状态。熟悉安全配置的用户可根据自己的实际情况，调整应用程序的安全和隐私等级等安全配置选项值。运用默认安全配置原则的一个例子是 Windows 系统：为提升默认配置安全，Windows 系统在 Windows 7 版本之后就默认开启 DEP(数据执行保护)安全选项，在 Windows 10 版本默认启用安全防护软件 Windows Defender。下面则是违背默认安全配置原则的例子：早期的一些路由器厂商将其路由器配置软件的账号和密码都默认设置为 admin。如果路由器用户未修改此默认配置，则攻击者使用 admin 这个默认账号和密码，通过 WiFi 就能轻易窃取用户的上网信息，造成安全隐患。

在软件设计中，遵循默认安全配置原则常采用如下措施：默认拒绝任何请求；默认关闭不经常使用的功能；默认检查口令的复杂性，当达到最大登录尝试次数后默认锁定用户等。

6. 完全控制原则

完全控制原则是指对于受保护对象的任何访问操作，都要进行授权检查并能标识其操作请求的源头。对于为提高性能而缓存的访问操作，更要进行细粒度的授权检查，以防范攻击者利用缓存绕过系统身份验证、发起攻击。

7. 开放设计原则

开放设计原则是指不将安全机制的设计作为秘密，不将系统安全寄托在保守安全机制设计秘密的基础上。依据此原则，即使将软件的设计甚至源代码开放，软件系统依然是安全的。目前，要完全做到软件的“开放设计”还是比较困难的，但是软件设计者应培养这样的设计理念，从而使得所设计的安全机制能经受更强的攻击、通过更严格的审查。

一个违背“开放设计”原则的典型例子就是使用私有加密算法，即设计者误以为使用自己设计的加密算法更加安全，而正确的做法是使用标准的加密算法（例如 RSA 和 AES 等加密算法）。数据的安全性不应该依赖于算法的保密性，而应依赖于易受保护的特定元素（例如密钥）的保密性：即只要保证密钥位数足够长且密钥不被泄露，就能保证加密的安全性。

8. 保护最弱环节原则

攻击者的资源和时间也是有限的，因此他们更倾向于攻击看起来更弱的（而不是更强的）安全机制。攻击者总是会寻找软件中最薄弱的环节，并试图从这一点突破防御，从而更容易、更快地获得收益，即使不一定是最大收益。打个比方，银行比小路边的便利店持有多得多的现金，但是普通抢劫犯却通常会在二者间选择便利店作为抢劫目标。这是因为便利店比银行的安全机制要薄弱得多，更容易抢劫成功而且更容易逃脱。

在进行软件安全性设计时，要对软件设计方案进行全面的风险分析，找到最容易被利用的那些风险，按严重程度对其进行排序，并按照严重程度从高到低处理风险和分配安全资源。其目的是将有限的时间和精力花费在最薄弱安全环节的修复上，以获得更大的安全收益。值得注意的是，有时候软件技术本身并不是最薄弱的环节，而使用软件的人（包括最终用户、技术支持人员或架构管理员等）是最薄弱的安全环节，因此在进行软件的安全性设计时应当考虑人的环节并采取相应的防范措施。

9. 安全机制的经济性原则

越复杂的系统存在安全风险的可能性也越高，可能带来更多的攻击面和更多的薄弱环节。简洁的软件系统更容易进行维护：检测和修复其漏洞的难度更低、速度更快，维护成本更低。因此，软件安全机制的设计应尽可能简洁以便降低成本，即遵循“安全机制的经济性原则”。

10. 安全机制心理可接受原则

安全机制的设计要尽可能符合用户的使用习惯，不给合法用户带来额外负担，不影响用户对资源的正常访问，即不明显降低软件的可用性。遵从“安全机制心理可接受”原则的安全机制不容易被用户关闭或绕过，从而发挥实际的安全保护效果。

以上十个安全设计原则可能在一个软件系统中不能同时共存，它们之间在某些方面可能存在矛盾，所以在实际的软件设计中，设计者需要对这些安全设计原则进行权衡，使得软件的重要安全需求或总体安全需求优先得到满足。

3.2.2 安全设计的方法与模式

为提高软件的安全性，软件设计人员可根据软件实际的安全需求，基于安全设计原则、灵活采用多种安全设计的方法。下面以 Web 应用为例，介绍一些常用的软件安全设计方法与模式。

1. 软件安全设计方法

1）服务器端验证

软件设计人员应认为用户的一切输入都是不可信任的。对于从 Web 客户端收集到的输入数据，必须在成功验证之后才能被服务器处理。采用服务器端验证代替客户端验证，可以更好地保障软件系统的安全性。

2）分页传输数据

当客户端需要传输大量数据到服务端时，采用一次性传输的方式将消耗系统资源、降低系统性能。因此，可对这些数据进行分页处理，当服务器端需要处理数据时才传输对应的数据页。这种分页传输数据到服务器的方式不仅能提升系统性能，而且可以降低数据被一次性泄露的风险。

3）使用安全协议进行传输

安全的网络传输协议可有效降低数据在传输过程中被泄露的风险，提高实时通信的可靠度。下面介绍三种常用的安全传输协议。

（1）SSL(Secure Socket Layer)协议。对数据传输进行加密认证，弥补了 TCP/IP 的不足，具有实施成本低和安全高效等优点。SSL 协议使用对称密钥加密传输数据，并对该密钥进行非对称加密，最后把加密后的数据和密钥一起发送。SSL 协议主要用于以下场景：认证客户端和服务器，以避免将数据错发给其他接收方；加密数据，以防止数据泄露和窃取；保证数据传输中不被篡改。

（2）HTTPS(HyperText Transfer Protocol over Secure Socket Layer，超文本传输安全协议)。以安全为目标的 HTTP，在 HTTP 的基础上加入 SSL 层，通过传输加密和身份认证来保证传输过程的安全性。

（3）SET(Secure Electronic Transaction，安全电子交易)。一种基于消息流的协议，用于保证电子交易的支付安全，即保证支付信息的机密性、支付过程的完整性、交易双方的合法性和可操作性。SET 协议的实现比较复杂，所涉及的安全技术包括公钥加密、数

字签名、电子信封和电子安全证书等。

在数据传输时使用安全协议进行加密认证，可提高传输的安全性，但是也会使传输的性能下降，因此可以在传输重要数据时才使用安全协议。

4）对会话进行管理

在使用完会话（Session）中的存储对象后，应立即删除并释放（或者在超时后自动释放）这些对象，以防止这些对象被冒用身份以发起攻击。

2. 软件安全设计模式

在各种软件的安全设计过程中，存在一些共性的、反复出现的安全问题。而针对这些共性问题，安全软件工程专家总结出了一些经过验证的、可重用的软件安全设计模式。软件设计人员通过使用这些安全设计模式，可以快速地解决同类的软件安全问题，建立具体有效的安全设计方案。接下来将介绍五种常见的软件安全设计模式，其模式名称、目的和关注点如表 3-3 所示。

表 3-3　五种安全设计模式

安全模式名称	目　　的	关　注　点
认证器模式	验证试图访问系统的用户是否是其所声称的身份	用户或系统鉴别
基于角色的访问控制模式	描述如何基于人的任务分配功能与权限	访问控制
安全的 MVC 模式	利用基于 MVC 模式的系统，增加用户交互的安全性	系统交互
传输层安全 VPN 模式	应用隧道与加密技术建立安全通道，对每个端点进行鉴别与访问控制	安全通信
安全日志与审计模式	对用户行为进行记录，并对日志进行分析	审计

1）认证器模式

对访问系统的所有用户进行身份认证，目的是验证试图访问系统的用户是否是其所声称的身份。所用的认证方式包括一次性口令、动态口令、用户＋口令认证、证书认证等。

2）基于角色的访问控制模式

对重要信息的访问实行强制访问控制，包括控制用户类别和信息类别、限制用户对资源的访问、对未经授权的用户拒绝访问。在系统中，通常设置系统管理员、安全管理员和安全审计员三个角色，其职责划分如下。

（1）系统管理员。负责系统的日常维护。

（2）安全管理员。负责系统的日常安全管理，例如用户账户管理和日志审查等。

（3）安全审计员。对系统管理员、安全管理员的操作日志进行审计分析，及时发现系统中存在的风险威胁。

以上三个角色各司其职，相互独立但又相互制约，合作维护整个系统的安全性。此外，系统中不允许存在具有所有权限的超级管理员。

3）安全的 MVC 模式

MVC 是 Model-View-Controller（模型-视图-控制器）的缩写，这三层分别代表系统的

业务逻辑层、表示层和控制器层。安全的 MVC 框架模式，通过使用切片和过滤器等方式对数据进行全局处理，在不同的层中解决不同的安全威胁，例如在业务逻辑层解决 SQL 注入等与业务逻辑相关的安全问题，在表示层解决与用户界面有关的安全问题。

4）传输层安全 VPN 模式

这种模式利用隧道和加密技术在传输层建立安全 VPN（虚拟专用网），以对每个端点进行鉴别和访问控制，实现安全通信。例如 IPSecVPN，是采用 IPSec 协议来实现远程接入的一种 VPN 技术，在公网上为两个私有网络提供安全通信通道，通过加密通道保证连接的安全。传输层安全 VPN 模式所用的主要安全技术如下。

（1）隧道技术。在公用网建立一条专用通道（即隧道）以传输数据包，类似于点对点连接技术。

（2）加解密技术。在对数据进行加解密时，综合使用对称加密算法（例如 DES、3DES 和 AES 等）和非对称加密算法（例如 RSA 和 DH 等）。通常使用非对称加密算法来加密“对称加密算法所用的密钥”。

5）安全日志与审计模式

软件系统中的安全日志，相当于飞机上的“黑匣子”，可以记录对软件系统的监控历史信息。安全审计数据所记录的是与安全相关的事件，包括对于敏感数据项的访问、对于目标对象的删除、访问权限的授予和撤销、改变主体或目标的安全属性、对标识定义和用户授权认证功能的使用、审计功能的启动和关闭等。安全审计分析类型包括潜在攻击分析、基于模板的异常检测、简单攻击试探和复杂攻击试探等。

3. 软件安全设计流程

软件安全设计人员在基于安全模式进行安全设计时，通常遵循一定的软件安全设计流程，主要包括以下三个阶段。

1）风险评估

此阶段包括以下三个步骤。①风险识别，即分析安全需求以确定业务类中的安全关键类和关键功能，基于专家知识库列举系统运行中的风险类型。②风险评估，即评估风险的危险等级并根据等级分配相应任务的开发时间和资源，首先解决危险等级高的风险。③风险描述，即将软件风险与人力和时间等项目资源相关联，形成风险描述文档。

2）安全模式选取

此阶段包括以下三个步骤。①安全模式选取，这是设计流程中最重要的环节，通过比对风险描述文档和安全模式库、根据对应规则选取安全模式。②安全模式评估，即将所选取的安全模式的风险评估为完全解决风险或部分解决风险。对于部分解决风险，还需通过多种模式相结合等方法来继续寻找安全模式；若最终依然无法完全解决风险，则需根据风险程度采取相应措施，包括直接删除该软件功能以移除风险。③系统框架重构，即在前两个步骤的基础上，把各种新功能加入到系统原有的高层设计中，形成最终的安全设计架构。

3）安全模式细化

将第二阶段选定的安全模式进行实例化，即在实例化框架中添加安全模式。根据加入安全模式后的系统架构和业务需求，重构系统的业务类图，生成详细的系统设计类图。

3.2.3　威胁建模

威胁建模是识别、分类和分析软件中潜在威胁的一种形式化方法。在软件安全开发的整个生命周期中都可以进行威胁建模，其作用包括：减少与安全相关的设计缺陷和编码缺陷的数量，降低软件中残留的安全缺陷的严重程度，从而减小软件的安全风险。

根据所关注的对象，威胁建模可分为以下三种类型。①关注资产的威胁建模。资产通常是指有价值的东西，包括攻击者想要的东西和我们想保护的东西，例如公司内部的用户数据。资产是威胁的攻击出发点，理论上以资产为中心是理所当然的，但实际上关注资产的威胁建模并没有想象中有效。②关注攻击者的威胁建模。通过识别潜在的攻击者（组织或个人），可根据攻击者的身份等特征来识别潜在的威胁可能有哪些。③关注软件的威胁建模。在实际中，前两种关注方法的实际建模效果并不理想，而最常用的仍是关注软件本身的威胁建模方法。

通常，威胁建模方式按照其应用阶段可划分为以下两种类型。①主动式建模，也称防御式建模，常用于软件开发早期阶段，特别是软件规格建立和软件设计阶段；其缺点在于，在早期阶段难以预测所有的威胁。②被动式建模，也称对抗式建模，常用在产品被创建和部署之后，有助于发现产品中需解决的安全缺陷；其威胁建模技术涉及模拟黑客攻击、渗透测试、代码审查和模糊测试等；其缺点是需要在部署后对软件产品进行更新或打补丁，可能会牺牲用户的使用体验和友好性，且未必比主动式建模方法更有效。在实际的软件开发中，通常使用主动式建模在设计阶段预测出尽可能多的安全威胁，而对于无法预测的威胁，则使用被动式建模在后期阶段解决。

接下来，本节将介绍两种主流的威胁建模方法。

1. STRIDE 威胁建模方法

微软公司提出的 STRIDE 是一种成熟的、得到广泛应用的系统安全威胁建模方法，对以下六种威胁进行建模：Spoofing（假冒）、Tampering（篡改）、Repudiation（抵赖）、Information Disclosure（信息泄露）、Denial of Service（拒绝服务）和 Elevation of Privilege（权限提升）。下面将介绍这六种威胁的含义，其汇总如表 3-4 所示。

（1）Spoofing。伪造身份并对目标系统进行访问，例如伪造用户名、系统名、无线网络名和电子邮件地址等。攻击者通过将自己伪造成合法用户，来绕过系统对未授权访问的防御措施，从而发起攻击。

（2）Tampering。对数据进行未授权的更改操作，破坏系统的完整性和可用性。

（3）Repudiation。攻击者否认自己的攻击行为，可能导致无辜的第三方受到指责。

（4）Information Disclosure。利用系统存在的缺陷，将一些敏感信息（例如客户信息、财务信息和业务操作信息等）泄露给外部的未授权实体。

（5）Denial of Service（DoS）。试图通过减少系统吞吐量和造成延迟等手段，阻止用户正常使用系统资源。大部分的 DoS 攻击是暂时性的，在攻击结束之后系统可通过重启而自动恢复；而对于永久性 DoS 攻击，系统则需要借助系统修复和备份才能恢复。

（6）Elevation of Privilege。通过盗窃高级账户的凭证，将某些用户账号转换为拥有

更高权限的账号，授予其临时或永久的额外权限。

表 3-4 STRIDE 的六种威胁

威胁类型	含义	举例
Spoofing(假冒)	攻击者冒充某人或某物	冒充其他用户或服务账号
Tampering(篡改)	未经授权修改数据	未授权修改存储的信息
Repudiation(抵赖)	否认自己的攻击行为	篡改日志
Information Disclosure(信息泄露)	敏感信息在传输、存储、处理等过程中被未经授权地访问	数据库中用户隐私信息被泄露
Denial of Service(拒绝服务)	无法正常提供服务	高危操作导致系统服务不可用
Elevation of Privilege(权限提升)	拥有了本不该有的权限	普通用户拥有了管理员的权限

以上六种威胁并非各自独立存在，在实际中遇到的威胁有可能同时属于以上多种。因此，以上的威胁分类只是帮助我们发现安全威胁的一个框架，而在使用 STRIDE 方法时并不需要将威胁进行严格划分。目前，微软公司已提出了 STRIDE 的升级版本，即 ASTRIDE(Advanced STRIDE)，在其中增加了新的威胁类型 Privacy(隐私)。

软件安全人员在使用 STRIDE 方法识别出软件的潜在安全威胁之后，需要对这些威胁进行评级，从而决定处理这些威胁的优先级。DREAD，作为 STRIDE 的附件，包括了评价安全威胁级别的以下五个维度，即 Damage Potential(破坏力)、Reproducibility(可重复攻击性)、Exploitability(漏洞利用难度)、Affected Users(影响用户数)和 Discoverability(漏洞隐蔽程度)。这五个评价维度对安全威胁的评级标准和例子，如表 3-5 所示。

表 3-5 DREAD 评分标准与示例

评价维度	威胁等级		
	高级别	中级别	低级别
Damage Potential (破坏力)	严重 (例如，获取管理员权限，非法上传文件)	一般 (例如，泄露敏感信息)	低 (例如，泄露不敏感信息)
Reproducibility (可重复攻击性)	容易复现 (例如，攻击者可再次攻击)	有条件复现 (例如，攻击者可重复攻击，但时间受限)	不可复现 (例如，攻击者很难重复攻击过程)
Exploitability (漏洞利用难度)	容易 (例如，初学者就能掌握攻击方法)	一般 (例如，熟练攻击者才有能力发起攻击)	难 (例如，漏洞利用条件非常苛刻)
Affected Users (影响用户数)	所有 (例如，所有用户，默认配置，关键用户)	部分 (例如，部分用户，非默认配置)	少量 (例如，极少数用户，匿名用户)
Discoverability (漏洞隐蔽程度)	显而易见 (例如，漏洞很明显，攻击条件易获得)	一般 (例如，漏洞未公开，需深入挖掘)	极其隐蔽 (例如，发现漏洞极其困难)

2. 攻击树威胁建模方法

攻击树威胁建模是对系统攻击的一种分类建模方法，采用树结构描述攻击逻辑（即攻击系统的各种潜在方法），使安全分析人员从系统可能遭受攻击的角度思考安全问题，从而确定哪些威胁最有可能以及如何有效地阻止威胁。

在攻击树中，根节点代表攻击者的最终攻击目标；叶节点表示具体的攻击事件，即攻击者可能采取的各种攻击手段；中间节点表示要实现最终目标所必须完成的中间步骤（或者子目标）。节点之间的关系可以用关系节点，即与节点（AND）和或节点（OR）来表示。图 3-1 示例了攻击者以“访问公司邮件”为目标的攻击树，图中的 AND 节点表示：要达到攻击目的“利用 WebMail”就必须同时实现“获取 WebMail 软件”和“找到漏洞”。从叶节点到根节点的一条路径表示实现最终攻击目标的一种可能的攻击方法。要实现此攻击方法，需要从叶节点到根节点上的条件依次得到满足。攻击者也可以从树的任何节点出发，向根节点目标发起攻击。

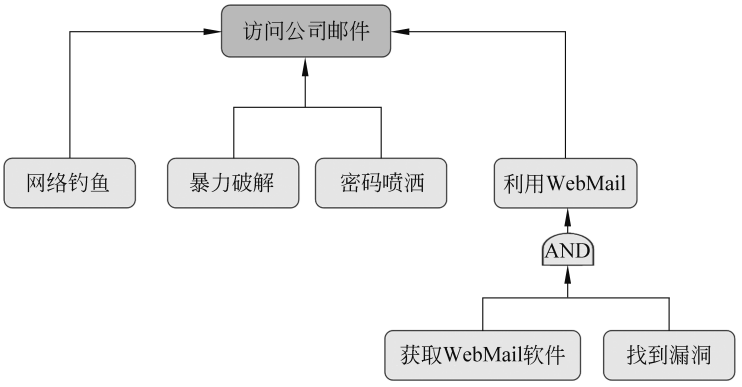

图 3-1　攻击树示例

在图 3-1 中，暴力破解使用特定的账户名，尝试不同的密码进行破解攻击；而密码喷洒使用特定的密码，尝试多个账户名进行破解攻击。

基于攻击树模型的软件安全分析是一个反向推理的过程。首先确定最终的攻击目标作为根节点；然后分析根节点事件发生所需的条件并将其作为根的子节点，分析子节点之间的关系并使用 AND 或 OR 节点来连接这些子节点；接下来按照根节点的分析扩展方法，依次对各个子节点进行分析和扩展；最后可对叶节点发生的概率进行量化，进而找出攻击者最有可能采取的攻击方法（即路径），进行重点防御。

3.3　实 例 分 析

3.3.1　软件安全需求分析实例

【实例描述】

项目团队拟开发一个短视频社交系统，其用户群体和开发目标如下，试对此短视频社

交系统进行安全需求分析。

1. 系统用户

(1) 经常性用户。包括有生活娱乐需求的短视频观众、提供娱乐性短视频的视频创作者或相关人员(统称视频创作者)、短视频系统维护人员(简称管理员)。他们都具有基本的计算机基础知识和操作计算机的能力。

(2) 间隔性用户。系统维护人员。他们是计算机专业人员,熟悉操作系统和数据库。

2. 系统目标

(1) 确保视频创作者上传娱乐性短视频内容能满足用户生活娱乐的需求,并符合国家新闻出版广电总局的要求。

(2) 提供良好的人机交互界面,操作简单。

(3) 向移动终端提供稳定、流畅的娱乐短视频。

(4) 提供视频上传者之间、视频上传者与用户之间、用户与短视频社交系统之间的社区互动功能。

(5) 跟踪和分析用户对短视频内容的偏好,主动、精确地进行视频推荐,以更好地满足用户的生活娱乐需求。

【实例分析】

使用本章介绍的软件安全需求获取方法,可以将上述短视频社交系统的安全需求分为三类:核心安全需求、通用安全需求和运维安全需求。读者可自行补充其他安全需求。

1. 核心安全需求

核心安全需求与软件安全的核心属性直接相关,可分为以下六类需求。

1) 保密性需求

保密性需求主要针对敏感数据泄露和被未授权实体访问的风险,需要考虑数据在其整个生命周期(从数据产生到应用结束)所面临的各种威胁。数据保密性要求:数据只能由授权实体存取和识别,防止非授权泄露;对于数据的保护机制,需要明确保护的时效性和保护范围;当打印或在屏幕显示用户的敏感信息(例如手机号、社交账号等)时,仅打印或显示其部分内容。

2) 完整性需求

完整性需求主要针对未经授权的修改问题,确保对系统(或软件)及其数据的修改都是经过授权的修改,从而保证系统按预期工作。完整性需求不仅要求系统的完整性、完备性和一致性,还要保证数据的完整性、完备性和一致性,防止非授权实体对系统或数据进行非法修改。当用户与应用系统进行交互时,需要防范用户的输入设备(例如键盘、鼠标等)被木马程序侦听,防止用户输入数据被截取或修改。为此,在系统处理用户输入的表单或参数数据之前,应当将这些输入数据与合法的数据集进行比较验证,而且必须进行用户身份的核实,从而保护数据的完整性并在一定程度上防范注入攻击。

3）可用性需求

可用性需求主要针对拒绝服务攻击，这种攻击通过大量并发的恶意请求占用系统资源，致使合法用户无法正常访问和使用目标系统。除此之外，不安全的编码（例如悬挂指针、内存分配不当或无限循环结构）会造成系统响应缓慢甚至崩溃，从而影响系统的可用性。为了提高系统的可用性，应当在需求中明确给出关键的系统功能（包括业务功能、基本功能和支持功能）被中断后的恢复时间；允许跨数据中心复制软件和数据，以提供系统负载均衡和冗余备份。

4）可认证需求

可认证需求用于确保提出认证申请的实体（包括人和硬件设备）的合法性和有效性。系统在用户注册时需要对用户进行实名认证并核实其手机号码等的真实性，在用户登录时对用户进行身份识别认证。还可以进行系统和用户的“双向身份认证”，即让应用系统和用户进行互相认证，不仅可以防范恶意用户攻击合法系统，还可以防范钓鱼网站等恶意系统侵害合法用户。

5）授权需求

在身份认证的基础上，授权需求用于确认一个经认证的实体在请求资源时所需要的权限水平，包括访问权限、优先级别及可执行的操作。在确定此系统的授权需求时，可使用主/客体关系矩阵模型，同时坚持“最小授权原则”，即对于主体仅授予完成工作所必需的访问权限：严格限制访问高度敏感的文件，只允许拥有最高许可证水平的用户访问；未经身份认证的用户（访客）仅拥有浏览（只读）权限；经过身份验证的用户（普通用户角色）默认拥有浏览和评论等权限。

6）监控与审计需求

监控与审计需求用于跟踪和记录系统中用户操作和活动的、可审计的完整轨迹（日志记录），其内容包括系统当前登录的用户 ID、用户类型、登录 IP 等。这些日志记录不仅可用于计算机调查取证，也可以用于系统的故障诊断。对于有授权流程的交易，系统应完整记录其授权的整个过程并将授权记录与交易记录分开存放。

2. 通用安全需求

通用安全需求是指系统中与通用功能相关的安全需求，包括以下几个方面。

1）安全架构需求

安全架构需求主要考虑系统软件体系结构本身的安全性、可靠性、兼容性和可扩展性等方面，关注系统如何进行负载均衡、采用什么集群架构等问题。可以通过调查问卷、访谈和开评审会等方式获取安全架构需求。

2）会话管理需求

会话管理需求要求在保证软件安全性的前提下进行有效会话：应当记录和跟踪每个用户的活动；在用户注销或关闭浏览器窗口时应明确地停止该用户的会话。系统可通过设定会话的时长对会话进行控制，对长时间无互动的会话进行超时断路等操作。

3）控制重复提交的需求

重复提交同一个娱乐短视频到应用系统，将会不必要地占用更多的系统资源。若重

复提交涉及交易，后果则会很严重，例如一笔用户付款转账的交易被提交两次则将可能导致该用户的账户被扣除双倍的付款额。重复提交可能是无意的，也有可能是有意的，所以当重复提交涉及对用户进行身份认证或其他安全操作时，系统应当提供反馈信息给用户的机制（例如短信或邮箱提醒等）。

4）配置参数管理需求

系统的配置参数和程序代码都需要被限制访问，以防止黑客的攻击。应当监控系统初始化和全局变量修改等活动，在程序或会话的起始事件和终止事件中完成对配置信息的安全保护，加密敏感的数据库连接，加密敏感的应用程序设置，在编程时对密码不进行硬编码等。

3. 运维安全需求

运维安全需求是指持续关注已投入运行系统的运行环境、运行状态和参数设置等情况，及时发现系统的运行故障和隐患，以保证系统正常、安全地运行。运维安全需求通常包括以下三个方面。

1）环境部署需求

可通过对用户的调查问卷和访谈、对遵从性标准和法规的分解来获得部署环境的需求。与环境部署安全相关的需求包括：系统应处于持续的安全监控环境之中，以确保它不易受到新安全威胁的影响；只有在获得所有必要的批准之后才能对生产环境进行变更；遵从公司对补丁管理的要求，对本系统的软件打补丁、及时修复漏洞。在软件正式发布之前，经过模拟环境测试之后，应当尽可能地修复所有的安全漏洞，特别是那些高危漏洞。

2）归档需求

可通过对用户的调查问卷和访谈来获得归档需求。这些需求包括：归档信息与哪些处理业务相关？采用的归档方式是远程或本地、在线或离线？采用何种存储介质？需要多少存储空间？

3）登录控制需求

用户登录是系统的关键功能之一，系统通过用户登录对用户身份进行认证。系统对用户登录的控制需求包括：对连续登录多次失败的用户，将其 IP 锁定一段时间（例如 24 小时），其中的登录失败次数和 IP 锁定时长等参数应当根据业务需求来设定，并在配置文件中有对应的配置项；对于首次登录系统的用户，系统强制将其引导到修改密码的页面，要求用户修改初始密码重新登录之后方可使用系统；系统将按照规则检查用户设定的密码类型和长度；系统将每个用户的登录情况记录入日志，以备审计。

3.3.2 软件威胁建模实例

【实例描述】

火车是人们出行出游的重要交通工具，通过订票系统订票已成为必不可少的订票方式。基于 Web 的火车票订票系统是面向互联网的开放系统，拥有数量庞大的访问用户，涉及用户个人的重要隐私和敏感信息。因此，在设计此系统时必须充分考虑系统的安全

性，尽量降低安全风险。试对基于 Web 的火车票订票系统进行威胁建模，以尽可能多地发现潜在的系统安全威胁。

【实例分析】

接下来分五个步骤，对基于 Web 的火车票订票系统进行威胁建模分析。

1. 明确安全目标

保护火车票订票系统安全的主要目标包括：①保护登录安全，防止用户的账号密码被盗用；②保护数据安全，杜绝用户敏感信息被泄露，防止攻击者对系统数据进行篡改；③保护支付安全，检测用户的支付环境是否安全；④对管理员设置车票信息等行为进行审计记录，并保证审计数据本身的安全；⑤保证系统能及时响应用户和管理员的操作。

2. 系统概要分析

用户在火车票订票系统进行登录操作，系统对在 Web 页面获得的用户账号和密码进行初步验证，然后将其发送到后台数据库进行比对，以验证登录用户是否系统的合法用户；用户在登录成功后可查询火车车次，并在有余票时进行下单，系统查询数据库是否有余票，并在有余票且用户成功支付之后创建订单并减少车票余量；管理员用户拥有普通用户的权限之外，可根据最新的列车安排设置车票，包括车次和站点等信息。

3. 系统功能分解

基于数据流图对概要设计进行细化，对系统功能进行分解，确定火车票订票系统中各模块的信任边界，细化数据流及其处理节点，以便在后续步骤中识别威胁和确认漏洞。系统通过互联网将经过验证的输入信息传送到受信任的服务器，通过外围防火墙建立信任边界；在访问敏感数据模块的入口处进行用户授权的检查，分析不同角色的数据流；对跨信任边界的数据流要进行授权检查。接下来描述的是一个大大简化了的火车票订票系统的设计，它只包含登录验证模块、订票模块和设置车票模块。

1）登录验证模块

用户在系统的 Web 页面输入登录的凭据（即账号和密码）并单击“登录”按钮，Web 页面检查该凭据是否符合规定的格式要求；Web 页面在确认登录凭据的格式无误后，将此凭据传输给后台服务器并发起后台验证请求；后台服务器将收到的凭据与数据库中的用户信息进行比对，并反馈验证响应消息给前端页面。后台服务器在验证用户信息无误后，反馈“用户登录成功”，并根据用户角色是普通用户或管理员用户，为用户分别赋予不同的系统访问权限。细化后的登录验证模块的数据流图如图 3-2 所示，图中的虚线框表示模块的信任边界（威胁往往与信任边界密切相关，甚至有一些人认为威胁只出现在信任边界上）。

2）订票模块

登录成功的用户可进行订票，通常的步骤包括：选择出发站点、到达站点、日期，查询当天某个可选车次是否有余票；当有余票时，用户填写乘客信息，成功支付后创建订单，并

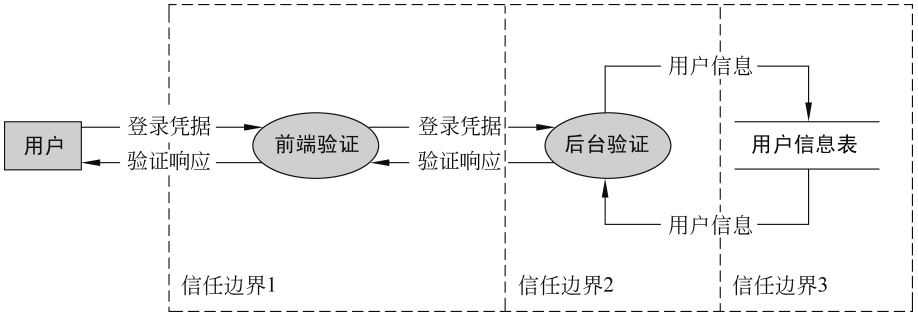

图 3-2　登录验证模块的数据流图

将包括车次、座位等订单信息返回给用户。细化后的订票模块的数据流图如图 3-3 所示。

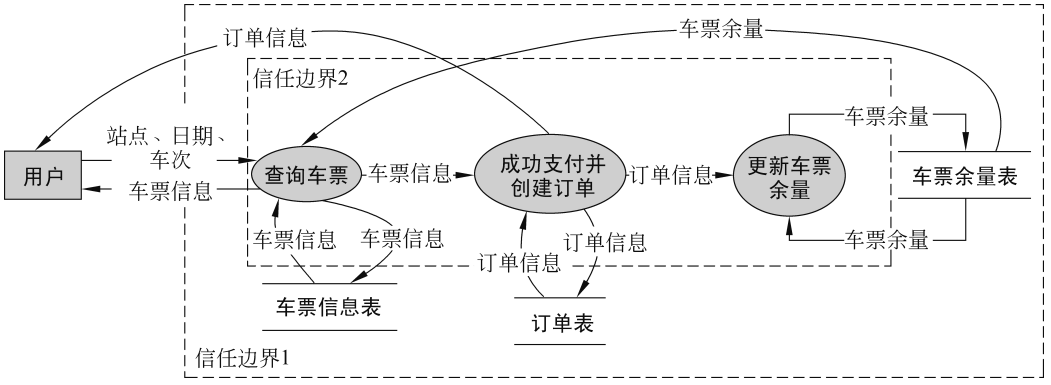

图 3-3　订票模块的数据流图

3）设置车票模块

管理员根据最新的列车运行安排设置新的车票信息，包括增加系统中的时间、车次和站点等信息；管理员的修改操作需要被记录入日志。细化后的管理员维护模块的数据流图如图 3-4 所示。

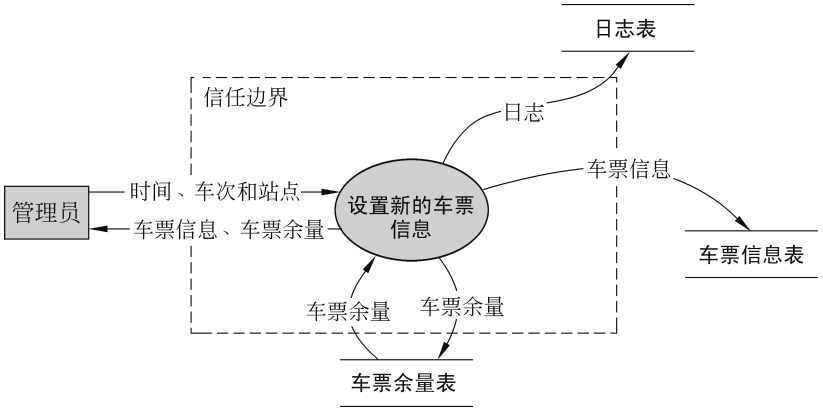

图 3-4　管理员维护模块的数据流图

4. 确定威胁

首先利用常见威胁列表检查火车订票系统，针对此系统的常见威胁如表 3-6 所示，注意检查输入输出边界和信任域的安全边界；然后使用问题驱动法，进一步确认相关威胁和攻击。经确认，针对该火车订票系统的安全威胁包括：①用户使用弱密码和共享账户；②对失败登录的账户未锁定；③使用未经验证的输入生成 SQL 查询；④以明文在数据库中保存密码；⑤以明文在网络上传递密码；⑥使用的输入验证不力且未对输出进行编码，从而导致注入 XSS；⑦会话周期过长，直接传输身份验证 Cookie；⑧未保护日志和审核数据文件，遗漏对应用程序重要操作的审计；⑨对输入至服务器上的数据验证不彻底，代码中的异常处理不完整。

表 3-6　火车票订票系统的威胁列表及其缓解措施

威胁缓解措施	安 全 威 胁
输入验证	缓冲区溢出、SQL 注入、XSS
认证	网络窃听、暴力攻击、字典攻击、Cookie 重播、凭证偷窃
授权	泄露机密数据、篡改数据
配置管理	检索配置数据、未授权访问管理界面和配置文件
敏感信息	访问敏感数据、网络窃听、篡改数据
会话管理	会话劫持、会话重播、中间人攻击
加密	破解算法、破解密钥
参数操纵	查询字符串操纵、表单操纵、Cookie 操纵、HTTP 头操纵
异常管理	拒绝服务、泄露系统信息
审计和日志	用户抵赖操作、掩盖踪迹

5. 制订威胁缓解计划或策略

通过重复上述第 2～4 步，迭代地进行火车票订票系统的威胁建模活动。随着项目的推进，将发现更多涉及威胁的细节。一旦确认系统的安全威胁，就要采取相应措施缓解威胁。因为从经济角度来讲，不可能解决所有存在的威胁，所以要评估这些威胁的安全风险级别以优先处理高风险的威胁。对于上述火车票订票系统的安全威胁，按其风险级别从高到低排序如下：①身份验证受到基于字典的暴力破解；②输入查询模块受 SQL 注入，使攻击者能够利用输入验证漏洞在数据库中执行命令；③凭证被偷窃，攻击者通过漏洞获取包括用户名和密码的敏感数据；④网络被窃听，攻击者通过嗅探以获取客户端凭证；⑤服务器遭受拒绝服务攻击，攻击者在系统运行的关键时期发起此类攻击以妨碍系统服务；⑥受到 Cookie 重播或捕获攻击，攻击者通过欺骗 Cookie 标识以另一个用户的身份访问应用程序；⑦攻击者试图通过漏洞注入 XSS；⑧用户抵赖其执行过的操作以逃脱审计，攻击者设法运行某个重要业务功能而没有留下痕迹。