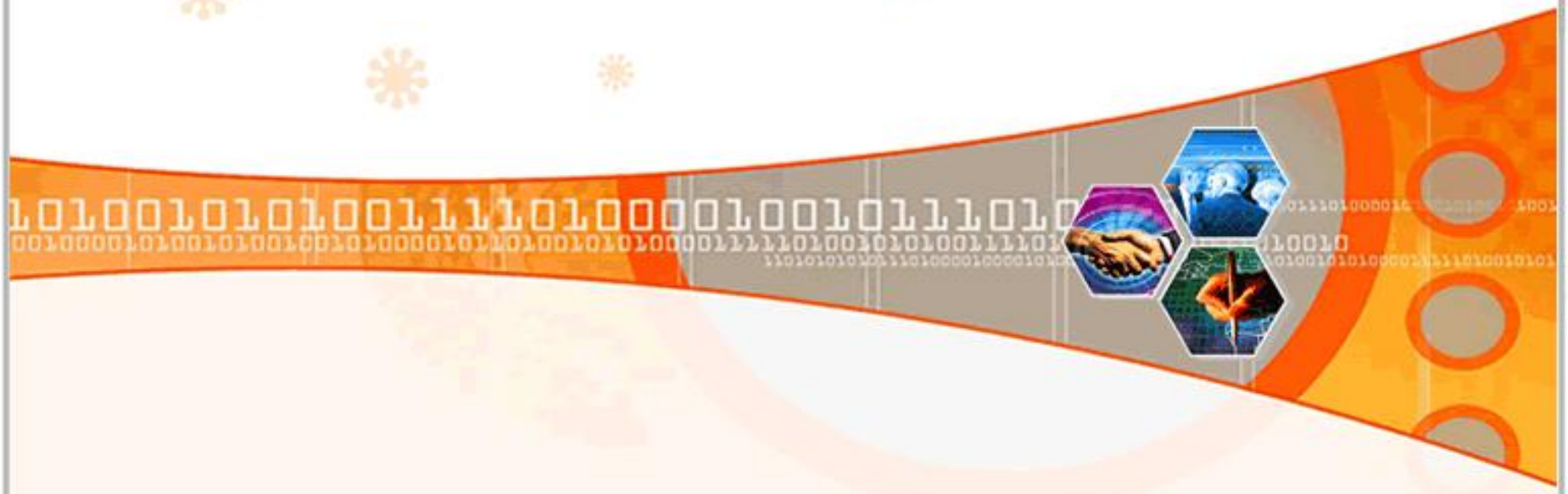
第1章

安全编程概述



简介

现代生活中，计算机的应用已经越来越广泛，给我们的生活带来了巨大的方便。计算机系统的安全问题也越来越受到重视。软件，是组成计算机应用的一个重要部分，当软件由于不安全而遭受攻击，或者运行期间出现错误时，会给用户带来巨大的损失。如犯罪分子利用软件漏洞来获取有价值的信息，用于牟取利益；又如软件因为开发时没有考虑运行时的具体情况，而造成运行的突然崩溃；等等。

面对越来越频繁的软件安全隐患带来的损失，对软件的开发者——软件工程师，提出了更高的要求，要求程序员能够编写出错误更加少的程序，并且能够及时修复软件出现的突发问题，切实为软件使用者服务。本书讲解的安全编程技术主要就是针对这些问题进行解决。安全编程，是软件质量的重要保证，在软件开发和程序设计中具有重要地位。

不过，实际的软件工程中，安全隐患的出现往往来源于多个方面，给软件系统带来的危害也是多方面的。安全问题的出现，由于原因众多，而某些安全问题又具有不间断发生，难于调试等特点，因此，很难用一个单纯的理论来完全地阐述安全编程问题。基于这个考虑，安全编程的内容只能针对各个侧面来进行阐述，如异常情况下的安全、线程操作中的安全、数据安全加密等等。

- **主要针对问题：**
  - ✓ 软件安全问题出现的原因
  - ✓ 软件安全问题的表现
  - ✓ 安全问题分类
  - ✓ 安全问题详细介绍
  - ✓ 本书内容

1.1 软件的安全问题

1.1.1 任何软件都是不安全的

- 安全问题的典型表现：
  - ✓ 使用某些交易软件的过程中，某些敏感信息，如个人身份信息、个人卡号密码等信息被敌方获取并用于牟利；
  - ✓ 访问某些网站时，服务器响应很慢，或者服务器由于访问量造成负载过大，造成突然瘫痪；
  - ✓ 自己的系统中安装了具有漏洞的软件，漏洞没有解决，敌方找到漏洞并对本机进行攻击，造成系统瘫痪
  - ✓ 自己花费精力完成了一幅漂亮的风景画，放到网上去，没有考虑版权，被他人随意使用却无法问责；

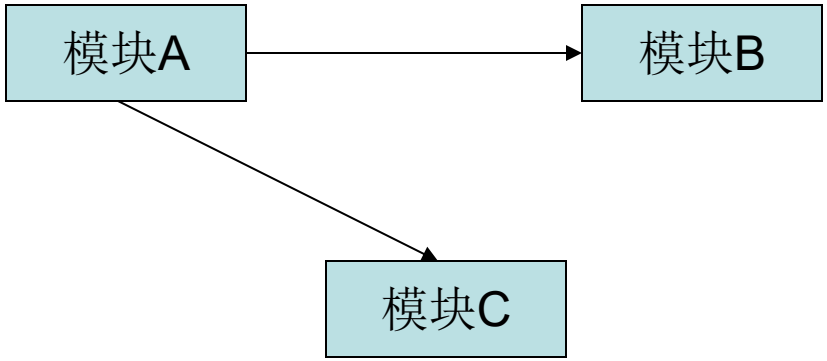
  ……

- 因此，对软件的开发提出了两个新的要求：
  - ✓软件复杂性加强；
  - ✓可扩展性要求的提高；

软件安全的挑战性

- ✓ 一方面，软件复杂了，安全问题也表现得很复杂，无法得到全面的考虑，而工程进度又迫使开发者不得不在一定时间内交付产品，代码越多漏洞和缺陷也就越来越多；
- ✓ 另一方面，软件的可扩展性要求也越来越高，系统升级和性能扩展成为很多软件必备的功能；可扩展好的系统，由于其能够用较少的成本实现功能扩充，受到开发者和用户的欢迎；但是由于针对可扩展性必须具备响应的设计，软件结构变复杂了，另外，添加新的功能，也引入了新的风险。

- 怎样解决这些安全问题？
  - ✓ 首先，大多数人可以想到的方法是软件测试，通过测试来减少软件中的缺陷。但是，由于软件系统规模越来越大，软件开发的进度要求越来越高，不可能在有限的时间内考虑所有安全方面的问题，即使进行了全方位的测试，也只能对所有的测试案例进行很小范围的覆盖。如下图：



模块A使用模块B和模块C，以黑盒测试为例，如果模块A的输入有X种，模块B的输入有Y种，模块C的输入有Z种，理论上讲，应该对X*Y*Z个组合进行全面的测试。但是，由于工程进度问题，实际上在测试时不可能兼顾全面，往往只是采用了一些具有代表性的测试案例来进行测试，但这些测试案例在设计的时候又不能保证能够具有最全面的代表性。如果想要将所有问题考虑到，除非进行穷举测试，而，这种穷举测试基本上是不可能完成的。

因此，软件测试无法完全保证软件的安全性。一方面是想要实现全面的测试，找出全部的错误，另一方面又要保证工程的进度，早日解决用户的问题，往往无法两全，只能在其中找到平衡点。

关于测试，另一个问题是，全面的测试，一般情况下是针对所有可能出现的隐患进行测试，但是这需要对软件的隐患具有全方位的预见性。而在有些情况下，很多隐患是在运行期间才显露出来的，软件的开发者很难在开发阶段预见到所有可能出现的隐患，容易让测试陷入盲目。

因此，测试只能减少软件安全问题的发生，但是不能完全解决安全问题。业界大都公认一个事实：几乎所有的软件都是带着安全隐患投入运行。

提示：

以网络软件为例，敌方可能通过因特网获得未授权的访问的信息，或者利用软件缺陷来控制用户系统并展开攻击。随着网络应用的更加丰富，用户对网络服务的依赖也相应的增加(如网上银行、网上股票、网上游戏等)，这也导致了攻击的方法的增加和复杂化，从而使得安全问题更加凸显出来。而软件工程师无法在开发阶段就预见到全部的攻击，提高了软件开发的难度。所谓“防不胜防”，就是这个道理。

- 另一个解决安全问题的方法可能就是在测试前就尽量多地解决安全隐患。
- 在设计、编码阶段，熟练的软件设计人员和软件工程师完全可以尽可能多地将安全问题进行考虑并加以解决。如果在程序设计的时候就能够尽量地考虑安全问题，对软件的安全性也就会有更好的保证，可以大大减小测试的负担。这就是《安全编程技术》所阐述的内容。
- 近年来，不管是在应用方面还是在研究方面，安全编程技术越来越受到了重视，本书将针对该话题中的若干方面进行讲述。

1.1.2 软件不安全性的几种表现

- 软件的不安全性，一般情况下的受害者就是其直接用户。从用户的角度来看，软件的不安全性主要体现在两个方面：

- 软件在运行过程中不稳定，出现异常现象、得不到正常结果、或者在特殊情况下由于一些原因造成系统崩溃。比如：
  - ✓ 由于异常处理不当，软件运行期间遇到突发问题，处理异常之后无法释放资源，导致这些资源被锁定无法使用；
  - ✓ 由于线程处理不当，软件运行中莫名其妙得不到正常结果；
  - ✓ 由于网络连接处理不当，网络软件运行过程中，内存消耗越来越大，系统越来越慢，最后崩溃；
  - ✓ 由于编程没有进行优化，程序运行消耗资源过大；等等。

- 敌方利用各种方式攻击软件，达到窃取信息、破坏系统等目的。比如：
  - ✓敌方通过一些手段获取数据库中的明文密码；
  - ✓敌方利用软件的缓冲区溢出，运行敏感的函数；
  - ✓敌方利用软件对数据的校验不全面，给用户发送虚假信息；
  - ✓敌方对用户进行拒绝服务攻击；等等。

通常情况下，大多数安全问题是在软件运行的过程中发生，而负责软件系统运行的技术管理人员或者软件的个人用户，并不是专业的软件开发人员。此时他们往往无法给出直接的应对方案，虽然可以依靠一些简单的方法，如：优化操作系统、优化网络、优化数据库管理系统或者设置额外的操作权限来对付这些剧增的安全问题，但是实际上，这些方法都是治标不治本的方法。此时，软件的生产单位就需要投入大量的成本，来进行软件的维护。

1.1.3 软件不安全的原因

软件出现安全隐患，并造成损失，一方面是由于攻击者的猖獗，但是从开发者角度，几乎都有一个共同的基本原因：那就是由于软件在设计、编码、测试和运行阶段，没有发现软件中的各种漏洞，导致软件的不安全。

从严格的定义上来讲，软件安全隐患一般可以分为两类：错误和缺陷。错误是指软件实现过程出现的问题，大多数的错误可以很容易发现并修复，如缓冲区溢出、死锁、不安全的系统调用、不完整的输入检测机制和不完善的数据保护措施等；缺陷是一个更深层次的问题，它往往产生于设计阶段并在代码中实例化且难于发现，如设计期间的功能划分问题等，这种问题带来的危害更大，但是不属于编程的范畴。但是业内一般将这两个概念放在一起讲，将错误和缺陷不是划分的绝对清楚，本书也沿袭这一说法。

- 下面来阐述软件不安全的原因。首先，站在软件的开发者主观的角度，软件的不安全的原因可以归纳为以下几种：
  - ✓ 软件的生产没有严格遵守软件工程流程。由于缺乏经验或者蓄意(如片面追求高进度)的原因，软件的设计者和开发者们没有一个统一的管理，可以在软件开发周期的任意时候，随意删除、新增或者修改软件需求规格说明书、威胁模型、设计文档、源代码、整合框架、测试用例和测试结果、安装配置说明书，使得软件的安全性保证大大减弱。

大多数系统软件或其它商业软件，结构都相当大并且复杂，而且由于考虑到软件的扩展性，它们的设计更加巧妙，复杂性可能会更加提高一些。在运行的过程中，这些系统又可以在大量不同的状态之间转换，这个特性使得开发和使用持续正常运行的软件，是一件很困难的事情，更不用说持续安全运行了。面对不可避免的安全威胁和风险，项目经理和软件工程师必须从开发流程做起，让安全性贯穿整个软件开发的始终。就大多数相对成功的软件工程案例而言，如果项目经理和软件工程师针对软件缺陷进行系统的训练，可以避免软件的许多安全缺陷。

- 编码者没有采用科学的编码方法。在软件开发的过程中没有考虑软件可能出现的问题，仅仅将能够想到的问题停留在实验室内进行解决。实际上，有些程序，在实验室阶段根本不会出现安全隐患，如下代码：

```
void function(char *input)
{
char buffer[16];
strcpy(buffer,input);
}
```

表示将input字符串拷贝到buffer中，如果没有考虑缓冲区溢出，即使在开发阶段的测试过程中让这个函数产生缓冲区溢出，也不会产生攻击效果。只有在精心设计之后，才可能对系统造成攻击。因此在开发阶段很难意识到这个问题，使得软件留下安全隐患。

- 测试不到位(不过有时是无法到位)。主要是测试用例的设计无法涵盖尽可能典型的安全问题。如下的登录表单：

一般测试用例只是设计输入正确的用户名和密码，看能否正常登录；再输入错误的用户名和密码，看能否得到相应的错误提示。但是攻击者如果输入某些和SQL注入有关的值，就有可能在不需要知道用户名和密码的情况下登录到系统，甚至知道系统中的其它信息或对系统中的内容进行修改。

- 从软件工程客观角度讲，软件的安全性隐患又来源于以下几个方面：

1. 软件复杂性和工程进度的平衡。如前一节所述，软件规模复杂了，不仅仅是编码工作量的提高，更重要的是其中需要考虑的问题更加复杂，测试用例规模也呈指数级增长。但是工程进度只是按照软件规模进行适当的延长，因此很多问题来不及解决，软件带着缺陷投入使用。

2. 安全问题的不可预见性。主要是软件工程师对运行的实际情况的不了解，在测试时作出过于简单的假设。有些问题，包括对软件的功能、输出和软件运行环境的行为状态，或者外部实体(用户，软件进程)的预期输入，都无法完全考虑到。而攻击者有足够的时间进行攻击方法的研究。
3. 由于软件需求的变动。软件规格说明书或设计文档无法一开始就确定下来；在现代软件工程中，很多软件的需求变动，导致其设计本来就是变动的，很多安全问题可能在变动的过程中被忽略。

4. 软件组件之间的交互的不可预见性。如客户可能在运行软件的过程中，自行安装第三方提供的组件，开发者根本无法知道客户的软件将要和谁交互，软件在运行的过程中出现安全问题。

因此，我们可以看到，不管采用了什么样的措施，软件的安全问题都无法完全避免。即使在需求分析和设计时可以避免(如通过形式化方法)，或者在开发时可以避免(比如通过全面的代码审查和大量的测试)，但缺陷还是会在软件汇编、集成、部署和运行时候被引入。不管如何忠实的遵守一个基于安全的开发过程，只要软件的规模和复杂性继续增长，一些可被挖掘出来的错误和其它的缺陷是肯定存在的。我们所能做的工作就是尽量让安全问题变少，而不能完全消灭安全问题。因此，本书所叙述的“安全编程技术”，不是为了消除安全隐患，而是为了尽量减少安全隐患。

1.2 在软件开发生命周期中考虑安全问题

- **软件开发生命周期可以有很多模型，如瀑布模型、原型模型等。但是一般说来，软件开发生命周期可以包括以下5个阶段：**

  ✓分析阶段。软件需求分析，实际上是回答“软件需要完成什么功能”。它的主要工作是，通过研讨或调查研究，对用户的需求进行收集，然后去粗取精、去伪存真、正确理解，最后把它用标准的软件工程开发语言(需求规格说明书)表达出来，供设计人员参考。

  该阶段首先是在用户中进行调查研究，和用户一起确定软件需要解决的问题，此阶段的重要工作有：

- 建立软件的逻辑模型；
- 编写需求规格说明书文档，根据用户需求，通过不断沟通，反复修改，并最终得到用户的认可；

✓ 设计阶段。一般说来，软件设计可以分为概要设计和详细设计两个阶段。该阶段的最终任务是将软件分解成一个个模块(可以是一个函数、过程、子程序、一段带有程序说明的独立的程序和数据，也可以是可组合、可分解和可更换的功能单元)，并将模块内部的结构设计出来。

该阶段的主要工作有：

- 利用结构化分析方法、数据流程图和数据字典等方法，根据需求说明书的要求，设计建立相应的软件系统的体系结构；
- 进行模块设计，给出软件的模块结构，用软件结构图表示，将整个系统分解成若干个子系统或模块，定义子系统或模块间的接口关系；
- 设计模块的程序流程、算法和数据结构，设计数据库；
- 编写软件概要设计和详细设计说明书，数据库或数据结构设计说明书，组装测试计划。

- ✓ 编码阶段。该阶段主要把软件设计转换成计算机可以接受的程序，选择某一种程序设计语言，编写出源程序清单。

  此阶段的主要工作有：

  - 基于软件产品的开发质量的要求，充分了解软件开发语言、工具的特性和编程风格；
  - 进行编码；
  - 提供源程序清单。

- 测试阶段。软件测试的目的是以较小的代价发现尽可能多的错误。要实现该目标，关键在于设计一套出色的测试用例。不同的测试方法有不同的测试用例设计方法，目前常见的测试方法有：
  - 白盒测试法。该测试法的对象是源程序，依据的是程序内部的的逻辑结构来发现软件的编程错误、结构错误和数据错误(如逻辑、数据流、初始化等错误)，其用例设计的关键，是以较少的用例覆盖尽可能多的内部程序逻辑结果。
  - 黑盒测试法。依据的是软件的功能或软件行为描述，发现软件的接口、功能和结构错误。黑盒法用例设计的关键同样也是以较少的用例覆盖模块输出和输入接口。

此阶段的主要工作是：

- 设计测试用例，进行测试；
- 写出测试报告，提交修改部门；
- 继续测试。

- **维护阶段。本阶段主要根据软件运行的情况，对软件进行适当修改，以适应新的要求；以及纠正运行中发现的错误。本阶段工作在已完成对软件的研制(分析、设计、编码和测试)工作并交付使用以后进行，一般所做的工作是编写软件问题报告、软件修改报告。**

维护阶段的成本是比较高昂的，设计不到位或者编码测试考虑不周全，可能会造成软件维护成本的大幅度提高。以一个中小规模软件为例，如果设计、编码和测试需要一年的时间，在投入使用后，其运行时间可能持续三年。那么维护阶段也就要持续三年。这段时间内，软件的维护者除了要解决研制阶段所遇到的各种问题，如排除障碍外，还要扩展软件的功能，提高性能。所以，事实上，和软件开发工作相比，软件维护的工作量和成本都要大得多。

在实际开发过程中，软件开发并不一定是从第一步进行到最后一步，而是在任何阶段，在进入下一阶段前一般都有一步或几步的回溯。如在测试过程中的问题可能要求修改设计，用户可能会提出一些需要来修改需求说明书等。

以下主要基于安全问题，针对软件工程中的各个阶段进行阐述。

1.2.1 软件设计阶段威胁建模

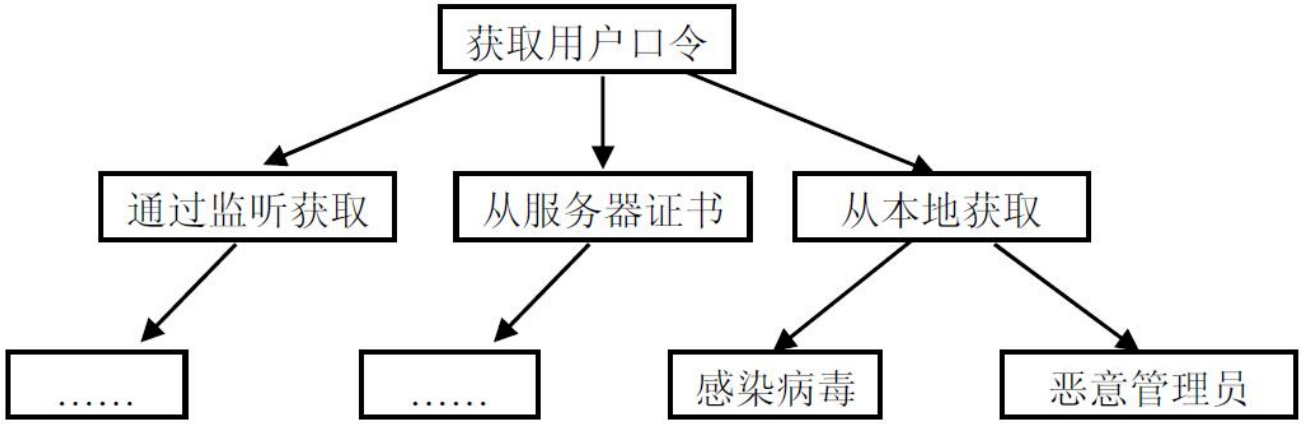
- 软件在设计阶段达到的安全性能，将是软件整个生命周期的基础。如果在设计阶段没有考虑某些安全问题，那么在编码时就几乎不被考虑。这些隐患将可能成为致命的缺陷，在后期以更高的代价的形式爆发出来。所以，安全问题，应该从设计阶段就开始考虑，设计要尽可能完善。
- 传统的软件设计过程中，将工作的重点一般放在软件功能的设计上，没有非常详细地考虑到安全问题。因此，在软件设计阶段，针对安全问题，应该明确以下方面：
  - ✓ 安全方面有哪些目标需要达到；
  - ✓ 软件可能遇到的攻击和安全隐患；等等。

该阶段，一般采用威胁建模的方法来在软件设计阶段加入安全因素的考量。威胁建模除了和设计阶段的其它建模工作类似的地方外．更加关心安全问题，是一种比较好的安全问题的表达方法。

如前所述，分析系统中可能存在的威胁，可能是一件比较繁重的工作，因为很多威胁是不可预见的。但是，在设计阶段就尽可能多地将威胁考虑到，在编写代码前修改方案，代价比较小。威胁建模过程一般如下：

- ✓ 在项目组中成立一个小组。在此过程中，需要从项目组内挑选对安全问题比较了解的人，这些人可能不一定要懂得怎样去编写程序，但是要懂得程序运行的过程中，可能会出现哪些安全问题，或者可能受到什么样的攻击。也可以请用户中的某些人来参与该小组。
- ✓ 分解系统需求。本过程中，可按照需求规格说明书和设计文档中的内容，站在安全角度，分析系统在安全方面的需求。当然，传统的软件工程中的一些工具也可以使用，如：数据流图(DFD)、统一建模语言(UML)等。关于这些内容，大家可以参考相应文献。
- ✓ 确定系统可能面临哪些威胁。系统可能遇到的威胁有很多种，在这里可以首先将威胁进行分类，如系统缓冲区溢出、身份欺骗、篡改数据、抵赖、信息泄露、拒绝服务、特权提升等。由于同类的安全问题可以用类似的方法解决，因此该过程可以减小后期工作量。

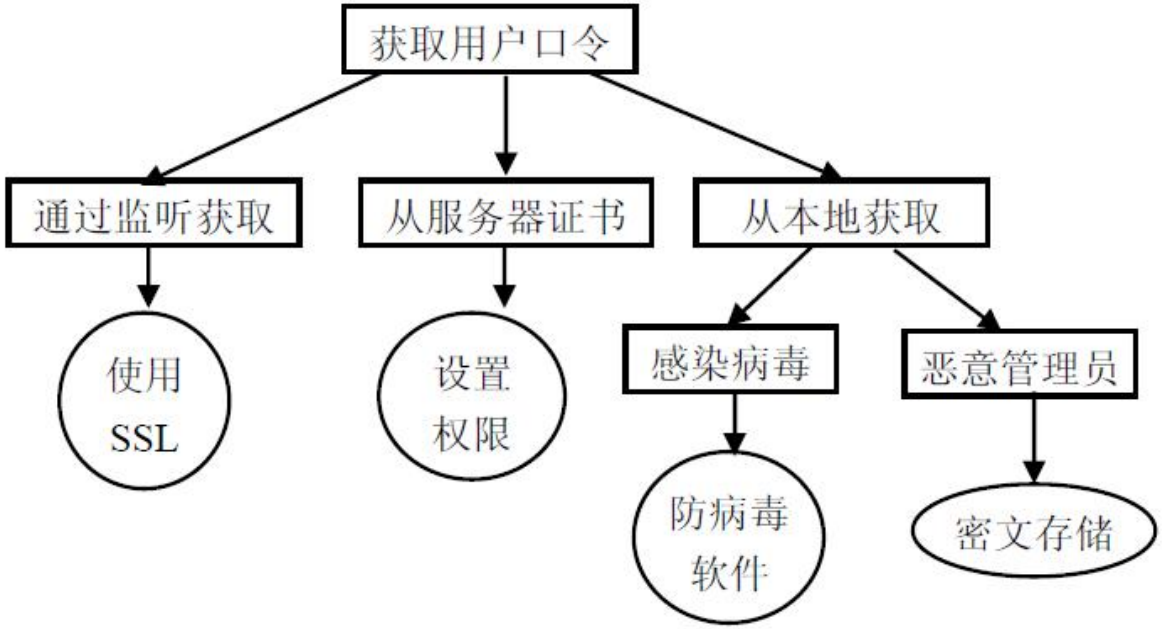
另外，对威胁进行分类之后，可以画出威胁树，其目的是对软件可能受到的威胁进行表达。如图是一个针对用户口令安全问题画出的威胁树：



以上威胁树中画出了用户口令不安全中的各种原因，我们就可以针对不同的原因采用相应的措施。

- ✓ 选择应付威胁或者缓和威胁的方法。很显然，针对不同的安全问题，我们可以选择应付威胁或者缓和威胁的方法。一般说来，可以应付或缓和威胁的方法有很多，但是考虑到实施的成本，根据威胁可能的危害程度，还是要有所选择。在面对威胁时，我们可以采用的方法有：
  - 不进行任何处理。这是不建议的方法。
  - 告知用户。如果某些威胁无法通过软件工程师自身在产品上施加某种技术来解决，可以告知用户。如提醒用户要杀毒等。
  - 排除问题。在软件中施加某种技术来避免出现安全问题。
  - 修补问题。某些问题如果无法预见和解决，可以提供修补接口，待出现问题之后进行扩展。不过这种方案的代价是比较大的，对软件的设计提出了较高的要求。

- 确定最终技术。在各种备选的方案中，确定最终选用的技术。一般可以将最终选用的技术，直接在威胁树中描述或者用图表画出来。如图所示的就是针对用户口令安全的威胁树进行的修改：



1.2.2 安全代码的编写

- 实际上，在设计阶段如果尽可能多地将问题考虑周到，在软件的代码编写阶段，只是针对这些问题进行实现而已。不过，在编码过程中也需要考虑一些技巧。如：
  - ✓ 内存安全怎样实现？
  - ✓ 怎样保证线程安全？
  - ✓ 如何科学地处理异常？
  - ✓ 输入输出安全怎样保障？
  - ✓ 怎样做权限控制？
  - ✓ 怎样保护数据？
  - ✓ 怎样对付篡改和抵赖？
  - ✓ 怎样编写优化的代码？

1.2.3 软件的安全性测试

- 测试是软件发布前所做的重要工作，一方面，需要对软件的可用性进行评测，另一方面，也要对软件的安全性进行最大限度的保障。所以，测试工作决定着软件的质量，是软件质量保证的关键手段。
- 在充分考虑安全性问题的前提下，安全性测试显得尤为重要。安全测试和普通的功能性测试主要目的不同。普通的功能测试的主要目的是：

- ✓ 确保软件不会去完成没有预先设计的功能；
- ✓ 确保软件能够完成预先设计的功能

- 而安全测试是安全的软件生命周期中一个重要的环节。实际上，安全测试就是一轮多角度、全方位的攻击和反攻击。因此，进行安全测试，需要精湛的系统分析技术和反攻击技术，其目的就是要抢在攻击者之前尽可能多地找到软件中的漏洞，以减少软件遭到攻击的可能性。因此，安全测试有如下特点：

- ✓非常灵活，测试用例没有太多的预见性；
- ✓没有固定的步骤可以遵循；
- ✓工作量大，并且不能保证完全地加以解决。

1.2.4 漏洞响应和产品的维护

- 在软件开发的过程中，即使在设计、代码编写和测试过程中考虑了安全因素。最终的软件产品仍可能存在漏洞。漏洞一般在用户使用的过程中被发现，此时，迅速确认、响应、修复漏洞，是非常重要的。
- 由于软件的维护是一个长期的过程，因此，软件的维护和跟踪要及时持续，也要花费较大的成本。大型软件公司都会有自己的安全响应队伍，专职处理安全事件，在发现漏洞后的第一时间采取措施，以保护客户的利益不被侵害。

- 一般来说，正常的漏洞响应可以大致分为以下四个阶段：
  - ✓ 发现漏洞通知厂商。在该阶段，漏洞首先由用户报告给厂商所设置的安全响应中心，响应中心经过初步的鉴定，如果确信是一个漏洞，安全响应队伍向漏洞上报者确认已经收到漏洞报告。
  - ✓ 确认漏洞和风险评估。安全响应队伍会联系上报者和相关产品的开发部门，以获得更多的技术细节，有时甚至会将上报者和开发团队召集在一起进行讨论。当漏洞被成功重现后，为漏洞定一个威胁等级。

- ✓ 修复漏洞。安全响应队伍和开发队伍协商决定解决方案，并确定响应工作的时间表。开发部门开始修复漏洞．补丁完成后．进行严格的测试。
- ✓ 发布补丁及安全简报对外公布安全补丁。通知所有用户修补该漏洞，在网站上发布安全简报，其中会特别感谢上报漏洞和协助修复漏洞的安全研究人员。

1.3 本书的话题

1.3.1 编程中的安全

- 如前所述，想要让软件的缺陷尽量少，并且在其运行期间可以预测其所有的安全隐患，这是非常困难的事；另一方面，如果要完全消除安全隐患，也是不可能的。传统的软件开发组织常常把软件的功能、任务时间表和开发成本放在关注的首位，而把软件的安全和质量放在其次，这在现代软件工程中，已经无法适应需求。因此，在编码过程中，必须考虑很多安全性。
- 健全的编码可以大大减少软件实现期间引入的漏洞，本书针对编码过程中的一些安全技巧进行讲解。主要针对如下问题进行阐述：

- ✓ 基本安全编程。主要是针对编程过程中最常见、最基本的安全问题进行讲解。包括：
  - 内存安全。主要包括编程过程中内存数据出现的常见安全问题，如缓冲区溢出、整数溢出、字符串格式化等。
  - 线程/进程安全。线程在软件开发过程中运用很广，但是不恰当的线程操作可能会造成安全隐患，该话题主要讲解线程安全问题，如线程同步、线程死锁等。
  - 异常/错误处理中的安全。异常是程序设计中必须处理的，本话题主要解决怎样处理异常能够保证系统的安全性。
  - 输入/输出安全。不恰当的输入可能给系统带来隐患，因此，如对输入的合法性检测等内容在本话题中提出。

- ✓ 应用安全编程。主要是针对常见的某些特定应用中出现的安全问题进行讲解。包括：
  - **数据库安全。数据库是大量软件中必然使用的系统，针对数据库安全的讲解主要包括数据库存储和访问上的安全。**
  - **国际化安全。软件的国际化很重要，但是由于国际化过程中的编码问题，有时会造成安全隐患，因此，本话题主要讲解国际化过程，以及解决国际化过程中的缓冲区溢出问题。**
  - **面向对象中的编程安全。目前，大部分软件项目都使用面向对象的方法进行编写，本话题主要解决面向对象编程中内存分配和数据安全，以及介绍一些面向对象的技巧来提高系统性能的方法。**

- 权限控制。很多系统中都会涉及授权和限制访问，权限控制是解决资源访问安全性的重要途径，本话题主要站在编程人员的角度，讲述怎样用编码方式实现较好的权限控制。
- Web编程安全。Web是一种比较流行的编程方式，Web编程中安全问题多种多样，这里主要解决网络编程中如跨站脚本、SQL注入、Web认证攻击、URL操作攻击等安全问题。
- 远程调用和组件安全。远程调用和组件是某些项目中的关键技术，也是某些编程体系结构中的亮点，本话题主要解决RPC、DCOM、EJB等组件在开发过程中遇到的安全问题。
- 避免拒绝服务攻击。拒绝服务攻击是一种比较古老的攻击，出现得比较频繁，危害也比较大。该部分主要讲述拒绝服务攻击的原理以及解决方法。

1.3.2 针对安全的编程

- **针对信息安全的编程主要集中在以下几个方面，本书也将进行讲解。**
  - ✓ 加密解密。在信息时代，数据的安全越来越受到了关注。对于保存在计算机上的某些数据，我们希望其信息不被人所知；对于在网络上传输的重要数据，我们希望即使被敌方窃听之后也不会泄密。此时，将信息进行加密，就成了保障数据安全的首要方法。该内容主要介绍常见的加密解密算法的实现。

- ✓ 数据的其它保护。由于数据加密算法所需要占用的资源和加密解密算法本身的复杂度，盲目将数据通过加密来进行保护，有可能会降低系统的运行速度。因此，不用加密方法来进行的数据保护，也具有较好的应用背景。该章针对一些特定的数据保护场合进行介绍。
- ✓ 数字签名。加密保护实际上防止的是被动攻击。在这这种攻击模式下，攻击者并不干预通信流量，只是尝试从中提取有用的信息。实际上，网络上的安全问题不仅仅只限于被动攻击，大量的主动攻击也是网络安全上需要考虑的重要问题。除了加密解密外，还需要对对信息来源的鉴别、对信息的完整和不可否认等功能进行保障，而这些功能通常都是可以通过数字签名实现。该章对数字签名的原理、实现方法进行了详细叙述。

1.3.3 其他话题

- 针对软件安全中的其他几个问题，本书也将进行了讲解。包括：
  - ✓ 软件的安全测试。虽然安全测试和安全编程属于两个不同的领域，但是同样是软件安全的重要保障。本章主要针对测试阶段的安全问题进行讲解。
  - ✓ 代码性能调优。代码性能的好坏有时候也关系到系统的安全。因此，在本书中也对代码性能调优进行了讲解。

小结

- 本章对安全编程技术进行了概述。首先讲解了软件安全问题出现的原因；然后阐述了软件安全问题的一些表现；站在软件开发者的角度，对安全问题进行了一些分类；接下来对软件工程中的安全问题进行了详细的讲解，提出了一些措施来在软件工程的各个阶段考虑安全问题。最后并介绍了本书的内容。

练习

1：软件安全问题的出现，直接的感受者一般是用户。
(1) 站在用户的角度，举出两个因为编程安全问题被忽略而造成隐患的现象。
(2) 站在程序员角度，分析其原因。

2：软件测试由于无法实现穷举测试，无法发现软件中的全部错误。
(1) 任举一个软件输入的例子，计算其穷举测试的代价。
(2) 怎样用尽量少的用例发现尽量多的问题？

3：有一个WEB站点，其数据库可能被攻击者攻击，请画出相关威胁树，并加上相应的解决方法。

4：一个软件，在开发阶段运行没有问题，但是在运行很长一段时间之后出现问题，有这样的情况吗？请你列举一个例子，并说明可能的原因。

5：如果你是项目经理，怎样在测试和产品交付之间找到平衡？