软件安全

第1章 课程介绍与软件安全概述

徐国胜
北京邮电大学

内容安排

- 0 课程介绍
- 1 概述
- 2 软件安全
- 3 漏洞
- 4 软件风险

教师、教材与交流

教师：徐国胜

——Email：guoshengxu@bupt.edu.cn
——电 话：18910970501
——微 信/QQ：guoshengxu/59845049
——微 信 个 人 公 众

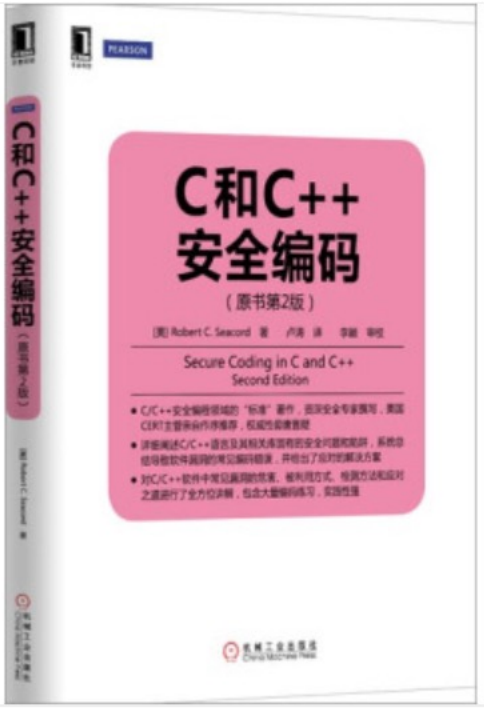
- **教材**

C和C++安全编码，第2版，机械工业出版社

0day安全：软件漏洞分析技术

- 王清
- 电子工业出版社

其他网络资源



- **交流**
  - 需要大家关注公众号，期末可以取消关注
  - 在云邮空间和教务网站上都有填QQ群号：298456519
  - 请学委把联系方式发给我，方便有一些问题和同学们交流

课程目标

- 介绍软件安全概念，普及软件安全意识；
- 使学生能够认识到软件开发及使用中存在的风险；
- 理解C/C++中安全编码缺陷根源、防范措施，学会初步的软件安全风险识别能力；
- 强调动手，使学生掌握初步的软件攻防能力，通过实验环节来实现。

课程内容安排

- 课程介绍与软件安全概述
- 软件安全基础
- 字符串安全
- 指针安全
- 动态内存管理安全
- 整数安全
- 格式化输出
- 多线程安全
- 文件I/O安全

考核要点

- 平时成绩：包括考勤、作业（课堂作业和课后论文）、资讯（提交统计和阅读统计）
- 实验考核：最多8次实验，取每次成绩的平均
- 期末考核：期末考试卷面成绩

平时考核

包括考勤，与课堂作业同步；课堂作业、少许课后作业（以学术研究为主要）和小资讯（每个人3条资讯，考核包括提交和阅读），没有期中考试。

（1）课堂作业采用数学作业纸，一般一张纸，当堂提交；（如果线上，请大家使用手机拍照发到邮箱，恢复线下则纸板提交）

（2）其余采用电子版提交。电子版提交地址：buptxu@sina.cn电子版作业提交需要大家按照约定时间来提交，延迟视为未提交。

（3）提交资讯的时间会做一个详细的分配，以免出现期末提交风暴！

（4）课堂演示（加分项），就是把实验部分完整的在课堂上进行先行展示！

提交要求

（1）除非单个文档的，否则需要打包，可以是zip，也可以是rar，并且文件名需要依次包含你的作业类别、编号、班级、学号、姓名和提交时间；
（2）小资讯是课程相关新资讯，本学期每个人都需要提交不少于3条，不同同学不可以提交相同的，提交资讯时，需要有标题，自己的评论，评论字数不限，但是务必要能够表达一个完整的意思，最后附上资讯原始的网址和能够体现这则新闻的图片一张。资讯使用word文档提交统一地址。资讯我这边使用个人公众号发布给大家看，请大家本学期关注。期末可以取消关注。要求大家仔细阅读小资讯，并且回复：学号（可以包括姓名和自己的评论）。
由于会有多门课程公用一个公众号，所以会在消息上显示是那个课程的，大家只需要负责回复本课程的就可以。
（3）作业和资讯命名格式：资讯/课堂作业/课后作业/实验-01-805-2019211592-张杉-20210302.doc/docx/zip/rar/jpg/png（类别次数编号/条目数-三位班级号-学号-姓名-提交时间.zip/rar）

资讯范例

包括4个部分，标题，图片，评论/描述，网址，放到一个Word中，按照命名要求，发到邮箱
buptxu@sina.cn
描述/评论必须表达一个完整的含义。

标题：比特币交易所会重启吗？听听周小川怎么说

图片：提供图片一张

描述：当前中国政府对新技术的的态度是谨慎的，应该说是还是比较得当的。

网址：http://www.sohu.com/a/225294468_313170

关于课堂演示

针对实验部分，请同学在课堂做先行演示，展示给大家！！！

请大家报名参与，根据做的情况有最高5分的平时加分（平时100分的前提下）：其中2、5 、6章各需要2个同学；3、7、8章各需要1个同学，4章需要3各同学。

详细演示再各别交流，原则参与同学都需要

详细要求随后会更详细的和大家介绍！

内容安排

- 0 课程介绍
- 1 概述
- 2 软件安全
- 3 漏洞
- 4 软件风险

1.1软件与软件安全

软件，一系列按照特定顺序组织的计算机数据和指令的**集合**。主要划分为三大类：

- **系统软件**为计算机使用提供最基本的功能
- **应用软件**是为了某种特定的用途而被开发的软件
  - 微软的Office软件
  - 数据库管理系统
- **支撑软件**是协助用户开发软件的工具性软件
  - java开发中的jdk套件

软件安全：采用工程的方法使得软件在敌对攻击的情况下仍能够继续正常工作。软件安全威胁来自于两个方面：

- **软件自身**：软件具有的漏洞和缺陷会被不法分子利用
- **外界**：黑客通过编写恶意代码并诱导用户安转运行

1.2软件安全威胁

软件开发中的不同阶段，会遇到不同的安全威胁

- **软件代码编写阶段** 敏感信息暴露等

  例如：将应用的敏感信息不加密就写在发布版本中易于被黑客读取的文件中

- **软件编译阶段** 编译方式固有漏洞

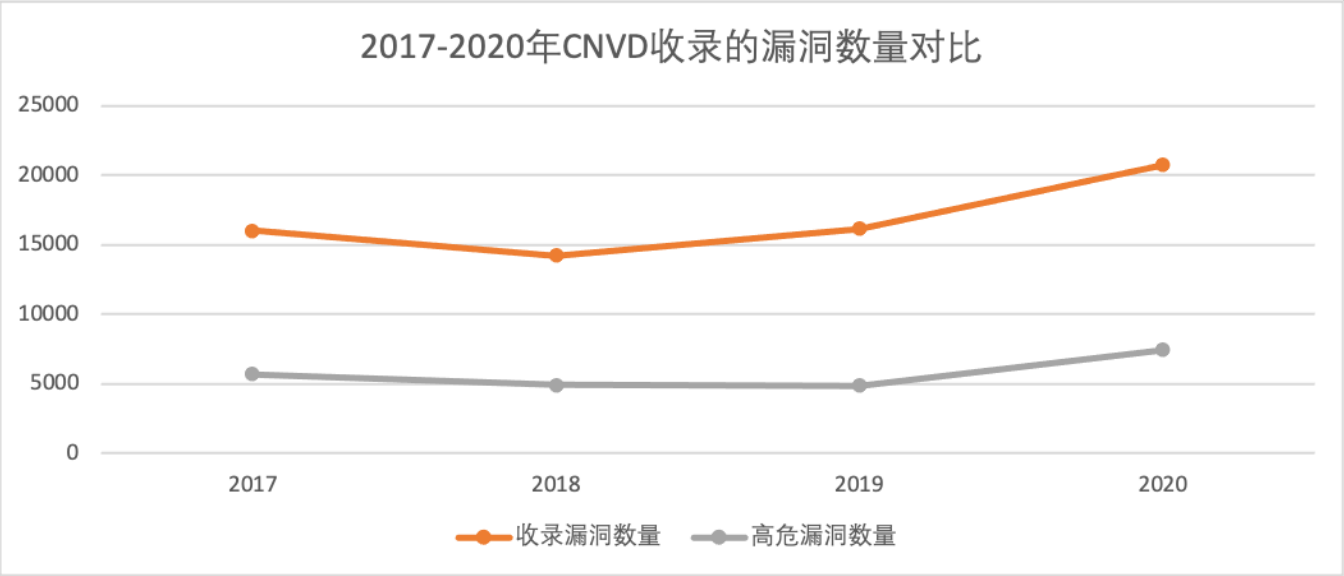
  例如：Android开发中最终编译生成的smali格式文件存在固有的代码注入风险

- **软件签名发布阶段** 签名绕过威胁等

  例如：在Android开发中旧版本的签名方式因为未能对所有软件文件进行校验，导致黑客可以通过修改未被校验到的文件使恶意dex文件注入

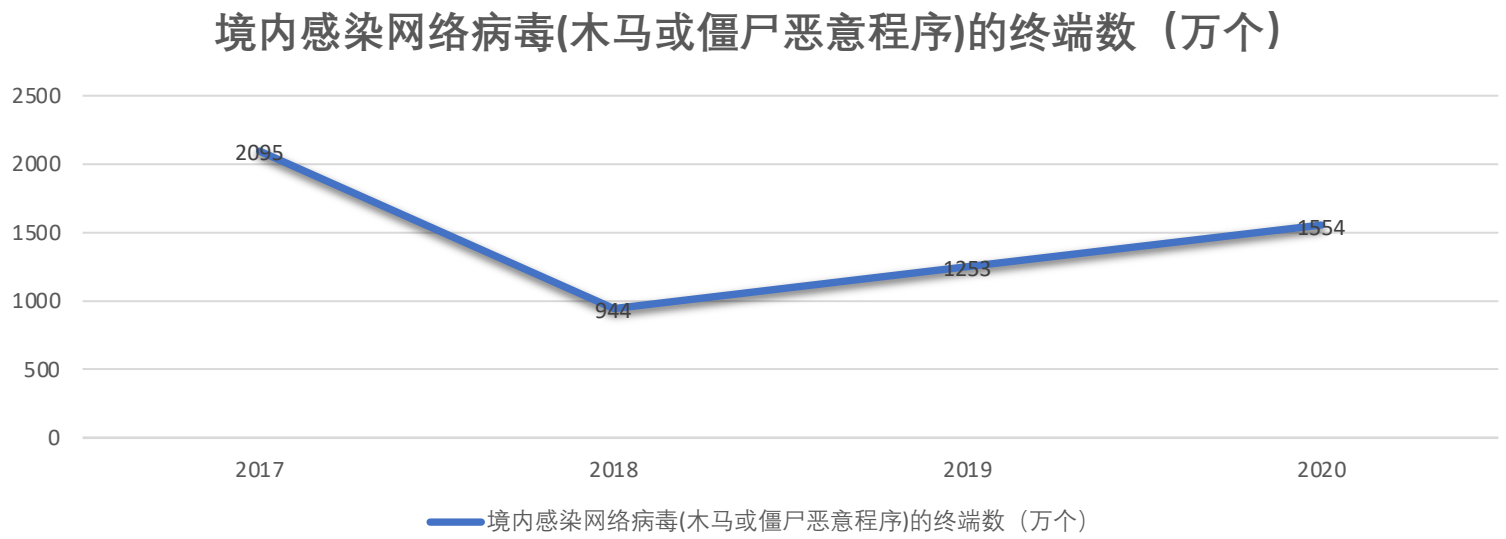
1.2软件安全威胁

在漏洞方面，2020 年，国家信息安全漏洞共享平台（CNVD）共收录通用软硬件漏洞**20721**个，较 2019年**增长 28.2%**。



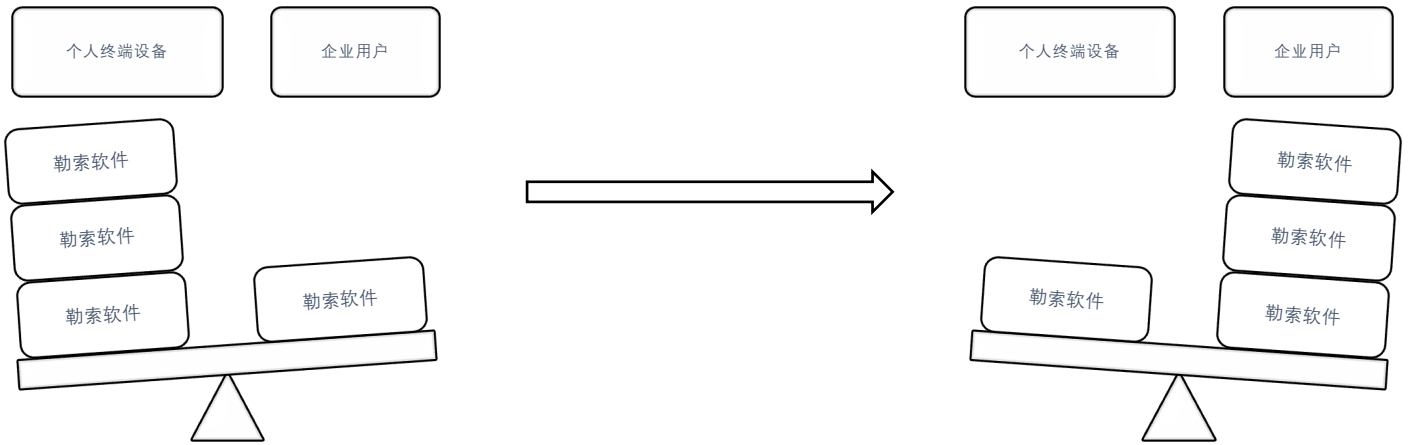
1.2软件安全威胁

2020年境内感染网络病毒(木马或僵尸恶意程序)的终端数约1554万个。



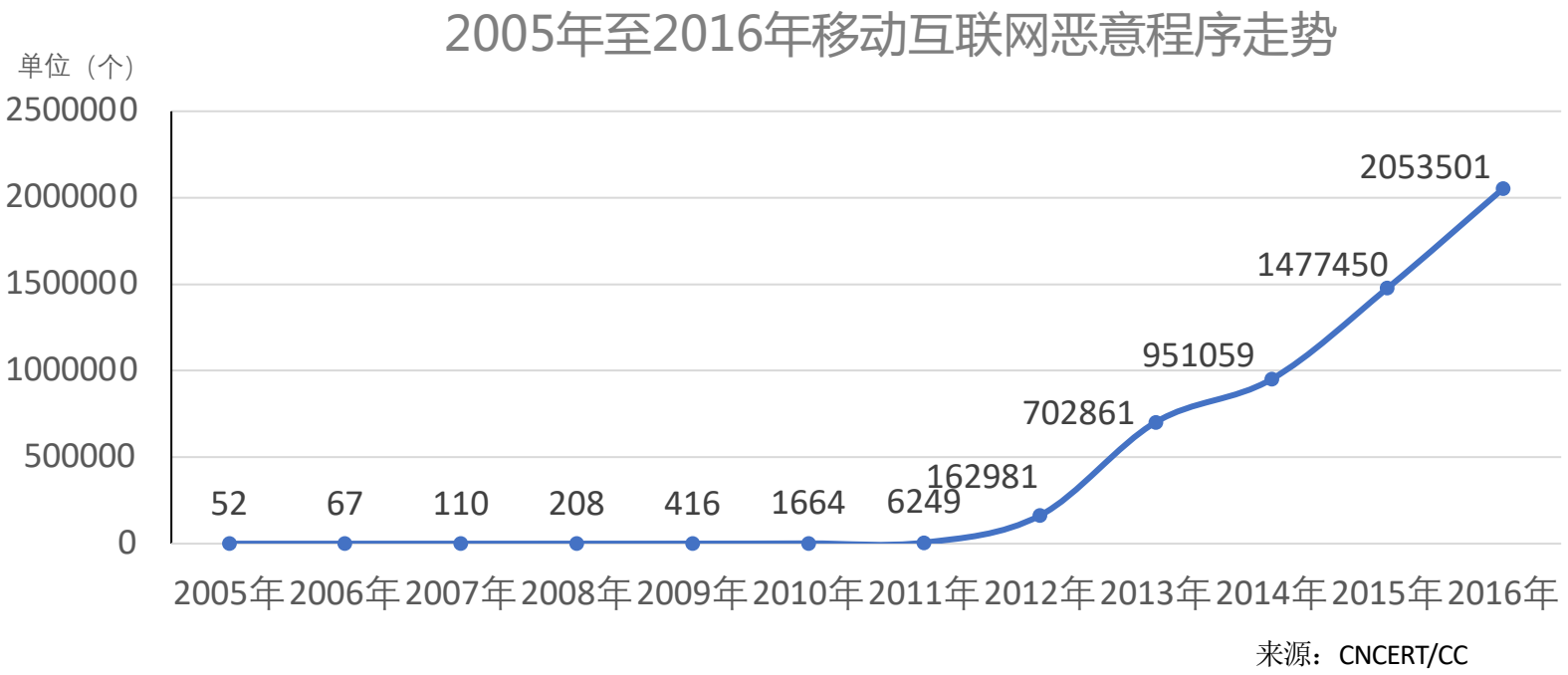
1.2软件安全威胁

勒索软件已逐渐由针对个人终端设备延伸道企业用户



1.2软件安全威胁

互联网恶意程序下载链接近 **67 万**条，较 2015 年增长近 **1.2 倍**，涉及的传播源域名 **22 万**余个，IP 地址 **3 万余**个，恶意程序传播次数达 **1.24 亿**次。



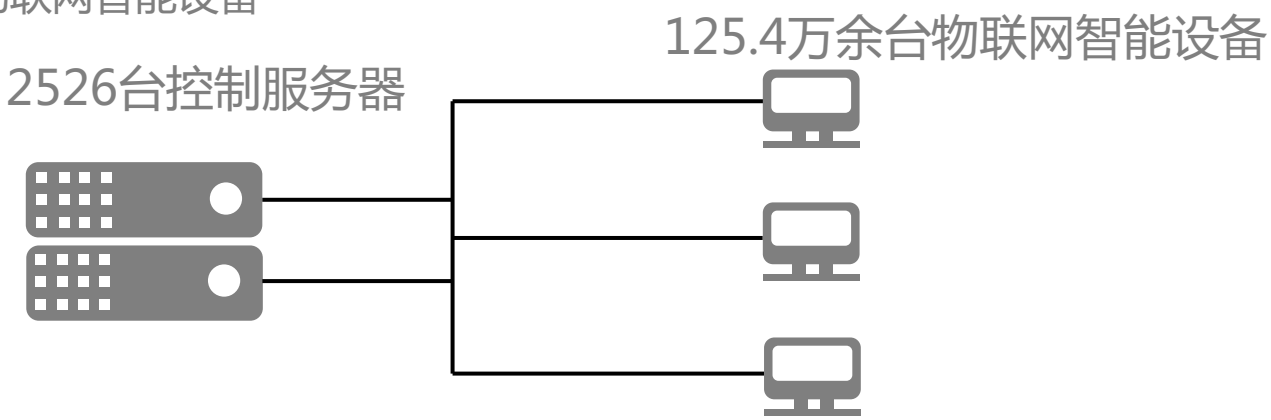
来源：CNCERT/CC

1.2软件安全威胁

数据泄露威胁日益加剧：

- 美国大选候选人希拉里的邮件泄露，直接影响到美国大选的进程
- 2016年我国免疫规划系统网络被恶意入侵，**20 万**儿童信息被窃取并在网上公开售卖

对Mirai 僵尸网络进行抽样监测：截至 2016 年年底，共发现 2526 台控制服务器控制 125.4 万余台物联网智能设备



1.3 软件的核心地位

- 软件是IT的灵魂
  - 硬件发展相对定型，硬件技术软件化
  - 业务的多样性由软件类别、功能的多样性支撑
  - 网络融合靠系统软件的支撑，比如：智能电视等

1.3 软件的核心地位

- 软件技术发生了深刻的变化
  - **软件开发者与用户之间关系发生了深刻的变化**
  - **软件开发、维护模式发生了深刻的变化**
  - **应用软件盈利模式正在发生深刻变化，应用软件媒体化**

内容安排

- 0 课程介绍
- 1 概述
- 2 软件安全
- 3 漏洞
- 4 软件风险

2.1 网络空间安全

- 研究信息获取、信息存储、信息传输和信息处理领域中信息安全保障问题的一门新兴学科
- 主要围绕网络空间中电磁设备、电子信息系统、网络、运行数据、系统应用中所存在的安全问题，开展理论、方法、技术、系统、应用、管理和法制等方面的研究

- 内涵：
  - 保密性：信息不泄漏给非授权的用户、实体或者过程的特性
  - 完整性：数据未经授权不能进行改变的特性，即信息在存储或传输过程中保持不被修改、不被破坏和丢失的特性。
  - 可用性：可被授权实体访问并按需求使用的特性，即当需要时应能存取所需的信息。
  - 真实性：内容的真实性
  - 可核查性：对信息的传播及内容具有控制能力，访问控制即属于可控性。
  - 可靠性：系统可靠性

- 范围
  - 网络安全：网络边界、网络协议等
  - 系统安全：信息系统软硬件等安全（软件安全：操作系统、数据库、应用软件等）
  - 内容安全： 或称为信息安全、数据安全，关注数据自身保护，涉及保密、内容合法性等
  - 信息对抗：为消弱、破坏对方电子信息设备和信息的使用效能，保障己方电子信息设备和信息正常发挥效能而采取的综合技术措施

- 原由
  - 信息系统的复杂性
    - 系统软硬件缺陷，网络协议的缺陷
  - 信息系统的开放性
    - 系统开放：计算机及计算机通信系统是根据行业标准规定的接口建立起来的。
    - 标准开放：网络运行的各层协议是开放的，并且标准的制定也是开放的。
    - 业务开放：用户可以根据需要开发新的业务。
  - 人的因素

- 特点
  - 攻防性：攻防技术交替改进
  - 配角性：安全是属性，安全不能脱离系统、应用或业务，安全是个形容词！
  - 相对性：安全性是相对的，相对主体的需求而言

- 安全业务
  - 本质是责任分解
  - 三分技术+七分管理

2.2 软件安全

- **软件安全关注计算机程序或者程序中所包含信息的完整性、机密性和可用性等**

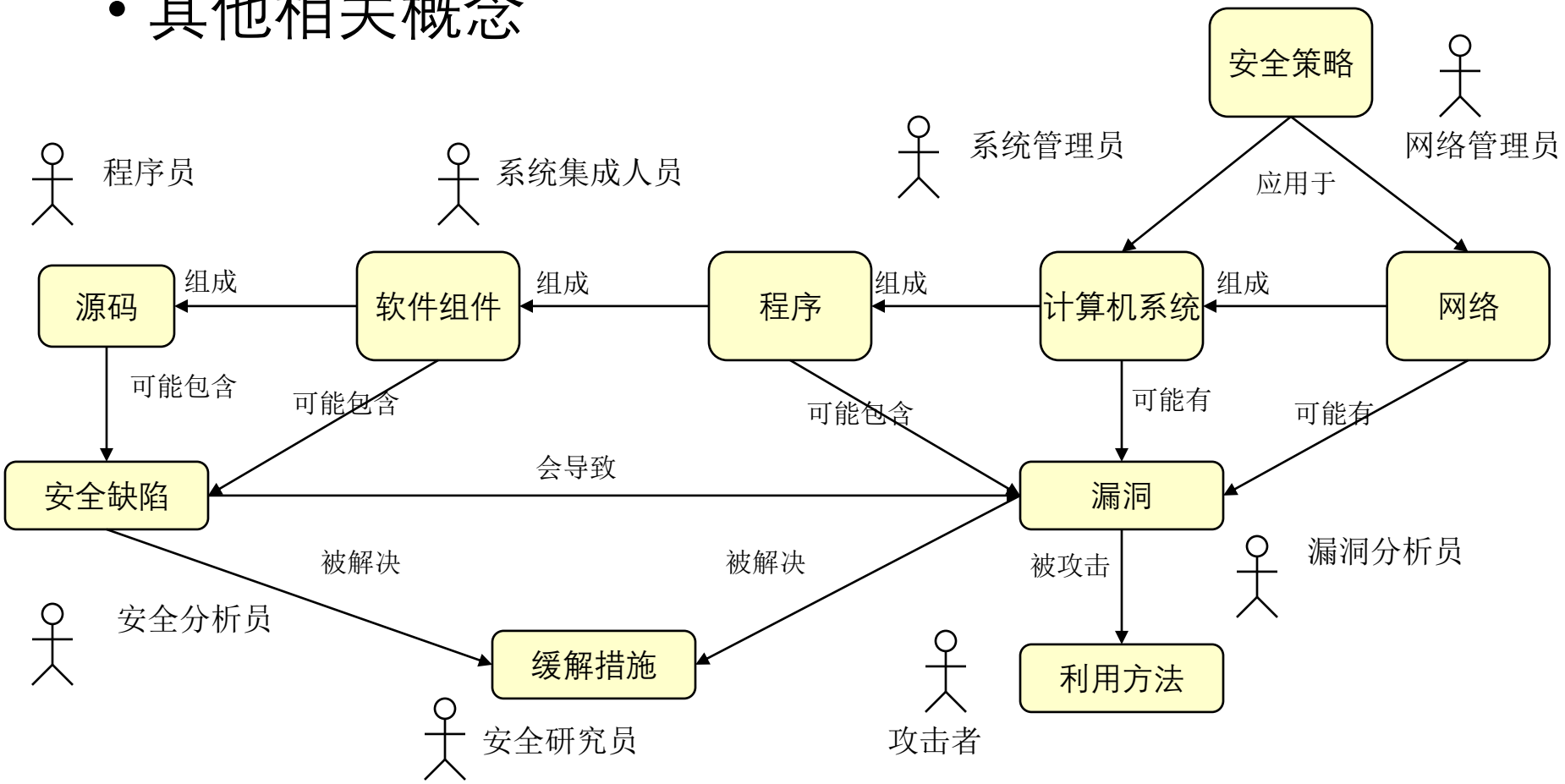
缺陷、威胁和风险

- 软件安全缺陷
  - 软件自身缺陷，设计者故意或过失而导致
  - **软件漏洞**是基本型态、**恶意代码**则是延伸的形态
  - 软件中的客观存在

- 软件安全威胁
  - 外在的因素，过去曾经是或将来会成为攻击来源的个人、团体、组织或外部势力
  - 威胁的例子包括:黑客、内部人员、罪犯、竞争情报从业者、恐怖分子和信息战士等
  - 软件外主观存在

- 软件安全风险
  - 内在的缺陷暴露在外在威胁下的状态即为**风险**
  - 内在的缺陷遭遇外在威胁则形成**安全事件**

- 其他相关概念


安全策略
网络管理员
程序员
系统集成人员
系统管理员
应用于
源码
组成
软件组件
组成
程序
组成
计算机系统
组成
网络
可能包含
可能包含
可能包含
可能有
可能有
会导致
安全缺陷
漏洞
被解决
被解决
被攻击
漏洞分析员
安全分析员
缓解措施
利用方法
安全研究员
攻击者

- 软件安全范围


设计阶段
编码测试阶段
运维阶段
软件生命周期
软件安全功能
防反汇编/反编译
防调试跟踪
防篡改
防盗版
防内存攻击
防API违规调用
防拒绝服务
可追踪
软件环境要素
软件运行环境
软件开发环境
软件开发者
软件自身

2.3 软件安全知识体系

- 软件漏洞
- 恶意代码
- 软件保护

- 软件漏洞
  - 漏洞原理与案例
  - 漏洞挖掘与利用
  - 漏洞检测与防范：安全编码

- 恶意代码
  - 恶意代码机理：传统病毒、宏病毒、木马、蠕虫等
  - 恶意代码检测：检测、消除、预防、免疫、数据备份及恢复、防范策略等

- 软件保护
  - 软件分析（破解）
    - 静态分析：控制流分析、数据流分析、数据依赖分析、别名分析、切片、抽象解析
    - 动态分析：调试、剖分、跟踪(Trace)、代码注入、HOOK、沙箱技术、反反调试

- 软件保护（反破解）
  - **防逆向分析：代码混淆、软件水印、原生代码保护、资源保护、加壳、资源及代码加密**
  - **防动态调试：函数检测、数据检测、符号检测、窗口检测、特征码检测、行为检测、断点检测、功能破坏、行为占用**
  - **运行环境检测、反沙箱**
  - **数据校验：文件校验、内存校验**
  - **代码混淆技术：静态混淆、动态混淆**
  - **软件水印：静态水印、动态水印**

内容安排

- 0 课程介绍
- 1 概述
- 2 软件安全
- 3 漏洞
- 4 软件风险

3.1 典型软件漏洞

- 缓冲区溢出
  - W32.Blaster.Worm
    - 2003年8月11日爆发
    - 在没有用户参与的情况下感染任何一台连接到互联网且未打补丁的计算机系统
    - 至少有800万个Windows系统被蠕虫感染[Lemos 04].
    - 经济损失> $500M

W32.Blaster.Worm

- Blaster:
  - 检测计算机是否已感染
    - 将“windows auto update” = “msblast.exe” 加入注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
  - 启动Windows
  - 产生一个随机的 IP地址
  - 试图感染具有该地址的计算机
  - 利用Windows 2000 或Windows XP上的DCOM RPC漏洞通过TCP 135端口发送数据。

```
1.  error_status_t _RemoteActivation(
2.        ..., WCHAR *pwszObjectName, ... ) {
3.    *phr = GetServerPath(
                 pwszObjectName, &pwszObjectName);
    ...
4.  }

5.  HRESULT GetServerPath(
      WCHAR *pwszPath, WCHAR **pwszServerPath ){
6.    WCHAR *pwszFinalPath = pwszPath;
7.    WCHAR wszMachineName[MAX_COMPUTERNAME_LENGTH_FQDN+1];
8.    hr = GetMachineName(pwszPath, wszMachineName);
9.    *pwszServerPath = pwszFinalPath;
10. }

11. HRESULT GetMachineName(
12.   WCHAR *pwszPath,
13.   WCHAR wszMachineName[MAX_COMPUTERNAME_LENGTH_FQDN+1])
14. {
15.   pwszServerName = wszMachineName;
16.   LPWSTR pwszTemp = pwszPath + 2;
17.   while ( *pwszTemp != L'\\' )
18.     *pwszServerName++ = *pwszTemp++;
      ...
19. }
```

SQL注入漏洞

select * from user where username=‘’ and password=‘’

引号中的就是帐号密码输入框传入的信息。

如果我们传入的信息为 ’ or ‘1’=‘1,那么这个语句就会变成

select * from user where username=‘’ or ‘1’=‘1’ and password=‘’ or ‘1’=‘1’

3.2 软件漏洞的危害

- 无法正常使用
- 引发恶性事件
- 关键数据丢失
- 秘密信息泄漏
- 被安装木马病毒

无法正常使用

对于软件安全漏洞来说，最直观的表现就是造成软件无法正常使用。

以win98为例，该操作系统曾经出现过再处理过多ping数据包时出现蓝屏的拒绝服务漏洞。

当我们用一台计算机连续不断地发送ping数据包到一台Windows 98系统的计算机时，该计算机就马上出现系统蓝屏死机的现象。

引发恶性事件

如果一个航天飞机上使用的导航控制软件出现了问题，在运行中错误地计算了飞机飞行的轨道，那么航天飞机上的宇航员就可能面临生命的危险。

计算机软件还可能被使用在注入国家安全局这样的情报部门，当一个软件发生信息泄露漏洞时，就会造成大量的国家绝密信息被泄漏，一旦敌对国家或者组织获得这些信息，那就可能造成一个国家陷入到极其可怕的境地中。

关键数据丢失

前几年，某杀毒软件曾将系统关键文件当作病毒进行了彻底删除，造成了用户的计算机系统无法正常启动，很多数据信息因此丢失，虽然引发这场危机的罪魁祸首是软件的误删除错误，而不是一个安全漏洞，但是我们能够看到软件出现问题后所造成的影响。

一些软件出现漏洞后，可能被恶意攻击者利用而进行对用户关键数据的删除，从而造成用户数据的丢失。有一些漏洞甚至会直接造成对用户数据的清除。

秘密信息泄漏

对于信息泄漏漏洞来说，用户的计算机系统等于是赤裸裸地摆在了别人的面前，里面的信息都能被其他人随意获取、查看。

对于个人是这样，对于单位、政府部门会造成更大的影响。一些敏感的有关国家军事安全、企业核心经济利益的信息，都有可能被一个软件的安全漏洞所泄漏，造成无法预知的后果。

被安装木马病毒

一个软件漏洞被发现后，最常见的利用就是恶意攻击者借此来散播木马病毒程序。我们的计算机系统中也许安装有最新的杀毒软件，最好的防火墙，但是只要计算机系统上的软件存在漏洞，那么恶意攻击者就能够在你的计算机系统上安装木马病毒程序。

3.2 软件漏洞出现的原因

- 小作坊式的软件开发
- 赶进度带来的弊端
- 被轻视的软件安全测试
- 淡薄的安全思想
- 不完善的安全维护

小作坊式的软件开发

严格地讲，任何一款计算机软件都必须依据软件工程的思想来进行设计开发。

但是，出于种种原因，很多软件的开发并没有按照软件工程的要求来实现。

有些小型公司采用了直接开发，或者边设计边开发的方法，这样只做出来的软件犹如小作坊里生产出来的产品，难免存在很多漏洞。

赶进度带来的弊端

很多大型的软件公司即使采用了软件工程思想来设计软件，但是由于事件紧迫，任务繁重，也会在一定程度上采用投机取巧或者省工省料的办法来开发软件。这个时候开发出来的软件，往往由于开发者过于疲劳或者赶进度，从而将不安全的因素带进软件，造成软件存在漏洞。

被轻视的安全测试

软件安全测试不但可以进行对软件代码的安全测试，还可以对软件成品进行安全测试。但是这样会增加成本，于是一些开发商要么不做，要么仅做最低级的测试。

这样只能保证软件能够正常使用，基本功能实现，其实，漏洞就这样被隐藏在软件内部。

淡薄的安全思想

软件安全思想主要是针对软件开发中最幸苦的编程人员来说的。由于编程人员对软件安全的认识并不一样，在做产品开发时，他们编写的代码也就对软件的安全出现了不同程度的影响。如果一个编程人员不具有一些基本的安全编程经验，他可能就会把最简单最常见的安全漏洞引入到软件内部。

不完善的安全维护

当软件出现安全漏洞时，这并不可怕。软件的开发商只需要认真检查软件出现漏洞的原因，找出修补方案就可以弥补漏洞带来的危害与损失。

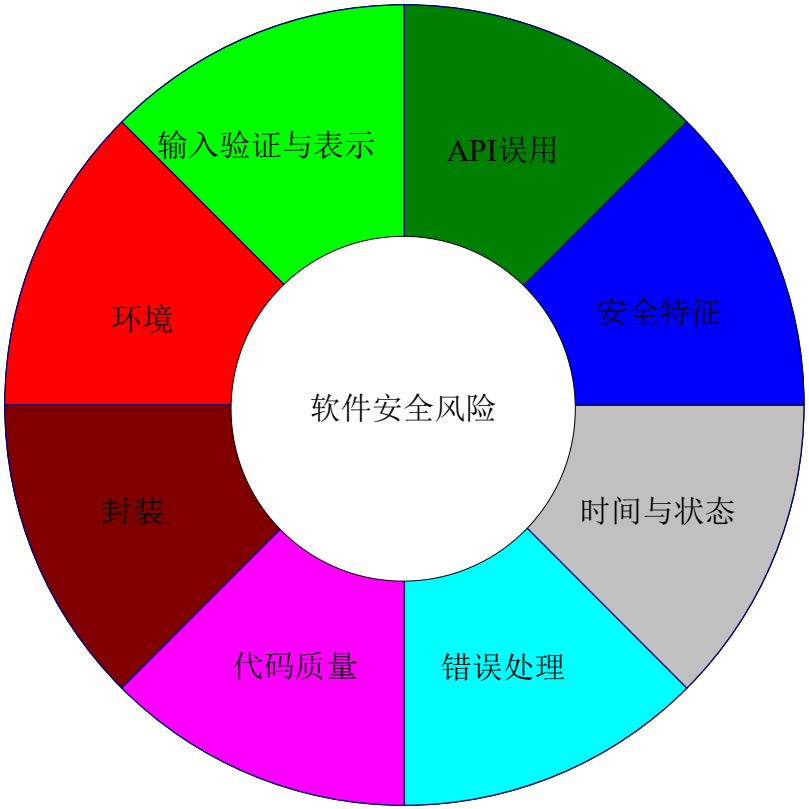
可是有些开发商知道软件出现漏洞也不修补甚至欺骗用户。这种情况下，软件就会一直存在漏洞，那么使用这样软件的用户就会面临着危险

内容安排

- 0 课程介绍
- 1 概述
- 2 软件安全
- 3 漏洞
- 4 软件风险

GaryMcgraw软件漏洞分类

- 1 输入验证与表示
- 2 API误用
- 3 安全特征
- 4 时间与状态
- 5 错误处理
- 6 代码质量
- 7 封装
- 8 环境



输入验证与表示

- ✓输入验证和表示问题通常是由特殊字符、编码和数字表示所引起的，这类安全问题的发生是对输入的信任所造成的。
- ✓**一些较严重的安全问题往往都是由于对输入的信息过度信任造成的，主要问题包括：**

缓冲区溢出（Buffer Overflow）	• 对所分配的内存块之外的内存进行写操作会覆盖数据，使程序崩溃，甚至导致执行恶意代码；
命令注入（Command Injection）	• 在没有指定完整路径的情况下执行命令可能使得攻击者可以通过改变$PATH或者其他环境变量来执行恶意代码；
跨站脚本（Cross-Site Scripting）	• 向浏览器发送非法的数据会使浏览器执行恶意代码,通过XML编码进行身份认证也会导致浏览器执行非法代码；

输入验证与表示

格式化字符串（Format String）	• 允许攻击者控制一个函数的格式化字符串，可能会引起缓冲区溢出；
HTTP应答截断（HTTP Response Splitting）	• 在在HTTP响应头中包含未经验证的数据将可能导致缓存中毒、跨站脚本、跨用户攻击或者页面劫持攻击
非法指针值（Illegal Pointer Value）	• 函数可能返回指向缓冲区外边的内存，接下来对该指针的操作有可能造成缓冲区溢出；
整数溢出（Integer Overflow）	• 不考虑整数溢出会造成逻辑错误或缓冲区溢出；
日志伪造（Log Forging）	• 写未验证数据到日记文件让攻击者伪造日记或注入恶意的内容
目录游历（Path Traversal）	• 允许用户通过输入来控制文件系统操作使用的路径，会使攻击者有机会访问或修改被保护的系统资源；

输入验证与表示

进程控制（Process Control）	• 装载来自于非信任源或者非信任环境的库可能会导致应用程序执行攻击者的恶意代码；
资源注入（Resource Injection）	• 允许用户输入控制资源ID可能会使攻击者访问或修改系统保护的资源；
配置操纵（Setting Manipulation）	• 允许外部控制系统设置有可能会导致服务崩溃或应用程序进行非法操作；
SQL注入（SQL Injection）	• 允许用户输入来构造一个动态的SQL请求，将有可能让攻击者控制SQL语句的意思，执行任意的SQL指令；
字符串结束错误（String Termination Error）	• 依赖于字符串终止符有可能造成缓冲区溢出；
未检查的返回值（Unchecked Return Value）	• 没有检查方法的返回值,可能会导致程序进入不可预料的状态。

API误用

- ✓API是调用者与被调用者之间的一个约定，大多数的API误用是由于调用者没有理解约定的目的所造成的。
- ✓举个例子，如果一个程序在调用chroot()后调用chdir()失败，那是因为它违背了约定所指定的如何在安全模式下改变动态root目录。

API误用

危险函数（Dangerous Function）	不能被安全使用的函数不应该使用；
目录限制（Directory Restriction）	不合理的使用系统调用chroot()可能使攻击者逃脱chroot的限制
堆检查（Heap Inspection）	堆检查经常出现类似密码等敏感信息因为没有及时从内存中移走从而被黑客利用。所以不应该使用realloc()重新分配藏有敏感信息的缓冲区；
认证误用（Often Misused: Authentication）	攻击者经常伪造DNS进行攻击；
异常处理误用（Often Misused: Exception Handling）	_alloca()函数会抛出stack overflow exception，从而会使程序当掉；
路径操纵误用（Often Misused: Path Manipulation）	传送一个大小不够的输出缓冲区给路径操纵函数会导致缓冲区溢出；

API误用

- API误用包括以下方面:

权限管理误用（Often Misused: Privilege Management）	未能坚持最小权限原则会增大其他弱点出现的风险；
字符串操纵（Often Misused: String Manipulation）	在进行多字节和unicode字符串转换时容易发生缓冲区溢出；
未检查的返回值（Unchecked Return Value）	不考虑方法的返回值会导致程序出现意外的状况。

安全特征

- 安全特征主要指认证，访问控制，机密性，密码，权限管理等方面的内容：

不安全随机数（Insecure Randomness）	标准的伪随机数生成器不能抵抗加密攻击；
最小权限违规（Least Privilege Violation）	程序在运行到需要将权限提高的函数时将权限提高，当不需要时应立即将权限降到最低
缺少访问控制（Missing Access Control）	程序未能在所有可能的执行路径执行访问控制检查
口令管理（Password Management）	用明文存储密码或者口令为空将导致系统的不安全。在HTTP重定向报文发送密码会导致密码泄露
隐私违规（Privacy Violation）	对私有信息的不恰当处理会给用户带来损害。

时间与状态

- ✓分布式计算是与时间和状态有关的。也就是说，为了让多个部件之间交互，需要状态共享，而这要花费时间。
- ✓ 大多数程序员将他们的工作人格化。他们认为控制器的一个线程会按照他们所想的方式来执行整个程序。然而，现在的计算机能在多任务之间快速切换，采用多核、多CPU或者说分布式系统，两个事件甚至能在同一时间发生。
- ✓这样一来，在程序员所想的程序执行模式和实际发生的情况之间就会产生问题。这些问题可能涉及到线程、进程、时间和信息之间的非法交互。这些交互通过共享的状态产生：如信号量、变量、文件系统以及任何能够存储信息的东西。

时间与状态

- 包括以下方面:

死锁（Deadlock）	不一致的程序锁定会导致死锁；
固定会话（Session Fixation）	通过认证后使用同一个会话会导致攻击者劫持已认证过的会话
文件访问竞争条件TOCTOU（File Access Race Condition: TOCTOU）	文件属性被验证和文件被使用之间的时间差可以引起文件使用权限提升的攻击
不安全临时文件（Insecure Temporary File）	创建和使用临时文件可能会导致应用程序和系统受到攻击；

错误处理

- ✓错误和错误处理代表了一类API。与错误处理有关的错误是很常见的。与API误用相比，和错误处理相关的安全漏洞一般是两种方式造成的：
  - ✓第一种是根本忘记处理错误或者只是简单的处理，并没有彻底解决
  - ✓第二种则是程序对可能的攻击者泄露了过多的信息或者涉及面太广没有人愿意去处理这些问题。

错误处理

- 包括以下方面:

捕获空指针异常（Catch NullPointerException）	捕获一个空指针异常不是一个有效的防止空指针发生的办法
空的捕获（Empty Catch Block）	忽视异常以及其他错误可能会导致攻击者引发意想不到的行为
过度捕获异常（Overly Broad Catch Block）	捕获过多的异常会使错误处理代码过度复杂，使代码更可能携带安全弱点

代码质量

- 低劣的代码质量会导致不可预测的行为。从用户的角度来看，这通常会表现为低劣的可用性。对于攻击者而言，低劣的代码使他们可以以意想不到的方式威胁系统：

双重释放（Double Free）	对同一个内存地址调用2次free()会导致缓冲区溢出
内存泄露（Memory Leak）	内存分配却没有释放，会致资源枯竭
空指针调用（Null Dereference）	程序使用一个空指针引用时，就会抛出NullPointerException
已废弃（Obsolete）	尽量不要使用已经废弃的函数；
未定义行为（Undefined Behavior）	函数行为未定义，直到其控制参数被设置为特定的值；
未初始化变量（Uninitialized Variable）	程序可能在变量没初始化之前使用它
未释放资源（Unreleased Resource）	程序有时可能不能释放系统资源；
释放后使用（Use After Free）	在内存已被释放后使用可能会导致程序崩溃；

封装

- 封装就是划定强力的分界线。在web浏览器中，这就意味着你的代码模块不能被其他代码模块滥用。在服务端，这意味着要区分校验过的数据和未经校验的数据，区分不同用户的数据，或者区分用户能看到的和不能看到的数据。

通过名字比较类	通过名字比较类可能会对导致程序将两个不同的类认为是相同的
用户间数据泄露	通过不同的会话访问同一个对象里的变量，数据可能被泄露。例如servlet以及通过共享池保存的对象
残留的debug代码	Debug代码可能会无意识的在应用程序中创建一些入口点；
系统信息泄露	系统数据和调试信息的泄露会给攻击者对系统发动攻击机会；
信任边界违规	将可信的以及不可信的数据混杂在同一个数据结构中可能会导致程序员错误地信任未验证的数据。

环境

✓环境包括的内容虽然是源代码之外的，但它们对产品的安全性仍然至关重要。因为其覆盖的内容并不与源代码直接相关，所以我们将它与其他内容分隔开来。

✓环境中可能存在风险例如:

- 不安全的编译器优化(Insecure Compiler Optimization)
- 不安全的传输(J2EE Misconfiguration: Insecure Transport)
- 弱访问许可(J2EE Misconfiguration: Weak Access Permissions)
- 配置文件中的密码(Password Management: Password in Configuration File)

本讲小结

- 什么是软件安全
  - 软件安全知识体系
- 什么是软件漏洞
  - 软件漏洞分类

谢谢大家！

谢谢大家