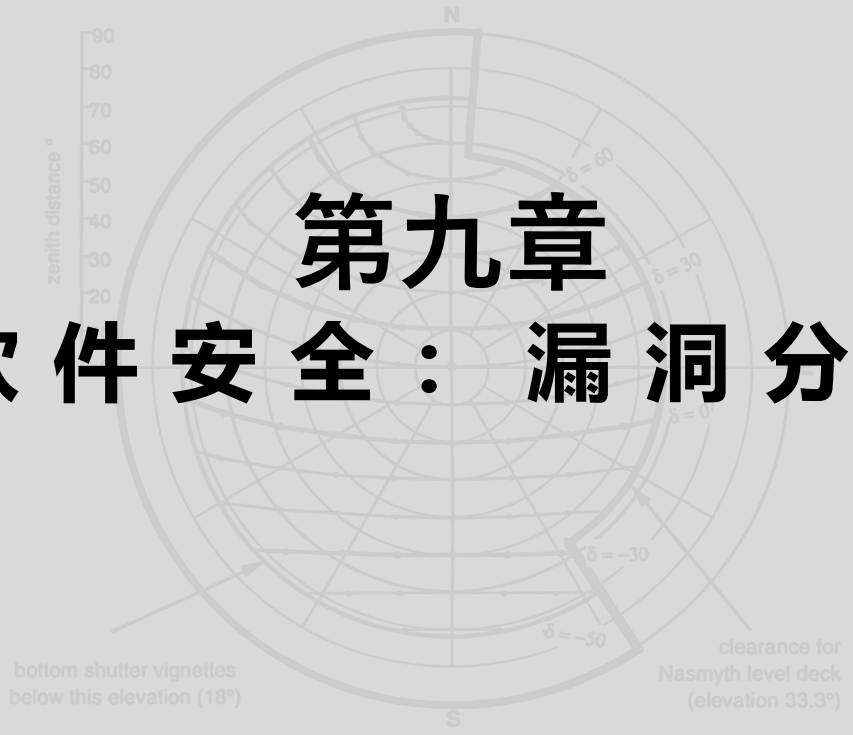
Computer System Security **CS3312**

计算机系统安全

2024年 春季学期

主讲教师：张媛媛 副教授

上海交通大学 计算机科学与技术系



第九章
软件安全：漏洞分类

(软件)漏洞的定义

IETF：漏洞是在系统设计、完成、操作、管理中出现的缺陷或弱点，能够被利用以违反系统的安全策略。

ENISA：漏洞是设计或实施中存在的弱点或错误，可以导致计算机系统、网络或协议等安全目标出现非预期的结果。

ISO/IEC27005：漏洞是一个或一组资产的弱点，通过利用它可以威胁到拥有该资产的机构的事务运营和持续性，包括支持机构计划的信息资源。

NIST IR 7298：漏洞是威胁源可以攻击或触发的信息系统、系统安全流程、内部控制或实施中的脆弱点。

NSTISSI(CNSS)：漏洞是可以被利用的信息系统、系统安全流程、内部控制或实施中存在的脆弱点。

Information Security Dictionary：漏洞是系统安全流程、系统设计、实施、内部控制等过程中的脆弱点，这些脆弱点可以被攻击以违反系统安全策略。

CVE：
漏洞是软件中的错误，能够被攻击者利用而获得对系统或网络的访问权，具体形式包括，攻击者能以另一用户身份执行命令，能够访问已经制定了严格访问规则的数据，能够假冒另一用户的身份或者能够实施拒绝式服务攻击。

OWASP：
漏洞是应用中的弱点，可能是设计或完成过程中引入的bug，可以被攻击者用来对应用的相关用户实施攻击，从而造成损失。

提取关键字	
成因	脆弱点、缺陷、**错误**
介质	攻击者利用
后果	破坏安全策略、取得控制权、攫取数据、拒绝服务

参见第五章
一个漏洞攻击的案例

软件漏洞分类

CWE对所有漏洞行为的抽象描述和分类,
适用于软件、硬件等各类系统漏洞：

1. Improper Access Control - (284)
2. Improper Interaction Between Multiple Correctly-Behaving Entities - (435)
3. Improper Control of a Resource Through its Lifetime - (664)
4. Incorrect Calculation - (682)
5. Insufficient Control Flow Management - (691)
6. Protection Mechanism Failure - (693)
7. Incorrect Comparison - (697)
8. Improper Check or Handling of Exceptional Conditions - (703)
9. Improper Neutralization - (707)
10. Improper Adherence to Coding Standards - (710)

1. 不当的访问控制

该软件不会限制或错误地限制未经授权的参与者对资源的访问。

访问控制涉及使用多种保护机制，例如：

- 身份验证（证明参与者的身份）
- 授权（确保给定的参与者可以访问资源），以及
- 问责制（跟踪已执行的活动）

当上述机制未得到适当应用或失败时，攻击者可以通过获取权限、读取敏感信息、执行命令、逃避检测等方式危及软件的安全性。

例如：

- Use of Hard-coded Password (259)

CVE-2010-4624：
MyBB (aka MyBulletinBoard) before 1.4.12 allows remote authenticated users to bypass intended restrictions on the number of [img] MyCodes by editing a post after it has been created.

2. 行为正确的实体之间的不当交互

当两个实体在彼此独立运行时具有正确的行为时，就会发生交互错误，但是当它们作为组件集成到更大的系统或流程中时，它们会引入可能导致弱点的错误行为。

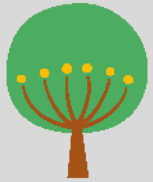
当一个系统或流程组合了多个独立的组件时，这通常会在系统级别产生新的、紧急的行为。但是，如果这些组件之间的交互没有得到充分考虑，一些紧急行为可能是不正确的，甚至是不安全的。

例如：

- Use of Incorrect Byte Ordering (198)

软件从上游组件接收输入，但在处理输入时不考虑字节顺序（例如大端和小端），导致使用不正确的数字或值。

3. 生命周期内对资源的不当控制



该软件在其创建、使用和发布的整个生命周期内不维护或不正确地维护对资源的控制。

资源通常有关于如何创建、使用和销毁的明确说明。
如果软件不遵循这些说明，则可能导致意外行为和潜在可利用状态。
即使没有明确的指示，也希望遵守各种原则，例如“在对象创建完成之前不要使用它”或“在对象被销毁后不要使用它”。

例如：

- Double Free (415)
- Use After Free (416)

4. 不正确的计算

该软件执行的计算会生成不正确或意外的结果，这些结果稍后会用于安全关键决策或资源管理。

当软件不正确地执行安全关键计算时，可能会导致资源分配不正确、权限分配不正确或比较失败等。不正确计算的许多直接结果可能导致更大的问题，例如保护机制失败甚至任意代码执行。

例如：

- Integer Overflow (190)
- Divide By Zero (369)

5. 控制流管理不足

代码在执行期间没有充分管理其控制流，从而创造了可以以意想不到的方式修改控制流的条件。

例如：

- Time-of-check Time-of-use (TOCTOU)
- Race Condition (367)

6. 保护机制失效

产品未使用或错误地使用保护机制，以充分防御针对产品的定向攻击。

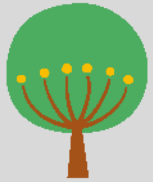
这个弱点涵盖了三种不同的情况：

- 当应用程序没有定义任何针对某一类攻击的机制时，就会出现“缺失”的保护机制。
- “不足”的保护机制可能会提供一些防御措施——例如，针对最常见的攻击——但它并不能针对预期的一切提供保护。
- 最后，当某种机制在产品中可用且正在积极使用时，会出现“忽略”机制，但开发人员尚未在某些代码路径中应用它。

例如：

- Use of Hard-coded Cryptographic Key (321)
- Public Key Re-Use for Signing both Debug and Production Code (1291)

7. 不正确的比较



该软件在与安全相关的上下文中比较两个实体，但比较不正确，这可能会导致漏洞。

这个弱点类涵盖了几种可能性：

- 比较过程中，错误地检查了一个因素；
- 应考虑多个因素，但根本不检查其中一些因素；
- 检查错误的因素。

例如：

- Comparison of Incompatible Types (1024)