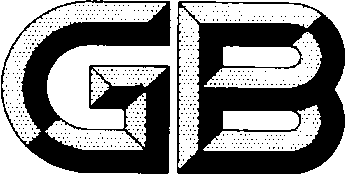
ICS 点击此处添加ICS号

点击此处添加中国标准文献分类号



中华人民共和国国家标准

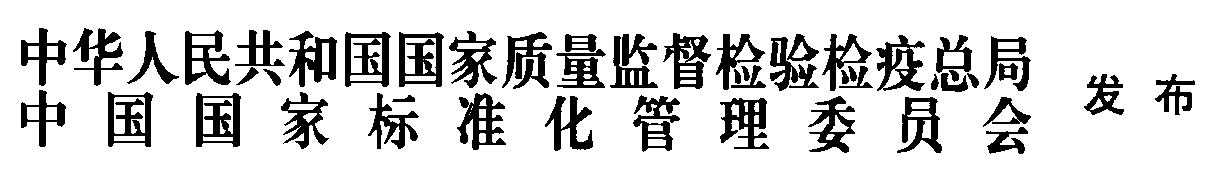
信息安全技术 应用软件安全编程指南

Secure Coding Guide for Application Software

|  |
| --- |
| (草案) |
| （本稿完成日期：2018年6月） |

XXXX-XX-XX发布

XXXX-XX-XX实施



GB/T XXXX—XXXX

场

目次

[目次 2](#_Toc515951488)

[前  言 3](#_Toc515951489)

[1 范围 4](#_Toc515951490)

[2 规范性引用文件 4](#_Toc515951491)

[3 术语、定义和缩略语 4](#_Toc515951492)

[3.1 术语和定义 4](#_Toc515951493)

[3.2 缩略语 7](#_Toc515951494)

[4 概述 8](#_Toc515951495)

[5 安全功能实现要求 8](#_Toc515951496)

[5.1 数据清洗 8](#_Toc515951497)

[5.2 数据加密与保护 10](#_Toc515951498)

[5.3 访问控制 11](#_Toc515951499)

[6 代码实现安全要求 14](#_Toc515951500)

[6.1 面对对象程序安全 14](#_Toc515951501)

[6.2 并发程序安全 16](#_Toc515951502)

[6.3 异常处理安全 17](#_Toc515951503)

[6.4 日志安全 17](#_Toc515951504)

[7 资源使用安全要求 18](#_Toc515951505)

[7.1 资源管理 18](#_Toc515951506)

[7.2 内存管理 19](#_Toc515951507)

[7.3 数据库管理 20](#_Toc515951508)

[7.4 文件管理 20](#_Toc515951509)

[7.5 网络传输 21](#_Toc515951510)

[8 配置管理安全要求 22](#_Toc515951511)

[8.1 构建安全的编译环境 22](#_Toc515951512)

[8.2 及时校验版本稳定性和安全性 22](#_Toc515951513)

[8.3 关闭目录自动列表功能 22](#_Toc515951514)

[8.4 应用程序的安全配置信息应当可读 22](#_Toc515951515)

[8.5 隔离开发环境和实际运行环境 22](#_Toc515951516)

[8.6 推荐使用软件变更管理系统 23](#_Toc515951517)

[参考文献 24](#_Toc515951518)

前  言

本标准依据GB/T 1.1-2009给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由全国信息安全标准化技术委员会(SAC/TC 260)提出并归口。

本标准起草单位：国家计算机网络应急技术处理协调中心、北京邮电大学、北京奇虎测腾安全技术有限公司。

本标准主要起草人：舒敏、王博、吴倩、黄元飞、张家旺、林星辰、陈禹、王鹏翩、李燕伟、高强、杨鹏、陈亮、范乐君、张淼、徐国爱、郭燕慧、李祺、杨昕雨、王晨宇、黄永刚、韩建、章磊。

信息安全技术应用软件安全编程指南

# 范围

本标准对应用软件安全编程进行规范和指导，帮助应用软件开发者在编程开发阶段尽可能考虑安全要素，以提升软件的安全性，避免在软件发布后由安全问题造成的重大损失。

本标准主要针对PC或(和)服务器架构的应用软件，从安全意识和设计的角度，提出通用的、不局限于特定编程语言的安全编程规范，主要针对软件开发人员的编程过程进行约束，包括软件开发人员在编程初期的环境搭建、系统配置、系统代码架构及程序的具体实现等过程。本标准所讨论的应用软件与GB/T 28452-2012中的应用软件范围一致，为针对特定应用开发的业务处理软件。本标准描述对各个应用领域的应用软件普遍适用的安全编程要求，特定领域的应用软件为满足具体业务的安全性需求，应采取的特定安全编程措施不在本标准的讨论范围之内。

# 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 25069-2010 信息安全技术术语

GB/T 28452-2012 信息安全技术 应用软件系统通用安全技术要求

GB/T 5271.8-2001 信息系统 词汇 第8部分：安全（ISO/IEC 2382-8-1998，EQV）

GB/T 16264.8-2005 信息技术 开放系统互连 目录 公钥和属性证书框架（ISO/IEC 9594-8-2001，EQV）

# 术语、定义和缩略语

## 术语和定义

### 应用软件 Application Software

信息系统的重要组成部分，是指信息系统中对特定业务进行处理的软件系统。

### 缓冲区溢出 Buffer Overflow

当应用程序向为某特定数据结构分配的内存空间边界之外写入数据时，即会发生缓冲区溢出。

注：缓冲区溢出被用来作为一种软件系统的攻击手段，通过往程序的缓冲区写入超出其长度的内容，造成缓冲区溢出，从而破坏程序堆栈，使程序转而执行其它指令，以达到攻击的目的。

### SQL注入 SQL Injection

当应用程序将用户输入的内容拼接到SQL语句中，一起提交给数据库执行时，就会产生SQL注入威胁。

### 跨站脚本攻击Cross Site Scripting

跨站脚本攻击是指攻击者利用网站程序对用户输入过滤不足的漏洞，输入可以显示在页面上对其他用户造成影响的HTML代码，从而盗取用户资料、利用用户身份进行某种动作或者向访问者散播恶意代码的一种攻击方式。

### 强类型Strongly-typed

为所有变量指定数据类型称为“强类型”。

### 口令Password

口令是指用于身份鉴别的秘密的字、短语、数或字符序列。

### 应用软件日志Application Software Log

应用软件日志是用于记录系统操作事件的记录文件和文件集合。

### 线程安全Thread Safe

线程安全是指某个函数、函数库在多线程环境中被调用时能够正确地处理多个线程之间的共享变量，使程序功能正确执行。

### 线程同步Thread Synchronization

线程同步是指多个线程通过特定手段来控制线程之间执行顺序的一种机制。注：例如，当有一个线程在对内存进行操作时，其它线程就不能对该内存地址执行操作，直到该线程操作完成，此时，其它线程被设置处于等待状态。

### 死锁Deadlock

死锁是指两个或两个以上的进程在执行过程中，因竞争资源或因彼此通信而造成的一种阻塞现象，此时这些相互等待的进程称为死锁进程，该系统处于死锁状态或者产生了死锁。

### 阻塞Block

阻塞即进程/线程的请求不能立即得到应答，进程/线程需要等待。造成阻塞的原因主要有：等待I/O、2）进程处于sleep中、3）等待解锁。

### 游标cursor

游标是一种用于操纵数据库查询返回的多行结果集的机制。

### 敏感信息Sensitive Information

由权威机构确定的必须受保护的信息，该信息的泄露、修改、破坏或丢失会对人或事产生可预知的损害。

### 秘密信息Secret Information

为了执行特定安全功能策略，只能由授权用户或被评对象安全功能知晓的信息。

### 信任边界Trust Boundary

通常，一个信任边界由编程人员直接控制的系统部件组成。编程人员直接控制的系统以外的所有连接和数据，包括所有其它客户端和系统，在允许进一步系统交互之前，应当被认为是不可信的，且需在边界被验证。

### 可信系统Trust System

指一种封闭、可控的计算系统，其任何操作行为总是和预期保持一致。

### 最小特权Minimum Privilege

主体的访问权限制到最低程度，即仅提供授权任务所必需的那些权利。

### 挂起

挂起进程在操作系统中可以定义为暂时被淘汰出内存的进程。

注：机器的资源是有限的，在资源不足的情况下，操作系统对在内存中的程序进行合理的安排，其中有的进程被暂时调离出内存，当条件允许的时候，会被操作系统再次调回内存，重新进入等待被执行的状态即就绪态，系统在超过一定的时间没有任何动作。

### 异常

异常是导致程序中断运行的一种指令流。如果不对异常进行正确的处理，则可能导致程序的中断执行。

### 错误

错误是系统运行中出现的非预期问题，可能导致系统崩溃或者暂停运行。

### Cookie

储存在用户本地终端上的数据。

### 密钥交换

通信双方在正式通信前，通过密码技术形成一个安全的会话密钥的过程。

### 数字签名

使用发送者的私钥加密摘要信息后形成的一串序列。

### 散列函数

把任意长度的输入变换成固定长度的输出的函数。

### 伪随机数

用确定性的算法计算出来似来自[0,1]均匀分布的随机数序列称为伪随机数。

### 消息摘要

一个唯一对应一个消息或文本的固定长度的值。

### 随机盐

## 缩略语

下列缩略语适用于本标准。

SSL：Secure Sockets Layer安全套接子层

TLS：Transport Layer Security传输层安全

SQL：Structured Query Language 结构化查询语言

HTTP：HyperText Transfer Protocol超级文本传输协议

XML：Extensible Markup Language 可扩展标记语言

LDAP：Lightweight Directory Access Protocol 轻量目录访问协议

PC：Personal Computer 个人电脑

UTF-8： 8-bit Unicode Transformation Format 针对Unicode的可变长度字符编码

# 概述

本指南从程序安全和环境安全两个方面规定应用软件的安全编程要求。其中，程序安全部分描述软件在资源使用、代码实现、安全功能方面的安全技术要求，环境安全部分描述软件的安全管理配置要求。图1为本标准描述的安全编程规范框架。

采用本标准中的要求，能够有效的降低应用软件的脆弱性。

图1：应用软件安全编程要求框架

**程序安全**

**数据净化**

输入验证

输出处理

**数据加密与保护**

**访问控制**

身份鉴别

口令安全

权限控制

**面向对象程序安全**

**并发程序**

**安全**

**错误处理**

**安全**

**日志安全**

**内存管理**

**数据库管理**

**文件管理**

**网络传输**

**管理配置**

**安全功能技术要求**

**环境安全**

**代码实现安全要求**

**资源使用安全要求**

# 安全功能实现要求

## 数据清洗

### 输入验证

#### 验证所有输入数据

1. 对输入的数据进行过滤或标准化处理，然后进行验证。
2. 确保对所有输入的数据进行验证，拒绝接受验证失败的数据。
3. 禁止试图对验证失败的数据进行修复，自动错误恢复代码很可能改变请求的初始意图或者截断验证逻辑。

#### 在可信系统上执行输入验证

输入验证过程应在可信系统上执行。

#### 建立可信边界

1. 在程序中定义清晰的信任边界，将可信和不可信数据（比如：数据库，文件流）分别存储。
2. 当数据要从不可信的一侧传输到可信一侧的时候，应使用验证逻辑进行判断。

#### 集中输入验证

把输入验证作为软件框架的一部分，为应用程序提供一个统一的输入验证策略。

#### 验证来自重定向输入的数据

攻击者可能向重定向的目标直接提交恶意代码，从而避开应用程序逻辑以及在重定向前执行的验证，所以对重定向输入数据应再次验证。

#### 检测数据类型

为参数指定明确的数据类型，严格检查用户输入的数据类型。

#### 检测输入长度

验证允许输入的最小和最大长度。

#### 明确所有输入字符集

为所有输入明确恰当的字符集，比如：UTF-8。确定系统是否支持UTF-8扩展字符集，如果支持，在UTF-8解码完成以后进行输入验证。

#### 验证HTTP请求中的所有组成

验证来自HTTP请求中的所有数据，恶意数据可以从表单域，URL参数，cookie，HTTP头以及URL自身传入。

#### 对来自命令行、环境以及配置文件的输入进行校验

禁止使应用软件的安全性依赖于配置和维护它的人，对来自命令行、环境以及配置文件的输入也应进行验证。

#### 防范元字符攻击

禁止攻击者控制发送给文件系统、浏览器、数据库或者其他子系统的命令，否则，攻击者能利用这种命令构造元字符攻击。

#### 抗原发抵赖

当在网络环境中出现数据交换时，可提供原发证据，实现抗原发抵赖。

### 输出编码

#### 编码所有不安全的输出数据

通过语义输出的编码方式，对所有返回到客户端的，来自于应用程序信任边界之外的数据进行编码。

除非明确对目标编译器是安全的，否则对所有字符进行编码。

SQL、XML和LDAP查询语句以及操作系统命令，可能存在潜在的危险字符，应语义净化所有不可信数据的输出。

#### 在可信系统上执行输出编码

所有的输出编码过程必须在可信系统上执行。

#### 定义输出编码规则

为每一种输出编码方法采用一个标准的、已通过测试的规则。

#### 抗接收抵赖

当在网络环境中出现数据交换时，可提供接收证据，实现抗接收抵赖。

## 数据加密与保护

### 加密规范

数据加密应符合GM/T 0028-2014标准中规定的要求。

#### 使用可信的密码算法

如编码中需要使用加密、数字签名、密钥交换或散列函数时，，应使用公开的、可信的密码算法。

#### 确保密码运算过程安全

密码运算应基于制定的算法和特定长度的密钥来进行密码运算。

#### 在可信系统上执行加密过程

所有秘密信息的加密过程必须在可信系统执行。

#### 安全地处理加密模块的失败操作

如果加密模块加密失败或报错，需重新加密。

#### 尽量少地共享秘密信息

1. 应当按照用途尽量减少需要保存的秘密信息。
2. 在将数据发送到客户端的时候，应基于任何通过客户端共享的数据都是不安全的假设对数据进行操作。

#### 规范密钥管理流程

建立并使用相关的安全策略和流程以实现加、解密的密钥管理。

#### 使用安全的随机数生成器

应使用密码学的伪随机数生成器，并使用信息熵最大的信息作为密码学伪随机数生成器的种子。

如果信息熵不可用，可以在使用密码学伪随机数生成器的时候改变其种子来降低威胁。

#### 密钥安全

1. 应规定安全的密钥强度，仅使用高于规定强度的密钥。
2. 应规定密钥有效期，禁止使用已经过期的密钥。
3. 禁止使用硬编码密钥，硬编码密钥将显著增加加密数据被攻击者破解的可能性。

### 数据保护

#### 确保用户数据的完整性

对应用软件中的用户数据进行完整性检查。

#### 保护服务器端的源代码

合理存储源代码，并结合权限控制策略，保护服务器端的源代码不被非法用户访问。

#### 加密存储敏感信息

即使在服务器端，仍然要加密存储敏感信息，比如，身份鉴别数据。总是使用已经被充分验证过的算法，更多指导信息请参见上节“加密规范”部分。

#### 删除用户可访问的源码中的注释

避免用户通过逆向或者直接获取网页源代码方式获取源代码注释，编程人员应及时删除注释。

#### 删除不需要的应用程序和系统文档

1. 删除服务器上不需要的应用程序和系统文档，关闭不需要的服务。
2. 禁止访问已删除的资源。

#### 保护缓存或临时拷贝的敏感数据

保护所有在服务器上缓存的或临时拷贝的敏感数据，并在不需要时尽快清除。

#### 禁止在客户端保存敏感信息

禁止在客户端上以明文形式或其他非加密安全模式保存口令、连接字符串或其他敏感信息。

#### 避免用户数据彻底丢失

当用户数据丢失或破坏时，最好确保可通过备份信息进行数据恢复。

#### 禁止表单中的自动填充功能

在设计WEB登录表单的时候，可考虑禁止浏览器的口令自动填充功能。

## 访问控制

### 身份鉴别

#### 提供安全的身份鉴别环境与流程

1. 访问需要保护的资源均要求身份鉴别，除了那些特定设为“公开”的内容以外，所有的网页和资源要求身份鉴别。
2. 在可信系统上执行所有的身份鉴别，所有的身份鉴别过程必须在可信系统上执行，且在每次用户登录时进行身份鉴别。
3. 在所有数据输入后再进行连续身份鉴别。对于连续身份鉴别机制，应当在用户输入完账号、口令以及验证码等信息后，才进行身份鉴别。

#### 最小化角色授权

* 1. 建立并使用标准的、已通过测试的身份鉴别策略。
  2. 为所有身份鉴别使用一个集中实现的方法，包括利用库文件请求外部身份鉴别服务。
  3. 一个账号对应一个人而不是一个组，使用软件的每个人应拥有唯一的用户名。

#### 避免依赖不可靠信息进行身份鉴别

在进行关键的安全操作时，不应信任cookie中的数据。

在进行关键的安全操作时，不应依赖反向DNS解析获取的主机信息。

#### 验证数字证书

必须检查证书的状态和证书持有者，只有有效的、未过期的且证书的实际持有者与证书中声明的持有者一致的证书才能被信任和使用。

#### 避免认证被绕过

严格控制用户访问系统的可选途径或通道，保证用户只能通过指定的途径或通道访问系统，避免认证被绕过。

应使用安全的认证算法，且算法的关键步骤没有被省略或跳过。

#### 根据业务安全要求选择身份鉴别方式

安全性要求高的系统建议采用多因素身份鉴别方式，基于口令的单因素身份鉴别是不安全的。

#### 避免在处理身份鉴别的过程中透露多余信息

处理每个认证请求所花费的时间相同。避免攻击者根据登录尝试失败的时间来判断登录尝试是否成功。

安全地处理未成功的认证。认证和注册的错误信息不能包含可被攻击者利用的信息，例如，判断一个特定的用户名是否有效的信息。

确保鉴别反馈的内容中不包含敏感信息。

#### 对认证尝试的频率进行限制，连续多次登录失败强制锁定账户

限制同一个账号能够进行认证尝试的频率和次数。

应设定用户登录失败次数的阈值，在用户登录失败次数达到阈值后应锁定用户账号，防止攻击者进行暴力破解。

#### 实现用户与主体的绑定

用户进程应与所有者用户相关联，使用户进程的行为可以追溯到进程的所有者用户。

系统进程应与当前服务要求者用户动态关联，使系统进程的行为可以追溯到当前服务要求者用户。

### 口令安全

#### 确保口令不可见

登录过程中，应确保口令不可见。

#### 使用强口令

1. 口令的复杂度（包括口令组成、口令长度等）应满足安全策略要求。
2. 避免使用弱口令、空口令或已泄露的口令。

#### 替换默认口令

对于默认的初始口令，强制用户初次登录时替换默认口令。

#### 不使用过期口令

1. 过期口令不可继续使用。
2. 应定期更改口令，关键系统可要求更频繁地更改。
3. 应明确口令更改时间周期。

#### 保护口令重置信息

1. 应使用保护口令信息的安全策略保护口令重置信息。
2. 口令重置操作应采取与账户创建、身份鉴别同等级别的安全控制策略。
3. 口令重置问题应当支持尽可能随机的提问。

#### 安全地存储口令

1. 避免明文存储口令。
2. 应使用不可逆的加密算法或单向散列函数对口令进行加密存储。
3. 在散列过程加入随机盐，将口令转化为不可还原或难以使用字典攻击猜测的形式。
4. 应将加密后的口令存储在配置文件、数据库或者其它外部数据源中。
5. 禁止在源代码中写入明文口令。

#### 在可信系统上执行口令加密

所有的口令加密过程必须在可信系统上执行。

#### 最小化口令的保存时间

尽可能地减少口令、加密密钥的保存时间。

#### 使用安全的口令传输

1. 禁止在不安全的信道中传输口令，也禁止接受来自不安全信道的口令。
2. 避免传递明文口令。
3. 传统协议，如FTP，TELNET，HTTP，POP以及IMAP，应在使用了安全传输协议（例如SSL）的情况下才可被用于传输口令。

#### 用户信息改变时使用单独的信道通知

1. 允许用户改变其口令，当用户改变其账号信息时（例如重置口令）需要发送确认信息，确认信息应使用单独的通道发送。
2. 可要求用户通过邮件等方式来确认信息的变更，但禁止在确认邮件中包含认证信息。
3. 当用户改变他们的联系信息时，应发送两次变更通知：分别包含旧的和新的信息。

#### 权限管理

#### 最小权限原则

1. 服务账户，或连接到外部系统的账号，应当具有尽可能小的权限。
2. 将许可权限尽可能地细化，使用细粒度的访问控制。

#### 使用可信系统对象做出访问授权的决定

使用可信系统对象以做出访问授权的决定。

#### 集中访问控制

与输入验证一样，访问控制是一个集中化程序框架的一部分，建立统一的访问控制策略。

使用一个单独的全站点部件以检查访问授权。这包括调用外部授权服务的库文件。

#### 对每个用户交互都要检测访问控制状态

确保访问控制策略检查了用户访问或操作的数据。

如果允许一次身份鉴别后，可进行较长时间的通话，则应周期性地重新验证用户的身份，以确保其权限没有改变。如果发生改变，注销该用户，并强制重新执行身份鉴别。

#### 关键信息限制只有授权的用户才能访问

只有授权的用户才能访问秘密信息或敏感信息。通常，这些信息包括但不限于：文件或其他资源、应用程序外部的直接控制、受保护的URL、受保护的功能、直接对象引用、服务、应用程序数据以及与安全相关的配置信息。

#### 对关键操作需要再次验证

在用户执行关键或者不可逆的操作（如修改口令）之前，再次验证用户身份，以减少不安全会话带来的损失。

#### 执行账户审计并强制失效长期不使用的账户

建议明确允许账户不使用的最长期限，支持账户的强制失效，并在账户停止时终止会话。

# 代码实现安全要求

## 面对对象程序安全

### 禁止非受信子类对具有不变性的类和方法进行扩展

子类对基类的扩展，必须保持由基类提供的不变性。

关键基类必须只允许受信的子类扩展，并同时防止被非受信的代码扩展。

### 改变基类须保存子类之间的依赖关系

修改一个基类时，必须保证不触动其子类所依赖的程序不可变性。

### 声明数据成员为私有并提供可访问的封装器方法

数据成员必须声明为私有。

应当使用暴露类成员的封装器访问方法，用该方式使得这些类成员可以在声明该类的包以外被访问。

使用封装器方法监视并控制对数据成员的修改。例如，使用防御性复制、输入验证和日志。

### 代码中禁止混用具有泛型和非泛型的原始数据类型

仅在需要考虑对遗留代码兼容性时，才允许使用原始数据类型。引入泛型后，建议避免使用原始数据类型来编写代码。

### 为可变类提供复制功能并通过此功能允许将实例传递给非受信代码

可变类必须提供一个可复制的构造函数，或者提供一个返回该实例副本的公有静态工厂方法。

对于非受信代码的输出及传递给非受信代码的输入，即使提供了复制功能，仍然需要对这两种情况的数据创建防御性副本。

### 返回引用之前防御性复制私有的可变的类成员

返回类的内部可变成员的引用，会破坏一个应用的安全性，为此，程序禁止返回内部可变类的引用。

返回一个指向可变内部状态的防御性复制的引用，能保证调用者不会修改初始的内部状态。

### 对可变输入和可变的内部组件创建防御性副本

禁止直接依赖可变输入，避免竞态攻击。例如，当一个数据成员包含一个通过验证和安全检查但在使用前被篡改的数值时，就会发生Time-of-Check Time-of-Use(TOCTOU)漏洞。

返回对象内部一个可变组件的引用，会为攻击者提供一个破坏对象状态的机会。因此，访问器方法必须返回内部可变对象的防御性副本。

### 不允许敏感类复制其自身

禁止复制包含秘密或者其他敏感数据的类。

那些不敏感但是需要维持其他不可变性的类，必须对恶意子类访问或者操作它们的数据及破坏其不可变性的可能性做出防御。

复制操作禁止改变源对象。

### 禁止在嵌套类中暴露外部类的私有字段

嵌套类禁止将外部类的私有成员暴露给外部的类或者包。

### 比较类而不是类名称

当判定一个对象是否属于特定的类，或两个对象的类是否相同时，应比较类对象，不能仅基于类名称进行判定。

### 小心处理构造函数抛出异常的情况

在一个构造方法开始创建对象但并未结束时，对象会被部分地初始化。只要对象没有被完全初始化，就必须对其他类不可见。

### 核查并删除代码中存在从未使用的变量或未执行的代码段

建议核查并删除代码中那些从未被使用的变量或从未执行的代码段。

### 确保虚函数被安全调用

不要从构造函数与析构函数中调用虚函数。

### 不要切分子类对象

不要将基类对象初始化为子类对象，除非通过引用、指针或类指针的方式。

### 谨慎删除多态对象

不要通过指向具有非虚析构函数的基类指针删除子类对象。

在没有虚析构函数时不要删除多态对象。

## 并发程序安全

### 确保共享数据的线程安全

确保所有的全局变量、线程间的共享可变数据是线程安全的。

对所有需确保线程安全的数据通过同步方法或代码块进行保护。

### 确保共享变量、数据的可见性

对共享变量、数据的读或写操作均应使用同步访问。

所有执行读或写操作的线程都必须在同一个锁上同步，以确保所有线程都能看到共享变量或数据的最新值。

### 确保同步对象上的锁按相同的顺序获得和释放

同步对象上的锁应按照相同的顺序获得和释放，以避免当两个或多个线程互相等待时被阻塞进而发生死锁。

### 确保导致阻塞的线程可以被终止

需确保能够终止处于阻塞状态的任务和线程。

### 确保线程池安全

使用线程池来处理请求，以抵御拒绝服务攻击。

禁止在一个有限的线程池中执行需要依赖其他线程的任务，以避免死锁。

提交至线程池的任务应是可被中断的。

所有线程池中的任务应提供将任务执行的异常情况通报应用程序的机制。

### 确保线程专有存储可被及时清理

应及时释放线程专有对象，防止内存泄漏与拒绝服务式攻击。

### 注意共享对象的存储周期

禁止在一个线程中访问其他线程的非静态局部变量。

### 在循环体中执行线程的挂起操作

仅在循环体中执行线程的挂起操作，并在每次挂起线程之前检查线程继续执行要满足的条件。

### 谨慎处理已加入或分离过的线程

不要加入或分离已进行过加入或分离操作的线程。

### 谨慎处理原子变量

禁止在一个表达式中两次访问同一个原子变量。程序无法确保在两次访问该变量之间，其值不被其他线程修改。

### 确保特权让渡过程成功

进行特权让渡时，应遵守正确的特权废除顺序。

### 不要使用异步取消的线程

不要使用异步取消的线程，防止产生资源泄露。

### 控制条件变量同步等待操作中的互斥数量

不要在条件变量同步等待操作中使用多于一个的互斥。

### 通知所有等待的线程

释放互斥锁时，建议线程通知所有正在等待该锁的线程，而不是单一线程。

## 异常处理安全

### 自行处理错误

1. 软件应自行处理程序错误，并且不依赖于服务器配置。
2. 静态对象的构造器和线程存储周期一定不能抛出异常。
3. 越过执行边界时不要抛出异常。

### 保持对象状态的一致性

异常处理时应保持数据的状态一致性。

### 异常处理时及时释放资源

异常处理时应及时回收并释放系统资源。

### 禁止忽略捕获的异常

1. 精确捕获异常，并对捕获的异常进行恰当的处理。
2. 避免在捕获异常后不做任何处理。

### 不在软件执行异常时暴露敏感信息

1. 向用户展示通用的错误提示信息。
2. 在系统发生异常状况时禁止向用户暴露的敏感信息包括但不限于：系统的详细信息、会话标识符、账号信息、调试或堆栈跟踪信息。

## 日志安全

### 保护日志文件

1. 应对日志文件进行安全存储，例如将日志文件独立保存于应用程序目录外，使用严格的访问权限来控制日志文件使用。
2. 应防止攻击者获取操纵日志的权限。
3. 使用消息摘要算法以验证日志记录的完整性。

### 在可信系统上执行所有的日志记录操作

所有的日志记录操作必须在可信系统上执行。

### 集中日志记录

日志记录应是一个集中化程序框架的一部分。

### 创建清晰的日志等级

将不同类别的操作与清晰的日志等级关联起来，日志等级不能交叠或者不明确。

### 在日志记录中使用时间戳

应在每个日志条目中增加精确的时间戳，同时确保时间戳的可靠性。

### 对每个重要的行为都记录日志

1. 确保系统在发生重要行为时创建日志，并将重要行为记录在最高级别的日志中。
2. 重要行为通常包括但不限于：重要数据更改、认证尝试（特别是失败的认证）、失败的访问控制、失效或者已过期的会话令牌尝试、系统例外、管理功能行为、失败的后端TLS连接、加密模块的错误。

### 应避免在记录日志时发生异常

1. 尽量避免在日志记录过程中发生异常。
2. 如果出现异常情况，应确保日志记录能正确地继续执行。

### 防止日志记录中不可信数据被执行

1. 确信日志记录中的不可信数据，不会在查看界面或者运行软件时以代码的形式被执行。
2. 禁止允许攻击者能够写任意的数据到日志里。

### 避免在日志中保存敏感信息

1. 禁止在日志中保存敏感信息，这些敏感信息包括但不限于：系统详细信息、会话标识符或口令。
2. 禁止记录包含口令的通信和错误。

# 资源使用安全要求

## 资源管理

### 每个资源具有唯一的标识

系统中的每个资源应具有唯一标识符。

### 初始化重要资源

使用重要资源前应进行正确初始化。

### 限制资源的分配和使用

1. 对分配的资源数量、使用权限、有效时间做限制。
2. 合理控制递归，防止消耗过多资源。

### 及时释放资源，不使用已过期或已释放的资源

1. 及时释放系统资源，且禁止再调用已释放或过期的资源。
2. 释放资源前应完全清除敏感信息。

### 检查外部资源的安全性

对外部资源，如下载的文件，应进行完整性和发布源检验，确保外部资源的安全性。

### 应提供足够大的资源池

系统提供的资源池应能满足高峰期的业务需求。

## 内存管理

### 一致的内存管理约定

1. 使用同样的模式分配和释放内存。
2. 在同一个模块中，在同一个抽象层次中，分配和释放内存。
3. 分配和释放必须配对。

### 防范缓冲区溢出

缓冲区溢出被用来作为一种软件系统的攻击手段，通过往程序的缓冲区写入超出其长度的内容，造成缓冲区溢出，从而破坏程序堆栈，使程序转而执行其它指令，以达到攻击的目的。为防范缓冲区溢出，要求：

1. 对不可信数据进行输入和输出控制。
2. 重复确认缓冲区空间的大小是否和指定的大小一样。
3. 字符串操作时，检查字符串长度和终止符。
4. 在循环中调用函数时，注意检查缓冲区空间大小，确保不存在超出分配空间的访问。
5. 不要在无关的智能指针中存储已有的指针值。

### 避免双重释放漏洞

避免对同一块内存释放两次。

### 禁止写入已释放的内存

已释放的内存，在再次分配前禁止写入。

### 及时释放内存

1. 及时释放内存。
2. 禁止依赖垃圾回收机制。

### 保护堆安全

1. 执行堆完整性检测。
2. 在可能的情况下，使用不可执行的堆栈。

### 避免使用已知有漏洞的函数

应尽量避免使用不进行自变量检查的、已知存在漏洞的字符串操作函数，如printf、strcat、strcpy，并使用其他函数替代。

### 允许用户适当个性化内存管理方式

1. 应允许用户根据自己的使用需求更改动态内存管理方式。
2. 尽量避免对过度整齐的数据类型使用默认操作。

## 数据库管理

### 禁止使用默认账户与口令

当应用程序访问数据库时，禁止使用默认的角色、账户与默认数据库口令访问数据库。

### 确保数据完整性

使用数据库进行数据存储时，应确保数据的完整性。

### 使用参数化的SQL语句和存储过程

1. 使用参数化SQL语句，禁止改变数据定向的上下文环境，并强制区分数据和命令。
2. 使用存储过程以实现抽象访问数据，并允许对数据库中表的删除权限。

### 访问数据库使用尽可能低权限

1. 当应用程序访问数据库时，仅提供给应用程序满足其需求的最低权限，以降低访问数据库的风险。
2. 规定不同信任级别用户连接数据库的角色，比如用户、只读用户、访问用户、管理员。

### 使用行级别的访问控制

禁止依赖应用程序访问控制来保护数据库的数据，限制每个请求使用户只能访问他们自己的数据。

### 为所有变量指定类型

需要对所有变量指定数据类型，并在编译期间进行严格检查。

### 验证来自数据库中的数据

对来自数据库的数据进行验证，确保从数据库读出的数据符合预期。

### 及时释放数据库资源

及时释放数据库资源，例如连接、游标等。

### 关闭不必要的数据库功能

关闭所有不必要的数据库功能，如不必要的存储过程或服务、应用程序包，最小化需安装的功能和选项。

## 文件管理

### 采用最小权限原则

文件应只能被指定的用户访问，而且该用户应该被限制为最小权限。

### 验证文件系统的所有信息

对来自文件系统的所有信息（包括但不限于文件路径、文件名的内容和长度）进行标准化，然后进行验证。

### 确保文件上传安全

1. 上传文档前进行身份鉴别。
2. 验证文件类型，仅允许上传满足业务需要的相关文件类型。除验证文件类型扩展名外，还需至少检查文件报头中的类型信息。
3. 访问上传的文件之前，进行恶意代码的扫描，并校验文件完整性。
4. 关闭在文件上传目录的运行权限。

### 禁止将绝对文件路径传递给用户

禁止传递目录或绝对文件路径给用户，建议使用预先设置的路径列表中的匹配索引值。

### 使用安全的临时文件

在程序初始化时采用严格的权限策略建立一个安全临时文件夹，把临时文件放在该文件夹内。

### 确保应用程序文件和资源是只读的

禁止用户修改应用程序文件和资源，权限仅限于可读。

### 避免文件访问竞争

使用文件句柄来保证针对某文件的多次操作确实是对同一个文件的操作。

### 确保文件系统资源会被释放

保证诸如文件句柄之类的文件系统访问结构在不再需要时会被及时释放。

禁止依赖Java和.NET等的垃圾回收机制来回收资源。

### 检测和处理文件相关的错误

不能忽略文件相关的错误。比如，Java文件操作往往通过返回值来表示操作是否失败，Java程序必须检查I/O方法的返回值。

## 网络传输

### 验证网络数据包

对来自网络的数据进行验证以确保数据包符合预期要求。

### 使用加密传输确保通信安全

1. 应采用加密传输方式保护敏感信息，特别是要求身份鉴别的访问内容、连接到外部系统的功能。
2. 宜使用配置合理的单一标准TLS或SSL对连接保护，并支持对敏感文件或非基于HTTP连接的不连续加密。
3. 没有成功的TLS连接不应当后退成为一个不安全的连接。
4. 为在TLS连接上传输的cookie设置“安全”属性。
5. TLS证书应当是有效的，有正确且未过期的域名，并且在需要时，和中间证书一起安装。

### 消息完整性验证

应对信道中传输的消息进行完整性验证。

### 使用随机数和时间戳组合防止重放攻击

应当使用时间戳和随机数组合的方式进行重放检测。

### 管理控制会话

1. 使用服务器或者框架的会话管理控制。应用程序应当只识别有效的会话标识符。
2. 会话标识符必须总是在一个可信系统上创建。
3. 使用安全的算法以保证会话标识符足够随机。
4. 建议用账户登出后应完全终止相关的会话或连接。
5. 在平衡风险和业务功能需求的基础上，最小化会话超时时间（通常小于几个小时）。
6. 如果一个会话在登录以前就建立，在成功登录以后，应关闭该会话并创建一个新的会话。
7. 任何重新进行身份鉴别的过程,都应建立一个新的会话标识符。
8. 不允许同一用户ID的并发登录。
9. 在身份鉴别的时候，如果连接从HTTP变为HTTPS，则生成一个新的会话标识符。在应用程序中，宜持续使用HTTPS，而非在HTTP和HTTPS之间转换。
10. 为服务器端的操作执行标准的会话管理，比如，通过在每个会话中使用强随机令牌或参数来管理账户。

# 配置管理安全要求

## 构建安全的编译环境

1. 使用安全的编译器，并验证编译器的有效性。
2. 建议使用源代码复杂性度量工具，防止编写出难以控制和难以测试的代码。
3. 去掉不必要的编译功能，如去掉调试开关。

## 及时校验版本稳定性和安全性

确保软件运行服务器、使用框架和系统部件均为相对安全的稳定版本，并安装了该版本的所有补丁。

## 关闭目录自动列表功能

建议禁止自动目录列表功能。

如果必须开启目录列表功能，则应对目录下的文件进行详细检查，确保不包含敏感文件。

## 应用程序的安全配置信息应当可读

应用程序的安全配置信息应以可读的形式输出，以支持审计。

## 隔离开发环境和实际运行环境

确保开发环境与实际运行环境的物理隔离，并只提供给授权的开发和测试团队访问。在实际工作中，可实行开发环境、测试环境及生产环境的分离控制，或开发环境、测试环境、用户验收测试环境及生产环境的分离控制。

## 推荐使用软件变更管理系统

建议使用软件变更管理系统以管理和记录在开发和产品迭代过程中代码的变更。

参 考 文 献

[1] Common Weakness Enumeration. http://cwe.mitre.org/.

[2] OWASP安全编码规范快速参考指南,Version 1.0，2012年8月.

[3] ISO/IEC TS 17961:2013,Information technology - Programming languages, their environments and system software interfaces - C secure coding rules

[4] ISO/IEC TR 24772-2013,Information technology-Programming languages–Guidance to avoiding vulnerabilities in programming languages through language selection and use.

[5] SEI CERT Coding Standards. https://www.securecoding.cert.org/confluence/display/seccode/SEI+CERT+Coding+Standards.

[6] MISRA C:2012Guidelines for the use of theC language in critical systems.

[7] MISRA C++: 2008 Guidelines for the Use of the C++ Language in Critical Systems.

[8] HPE Security Fortify 分类法：软件安全错误. http://www.hpenterprisesecurity.com/vulncat/zh\_CN/vulncat/intro.html.

[9] RFC3280 互联网X.509公钥基础设施证书和CRL轮廓(RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_