# 应用系统安全开发规范

# 1、目的

本规范的目标是提供一套完善的、系统化的、实用的WEB应用信息系统安全开发方法供应用信息系统研发人员使用，以期达到提高WEB应用信息系统安全的目的。本规范主要包括：WEB应用信息系统开发规范、WEB应用信息系统编码规范，通过这两方面的内容，提升WEB应用信息系统整体的安全性；本规范主要以Java编程语言为例。

# 2、适用范围

适用于产品的设计、开发阶段中的安全编码规范。

# 3、定义及说明

表1 列出了一些WEB应用信息系统缺陷类别，并针对每类缺陷列出了由于设计不当可能会导致的潜在问题。针对这些潜在的问题，本规范中有相应的解决措施。

表1 WEB应用程序缺陷和由于不良设计可能导致的问题

|  |  |
| --- | --- |
| 缺陷类别 | 由于不良设计可能导致的问题 |
| 身份验证 | 身份伪造、口令破解、权限提升和未授权访问。 |
| 会话管理 | 通过捕获导致会话劫持和会话伪造。 |
| 权限管理 | 访问机密或受限数据、篡改和执行未授权操作。 |
| 配置管理 | 未授权访问管理界面、更新配置数据、访问用户帐户和帐户配置文件。 |
| 敏感数据 | 敏感信息泄漏和数据篡改。 |
| 加密技术 | 未授权访问机密数据或帐户信息。 |
| 安全审计 | 未能识别入侵征兆、无法证明用户的操作，以及在问题诊断中存在困难。 |
| 输入检验 | 通过嵌入查询字符串、窗体字段、Cookie 和 HTTP 标头中的恶意字符串所执行的攻击。包括命令执行、跨站点脚本编写 (XSS)、SQL 注入和缓冲区溢出攻击等。 |
| 参数操作 | 路径遍历攻击、命令执行、此外还有跳过访问控制机制、导致信息泄露、权限提升和拒绝服务。 |
| 异常管理 | 拒绝服务和敏感的系统级详细信息泄露。 |

**术语用词约定：**

**规则：**强制必须遵守的原则（标红）

**建议：**需要加以考虑的原则

**说明：**对此规则或建议进行相应的解释

**实施指导：**对此规则或建议的实施进行相应的指导

# 4、常见的WEB应用信息系统的安全漏洞

WEB应用的安全漏洞有很多，无法穷举。针对众多的WEB漏洞，OWASP的专家们结合各自在各领域的应用安全工作经验及智慧，提出了十大WEB应用程序安全漏洞，帮助人们关注最严重的漏洞。（OWASP即开放WEB应用安全项目，是一个旨在帮助人们理解和提高WEB应用及服务安全性的项目组织）

表2 十大WEB应用程序安全漏洞列表

 OWASP TOP 10 2021

|  |  |  |
| --- | --- | --- |
| 序号 | 漏洞名称 | 漏洞描述 |
| 1 | 权限控制失效 | 访问控制是指管理用户访问资源的安全策略。失效的访问控制漏洞发生在应用程序未能正确执行授权策略，导致攻击者能够访问本不应该有权限访问的资源。 |
| 2 | 加密机制失效 | 加密机制失效指的是未能正确实施加密措施来保护敏感数据。加密不当会导致数据在存储或传输过程中被窃取或篡改。常见的问题包括使用过时的加密算法、密钥管理不当以及未加密敏感数据（如密码、个人信息等）。 |
| 3 | 注入 | 注入漏洞是指攻击者利用程序弱点，通过将恶意字符或代码写入数据库，获取敏感数据或在服务器执行命令。常见的注入方式包括SQL注入、SSI注入、LDAP注入、XPath注入、XML注入、代码注入等。 |
| 4 | 不安全的设计 | 不安全的设计是2021年的一个新类别，重点关注与设计缺陷相关的风险。如果行业想提升安全性，就需要更多地使用威胁建模、安全设计模式和原则以及参考架构。 |
| 5 | 安全配置错误 | 安全配置错误是指应用程序或系统的安全配置不正确或未得到妥善维护，导致攻击者可以利用这些配置漏洞来攻击系统。 |
| 6 | 易受攻击与过时组件 | 易受攻击与过时组件是指应用程序中使用了已知存在漏洞或已过时的组件，这些组件可能会被攻击者利用来攻击系统。 |
| 7 | 身份识别和身份验证错误 | 该漏洞涉及身份识别和身份验证机制的失效，可能导致攻击者能够绕过正常的认证流程，获取未授权访问权限。 |
| 8 | 软件和数据完整性故障 | 该漏洞指的是软件或数据在未经授权的情况下被篡改或损坏，可能导致应用程序功能异常或敏感数据泄露。 |
| 9 | 安全日志和监控故障 | 安全日志和监控是检测和响应安全事件的重要手段。如果安全日志记录不完整、不准确或监控机制失效，那么攻击者就可能利用这些漏洞进行未授权访问或执行恶意操作而不被发现。 |
| 10 | 服务端请求伪造 | SSRF漏洞允许攻击者利用服务器向内部或外部网络发送恶意请求，可能导致敏感信息泄露、服务拒绝攻击或其他安全风险。 |

# 5、WEB安全开发规范

## 5.1 身份验证

### 5.1.1 口令

关于Web应用及容器涉及到的口令，请遵循口令安全要求。详细内容请见下表。

|  |  |  |
| --- | --- | --- |
| 要求项 | 要求概述 | 详细描述 |
| 口令安全策略管理 | 设置口令时，默认检测口令复杂度 | 系统默认检测口令复杂度，口令至少满足如下要求：  1.长度≥8位。  2.复杂度≥两种字符，支持字母、数字、特殊字符。  3.密码中不能包含有连续四位及以上顺序(或逆序)数字或字母；（如：1234、abcd等）。  4.密码中不能包含有连续四位及以上重复字符，字母不区分大小写；（如：8888、AAAA、$$$$等）。  5.不能将帐号名作为密码的一部分存在于密码，帐号密码也不能一样；（如：帐号为1355，密码不允许设置为\*\*\*1355\*\*\*\*）。  6.常用的弱密码不区分大小写不能作为密码；（如：000000、111111、11111111、112233、123123、123321、123456、12345678、654321、666666、888888、abcdef、abcabc、abc123、a1b2c3、aaa111、123qwe、qwerty、qweasd、admin、password、p@ssword、passwd、iloveyou、5201314等）。  若设置的口令不符合上述规则，必须进行警告。 |
| 可设置口令出错锁定阈值 | 系统必须提供锁定用户的机制。可选择如下两种方式之一：  方式一：当重复输入错误口令次数（默认3次，次数系统可以设置）超过系统限制时，系统要锁定该用户。  方式二：系统还可以设置下次允许输入口令的间隔时间加倍，采用这种方式时，用户可以不设置自动锁定。  \*\*\*确认选择其中之一 |
| 可设置自动解锁时间（只适用于由于口令尝试被锁定的用户） | 1、对于口令尝试N次失败被锁定的用户，系统要能够设置自动解锁时间，建议默认解锁时间为5分钟。  2、用户被锁时间达到预定义时间，可自动解锁该用户，或者也可通过安全管理员手工解锁该用户。  3、在锁定时间内，仅能允许应用安全管理员角色所属账号手动解锁该用户。 |
| 口令安全使用规则 | 操作界面中的口令不能明文显示 | 键入口令时不能明文显示出来(操作界面中的输入口令可不显示或用\*代替)，包括在终端上打印或存储在日志中时也不能明文显示口令，即使是内存中的明文口令（如登录期间），也应在使用后立即覆盖。 |
| 口令输入框内容禁止拷贝 | 口令输入框不支持拷贝功能。 |
| 缺省口令符合复杂度要求 | 对于系统内置账号的缺省口令，口令应符合复杂度的要求，并在客户资料中提醒用户修改。 |
| 用户可修改自己的口令 | 1、用户修改自己口令时必须验证旧口令；  2、不允许修改除自身账号以外的账号的口令（管理员除外）。 |
| 口令不能在网络中明文传输 | 口令等认证凭证在传输过程中必须加密，安全的加密算法。 |
| 口令在本地存储时必须使用不可逆的加密算法加密 | 1、口令不能够明文写入日志文件、配置文件以及cookie中；  2、口令文件必须设置访问控制，普通用户不能读取或拷贝加密的内容。 |

\*\*\*异地登陆

### 5.1.2 认证

**规则5.1.2.1**：

对用户的最终认证处理过程必须放到应用服务器进行。

说明：不允许仅仅通过脚本或其他形式在客户端进行验证，必须在应用服务器进行最终认证处理（如果采用集中认证，那么对用户的最终认证就是放在集中认证服务器进行）。

**建议5.1.2.2：**

管理页面建议实施强身份认证。

说明：如双因素认证、SSL双向证书认证、生物认证等；还可以通过应用程序限制只允许某些特定的IP地址访问管理页面，并且这些特定的IP地址可配置。

**规则5.1.2.3：**

网页上的登录/认证表单必须加入验证码。

说明：使用验证码的目的是为了阻止攻击者使用自动登录工具连续尝试登录，从而降低被暴力破解的可能。如果觉得验证码影响用户体验，可以对接极验的无感知验证码功能。

**建议5.1.2.4：**

同一客户端在多次连续尝试登录失败后，服务端需要进行用户帐号或者是客户端所在机器的IP 地址的锁定策略，且该锁定策略必须设置解锁时长，超时后自动解锁。

说明：登录失败应该提示用户：如果重试多少次不成功系统将会锁定。在锁定期间不允许该用户帐号（或者客户端所在机器的IP 地址）登录。

允许连续失败的次数（指从最后一次成功以来失败次数的累计值）可配置，取值范围为：0-99 次，0表示不执行锁定策略，建议默认：5 次。

锁定时长的取值范围为：0-999 分钟，建议默认：30 分钟，当取值为0 时，表示无限期锁定，只能通过管理员手动解锁（需要提供管理员对服务器锁定其它用户帐号/IP进行解锁的功能界面）。建议优先使用帐号锁定策略。

注意：应用程序的超级用户帐号不能被锁定，只能锁定操作的客户端所在的IP，这是为了防止系统不可用。特别说明：锁客户端IP策略存在缺陷，当用户使用proxy上网时，那么锁定客户端IP会导致使用该proxy上网的所有用户在IP锁定期间都不能使用该Web应用；锁定用户帐户的策略也存在缺陷，当攻击者不断尝试某帐户的口令，就给该帐户带来拒绝服务攻击，使该帐户不可用。

**规则5.1.2.5：**

用户名、密码和验证码必须在同一个请求中提交给服务器，必须先判断验证码是否正确，只有当验证码检验通过后才进行用户名和密码的检验，否则直接提示验证码错误。

说明：如果验证码和用户名、密码分开提交，攻击者就可以绕过验证码校验（如：先手工提交正确的验证码，再通过程序暴力破解），验证码就形同虚设，攻击者依然可以暴力破解用户名及口令。

**规则5.1.2.6：**

所有登录页面的认证处理模块必须统一。

说明：可以存在多个登录页面，但是不允许存在多个可用于处理登录认证请求的模块，防止不一致的认证方式。

**规则5.1.2.7：**

所有针对其他第三方开放接口的认证处理模块必须统一。

**规则5.1.2.8：**

认证处理模块必须对提交的参数进行合法性检查。

说明：具体输入校验部分请查看6.1输入校验。

**规则5.1.2.9：**

认证失败后，不能提示给用户详细以及明确的错误原因，只能给出一般性的提示。

说明：可以提示：“用户名或密码错误，登录失败”；不能提示：“用户名不存在”、“口令必须是6 位”等等。

**规则5.1.2.10：**

禁止在系统中预留任何的后门帐号或特殊的访问机制。

**规则5.1.2.11：**

对于重要的管理事务或重要的交易事务要进行重新认证，以防范会话劫持和跨站请求伪造给用户带来损失。

说明：重要的管理事务，比如重新启动业务模块；重要的交易事务，比如转账、余额转移、充值等。重新认证，比如让用户重新输入支付密码等。

**规则5.1.2.12：（V\_1.0.2新增）**

对于手机号验证码登录的处理逻辑，需要先发送验证码，再校验验证码，最后判断用户是否存在。

说明：如果先判断用户是否存在，再校验验证码。或者先判断用户是否存在，再发送验证码，这两种逻辑均存在账号枚举的风险。

### 5.1.3 验证码

**规则5.1.3.1：**

为了平衡安全性和用户体验，公司统一采购了极验验证码，涉及到需要验证码的接口，需统一对接腾讯或极验验证码。

**规则5.1.3.2：**

验证码在一次使用后要求立即失效，新的请求需要重新生成验证码。

说明：进行验证码校验后，立即将会话中的验证码信息清空，而不是等到生成新的验证码时再去覆盖旧的验证码，防止验证码多次有效；注意：当客户端提交的验证码为空，验证不通过。

**规则5.1.3.3:（V\_1.0.2新增）**

验证码有效期不宜过长，必须设置有效期。

禁止通过API接口返回验证码明文字符串，包括但不限于：Header、Cookie、Json等。

禁止设置万能验证码。

## 5.2 会话管理

**规则5.2.1：**

使用会话cookie/维持会话。

说明：目前主流的Web容器通过以下几种方式维持会话：

隐藏域、URL重写、持久性cookie、会话cookie，但通过隐藏域、URL重写或持久性cookie方式维持的会话容易被窃取，所以要求使用会话cookie维持会话。如果条件限制必须通过持久性cookie维持会话的话，那么cookie信息中的重要数据部分如身份信息、计费信息等都必须进行加密。（cookie有两种：会话cookie和持久性cookie；会话cookie，也就是非持久性cookie，不设置过期时间，其生命期为浏览器会话期间，只要关闭浏览器窗口，cookie就消失了；会话cookie一般不存储在硬盘上而是保存在内存里。持久性cookie，设置了过期时间，被浏览器保存到硬盘上，关闭后再次打开浏览器，持久性cookie仍然有效直到超过设定的过期时间。）

**规则5.2.2：**

会话过程中不允许修改的信息，必须作为会话状态的一部分在服务器端存储和维护。

说明：会话过程中不允许修改的信息，例如，当用户通过认证后，其用户标识在整个会话过程中不能被篡改。禁止通过隐藏域或URL重写等不安全的方式存储和维护。应该通过session对象进行存储和维护。

**规则5.2.3：**

当Web应用跟踪到非法会话，则必须记录日志、清除会话并返回到认证界面。

说明： 非法会话的概念就是通过一系列的服务端合法性检测（包括访问未授权资源，缺少必要参数等情况），最终发现的不是正常请求产生的会话。

**规则5.2.4：**

禁止使用客户端提交的未经审核的信息来给会话信息赋值。

说明：防止会话信息被篡改，如恶意用户通过URL篡改手机号码等。

**规则5.2.5：**

当用户退出时，必须清除该用户的会话信息。

说明：防止遗留在内存中的会话信息被窃取，减少内存占用。

**规则5.2.6：**

必须设置会话超时机制，在超时过后必须要清除该会话信息。

说明：建议默认会话超时时间为30分钟。如果没有特殊需求，禁止使用自动发起请求的机制来阻止session超时。

**规则5.2.7：**

在服务器端对业务流程进行必要的流程安全控制，保证流程衔接正确，防止关键鉴别步骤被绕过、重复、乱序。

说明：客户端流程控制很容易被旁路（绕过），因此流程控制必须在服务器端实现。

**实施指导：**

可以通过在session对象中创建一个表示流程当前状态的标识位，用0、1、2、3、…、N分别表示不同的处理步骤，标识位的初始值为0，当接收到步骤N的处理请求时，判断该标识位是否为N-1，如果不为N-1，则表示步骤被绕过（或重复或乱序），拒绝受理，受理完成后更改标识位为N。

**规则5.2.8：**

所有登录后才能访问的页面都必须有明显的“注销（或退出）”的按钮或菜单，如果该按钮或菜单被点击，则必须使对应的会话立即失效。

说明：这样做是为了让用户能够方便地、安全地注销或退出，减小会话劫持的风险。

## 5.3 权限管理

**规则5.3.1：**

对于每一个需要授权访问的页面或servlet的请求都必须核实用户的会话标识是否合法、用户是否被授权执行这个操作。

说明：防止用户通过直接输入URL，越权请求并执行一些页面或servlet；建议通过过滤器实现。

**规则5.3.2：**

授权和用户角色数据必须存放在服务器端，不能存放在客户端，鉴权处理也必须在服务器端完成。

说明：禁止将授权和角色数据存放在客户端中（比如cookie或隐藏域中），以防止被篡改。

**规则5.3.3：**

一个帐号只能拥有必需的角色和必需的权限。一个组只能拥有必需的角色和必需的权限。一个角色只能拥有必需的权限。

说明：做到权限最小化和职责分离（职责分离就是分清帐号角色，系统管理帐号只用于系统管理，审计帐号只用于审计，操作员帐号只用于业务维护操作，普通用户帐号只能使用业务。）这样即使帐号被攻击者盗取，也能把安全损失控制在最小的限度。

**规则5.3.4：**

对于运行应用程序的操作系统帐号，不应使用“root”、“administrator”、“supervisor”等特权帐号或高级别权限帐号，应该尽可能地使用低级别权限的操作系统帐号。

**规则5.3.5：**

对于应用程序连接数据库服务器的数据库帐号，在满足业务需求的前提下，必须使用最低级别权限的数据库帐号。

## 5.4 敏感数据保护

敏感数据包括但不限于：口令、密钥、证书、会话标识、License、隐私数据（如短消息的内容）、授权凭据、个人数据（如姓名、住址、电话等）等，在程序文件、配置文件、日志文件、备份文件及数据库中都有可能包含敏感数据。

**规则5.4.1：**

禁止在代码中存储敏感数据。

说明：禁止在代码中存储如数据库连接字符串、口令和密钥之类的敏感数据，这样容易导致泄密。用于加密密钥的密钥可以硬编码在代码中。

**规则5.4.2：**

禁止用户手机号、身份证信息、密钥或帐号的口令以明文形式存储在数据库或者文件中。

**规则5.4.3：**

禁止在cookie 中以明文形式存储敏感数据。

说明：cookie信息容易被窃取，尽量不要在cookie中存储敏感数据；如果条件限制必须使用cookie存储敏感信息时，必须先对敏感信息加密再存储到cookie。

**规则5.4.4：**

禁止在隐藏域中存放明文形式的敏感数据。

**规则5.4.5：**

禁止用自己开发的加密算法，必须使用公开、安全的标准加密算法。

使用公司规定的加密算法

**实施指导：**

|  |
| --- |
| **场景 1：**后台服务端保存数据库的登录口令  后台服务器登录数据库需要使用登录数据库的明文口令，此时后台服务器加密保存该口令后，下次登录时需要还原成明文，因此，在这种情况下，不可用不可逆的加密算法，而需要使用对称加密算法或者非对称加密算法，一般也不建议采用非对称加密算法。  推荐的对称加密算法：aes256。  **场景 2：**后台服务端保存用户的登录口令  在该场景下，一般情况是：客户端提交用户名及用户口令，后台服务端对用户名及用户口令进行验证，然后返回验证的结果。此时，在后台服务端，用户口令可以不需要还原，因此建议使用不可逆的加密算法，对密码字符串进行加密。  推荐的不可逆加密算法： pbkdf2\_sha256 |

**规则5.4.6：**

禁止在日志中记录明文的敏感数据。

说明：禁止在日志中记录明文的敏感数据（如口令、会话标识jsessionid等）， 防止敏感信息泄漏。

**规则5.4.7：**

禁止带有敏感数据的Web页面缓存。

说明：带有敏感数据的Web页面都应该禁止缓存，以防止敏感信息泄漏或通过代理服务器上网的用户数据互窜问题。

**规则5.4.8：**

敏感信息应加密存储。

说明：公司的敏感信息（如手机号）应在数据库中加密存储。

## 5.5 敏感数据传输

**规则5.5.1：**

带有敏感数据的表单必须使用HTTP-POST 方法提交。

说明：禁止使用HTTP-GET 方法提交带有敏感数据的表单（form），因为该方法使用查询字符串传递表单数据，易被查看、篡改。

**规则5.5.2：**

在生产环境对外提供服务的应用，必须使用https进行通道加密，应使用TLS 1.2及以上相对高版本的协议，取消对低版本协议的支持。

**规则5.5.3：**

禁止在URL中携带会话标识（如jsessionid）。

说明：由于浏览器会保存URL历史记录，如果URL中携带会话标识，则在多人共用的PC上会话标识容易被其他人看到，一旦该会话标识还在其生命有效期，则恶意用户可以冒充受害用户访问Web应用系统。

**规则5.5.4：**

禁止将对用户保密的信息传送到客户端。

说明：这些信息一旦传送到客户端，那么用户也就可以获取到了。

## 5.6  安全审计

本节的安全审计是针对Web业务应用，不包括对操作系统、Web容器的安全审计。对于操作系统和Web容器的安全审计，可以参考对应的操作系统安全基线和Web安全配置规范。

**规则5.6.1：**

应用服务器必须对安全事件及操作事件进行日志记录。

说明：安全事件包括登录、注销、添加、删除、修改用户、授权、取消权限、鉴权、修改用户口令等；操作事件包括对业务系统配置参数的修改，对重要业务数据的创建、删除、修改、查询等；对于上述事件的结果，不管是成功还是失败，都需要记录日志。

**建议5.6.2：**

安全日志应该有备份及清理机制。

说明：备份及清理机制包括定期备份及清理安全日志和监控用于存放安全日志的磁盘空间的使用情况。可以配置定期备份及清理的时间，可以配置以用于存放安全日志的磁盘空间使用率达到多少时进行备份及清理。

**规则5.6.3：**

安全日志必须包括但不限于如下内容：事件发生的时间、事件类型、客户端IP、客户端机器名、当前用户的标识、受影响的个体（数据、资源）、成功或失败标识、启动该事件的进程标识以及对该事件的详细描述。

**建议5.6.4：**

通过网络形式保存安全日志。

说明：在生成安全日志时，即时将日志保存到网络上其他主机，而且生成安全日志的应用程序不能再访问存放在其他主机的日志。

**规则5.6.5：**

严格限制对安全日志的访问。

说明：只有Web应用程序的管理员才能查询数据库表形式或文件形式的安全日志；除数据库超级管理员外，只有应用程序连接数据库的帐号可以查询（select）及插入（insert）安全日志表；除操作系统超级管理员外，只有应用程序的运行帐户才能读、写文件形式的安全日志（但不允许删除）。确保日志的安全，限制对日志的访问，这加大了攻击者篡改日志文件以掩饰其攻击行为的难度。

**规则5.6.6：**

对日志模块占用资源必须有相应的限制机制。

说明：限制日志模块占用的资源，以防止如自动的恶意登陆尝试导致的资源枯竭类DOS攻击；比如限制日志记录占用的磁盘空间。

**规则5.6.7：**

禁止日志文件和操作系统存储在同一个分区中，同时，应使用转储、滚动、轮循机制，来防止存储日志的分区写满。

说明：所需空间和具体业务、局点容量、日志保存周期相关，要根据实际情况估算。

# 6、WEB应用安全编码规范

## 6.1 输入校验

**规则6.1.1：**

必须对所有用户的输入进行校验，一旦数据不合法，应该告知用户输入非法并且建议用户纠正输入。

说明：用户产生的输入是指来自text、password、textareas或file表单域的数据；必须假定所有用户产生的输入都是不可信的，并对它们进行合法性校验。

必须对所有服务器产生的输入进行校验，一旦数据不合法，必须使会话失效，并记录告警日志。

说明：服务器产生的输入是指除用户产生的输入以外的输入，例如来自hidden fields、selection boxes、check boxes、radio buttons、cookies、HTTP headers、热点链接包含的URL参数的数据或客户端脚本等；必须假定所有服务器产生的输入都是被篡改过的、恶意的，并对它们进行合法性校验，如果不合法，说明有人恶意篡改数据。举例：假如用户资料填写表单中的“性别”为必填项，用radio button（‘男’和‘女’对应实际值分别为‘1’和‘0’）来限制用户的输入，如果应用程序收到的“性别”值为‘2’，那么可以断定有人恶意篡改数据。

**规则6.1.2：**

禁止将HTTP标题头中的任何未加密信息作为安全决策依据。

说明：Web 应用程序必须确保不以HTTP 标题头中的任何未加密信息作为安全决策依据，因为攻击者要绕过这一标题头很容易。例如，标题头中的referer 字段包含来自请求源端的Web 页面的URL。不要根据referer字段的值做出任何安全决策（如检查请求是否来源于Web 应用程序生成的页面），因为该字段是很容易被伪造。

**规则6.1.3：**

不能依赖于客户端校验，必须使用服务端代码对输入数据进行最终校验。

说明：客户端的校验只能作为辅助手段，减少客户端和服务端的信息交互次数，如果依赖客户端校验，存在绕过校验的风险。

**规则6.1.4：**

对于在客户端已经做了输入校验，在服务器端再次以相同的规则进行校验时，一旦数据不合法，必须使会话失效，并记录告警日志。

说明：肯定存在攻击行为，攻击者绕过了客户端的输入校验，因此必须使会话失效，并记录日志。

**规则6.1.5：**

如果输入只允许包含某些特定的字符或字符的组合，则使用白名单进行输入校验。

说明：对于一些有规则可循的输入，如email地址、日期、小数等，使用正则表达式进行白名单校验，这样比使用黑名单进行校验更有效。

**规则6.1.6：**

如果输入为字符串参数则必须进行字符型合法性判断。

说明：可定义一个合法字符集。

实施指导：

|  |
| --- |
| String text = request.getParameter("text");  String characterPattern = "^[A-Za-z]\*$"; //开发者自行定义字符规则(方括号内的字符集)  if (!text.matches (characterPattern))  {  out.println (“Invalid Input”);  } |

**规则6.1.7：**

校验输入数据的长度。

说明：如果输入数据是字符串，必须校验字符串的长度是否符合要求，长度校验会加大攻击者实施攻击的难度。

**规则6.1.8：**

校验输入数据的范围。

说明：如果输入数据是数值，必须校验数值的范围是否正确，如年龄应该为0～150之间的正整数。

**规则6.1.9：**

禁止通过字符串拼接用户输入构造可执行SQL 语句。

说明：禁止通过字符串串联直接使用用户输入构造可执行SQL 语句，如：string sql = "select status from Users where UserName='" + txtUserName.Text + "'";这样很容易被SQL注入攻击。

**规则6.1.10：**

对于java/JSP语言，使用预编译语句PreparedStatement代替直接的语句执行Statement。

说明：使用预编译语句PreparedStatement，类型化SQL 参数将检查输入的类型，确保输入值在数据库中当作字符串、数字、日期或boolean等值而不是可执行代码进行处理，从而防止SQL注入攻击。而且，由于PreparedStatement 对象已预编译过，所以其执行速度要快于Statement 对象。因此，多次执行的SQL 语句经常创建为PreparedStatement 对象，还可以提高效率。

**规则6.1.11：**

用于重定向的输入参数不能包含回车和换行字符，以防止CRLF注入攻击。

说明：注意，“回车”字符有多种表示方式（CR = %0d = \r ），“换行”字符有多种表示方式（LF = %0a = \n）。

**规则6.1.12：**

如果服务端代码执行操作系统命令，禁止从客户端获取命令。

说明：如果服务端代码中使用Runtime.getRuntime().exec(cmd)或ProcessBuilder或subprocess等执行操作系统命令，那么禁止从客户端获取命令；而且最好不要从客户端获取命令的参数，如果必须从客户获取命令的参数，那么必须采用正则表达式对命令参数进行严格的校验，以防止命令注入（因为，一旦从客户端获取命令或参数，通过;&|<>符号，非常容易构造命令注入，危害系统）。

## 6.2 输出编码

**规则6.2.1：**

对于不可信的数据，输出到客户端前必须先进行HTML编码。

说明：不可信的数据（也就是其他业务系统生成的未经本应用程序验证的表数据或文件数据），通过对输出到客户端的数据进行HTML实体编码，可以防止浏览器将HTML渲染成标签，从而防止跨站脚本攻击。

## 6.3 上传下载

**规则6.3.1：（V\_1.0.2新增）**

必须在服务器端采用白名单方式对上传的MIME类型做严格的限制，上传的文件不可被覆盖，上传的文件大小必须进行严格的限制。

**规则6.3.2：**

禁止以用户提交的数据作为读/写/上传/下载文件的路径或文件名，以防止目录穿越和IDOR攻击。

说明：

建议对写/上传文件的路径或文件名采用随机方式生成，或将写/上传文件放置在有适当访问许可的专门目录。对读/下载文件采用映射表（例如，用户提交的读文件参数为1，则读取file1，参数为2，则读取file2）。防止恶意用户构造路径和文件名，实施目录穿越和攻击。IDOR攻击

**规则6.3.3：**

禁止将敏感文件（如日志文件、配置文件、数据库文件等）存放在Web内容目录下。

说明：Web内容目录指的是：通过Web可以直接浏览、访问的目录，存放在Web内容目录下的文件容易被攻击者直接下载。

## 6.4 异常处理

**规则6.4.1：**

使用全局异常处理方案，禁止直接将服务端的异常直接输出到客户端，暴露不必要的信息，只能向客户端返回一般性的错误提示消息。

说明：应用程序出现异常时，禁止将数据库版本、数据库结构、操作系统版本、堆栈跟踪、文件名和路径信息、SQL 查询字符串等对攻击者有用的信息返回给客户端。建议重定向到一个统一、默认的错误提示页面，进行信息过滤。

**规则6.4.2：**

应用程序捕获异常，并在日志中记录详细的错误信息。

说明：记录详细的错误消息，可供入侵检测及问题定位。

**规则6.4.3：（V\_1.0.2新增）**

使用准确的异常类型，以便能够更精确地识别问题和处理异常。

准确描述自定义类异常信息，在抛出和捕获异常时，必须提供有意义的异常消息，以便在日志中记录或在错误页面上显示。

**规则6.4.4：（V\_1.0.2新增）**

在设计和实现接口时，正确使用HTTP状态码，未正确使用状态码，将导致监控告警误报或漏报、问题定位难度增加

说明：

100 Continue: 客户端可以继续发送请求的剩余部分。  
101 Switching Protocols: 服务器正在根据客户端的请求切换协议。  
103 Early Hints: 用于提示客户端可以预加载某些资源。  
200 OK: 请求成功，具体的响应取决于请求方法（GET、POST 等）。  
201 Created: 请求成功且资源已被创建。  
202 Accepted: 请求已接受但尚未处理完成。  
204 No Content: 请求成功但无返回内容。  
301 Moved Permanently: 请求的资源已被永久移动到新位置。  
302 Found: 请求的资源临时从不同 URI 响应。  
303 See Other: 客户端应使用 GET 方法获取资源。  
304 Not Modified: 请求的资源未修改，客户端可以使用缓存的版本。  
307 Temporary Redirect: 请求临时重定向到另一个 URI，且方法不变。  
308 Permanent Redirect: 请求永久重定向到另一个 URI，且方法不变。  
400 Bad Request: 请求格式错误或参数无效。例如，缺少必需参数、参数格式不符。  
401 Unauthorized: 用户未认证或认证失败。  
403 Forbidden: 用户已认证，但无权访问该资源。  
404 Not Found: 请求的资源不存在。  
405 Method Not Allowed: 请求方法（如 GET、POST）不被允许。  
429 Too Many Requests: 客户端在短时间内发送了过多请求（限流）。  
500 Internal Server Error: 服务器内部错误，未明确分类的错误。  
501 Not Implemented: 请求的方法尚未实现。  
502 Bad Gateway: 服务器作为网关或代理，从上游服务器收到无效响应。  
503 Service Unavailable: 服务器暂时无法处理请求，通常用于维护或过载情况。  
504 Gateway Timeout: 服务器作为网关或代理，未能在规定时间内从上游服务器获得响应。

## 6.5 代码注释

**规则6.5.1：**

在注释信息中禁止包含物理路径信息。

**规则6.5.2：**

在注释信息中禁止包含数据库连接信息。

**规则6.5.3：**

在注释信息中禁止包含SQL语句信息。

**规则6.5.4：**

对于静态页面，在注释信息中禁止包含源代码信息。

**规则6.5.5：（V\_1.0.2新增）**

在注释中禁止包含AK、SK信息。

**规则6.5.6:（V\_1.0.2新增）**

在注释中不得包含测试账号/密码，超管账号/密码。

**规则6.5.7：**

发布到生产环境的前端页面，应清除所有注释内容。

## 6.6 归档要求

**规则6.6.1：**

版本归档时，必须删除开发过程中的临时文件、备份文件、无用目录等。

说明：恶意用户可以通过URL请求诸如.bak之类的文件，Web服务器会将这些文件以文本方式呈现给恶意用户，造成代码的泄漏，严重威胁Web应用的安全。

实施指导：

|  |
| --- |
| 在web应用的根目录下执行以下命令：  find ./ -name "\*.old" -o -name "\*.OLD" -o -name "\*.bak" -o -name "\*.BAK" -o -name "\*.temp" -o -name "\*.tmp" -o -name "\*.save" -o -name "\*.backup" -o -name "\*.orig" -o -name "\*.000" -o -name "\*~" -o -name "\*~1" -o -name "\*.dwt" -o -name "\*.tpl" -o -name "\*.zip" -o -name "\*.7z" -o -name "\*.rar" -o -name "\*.gz" -o -name "\*.tgz" -o -name "\*.tar" -o -name "\*.bz2"  分析查找到的文件是否临时文件、备份文件、无用文件，如果是则删除。 |

**规则6.6.2：**

归档的页面程序文件的扩展名必须使用小写字母。

说明：很多Web server对大小写是敏感的，但对后缀的大小写映像并没有做正确的处理。攻击者只要在URL中将JSP文件后缀从小写变成大写，Web服务器就不能正确处理这个文件后缀，而将其当作纯文本显示。攻击者可以通过查看源码获得这些程序的源代码。因此，归档的页面程序文件的扩展名必须使用小写字母，如jsp、html、htm、asp等页面程序文件的扩展名分别为jsp、html、htm、asp。

**规则6.6.3：**

归档的程序文件中禁止保留调试用的代码。

说明：这里的“调试用的代码”是指开发过程中进行临时调试所用的、在Web应用运行过程中不需要使用到的Web页面代码或servlet代码。例如：在代码开发过程中为了测试一个添加帐号的功能，开发人员临时编写了一个JSP页面进行测试，那么在归档时，该JSP页面必须删除，以免被攻击者利用。

**规则6.6.4:（V\_1.0.2新增）**

禁止将内部代码上传到github个人账号。

线上环境禁止将webpack配置成生成source-map，当Webpack配置为生成source-map时，攻击者可以通过它来查看原始的源码，从而获取敏感信息。

## 6.7 安全配置（V\_1.0.2新增）

**规则6.7.1：**actuator安全配置：

Spring Boot Actuator 提供了很多有用的监控和管理端点，但这些端点如果没有适当的安全措施，可能会泄露敏感信息。

禁用不必要的端点，禁用默认启用的端点，仅暴露需要的端点。例如，只启用 health 和 info 端点

配置端点访问权限，通过 Spring Security 配置对 Actuator 端点的访问权限。限制只有授权用户可以访问特定的 Actuator 端点。

日志记录 Actuator 端点的访问，为了更好地审计，记录访问 Actuator 端点的行为

**规则6.7.2：Swagger 安全配置**

Swagger 通过自动生成 API 文档简化了 API 的测试和调试，但暴露 API 细节可能会带来安全风险。

禁用生产环境中的 Swagger，为防止在生产环境暴露 API 文档，可以通过 Spring 配置文件或者代码来禁用 Swagger。

**实施指导**

@Configuration  
@Profile("!prod") // 仅在非生产环境下启用 Swagger  
public class SwaggerConfig {  
  
 @Bean  
 public Docket api() {  
 return new Docket(DocumentationType.SWAGGER\_2)  
 .select()  
 .apis(RequestHandlerSelectors.basePackage("com.example"))  
 .paths(PathSelectors.any())  
 .build();  
 }  
}

**规则6.7.3: 安全请求头配置**

安全请求头是确保 Web 应用程序安全的有效措施之一。可以通过配置 HTTP 响应头来增强 Web 应用的安全性，防止常见攻击，如 XSS 和 Clickjacking。

X-XSS-Protection: 防止 XSS 攻击。

**实施指导**

X-XSS-Protection: 1; mode=block

X-Content-Type-Options: 防止浏览器对文件类型进行猜测，避免 MIME 类型混淆攻击。

**实施指导**

X-Content-Type-Options: nosniff

Strict-Transport-Security (HSTS): 强制使用 HTTPS 协议，防止中间人攻击。

**实施指导**

Strict-Transport-Security: max-age=31536000; includeSubDomains

Content-Security-Policy (CSP): 防止 XSS 和其他代码注入攻击。

**实施指导**

Content-Security-Policy: default-src 'self'; script-src 'self'; object-src 'none'; style-src 'self'

X-Frame-Options: 防止 Clickjacking 攻击。

**实施指导**

X-Frame-Options: DENY

Referrer-Policy: 控制请求的 Referer 信息，避免泄露敏感信息。

**实施指导**

Referrer-Policy: no-referrer

可以通过 Spring Security 配置这些头部信息：

**实施指导**

@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .headers()  
 .contentSecurityPolicy("default-src 'self'; script-src 'self'; style-src 'self';")  
 .xssProtection().block(false)  
 .contentTypeOptions().noSniff()  
 .frameOptions().deny()  
 .httpStrictTransportSecurity().maxAgeInSeconds(31536000).includeSubDomains(true)  
 .referrerPolicy(ReferrerPolicy.NO\_REFERRER)  
 .and()  
 .authorizeRequests()  
 .anyRequest().authenticated();  
 }  
}

**规则6.7.4：CORS配置**

CORS（跨源资源共享）允许 Web 应用从不同的域进行请求。在处理 CORS 时，必须仔细配置，以确保只允许信任的域进行跨域请求。

通过 Spring Boot 配置全局 CORS 策略：

**实施指导**

@Configuration  
public class WebConfig implements WebMvcConfigurer {  
  
 @Override  
 public void addCorsMappings(CorsRegistry registry) {  
 registry.addMapping("/\*\*")  
 .allowedOrigins("https://trusted-domain.com") // 允许的来源  
 .allowedMethods("GET", "POST", "PUT", "DELETE") // 允许的方法  
 .allowedHeaders("\*") // 允许的请求头  
 .allowCredentials(true); // 是否允许发送凭证（如 Cookie）  
 }  
}

配置 CORS 策略以防止公开访问，如果应用程序不希望公开某些资源，可以针对特定路径或域限制 CORS 配置。

**实施指导**

@Configuration  
public class WebConfig implements WebMvcConfigurer {  
  
 @Override  
 public void addCorsMappings(CorsRegistry registry) {  
 registry.addMapping("/restricted/\*\*")  
 .allowedOrigins("https://trusted-domain.com")  
 .allowedMethods("GET");  
 }  
}

**规则6.7.5：HttpOnly配置：**

HttpOnly 属性是为了防止 JavaScript 访问敏感的 Cookie 数据，从而减少 XSS 攻击的风险。

启用 HttpOnly 属性，确保所有存储敏感信息的 Cookie 都设置为 HttpOnly，以确保它们不能通过 JavaScript 访问。

**实施指导**

@Configuration  
public class CookieConfig {  
  
 @Bean  
 public CookieSerializer cookieSerializer() {  
 DefaultCookieSerializer serializer = new DefaultCookieSerializer();  
 serializer.setUseHttpOnlyCookie(true); // 启用 HttpOnly 属性  
 serializer.setCookieName("SESSIONID");  
 return serializer;  
 }  
}

## 6.8 开源框架/组件/中间件使用规范（V\_1.0.2新增）

**规则6.8.1：**

禁止在生产环境使用Swagger、Tomcat/Java等远程调试模式，防止信息泄露和远程代码执行漏洞。

**规则6.8.2:**

中间件需修改默认口令、有默认密钥的必须重置，有未授权访问的服务应当修改绑定地址到127.0.0.1 ，禁止在内网或公网提供服务，例如： ElasticSearch、Apache Solr、Spring、Actuator、Redis。

**规则6.8.3:**

慎用或不用高风险组件解析请求参数，如： Fastjson、log4j、log4j2、XMLDecoder、XStream、Shiro ，全面禁 用 Fastjson1。

**规则6.8.4:**

所选择的任何框架、中间件、组件都需严格确认没有已知的安全问题，例如 Maven/Gradle 依赖可使用 MurphySec Code Scan、Checkmarx 等检测后不存在安全风险方可使用，可借助CNVD（<https://www.cnvd.org.cn/>）、CVE（<https://cve.mitre.org/index.html>） 漏洞库确认所使用的版本是否存在安全问题。

**规则6.8.5:**

新项目禁用或不用一些已过时或存在非常多安全问题的技术，如： Struts2、JDK<=1.6、Apache Axis、JSP、JSF、DWR、Java Applets 等。

**规则6.8.6:**

禁用有开源协议风险的组件，如：GPL（ GNU General Public License ，GNU通用公共许可证）、LGPL（ GNU Lesser General Public License ，GNU较宽松通用公共许可证）等

## 6.9 编码/加密规范：（V\_1.0.2新增）

**规则6.9.1：**

加密算法强度规范：

少用或不用弱加密算法： DES、RC4、MD5、SHA-1。

Hex、URLEncoder、Base64 是编码，不是加密算法，请勿滥用。

选择强加密算法，如：RSA、AES、SM（国密）。

密钥长度需符合安全规范，例如，使用RSA 2048而不是RSA 1024。

**规则6.9.2:**

密钥存储规范：

禁止硬编码，避免将密钥硬编码到应用程序代码中，因为容易被攻击者发现。

严格控制密钥访问权限，避免直接存储于缓存服务中，可使用访问控制列表（ACLs）或身份验证和授权来限制访问。

定期轮换密钥，以减少密钥泄露或滥用的风险。

定期备份存储的密钥，以防止丢失或损坏，备份应存储在安全的位置。

禁止将密钥直接存储于Git或其他版本管理工具中，尤其是Github、Gitee等开源平台。

使用密钥管理服务， 如果可能的话，使用专门的密钥管理服务（Key Management Service，KMS）来生成、存储和管理密钥。