金融科技SDL安全设计Checklist v1.0

检查类型	检查项(Checklist)
输入验证	校验跨信任边界传递的不可信数据(策略检查数据合法性，含白名单机制等)
	格式化字符串时，依然要检验用户输入的合法性，避免可造成系统信息泄露或者拒绝服务
	禁止向Java Runtime.exec()方法传递不可信、未净化的数据(当参数中包含空格，双引号，以-或者/符号开头表示一个参数开关时，可能会导致参数注入漏洞)，建议如果可以禁止JVM执行外部命令，未知漏洞的危害性会大大降低，可以大大提高JVM的安全性。
	验证路径之前应该先将其标准化为实际路径（特殊的文件名，比如“..”，symbolic links、hard links、shortcuts）
	从ZipInputStream提取文件，如果不在程序预期计划的目录之内时，应拒绝将其提取出来，或者将其提取到一个安全的位置
	从ZipInputStream提取文件，若解压之后的文件大小超过一定的限制时，必须拒绝将其解压
	在处理以前，验证所有来自客户端的数据，包括：所有参数、URL、HTTP头信息（比如：cookie名字和数据值），确定包括了来自 JavaScript、Flash 或其他嵌入代码的post 返回信息
	如果任何潜在的危险字符必须被作为输入，请确保您执行了额外的安全控制，比如：输入转义、输出编码、特定的安全 API等。部分常见的危险字符，包含但不限于： < > " ' % ( ) & + \ \' \"
	如果您使用的标准验证规则无法验证下面的输入，那么它们需要被单独验证，比如验证空字节 (%00)； 验证换行符 (%0d, %0a, \r, \n)； 验证路径替代字符“点-点-斜杠”（../或 ..\）；如果支持 UTF-8 扩展字符集编码，验证替代字符： %c0%ae%c0%ae/ (使用规范化验证双编码或其他类型的编码)
	严格验证来自重定向输入的数据（一个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任何验证）
	验证数据类型
	验证数据范围
	验证数据长度
输出编码	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
	通过语义输出编码方式，对所有从服务端返回到客户端的数据进行编码。比如HTML编码、URL编码等，编码形式需根据具体的应用场景选择
	除非对目标编译器是安全的，否则请对所有字符进行编码
	针对 SQL、XML 和 LDAP 查询，语义净化所有不可信数据的输出
	对于操作系统命令，净化所有不可信数据输出
异常处理	禁止在异常中泄露敏感信息（敏感数据的范围应该基于应用场景以及产品威胁分析的结果来确定。典型的敏感数据包括口令、银行账号、个人信息、通讯记录、密钥等）
	禁止在异常中泄露应用服务器的指纹信息（如版本，路径，架构）
	方法发生异常时要恢复到之前的对象状态（业务操作失败时，进行回滚业务；或者避免去修改对象状态，维持对象状态一致性）
I/O操作	临时文件使用完毕应及时删除
	不要将Buffer对象封装的数据暴露给不可信代码
	在多用户系统中创建文件时指定合适的访问许可，以防止未授权的文件访问
	当一个外部进程通过其输出流对外输出信息或错误时，必须及时清空其输出流，以防止输出流中的缓冲区被耗尽而导致外部进程被阻塞。
	白名单控制共享目录操作文件权限，比如读/写/可执行权限
运行环境	不要使用危险的许可与目标组合（比如不要将AllPermission许可赋予给不信任的代码，不要将ReflectPermission许可和suppressAccessChecks目标组合使用，不要将java.lang.RuntimePermission许可与createClassLoader目标组合）
	不要禁用JVM字节码验证，如果使用的字节码，如class文件被恶意篡改过，将会存在安全风险
	建议监控平台不要对互联网开放，仅限于内网环境访问；如果监控平台存在远程执行漏洞，将会给所监控的应用带来安全风险
	建议将所有安全敏感代码(例如进行权限控制或者用户名密码校验的代码)都放在一个jar包中
	生产代码不能包含任何调试代码或接口
身份验证	除了那些特定设为“公开”的内容以外，对所有的网页和资源都要求进行身份验证，并正确设计身份验证功能
	所有的身份验证过程必须在服务器后端上执行
	在任何可能的情况下，建立并使用标准的、已通过安全测试的身份验证服务(比如 C4A)
	所有的身份验证控制应当安全的处理未成功的身份验证，比如给出错误模糊提示，隐藏敏感信息
	登录入口应具有防止暴力猜解及撞库猜解（利用已泄漏的密码字典进行批量登录尝试）的措施，超过设定失败次数需要启用锁定或图片随机码进行访问限制
	采用https post请求方式传输身份验证的凭据信息
	身份验证的失败提示信息采用模糊处理，比如可以使用“用户名或密码错误”，而不要使用“用户名错误”或者“密码错误”明确提示。
	涉及敏感信息或功能的外部系统连接应配置身份验证功能，并进行有效身份验证控制
	在执行关键操作（如个人信息密码修改操作）时，应对用户身份进行再次验证
	为高度敏感或重要的交易账户使用多因子身份验证机制，如支付密码、短信验证码等
短信验证码	一次一用
	发送频率控制（建议60s获取一次）
	验证码有效期（建议60s内有效，发短信时进行友好提示）
	复杂度（短信验证码建议6位数字）
	安全提示：是否是个人自己操作等风险提示信息
	在前端校验（客户端的校验只能作为辅助手段，很容易被绕过），必须使用服务端代码对输入数据进行最终校验
图形验证码	一次一用
	验证码有效期(10分钟内有效，可根据场景兼容安全和体验灵活设置)
	复杂度（4位及以上数字、字母交替），根据需要也可采用当下流行的拖拽验证码或计算值的验证方式
	服务器端进行认证
	从用户体验和安全角度出发，可设计为当用户输3次错误密码后自动弹出验证码输入框进行验证操作

检查类型	检查项(Checklist)
密码管理	禁止使用私有或者弱加密算法（比如禁止使用DES，SHA1等，推荐使用AES: 128位，RSA: 2048位，DSA: 2048位）
	采用基于哈希算法和加入盐值（salt）方式安全存储口令信息
	密码输入框，可设计为显示密码和隐藏密码切换功能
	密码重设和更改操作，需要进行二次合法身份验证
	密码重设时，应对注册手机号和邮箱进行有效验证，链接只能发送到预先注册的邮件地址或预先绑定的手机号
	临时密码和链接应设计一个短暂的有效期（比如5分钟），防止暴力破解
	当密码重新设置时，应短信通知用户是否是本人在操作，告知安全风险
	密码复杂度设置：建议8个字符以上，包含字母、数字及特殊字符等
	密码设置场景中应具有密码复杂度检查功能
	密码不能输出到日志和控制台
	数据库连接配置中的用户密码要以加密的形式存储
	建议设计密码定期修改提醒机制
会话安全	用户登出后应立即清理会话及其相关登录信息
	注销功能应当完全终止相关的会话或连接
	增加Cookie 安全性，添加“HttpOnly”和“secure”属性（当“secure”属性设置为true时表示创建的 Cookie 会被以安全的形式向服务器传输，也就是只能在HTTPS 连接中被浏览器传递到服务器端进行会话验证，在HTTP 连接中不会传递该信息，也就不会存在Cookie被窃取的问题；设置了"HttpOnly"属性，通过程序(JS脚本、Applet等)将无法读取到Cookie信息，这样也能减少XSS跨站脚本攻击风险）
	会话cookie应设计有效期，超时后立即失效
	当设计允许用户在多渠道终端同时登录时，建议应进行常用设备登录限制
	为包含已验证的会话标识符的 cookie 设置域和路径，为站点设置一个恰当的限制值。默认cookie的域是当前域名，默认cookie的路径是当前页面的目录路径。如果想要跨域或者在其他的路径下访问cookie就必须要重新设置这两个属性，domain和path。
	注销功能应当可用于所有受身份验证保护的网页
	在平衡风险和业务功能需求的基础上，设置一个尽量短的会话超时时间。通常情况下，应当不超过几个小时。
	不要在URL、错误信息或日志中暴露会话标识符，会话标识符应当只出现在http头信息中，不要将会话标识符以 GET 参数进行传递
	定期生成一个新的会话标识符并周期性地使上一个会话标识符失效（这可以缓解那些原标识符被获得的特定会话劫持情况）
	在身份验证的时候，如果连接从 HTTP 变为 HTTPS，则会生成一个新的会话标识符。在应用程序中，推荐持续使用 HTTPS，不应在 HTTP 和 HTTPS 之间来回转换，有效避免切换过程会话被劫持篡改。
	为服务器端的操作执行标准的安全会话管理，为每个会话执行合法的身份验证和权限控制，防止存在CSRF跨站点请求伪造漏洞
访问控制	将具有特权的逻辑从其他应用程序代码中隔离开
	限制只有授权的用户才能访问文件资源
	限制只有授权的用户才能访问受保护的URL
	限制只有授权的用户才能访问受保护的功能或服务
	建议只有授权的用户才能访问直接对象引用
	限制只有授权的用户才能访问受保护的应用程序数据
	限制只有授权的用户才能访问与安全相关的配置信息
	限制只有授权的外部应用程序或接口才能访问受保护的本地程序或资源
	服务器端执行的访问控制规则和前端实施的访问控制规则必须匹配
	服务器中创建文件时需指定合理的访问权限（读/写/可执行）
	当权限重新设置发生变更时，应记录好日志，并短信通知用户是否是本人在操作，告知可能存在的安全风险
日志规范	不要在日志中保存敏感信息，包括系统指纹信息、会话标识符、账号密码、证件、ID等
	确保日志记录包含了重要的日志事件数据
	记录所有失败和成功的输入验证
	记录所有失败和成功的身份验证记录
	记录所有失败和成功的访问和操作记录
	记录明显的修改事件，包括对于状态数据的修改
	记录连接无效或者已过期的会话令牌尝试
	记录所有的管理功能操作行为，包含但不限于安全配置设置的变更
	记录所有失败和成功的后端连接
	记录加密模块的错误信息
敏感信息	临时产生的敏感数据（写入内存或文件），应具有及时清除和释放机制
	不要在 HTTP GET 请求参数中包含敏感信息，如用户名、密码、卡号、ID等
	禁止表单中的自动填充功能，因为表单中可能包含敏感信息，包括身份验证信息
	不要在客户端上以明文形式保存密码或其他敏感信息
	为所有敏感信息采用SSL加密传输
	禁止将敏感信息（包含加密秘钥等）硬编码在程序中
	禁止明文存储用户的密码、身份证号、银行卡号、持卡人姓名等敏感信息
	不要在日志中保存敏感信息，包含但不限于系统详细信息、会话标识符、密码等
	禁止在异常中泄露应用服务器的指纹信息，如版本，路径，组件版本等
	禁止将源码或sql上传到开源平台或社区，如github、开源中国等
	请求中含有敏感参数（如订单号、ID等），应进行混淆方式处理，防止产生参数遍历获取信息风险
	敏感信息需要展示在web页面上时，应在后台进行敏感字段脱敏处理
	请求返回数据不应包含请求之外的业务数据，特别是敏感信息数据

检查类型	检查项(Checklist)
密码找回安全	服务器端要做认证，避免绕过前端控制
	增加二次认证因子，如验证码
	涉及登录验证token之类的，不要直接将验证内容直接返回给用户
	认证凭证加密，推荐强算法（推荐使用AES: 128位，RSA: 2048位，DSA: 2048位）
	认证凭证中的参数应进行混淆处理
	在多个验证操作中，要对各验证机制进行排序，以防出现跳过前面验证机制直接到最后一步认证的安全风险
	手机短信码验证，需同时校验手机号和短信是否对应
	输入框中，应校验输入数据合法性，防止产生XSS跨站脚本攻击
SQL注入	永远不要信任用户的输入，要对用户的所有输入进行校验，包含SQL语句的过滤和转义
	永远不要使用动态拼装SQL，可以使用参数化的SQL或者使用存储过程进行数据查询存取
	永远不要使用管理员权限进行数据库连接，为每个应用使用单独的非特权权限，且配置有限的数据库连接数
	不要把敏感信息明文存放，采用加密或者哈希、混淆等方式对敏感信息进行脱敏存储
	应用的异常信息应不带有敏感信息，给出尽可能少的提示；建议使用自定义的错误信息对原始错误信息进行包装，可把异常信息存放在独立的数据库表中
XML注入	不要使用字符串/StringBuffer/StringBuilder/StringFormat组装XML
	建议对XML元素属性或者内容进行转义
XSS跨站脚本攻击	对输入的数据进行过滤和转义，包含但不限于< >" ' % ( ) & + \ \' \"等危险特殊字符
	数据添加到html元素属性或者内容中时，对数据进行HTML转义
	数据添加到script脚本中时，对数据进行script转义
	数据添加到style中时，对数据进行css转义
CSRF跨站请求伪造	建议在每个关键表单中引入了CSRF Token验证(会话中生成的随机串，提交后校验)
	在关键表单提交时要求用户进行二次身份验证（录入密码、插KEY、输入图片验证码、短信验证码）
	对请求refer做验证（比如跨域、系统内部应用）
文件上传安全	上传操作应设计身份验证机制，并进行合法身份校验
	只允许上传满足业务需要的相关文档类型
	通过检查文件头信息，比如JPEG (jpg)文件头信息（十六进制）：FFD8FF，验证上传文档是否是所期待的类型
	不要把文件保存在与应用程序相同的 Web 环境中，建议将文件保存在专用的文档服务器中，单独给文档服务器配置域名访问更好
	限制上传任意可能被 Web 服务器解析的文件 ，比如jsp、php等
	上传文件以二进制形式下载，建议不提供直接访问（防止木马文件直接执行）
	禁止授予上传文件存储目录的可执行权限
组件安全	在使用随机数函数时，推荐使用强随机数函数（例如java.security.SecureRandom类）
	精简组件中不需要的功能、方法，以免带来未知的安全风险
	不可将系统内部使用的锁对象暴露给不可信代码
	建议使用SSL Socket代替Socket来进行安全数据交互
	封装本地方法调用（所有的本地方法都应该被定义为私有的，然后仅通过一个封装方法来调用）
	使用安全管理器（比如java.security或第三方安全组件）来保护敏感操作
	编写自定义类加载器必须覆盖getPermissions()函数时，在为代码源分配任意权限前，应调用超类super.getPermissions（）函数，实现除了自定义策略外，系统全局的默认安全策略也被应用。
	避免完全依赖URLClassLoader和java.util.jar提供的默认自动签名认证机制，应从加载类的代码源（Code-Source）中获取证书链，然后检查证书是否属于本地密钥库（KeyStore）中的受信任签名者
接口安全	调用方来源IP控制，比如可通过防火墙、主机host deny、Nginx deny等技术措施进行实施
	调用方身份认证，比如key、secret、证书等技术措施进行实施
	调用参数认证，需设计参数容错机制，避免出现参数可遍历敏感数据安全问题
	采用数字签名保障接口身份来源可信，数据防篡改
	调用方权限控制设置
	调用频率、有效期进行控制
	调用行为实时检测，对异常阻拦
	幂等性校验，保持数据一致性
	采用应用接入安全网关，实现APPID/KEY身份认证，加密传输，摘要签名安全保障
Dubbo调用安全	采用token验证访问控制，防止消费者绕过注册中心访问提供者；在注册中心控制权限以决定要不要下发令牌给消费者
	采用filter IP白名单访问控制，同时也可预防生产系统和测试系统之间Dubbo混乱调用问题
	在必要情况下（如敏感信息操作），连接注册中心Dubbo时要进行用户名和密码校验
Redis调用安全	应启用客户端IP访问控制验证功能
	应启用客户端身份验证功能
	敏感信息不要明文存储于Redis