安全开发手册

安全规范及最佳实践

支付宝系统安全中心

2014/1/1

本文档为支付宝系统安全 SDL 团队在不断参与项目过程中总结的安全解决方案，并会持续的进行更新。

版本	备注	修改人
V0.1	初稿	徐庶

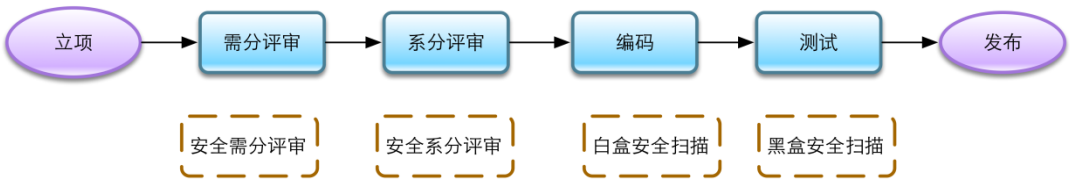
目录

1 目标

主要介绍支付宝开发过程中常见的安全注意点以及解决方案，从而保障开发过程中的安全性。

2 安全开发流程

支付宝目前执行 SDL 安全开发流程，详细流程点击查看，主要流程见下图



安全工程师会对项目进行安全需分以及安全系分评审，并给出安全建议。在开发、测试阶段需要完成建议的实现以及对应的测试，交付时需要进行白盒安全扫描，测试时需要进行黑盒安全扫描。

安全扫描平台	scan.alipay.net

3 常见 web 安全及规范

如果使用 sofa 框架进行开发，编码规范参考[sofa 安全编码规范]

3.1 XSS

对于所有使用 velocity 的变量渲染，都是默认进行编码转义，默认安全。

并且提供了转义宏以供特殊场景使用，

宏	备注	注意点
SLITERAL	不转义、原样输入	**必须经安全工程师确认**
SHTML	保留 html 标签、去除 JS 等恶意代码	非 HTML 场景不使用

SURL	变量为 URL 时使用	变量必须在””中

详细见[sofa 安全编码规范]。

3.2 CSRF

CSRF 针对表单提交以及 JSON/JSONP 提交提供了不同的解决方案。

针对表单的 CSRF 解决方案是在表单所在页面添加 token ,提交时与服务端保存的进行验证。

针对 JSON/JSONP 的 CSRF，是通过在 cookie 中增加 ctoken，每次请求时带上 ctoken 作为参数，服务端检查如果没有 ctoken 直接拦截，有的话与 cookie 中的进行比较。

详细使用方法见[sofa 安全编码规范]。

3.3 SQL 注入

为防止 sql 注入带来的风险，采用预编译的方式进行 sql 语句的调用，sofa 框架使用的是 ibatis，对于变量的占位符使用**#variable#**，不使用？variable？

如果使用其他框架，全部使用预编译方式进行 sql 执行。

3.4 URL 跳转

URL 跳转，sofa 框架中预设了跳转白名单，限制在集团公司网站内。

跳转地址不能通过 URL 上参数简单的进行指定，如果通过 URL 指定，需要对 URL 上所有参数进行签名，防止参数被篡改。

3.5 权限控制

应用需要关注的权限主要分成 2 种，一种是垂直权限，一种是水平权限。

垂直权限，不同角色之间权限要严格区分进行验证

水平权限

1. 不可通过变更参数查看他人信息，在需要用户登陆的情况下，数据库查询时可添加 userid 限制为会话中 userid 来保障水平权限
2. 在***未登陆，不能区分操作人的情况***下，会通过对所有参数进行摘要或者加密串的方式来实现内容的访问控制。此时必须要
   a) 通过摘要---必须参数中包含时间戳
   b) 通过加密串一加密内容必须包含时间戳

   验证时通过对比时间戳来达成服务的有效性校验，防止泄露、遍历以及搜索引擎可能的收录。

4 信息安全

应用系统中涉及的敏感信息主要分成 2 类，

一类是会员相关信息，在前端系统以及后台管理系统中都会存在

一类是公司经营相关的数据 ,例如费率、经营信息 ,该类数据主要集中在后台系统。

4.1 展示

敏感信息的范围以及前台展示规范详细参看《支付宝会员信息展示规范》，隐藏规则同时可以参看附录一。

4.2 传输

如果需要进行信息传输，参考下面表格，确保数据的完整性、保密性和可用性。

交互形式	传输通道	传输方式	注意点
页面交互，外部接口	HTTPS	**数据进行签名**，涉及用户敏感信息需	防止请求被搜索引擎收录、被他人遍历等风险，需要有失效

		要对敏感字段进行加密	时间，请求中增加时间因子。有效时间 15 分钟 ,对于具备单次有效特征的请求，业务完成后立即失效。（例如信任登陆，登陆结束即失效；支付行为，支付完成后即失效）
数据交互	专线/VPN/SFTP		

允许	禁止
敏感字段进行加密传输	通过 URL 传输未经过加密的敏感信息，用户初始提交的除外

4.3 存储

关于敏感数据加密存储：

a) 用户密码存储必须加盐存储，各用户盐值不同。

b) 信用卡数据一律存信用卡数据中心（CCDC）

c) 借记卡、证件号必须加密存储，且建议数据落地于卡数据中心（cifin），最终数据会落地于卡数据中心的，业务处理中间过程中需存储借记卡、证件号于数据库时使用与 cifin 相同的算法和密钥加密。

如果数据最终不落地于 cifin，统一使用密钥 dbencryption-aes128 进行 AES 加密。

Cifin 加密存储数据使用的密钥信息：

字段	密钥名称	算法和密钥长度
卡数据中心银行卡号加密	cifin-cardNo-aes128	AES-128
卡数据中心证件号加密	cifin-certNo-aes128	AES-128
卡数据中心手机号加密	cifin-mobile-aes128	AES-128
卡数据中心信用卡有效期加密	cifin-validDate-aes128	AES-128

5 信用卡

鉴于信用卡号的敏感性以及 PCI 合规要求 ,对于信用卡的处理 ,存储进行特别要求。

1. 处理信用卡信息系统需要部署在 CCDC 安全域中
2. 信用卡号由 CCDC 系统统一管理，不存储 CVV2 信息
3. 不得在日志中打印完整信用卡号
4. 页面上信用卡号展示前六后四，CVV2/CVC2 以及有效期不展示。

详细可参看文档 PCI-DSS V3

6 日志管理

详细内容见《支付宝日志规范》。

7 安全产品

在关键业务点、存在利用价值的页面(返回部分隐藏名字的 JSON，促销活动查看是否能看兑奖的 JSON)或者其他提供服务的接口处，为了防止外部的攻击，需要配置相应的安全产品进行防护

安全产品	简介	使用场景
图片校验码	通过展示图形校验码 ,增加机器识别的难度，增加攻击成本 ,从而降低、缓解机器程序的攻击。	为了防止程序恶意的进行业务攻击，在业务的入口或者业务最终提交处，可以增加校验码。 例如：登陆口，转账确认提交。 校验码可以与 RDS 配合使用，RDS 判断为机器行为时启用图片校验码。
RDS	人机识别系统 ,通过收集的行为数据来判断是机器程序行为还是人为行为。提供的判断	对于以下场景，建议接入 RDS 进行安全防护 1. 登陆、注册功能

	服务返回机器或者人。以供调用系统后续采用不同的处理措施。	2. 能返回指定用户信息（包括输入账号，返回部分隐藏的姓名；） 3. 校验密码（登陆密码、支付密码等各类密码） 4. 营销类活动（秒杀类活动可以不接入 RDS 而通过增加随机回答问题的方式进行防护；非秒杀类，持续进行的营销，抽奖活动可接入 RDS）
HUMMOCK/TMD	在 apache/nginx 层对于异常访问流量进行拦截，防止 CC 攻击。通过一段时间内的访问次数、频率来判断是否是攻击行为。	使用 tengine 或者 apache 时都需要进行配置。 可以针对某一个链接制定单独的策略。

8 无线

8.1 无线基础安全

8.1.1 Intent 组件安全

安全关注点

当一个正常的应用组件发送隐式的 intent 时 ,如果没有对接收组件的权限进行控制（安全的权限级别为 signature 和 signatureOrSystem），那么恶意应用的导出组件也能定义相应的 intent filter（除了系统 Action 外，可以包含 Intent 中所有的 Action、Data、Category）来接收，甚至截获、篡改该 intent

解决方案

1) 应用组件之间的通信（ICC）尽量使用显式 intent；intent 中不要携带敏感数据（如密码、token、卡号等）。
2) 从 intent 接收者的角度看，intent Filter 并不是一个安全边界。开发者应该意识到：声明一个 intent filter 时会将该组件导出，并暴露在攻击之下。因此为了组件更安全，应该避免导出组件，除非这个组件是专门设计用来处理外部程

序请求或者接收系统消息的，同时关键的逻辑或状态转换动作不要放在导出组件里。

3) 如果一个组件必须要导出(如接收系统广播)，只有指定发送 intent 给它的组件所需要的权限才能保证安全(signature 或 signatureOrSystem 才被认为是安全的权限级别），并且在处理接收到的消息之前，需要对接收到的 intent（包括发送者身份、动作、数据等）做合法性检查，看是否是期望接收到的消息。

4) 内部使用的组件可设置为私有，<android:exported="false">。

5) 对于关键广播的发送组件，需要明确指定广播接收者应该具有的接收权限（ signature 或 signatureOrSystem ）。对于广播接收者，可以定义发送消息给它的发送者必须具有的发送权限。避免使用 sticky broadcast。

6) 为防止重要数据泄露，应用的 Content Provider 必须在 manifest 文件中指定 read 权限。

8.1.2 IOS 目录遍历

安全关注点

如果攻击者能够控制文件名的部分内容，那么 NSFileManager 和 NSFileHandle 都可能遭受目录遍历的攻击（ Directory Traversal ）

解决方案

对程序中接受用户输入的数据进行严格的格式、内容（包括长度、类型、是否包含非法字符等）过滤，并在页面上展示时进行相应的编码

8.1.3 IOS keychain 风险与防护

安全关注点

在一个越狱过的 iOS 设备上，可以写一个应用加入到所有应用的组中，这样它就可以访问所有的 keychain 项

利用 DFU mode 漏洞，挂载自制的 ramdisk 对 apple A4 芯片的设备进行暴力破解用户密码和 keychain 数据项

解决方案

Keychain 中的条目创建时默认的保护层级是 kSecAttrAccessibleAlways，即在任何时刻都可以被解密并移植到其他设备上。如果使用了-ThisDeviceOnly 保护类，keychain 中条目将会被一个从设备密钥衍生出的密钥加密保存，这保证了只有在该设备上才能对该条目进行解密

8.1.4 IOS 本地 sql 注入风险与防护

安全关注点

iOS 应用通常会在客户端利用 SQLite 数据库存储一些应用数据，如果 sql 语句采用拼接参数的方式，并且没有对用户输入的数据进行过滤，可能会导致 sql 注入

解决方案

防范 sql 注入的方法是进行变量绑定和参数化的查询

8.1.5 Andorid 日志输出

安全关注点

如果应用的安装包在上线时没有关闭日志输出开关，则上线后可能会在日志中输出一些敏感的数据信息

解决方案

可以对 android.util.Log 类进行封装，添加一个日志输出开关，在测试环境下把日志开关设置为 true 进行输出调试用，而在上线时将开关关闭

8.1.6 IOS 日志输出

安全关注点

在开发和测试环境中，NSLog 经常用来输出日志信息来调试应用，但在线上发布的产品，需要避免在 NSLog 中输出敏感数据

解决方案

Xcode 中添加一个自定义的 Debug 编译标记，同时在代码中定义一个条件编译的宏

8.1.7 Android 防止逆向破解

安全关注点

如果 android 应用在编译时没有做反编译的混淆处理，应用的 apk 包在被恶意攻击者反编译后，可能会暴露数据传输、加密、存储等关键代码或业务逻辑。

apk 包还可能被注入恶意代码后重新编译、打包、签名，如果服务端未对应用的签名进行校验的话，可能会存在恶意的假冒客户端与服务端交互，并对用户进行钓鱼欺诈或资金窃取。

解决方案

1）代码混淆保护。应用发布前，需要对 Java 开发的应用进行代码混淆后再编译，提高逆向破解的难度。目前可利用 android sdk 在 eclipse 中集成的 proguard 插件，在应用编译前，在 eclipse 工程目录下的 proguard.cfg 文件中添加对核心代码类的混淆规则

2）建议对关键业务逻辑的代码用 C 和 JNI 来实现，可以对 so 文件进行加壳，如 UPX；JNI 的 so 库文件必须对调用它的应用进行签名校验，避免核心流程被人直接调用

3）对于应用自身的 apk 文件和 so 库文件在执行之前需要校验签名是否正确

8.1.8 OC 缓冲区溢出

安全关注点

缓冲区溢出包括栈溢出和堆溢出，指堆、栈预留分配的内存不够存储输入的数据，并且没有被截断，将会导致数据溢出，覆盖其他数据。如果输入的数据是精心构造的恶意代码，就可能执行恶意操作

解决方案

在编译应用时添加-fstack-protector-all 标记，避免栈溢出

在使用缓冲区时首先计算将要存储对象的大小，避免使用硬编码缓冲区大小（如果代码变更，需要对所有硬编码的数值进行修改，如有遗漏，就可能造成溢出）

String 操作尽量用 NSString 对象，如果要用 c 风格函数，应该用 strl、strn 家族的函数，例如 strlcat、strlcpy、[v]snprintf

8.1.9 OC 整数溢出

安全关注点

缓冲区大小由用户决定，假如恶意用户输入一个 integer 类型无法表示的整数，可能造成应用崩溃，或者造成其他问题

解决方案

错误的代码

```
size_t bytes = n * m; //可能造成溢出
if (bytes < n || bytes < m) { /* 错误的判断方法 */
    ... /* allocate "bytes" space */
}
```

正确的代码

```
#define SIZE_MAX 2147483647
```

```
    size_t bytes = n * m;
if (n > 0 && m > 0 && SIZE_MAX/n >= m) {
    ... /* allocate "bytes" space */
}
```

8.1.10 OC 格式化字符串

安全关注点

格式化字符串漏洞的主要原因是我们在使用[v][f]printf 等函数时没有严格检查输入参数的个数，比如正常的 printf 函数格式应该是： printf(“格式控制字符串”，变量列表)； 而我们经常会省略掉后面的变量列表或者变量个数，导致跟前面的格式控制符没有一一对应，如果格式控制的字符串是可以被控制的，那么可以利用特殊的格式控制符%x 读取内存数据，用%n 写入数据，甚至经过精心构筑后可以向任意地址写入任意数据

解决方案

在使用以下 Object-C 类和方法进行格式化字符串输出时，可参照 https://developer.apple.com/library/ios/#documentation/CoreFoundation/Conceptual/CFStrings/formatSpecifiers.html

8.1.11 OC 重复释放指针

安全关注点

指针指向的对象的重复释放会造成程序崩溃等

解决方案

为避免 C 编程带来的风险，可以在 iOS 程序开发完之后，采用下面的软件进行静态代码扫描，FlawFinder，Clang Static Analyzer

8.1.12 OC Use-after-free

安全关注点

使用已经释放的对象指针，同样会造成程序崩溃或者被恶意攻击利用

解决方案

为避免 C 编程带来的风险，可以在 iOS 程序开发完之后，采用下面的软件进行静态代码扫描，FlawFinder，Clang Static Analyzer

8.1.13 Android Webview 安全

安全关注点

Android 应用可以使用 webview 组件来展示 web 页面。Android 应用可以在应用代码中利用 addJavascriptInterface 接口给 javascript 代码调用。webview 中可以启用 javascript，调用本地暴露的接口。

解决方案

使用 addJavascriptInterface 接口允许 js 脚本控制 android 应用 如果嵌入的 html 页面的部分内容可由用户控制，恶意攻击者就会在 html 页面嵌入能够执行本地应用的恶意代码

8.1.14 IOS XSS 漏洞

安全关注点

UIWebView 是 iOS web 页面的展示渲染引擎，支持多种文件格式，同时还支持 javascript。当用户控制的变量未经过滤和编码在 WebView 展示时，就可能产生 XSS 漏洞，窃取用户的隐私数据，比如会话 token、本地存储的敏感数据等

解决方案

对 XSS 漏洞的防护措施主要有（1）对用户输入的非法字符过滤，比如<、>、’等。（2）在页面展示用户输入的变量时，进行相应的编码，比如 html 编码展示

8.1.15 XML 文件解析

安全关注点

攻击方式是让 XML 解析器解析外部的 XML 文档

解决方案

不打开 NSXMLParser 的 setShouldResolveExternalEntities:YES 选项来解析外部的 xml 文档。

不使用 libxml2 解析外部的 xml 文档

8.2 无线敏感信息

8.2.1 Android 敏感数据的传输

安全关注点

为保证数据传输过程中的安全，包括数据的机密性和完整性，客户端与服务端之间的通信采用 https 协议。SSL/TLS 通信中，客户端通过数字证书判断服务器是否可信，并采用与服务器协商的会话密钥进行加密通信。在有些情况下客户端发现证书无法与系统可信根 CA 形成信任链，出现了 CertificateException 等异常，这时就会中断检查服务器证书的有效性，或选择接受所有的证书。这种实现可能导致中间人攻击。

因为 https 协议本身在各个 OS 实现过程中也可能出现漏洞，建议对某些敏感字段做应用层加密签名。

解决方案

1）代码中更精确地直接验证服务端的证书是否是某张特定的证书，这种方法称为“证书锁定”。实现证书锁定的方法有两种：一种是实现 X509TrustManager 接口，另一种则是使用 KeyStore。

2）除 https 协议本身提供的身份认证、数据加密、签名的保护机制外，建议对一些重要数据（如用户的密码、交易订单等）进行业务层的数据加密和签名

8.2.2 Android 敏感数据的存储

安全关注点

敏感数据如果以明文保存，降低了敏感数据获取的难度，并在特定场合有信息泄露的风险。

解决方案

敏感数据的重要字段应该在数据库中以密文或散列方式存储，并标注为敏感字段。

8.2.3 Android 敏感数据的展现

安全关注点

如果把敏感数据直接展示在页面上，会造成信息泄露。

解决方案

7) 对需要展现在页面上的敏感数据采取截断处理。如：用“*”掩盖用户银行卡号、信用卡号部分内容，只留末几位进行显示。
8) 对必须要完整展现的内容（如金额等），应该使用混淆技术或图片方式进行输出。

8.2.4 Android 数据的本地存储

安全关注点

Android 应用开发中可以利用 Shared Preferences、File、SQLite 提供的 API 在设备内部、外部存储（SD 卡或其他应用共享的存储）或应用目录下创建并访问文件、数据库。注意这些数据的访问控制权限、sqlite 注入风险、webview 缓存密码的设置

解决方案

1）文件访问控制：GetSharedPreferences、openFileOutput 、openOrCreateDatabase 等方法的参数中通常包含一个 int 型的文件创建模式 - mode，它的取值主要有以下几种 MODE_PRIVATE 应用程序私有，MODE_WORLD_READABLE 所有应用可读，MODE_WORLD_WRITEABLE 所有应用

可写。建议采用 MODE_PRIVATE，避免其他应用能够访问。如果文件或数据库表中需要保存敏感数据，必须进行强加密存储或者存储 hash 值，加密密钥不能同时存储在移动设备中。应用卸载时，必须删除这些机密数据。

2）SQLite sql 注入风险，防御 sql 注入的方法是采用参数化的查询方式，而不是拼接字符串。

3）webview 缓存密码的设置，为了避免 webview 默认保存用户的密码，开发者在初始化 webview 实例后，请加上一行代码：theWebView.getSettings().setSavePassword(false)，将其设置为不保存密码，就不会提示用户选择保存密码。

8.2.5 IOS 敏感数据的传输

安全关注点

为保证数据传输过程中的安全，包括数据的机密性和完整性，客户端与服务端之间的通信应该利用 iOS SDK SSL 库中的 NSURLRequest、NSURLConnection、CFNetwork 等类，采用 https 通信

攻击者还可以向自己的手机安装可信证书，然后利用 Charles 或者 burp suite 进行 https 中间人抓包分析，防御此类分析可采用 ssl pinning 技术

解决方案

尽量不要使用私有协议进行通信

客户端与服务端之间的通信应该利用 iOS SDK SSL 库中的 NSURLRequest、NSURLConnection、CFNetwork 等类，采用 https 通信

禁止应用与使用自签名或无效证书的服务端进行通信

将服务端证书打包到客户端应用中，在 NSURLConnectionDelegate protocol 的 connection:willSendRequestForAuthenticationChallenge 中比较获取的服务端证书与客户端内置的证书是否一致

8.2.6 IOS 数据的本地存储

安全关注点

如果在 iOS 设备的本地配置文件、数据库中存储用户的敏感数据，比如永久的会话 token、用户明文密码、银行卡信息，可能造成数据的泄露，同时也违反央行、PCI 合规

解决方案

为了保护本地存储的重要数据，建议采用 iOS 提供的数据保护 API 和加密机制

9 后台管理系统

9.1 登陆控制

详细内容参看文档，[后台权限管理模块基本准则]

1. 双因素认证
   a) 静态密码+token
   b) 动态密码+证书

9.2 权限控制

1. 所有操作都要有对应的权限值
2. 查看/更改操作的权限分离，不同业务，不同安全级别不能复用之前权限值
3. 根据业务要求能够实现权限互斥

9.3 操作审计

1. 所有操作都有日志记录，包含
   a) 操作人
   b) 操作时间

c) 操作内容

d) 操作返回结果

10 加密与签名

场景	通用场景推荐算法
加密	AES128
摘要	SHA256
签名	RSA2048

均使用 KMI 进行密钥管理。

更加详细的信息参见《小金微服密钥管理规范》

11 其他安全方案最佳实践

11.1 HTTP HEADER

header	默认策略
P3P	默认不添加

非标准 header，只在部分现代浏览器中支持，详细见附录 2。

X-XSS-Protection	X-XSS-Protection: 1; mode=block	默认开启
X-Frame-Options	X-Frame-Options: SAMEORIGIN	默认 SAMEORIGIN
Strict-Transport-Security	Strict-Transport-Security: max-age=31536000;	默认开启

Content-Security-Policy	Content-Security-Policy: default-src 'self'	默认不开启
X-Content-Type-Options	X-Content-Type-Options: nosniff	默认禁用

11.2 Cookie

有两个 Http 头部和 Cookie 有关：Set-Cookie 和 Cookie。

1. Set-Cookie 由服务器发送，它包含在响应请求的头部中。它用于在客户端创建一个 Cookie
2. Cookie 头由客户端发送，包含在 HTTP 请求的头部中。注意，只有 cookie 的 domain 和 path 与请求的 URL 匹配才会发送这个 cookie

Set-Cookie 响应头的格式如下所示：

Set-Cookie: <name>=<value>[; <name>=<value>]...

[; expires=<date>][; domain=<domain_name>]

[; path=<some_path>][; secure][; httponly]

属性	含义	默认策略
name	cookie 名字	自定义
expires	过期时间	非特殊需求，默认不设置，为内存 cookie。 需要在 cookie 中长期记录信息的自定义设置。
domain	有效域	自定义
path	Cookie 生效的目录	自定义
secure	是否置能通过 HTTPS 传输	默认有
http only	是否只可以脚本获取修改	默认有

11.3 会话管理

会话是用户使用服务的上下文，客户端用 cookie 进行标示，服务端通过 session 等记录上下文状态，

1. 会话必须在服务端进行有效期限制，15 分钟有效
2. 每次重新登陆 sessionID 都重新生成
3. 要具备会话清理的功能，通过 sessionID 以及用户维度进行主动将会话失效的功能
4. 会话标示 cookie 要具备随机性，并且有效期内的唯一性
5. 客户端 cookie 中不等包含用户信息，特别是用户敏感信息

11.4 文件上传与下载

文件的上传，查看，下载有安全注意事项

1. 对上传文件类型进行限制，
   a) 图片类：JPEG,PNG,GIF
   b) 文档类：doc,docx,xls,xlsx,pdf,ppt,pptx
2. 文件大小限制，无特殊需求，上传文件不超过 5M
3. 图片文件需要进行重绘，防止图片中嵌入恶意代码。如果使用 sofa 框架，默认会进行图片重绘。
4. 文件不直接存储在磁盘上，使用 tfs/sfs 进行文件的统一保存
5. 上传的文件如果不需要外网访问，或者对文件的访问有应用隔离的需求，则使用 **SFS** 进行文件保存，例如身份证图片等敏感文件。
6. 非公开文件的访问都需要进行**应用层的水平权限控制**，数据库中记录文件 ID 和账户的绑定关系。不可通过改变 ID 进行文件遍历。

11.5 身份验证

11.5.1 短信校验码

维度	方案
有效期	1 分钟
尝试次数	1 分钟内只能触发 1 次
位数	不少于 6 位

另外为了防止进行短信轰炸，建议对短信总次数进行限制，如果登陆态下触发，以账号为维度进行限制；非登陆态，以 IP 或者其他识别 ID 为维度进行限制。

11.5.2 密码校验

1. 登陆密码与支付密码必须不同
2. 排除弱密码（弱密码包括常见密码、生日和姓名相关密码）
3. 密码尝试次数限制，错误不得超过 5 次，超过冻结 3 小时。（可信环境下可适当增加，错误不得超过 8 次）

12 附录

12.1 附录 1：隐藏展示规则

缺省信息隐藏规则：

显示前 1/3 和后 1/3，其他用*号代替（短信使用一个*）。内容长度不能被 3 整除时，显示前 ceil[length/3.0]和后 floor[length/3.0]。

以下是针对特定类型的信息隐藏规则，如在下表中不存在，则使用缺省的信息隐藏规则：

敏感信息类型	信息范围	展示规范
密码/口令及相关	1) 登录密码、 2) 支付密码、 3) 手机校验码、 4) 密码保护问题答案、 5) 支付盾 PIN 码、 6) 宝令动态密码、 7) 汇票密码、 8) 3D 密码、 9) 银行卡 PIN、 10)sessionId 等	禁止展示
密钥	1) 数据加密密钥、 2) 签名私钥、 3) Md5/HMAC 消息认证密钥等	禁止展示
信用卡信息	1) 信用卡卡号	前 6 和后 4 位
	2) 信用卡 CVV2/CVC2 3) 信用卡有效期	禁止展示
借记卡信息	借记卡卡号	**短信和收银台选择页面** 显示后 4 位。如： “尾号 7750” **其他** 显示前 6 位 + * （实际位数）+ 后 4 位。 如： 622575******1496
个人信息	1) 身份证号、军官证号、护照号	***身份证号*：** **推荐** 显示前 1 位 + *（实际位数）+后 1 位，如：5**********9 **最低要求** 前 5 和后 3 位 ***军官证号，护照号：*** 使用缺省信息隐藏规则

	2）姓名	如果要隐藏，隐藏第一个字
	3) 手机号	如需要部分隐藏，区号不算，隐藏中间四位 **网站和手机客户端** 大陆:显示前 3 位 + **** + 后 4 位。如：137****9050 香港、澳门：显示前 2 位+ ****+后 2 位。如：90****85 台湾：显示前 2 位+ ****+后 3 位。如：90****856 其他海外地区：使用缺省隐藏规则 **短信** 大陆：显示前 3 位 + * + 后 4 位。如：137*9050 香港、澳门：显示前 2 位+ *+后 2 位。如：90*85 台湾：显示前 2 位+ *+后 3 位。如：90*856 其他海外地区：使用缺省隐藏规则
	4) 固定电话号码	如需要部分隐藏，推荐的规范：显示区号和后 4 位
	5) 邮箱	如需要部分隐藏， **网站、手机客户端** @前面的字符显示 3 位，3 位后显示 3 个 *，@后面完整显示如：con***@163.com 如果少于三位 ,则全部显示 ,@前加***，例如 tt@163.com 则显示为 tt***@163.com

		短信 短信必须控制在 60 字内，所以，需要隐藏更多字 1） @前面的字符显示规则： 如果@前面的字符数小于等于 3 位，则全部显示字符，然后再加上* 如果@前面的字符数多于 3 位，则显示前三位字符，然后再加上* 2） @后面的字符显示规则： 如果@后面第 1 个字符到第 1 个"."之前的字符数小于等于 7 位，则全部显示，并以.*结尾 如果@后面第 1 个字符到第 1 个"."之前的字符数大于 7 位 则显示前 7 位字符，并以*结尾 3） 示例： 如果账户为 tjyihui@126.com 则显示：tjy*@126.* 如果账户为 mm@hotmail.com 则显示： mm*@hotmail.* 如果账户为 iceziling@netvigator.com 则显示：ice*@netviga*
	6) 地址信息	暂无要求
	7）企业工商注册号	显示后三位
	8）淘宝昵称	显示首/尾各 1 位，中间加** 例如：风**扬，m**d

POM 中引用最新版本的安全 JAR 包

```
<dependency>
    <groupId>com.alipay.service</groupId>
    <artifactId>alipay-service-security</artifactId>
    <version>查询使用最新版本号</version>
    <type>pom</type>
</dependency>
```

使用

com.alipay.service.security.utils. SensitiveDataUtil 类进行信息隐藏

12.2 附录 2：HTTP HEADER

header	注释	默认策略
P3P	This header is supposed to set P3P policy, in the form of P3P:CP="your_compact_policy". However, P3P did not take off, most browsers have never fully implemented it, a lot of websites set this header with fake policy text, that was enough to fool browsers the existence of P3P policy and grant permissions for third party cookies P3P: CP="This is not a P3P policy! See http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info." Status: Permanent	

非标准 http header

X-XSS-Protection	用于启用浏览器的 XSS 过滤功能，以防止 XSS 跨站脚本攻击。 利用 IE8/9 浏览器 XSS 跨站攻击脚本筛选过滤特性 (X-XSS-Protection) IE8 Security Part IV: The XSS Filter **X-XSS-Protection: 1; mode=block** 0 禁用 XSS 过滤功能 1 启用 XSS 过滤功能	默认开启
X-Frame-Options	该响应头中用于控制是否在浏览器中显示 frame 或 iframe 中指定的页面，主要用来防止 Clickjacking（点击劫持）攻击。如果某页面被不被允许的页面以或的形式嵌入，IE 会显示类似于“此内容无法在框架中显示”的提示信息，Chrome 和 Firefox 都会在控制台打印信息。由于嵌入的页面不会加载，这就减少了点击劫持（Clickjacking）的发生。 IE8 Security Part VII: ClickJacking Defenses Combating ClickJacking With X-Frame-Options 使用 X-Frame-Options header 拒绝被嵌入框架(iframe…) The X-Frame-Options response header 测试页面 X-Frame-Options: SAMEORIGIN DENY 禁止显示 frame 内的页面（即使是同一网站内的页面） SAMEORIGIN 允许在 frame 内显示来自同一网站的页面，禁止显示来自其他网站的页面 ALLOW-FROM origin_uri 允许在 frame 内显示来自指定 uri 的页面（当允许显示来自于指定网站的页面时使用）	默认 SAMEORIGIN
Strict-Transport-Security	用于通知浏览器只能使用 HTTPS 协议访问网站。用于将 HTTP 网站重定向到 HTTPS 网站。 Strict-Transport-Security: max-age=31536000; includeSubDomains max-age 用于修改 STS 的默认有效时间。	默认开启

	includeSubDomains 用于指定所有子域名同样使用该策略。 默认开启: options.hsts = { maxAge: 31536000 } 缓存一年	
Content-Security-Policy	用于控制当外部资源不可信赖时不被读取。用于防止 XSS 跨站脚本攻击或数据注入攻击（但是，如果设定不当，则网站中的部分脚本代码有可能失效)。 之前的字段名为 X-Content-Security-Policy Content Security Policy 介绍 Content-Security-Policy: default-src 'self' default-src 'self' 允许读取来自于同源（域名+主机+端口号）的所有内容 default-src 'self' *.example.com 允许读取来自于指定域名及其所有子域名的所有内容	默认不开启
X-Content-Type-Options	互联网上的资源有各种类型，通常浏览器会根据响应头的 Content-Type 字段来分辨它们的类型。例如："text/html" 代表 html 文档，"image/png"是 PNG 图片，"text/css"是 CSS 样式文档。然而，有些资源的 Content-Type 是错的或者未定义。这时，某些浏览器会启用 MIME-sniffing 来猜测该资源的类型，解析内容并执行。 例如，我们即使给一个 html 文档指定 Content-Type 为 "text/plain"，在 IE8-中这个文档依然会被当做 html 来解析。利用浏览器的这个特性，攻击者甚至可以让原本应该解析为图片的请求被解析为 JavaScript。通过下面这个响应头可以禁用浏览器的类型猜测行为： **X-Content-Type-Options: nosniff** 这个响应头的值只能是 nosniff，可用于 IE8+和 Chrome。	禁用