开放平台第三方应用安全开发指南

更新时间：2020-02-06 22:17:51

《开放平台第三方应用安全开发指南》给出常见开发场景下，帮助开发人员完善应用安全性的开发建议，同时也对常见的安全漏洞进行描述，并提供对应的修复方案。

1. 常见开发场景安全开发指南

1.1. 敏感信息使用场景

敏感信息指用户的 **身份证号、银行卡号、手机号** 等身份信息。重要敏感信息的脱敏规范如下。

敏感信息类型	展示规范
身份证	显示前 1 位 + *(实际位数) + 后 1 位，如： 3****************3
银行卡	显示前 6 位 + *(实际位数) + 后 4 位，如：622575******1496
手机号	显示前 3 位 + **** + 后 2 位，如：137******50

1.1.1. 敏感信息用于展示的场景

- **原则：敏感信息的展示请严格按照脱敏规范进行脱敏**
  - 说明：脱敏的逻辑必须在服务端完成，不能使用 Javascript 在客户端进行脱敏，包括代码注释、隐藏域、url 参数、cookies 等处的数据也必须脱敏。
  - 说明：不能使用可逆的编码/加密方式，如 base64 编码等代替脱敏规范。
  - 说明：若敏感信息明文展示在应用中，没有按照脱敏规范完成脱敏。支付宝开放平台将有权暂停敏感数据相关接口的开放。

1.1.2. 敏感信息用于身份校验的场景

- **原则：不要直接将敏感信息的明文信息在客户端与服务端之间传递**
  - 说明：可以将敏感信息在服务端关联到用户标识 ID，在客户端保存用户标识 ID 并提交到服务端，服务端根据 ID 取出对应信息后进行校验。

1.2. HTML 页面渲染

- **原则：所有在页面渲染的敏感数据 (身份证、银行卡号、手机号) 必须进行脱敏**
  - 说明：参考章节 1.1.1. 敏感信息用于展示的场景

- **原则：禁止在 Cookie 中** 明文写入 **敏感数据**
  - 说明：参考章节 1.1.2. 敏感信息用于用于身份校验的场景

- **原则：禁止向 HTML 页面输出未经安全过滤或未正确转义的用户数据**

- **原则：禁止向 HTML 页面输出未经安全过滤或未正确转义的用户数据**
  - 说明：用户数据不仅仅包括用户正常数据，同时包括攻击者可修改，伪造的其他数据。
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

- **原则：HTML 页面动态输出 JSON、JavaScript 必须对其中的字符串值做 XSS 防御处理**
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

- **原则：默认设置 HTTP Header 中的 `HttpOnly` 属性为 true**
  - 说明：设置 `HttpOnly` 为 true，将可以禁止 JavaScript 读取页面 Cookie 信息，一定程度上防范 XSS 攻击。
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

- **原则：如果网站使用 HTTPS 协议，默认设置 HTTP Header 中的 `secure` 属性为 true**
  - 说明：设置 `secure` 为 true，Cookie 信息将不会在 HTTP 连接中传输，一定程度上防范 XSS 攻击。
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

- **原则：服务器向响应 HTTP Header 写入用户输入数据时必须做 CRLF 过滤或者转义**
  - 说明：参考章节 2.3 HTTP Header 注入漏洞

1.3. 接口调用操作

- **原则：AJAX 接口必须执行 CSRF 过滤**
  - 说明：参考章节 2.2 CSRF 漏洞

- **原则：AJAX 接口输出 JSON 字符串禁止通过字符串拼接构造，且输出的 JSON 需要经过安全过滤**
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

- **原则：AJAX 接口返回头必须设置 `Content-Type` 为 `application/json;charset=utf-8`**

1.4. 表单提交操作

- **原则：统一使用 `POST` 方式提交表单**
  - 说明：Get 请求可以通过构造 img 等标签发起，造成 CSRF
  - 说明：参考章节 2.2 CSRF 漏洞

- **原则：Form 表单提交必须执行 CSRF 过滤**
  - 说明：参考章节 2.2 CSRF 漏洞

- **原则：用户输入的富文本浏览器展示之前必须由服务器端做安全过滤**
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

1.5. 数据库操作

- **原则：用户密码存储须加盐存储，各用户盐值不同**
- **原则：若涉及证件号等敏感信息的存储，须使用 AES-128 算法加密存储**
- **原则：使用的 SQL 必须预编译，不允许通过字符串拼接的方式合成**

- **原则：编写的 SQL 必须预编译，不允许通过字符串拼接的方式合成**

  - 说明：部分特殊场景，必须通过拼接合成，则拼接的变量必须经过处理，只允许 `[a-zA-Z0-9_-.]+` 字符。
  - 说明：参考章节 2.5 SQL 注入漏洞

1.6. URL 重定向

- **原则：URL 重定向的目标地址必须执行白名单过滤**

1.7. 跨域操作

1.7.1. JSONP 跨域

- **原则：JSONP 接口 Callback 必须验证有效性**
- **原则：JSONP 接口输出 JSON 字符串禁止通过字符串拼接构造，且输出的 JSON 需要经过安全过滤**
  - 说明：参考章节 2.1 跨站脚本（XSS）漏洞

- **原则：JSONP 接口必须对 REFERER 进行白名单校验，或执行 CSRF 检查**
- **原则：JSONP 接口返回头必须正确设置 `Content-Type` 为 `application/javascript;charset=utf-8`**

1.7.2. CORS 跨域

- **原则：支持 CORS 跨域的接口，返回头 `Access-Control-Allow-Origin` 必须使用白名单验证，禁止直接返回 `*`**

1.8. 文件上传与下载

- **原则：限制可下载文件所在的目录为预期范围，并通过指定文件编号的方式来定位待下载文件**
  - 说明：参考章节 2.4 目录遍历漏洞
  - 说明：参考章节 2.6 文件下载漏洞

- **原则：保存上传文件的目录不提供直接访问**
- **原则：对上传文件的大小和类型进行校验，定义上传文件类型白名单**
  - 说明：参考章节 2.7 文件上传漏洞

1.9. 加密与签名

不同场景使用的加密算法请参考下表。

场景	算法
加密场景	AES128
摘要场景	SHA256
签名场景	RSA2048

2. 常见安全漏洞及修复方案

2.1 跨站脚本（XSS）漏洞

漏洞描述

跨站脚本攻击(Cross Site Scripting, XSS)发生在客户端，可被用于进行窃取隐私、钓鱼欺骗、偷取密码、传播恶意代码等攻击行为。 恶意的攻击者将对客户端有危害的代码放到服务器上作为一个网页内容， 使得其他网站用户在观看此网页时，这些代码注入到了用户的浏览器中执行，使用户受到攻击。一般而言，利用跨站脚本攻击，攻击者可窃会话 Cookie 从而窃取网站用户的隐私。

漏洞危害

- 钓鱼欺骗：最典型的就是利用目标网站的反射型跨站脚本漏洞将目标网站重定向到钓鱼网站，或者注入钓鱼 JavaScript 以监控目标网站的表单输入。
- 网站挂马：跨站时利用 IFrame 嵌入隐藏的恶意网站或者将被攻击者定向到恶意网站上，或者弹出恶意网站窗口等方式都可以进行挂马攻击。
- 身份盗用：Cookie 是用户对于特定网站的身份验证标志，XSS 可以盗取到用户的 Cookie，从而利用该 Cookie 盗取用户对该网站的操作权限。如果一个网站管理员用户 Cookie 被窃取，将会对网站引发巨大的危害。
- 盗取网站用户信息：当能够窃取到用户 Cookie 从而获取到用户身份时，攻击者可以获取到用户对网站的操作权限，从而查看用户隐私信息。
- 垃圾信息发送：比如在 SNS 社区中，利用 XSS 漏洞借用被攻击者的身份发送大量的垃圾信息给特定的目标群。
- 劫持用户 Web 行为：一些高级的 XSS 攻击甚至可以劫持用户的 Web 行为，监视用户的浏览历史，发送与接收的数据等等。
- XSS 蠕虫：XSS 蠕虫可以用来打广告、刷流量、挂马、恶作剧、破坏网上数据、实施 DDoS 攻击等。

解决方案

- 对参数做 html 转义过滤，要过滤的字符包括：单引号、双引号、大于号、小于号，& 符号，防止脚本执行。
- 在变量输出时进行 HTML ENCODE 处理。

2.2 CSRF 漏洞

漏洞描述

跨站请求伪造(Cross-Site Request Forgery, CSRF)，恶意网站通过脚本向当前用户浏览器打开的其它页面的 URL 发起恶意请求，由于同一浏览器进程下 Cookie 可见性，导致用户身份被盗用，完成恶意网站脚本中指定的操作。

漏洞危害

- 信息泄露：如登录ID，隐私信息等。
- 恶意操作：如加好友，加购物车，删除数据等。

解决方案

CSRF漏洞修复方案主要思路有两类：

- 验证请求是信任页面发起，这类修复方案有：
  - 在表单中填充一次性随机的 csrf token 防止攻击者伪造 form 表单进行 CSRF。同时将此串 token 置入 session，在

后端再进行一次一致性校验。

    - referer 验证。

- 验证请求是合法用户发起，这类修复方案有：
    - 验证码
    - 密码验证
    - OTP 验证

2.3 HTTP Header 注入漏洞

漏洞描述

Web 程序代码中把用户提交的参数未做过滤就直接输出到 HTTP 响应头中，导致攻击者可以利用该漏洞来注入到 HTTP 响应头中实现攻击。

解决方案

- 对参数做合法性校验以及长度限制，谨慎的根据用户所传入参数做 HTTP 响应的 Header 设置。
- 在设置 HTTP 响应头时，过滤回车换行 `%0d%0a、%0D%0A` 字符。

2.4 目录遍历漏洞

漏洞描述

目录遍历是由于 Web 服务器或 Web 应用程序对用户输入文件名称的安全性验证不足而导致的一种安全漏洞，使得攻击者通过 HTTP 请求和利用一些特殊字符就可以绕过服务器的安全限制，访问任意受限的文件(可以是 Web 根目录以外的文件)，甚至执行系统命令。

解决方案

- 严格检查文件路径参数，限制在指定的范围。
- 严格限制文件路径参数，不允许用户控制文件路径相关的参数，限定文件路径范围。

2.5 SQL 注入漏洞

漏洞描述

SQL 注入攻击(SQL Injection)，被广泛用于非法获取网站控制权，是发生在应用程序的数据库层上的安全漏洞。 在设计不良的程序当中，忽略了对输入字符串中夹带的 SQL 指令的检查，那么这些夹带进去的指令就会被数据库误认为是正常的 SQL 指令而运行，从而使数据库受到攻击，可能导致数据被窃取、更改、删除，以及进一步导致网站被嵌入恶意代码、被植入后门程序等危害。

漏洞危害

- 机密数据被窃取
- 核心业务数据被篡改
- 网页被篡改
- 数据库所在服务器被攻击变为傀儡主机，甚至企业网被入侵

解决方案

- 所有的查询语句都使用数据库提供的参数化查询接口，参数化的语句使用参数而不是将用户输入变量嵌入到 SQL 语句中。
- 对进入数据库的特殊字符 `'"\<>&*;` 等进行转义处理，或编码转换。
- 确认每种数据的类型，比如数字型的数据就必须是数字，数据库中的存储字段必须对应为 int 型。
- 数据长度应该严格规定，能在一定程度上防止比较长的 SQL 注入语句无法正确执行。
- 网站每个数据层的编码统一，建议全部使用 UTF-8 编码，上下层编码不一致有可能导致一些过滤模型被绕过。
- 严格限制网站所用数据库账号的权限，给此用户仅提供能够满足其工作的权限，从而最大限度的减少注入攻击对数据库的危害。
- 避免网站显示 SQL 错误信息，比如类型错误、字段不匹配等，防止攻击者利用这些错误信息进行一些判断。

2.6 文件下载漏洞

漏洞描述

Web 应用程序在处理文件下载时，接受用户指定的路径和文件名进行下载，攻击者利用此漏洞来下载服务器的其它文件甚至任意文件（源代码、数据库甚至 passwd 等）。

解决方案

- 限制可下载文件所在的目录为预期范围。
- 通过指定文件编号的方式来定位待下载文件。

2.7 文件上传漏洞

漏洞描述

文件上传的 Web 程序未对文件类型和格式做合法性校验，导致攻击者可以上传 Webshell 或者非期望格式的文件。

解决方案

- 对上传文件的大小和类型进行校验，定义上传文件类型白名单。
- 保存上传文件的目录不提供直接访问。

< 上一篇：获取 PID

下一篇：创建订单+JSAPI唤起收银台支付 >

相关问题

应用查询成员列表能不能根据支付宝ID或者是登录ID查询成员是不是开发者？...

第三方应用模板小程序，成员管理增删改开发成员能不能细分到对应商家？...

第三方应用模板小程序，成员接口只能增删改开发成员和体验成员？

小程序模板开发中，碰到第三方应用授权通知怎么测试？

第三方应用提审时营业执照大小问题