神策数据 JS SDK 源码安全审计报告

■ 文档编号		■ 密级	商业机密
■ 版本编号	V1.0	■ 日期	2019 年 08 月 02 日

■ **版本变更记录**

时间	版本	说明	修改人
2019-8-2	V1.0	文档创建	刘红涛
2019-8-28	V2.0	更新文档	武立鑫

■ **适用性声明**

本文档为北京神州绿盟信息安全科技股份有限公司（以下简称“绿盟科技”）在神策数据安全服务中所提供的源代码安全审计报告，适用于相关技术人员在针对发现的漏洞进行安全修复时参考。

目录

一. 执行摘要

经神策数据授权，绿盟科技对其 JS SDK 进行源代码安全审计。审计发现的问题总结如下：

表 1.1 发现问题汇总

威胁程度	漏洞名称	风险分析
风险提示	Cookie 未设置 httponly 属性	Cookies 包含一个 httponly 属性选项，可以使得 cookie 不能被 js 读取而只能用于 http 请求，防护 xss 攻击。Httponly 属性的缺失，将增大 xss 攻击的风险。

审计发现的风险分布如下图所示。

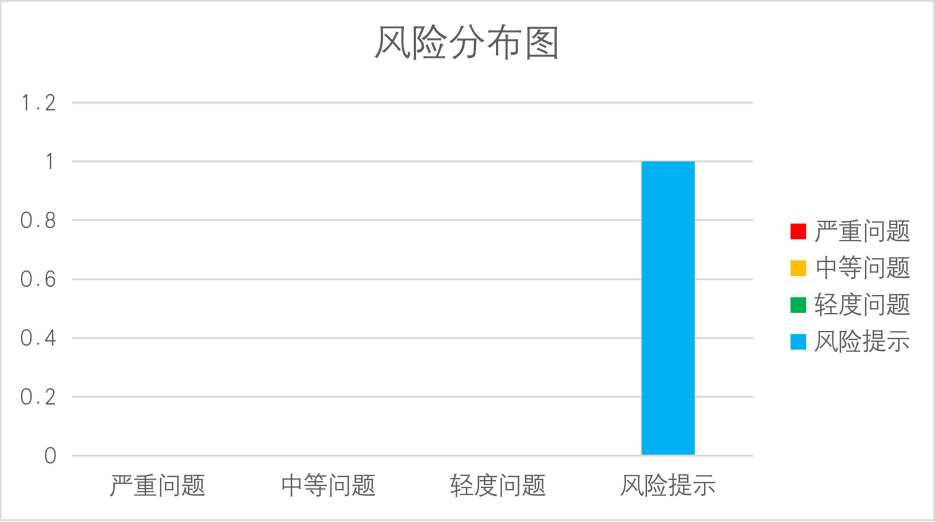

图 1.1 安全问题分布图

通过本次代码审计活动，我们发现 JS SDK 设计了较好的安全防护策略和技术防控 功能。但是因为种种原因，在开发过程中由于业务需求，会存在一些风险点。我们建议开发商在保证业务功能正常使用的基础上有针对性的通过其他方式去避免风险，切实保障系统业务稳定运行以及用户信息安全。

二. 服务综述

2.1 审计概述

在对神策数据 JS SDK 进行源代码安全审计活动前，我们会审阅系统相关文档，对技术架构的安全性进行审计和评估，并且熟悉系统实现的业务流程，评估其可能存在的安全风险。在审计活动起始阶段，我们会使用源代码安全审计工具进行全面的审计，发现其存在的不安全编码并进行人工分析和确认。自动化审计完成后，审计实施人员对重要业务场景进行深入分析。不论是客户端代码还是服务端代码，除常规的安全漏洞如 SQL 注入、XSS、CSRF 等漏洞外，我们还会参照相关行业标准及规范，对其合规性进行审计。

图 2.1 源代码安全审计技术模型

2.2 审计依据

为保证项目实施的标准化和规范化，本次源代码安全审计活动主要参考如下依据：

国内通用标准、指南或规范

- GB/T 22239 信息安全技术 网络安全等级保护基本要求
- ISO/IEC 27001:2013 信息技术-安全技术-信息系统规范与使用指南
- ISO/IEC 13335-1: 2004 信息技术-安全技术-信息技术安全管理指南
- ISO/IEC TR 15443-1: 2005 信息技术安全保障框架

- GB/T 20984-2007 信息安全技术 信息安全风险评估规范
- GB/T 19715.1-2005 信息技术-信息技术安全管理指南
- GB/T 19716-2005 信息技术-信息安全管理实用规则
- GB/T 18336-2015 信息技术-安全技术-信息技术安全性评估准则
- GB 17859-1999 计算机信息系统安全保护等级划分准则
- GB/T 20984-2007 信息安全技术 信息安全风险评估规范
- 绿盟科技安全编码规范

国际标准、指南或规范

- OWASP Top 10（2013）
- OWASP Top 10（2017）
- OWASP Development Guide
- OWASP Testing Guide v4
- OWASP Application Security Verification Standard
- OWASP Secure Coding Practices

2.3 项目实施

2.3.1 审计对象

表 2.1 审计对象

单位	系统名称	审计对象
神策数据	神策分析 SDK	JS SDK

2.3.2 审计时间

表 2.2 项目实施时间

源代码安全审计时间	
起始时间	2019 年 7 月 29 日
结束时间	2019 年 8 月 2 日

2.3.3 审计人员

表 2.3 源代码安全审计实施人员

安全审计人员名单					
姓名	刘红涛	**所属部门**	华东安全服务交付部	**联系方式**	liuhongtao@nsfocus.com

2.3.4 审计工具

本次源代码安全审计活动所使用的工具主要有：

表 2.4 源代码安全审计工具

工具名称	工具版本	简要介绍
HP Fortify SCA	4.02	源代码安全审计工具
Eclipse	Oxygen 2	开发 IDE 工具
IntelliJ IDEA	2018.1	开发 IDE 工具
Sublime Text	3.0	代码查看工具

三. 源代码安全审计结果

3.1 Cookie 未设置 httponly 属性

3.1.1 漏洞概述

风险描述	Cookies 包含一个 httponly 属性选项，可以使得 cookie 不能被 js 读取而只能用于 http 请求，防护 xss 攻击。Httponly 属性的缺失，将增大 xss 攻击的风险
威胁级别	风险提示
可利用性	容易
测试结果	仅作风险提示

3.1.2 代码安全分析

分析查看 JSManifest.xml 文件中声明的组件可导出权限，判断是否可导出且存在敏感数据。

代码位置： src/main/JSManifest.xml

```
    _.cookie = {
                        get: function (e) {
                            for (var t = e + "=", r = document.cookie.split(";"), n = 0; n <
r.length; n++) {
                                for (var s = r[n];
                                    " " == s.charAt(0);) s = s.substring(1, s.length);
                                if (0 == s.indexOf(t)) return
_.decodeURIComponent(s.substring(t.length, s.length))
                            }
                            return null
```

```
                },
                set: function (e, t, r, n, s) {
                    n = "undefined" == typeof n ? sd.para.cross_subdomain : n;
                    var a = "",
                        i = "",
                        o = "";
                    if (r = null == r ? 73e3 : r, n) {
                        var c = _.getCurrentDomain(location.href);
                        "url\u89e3\u6790\u5931\u8d25" === c && (c = ""), a = c ? "; domain=" + c : ""
                    }
                    if (0 !== r) {
                        var u = new Date;
                        "s" === String(r).slice(-1) ? u.setTime(u.getTime() + 1e3 * Number(String(r).slice(0, -1))) : u.setTime(u.getTime() + 24 * r * 60 * 60 * 1e3), i = "; expires=" + u.toGMTString()
                    }
                    s && (o = "; secure"), document.cookie = e + "=" + encodeURIComponent(t) + i + "; path=/" + a + o
                },
                remove: function (e, t) {
                    t = "undefined" == typeof t ? sd.para.cross_subdomain : t, _.cookie.set(e, "", -1, t)
                },
                getCookieName: function (e) {
                    var t = "";
                    return sd.para.cross_subdomain === !1 ? (t = _.url("sub", location.href), t = "string" == typeof t && "" !== t ? "sajssdk_2015_" + e + "_" + t : "sajssdk_2015_root_" + e) : t = "sajssdk_2015_cross_" + e, t
                },
                getNewUser: function () {
```

```
                    var e = "new_user";
                    return null !== this.get("sensorsdata_is_new_user") || null !==
this.get(this.getCookieName(e))
                }
```

代码安全分析：前端 JS 代码读取 Cookie 中的数据，服务器端未设置 httponly 属性。

结合业务安全分析：神策 JS SDK 主要是用于收集用户在 Web 页面上的交互行为，每一条行为已 “事件” 的形式发送，每一个 “事件” 内，都需要存储用户的唯一标识，用于标识是哪个用户产生的 “事件”。目前业界通用的做法是将用户唯一标识存储于 cookie 中，神策 JS SDK 采用同样的方式，根据随机数、日期、屏幕宽高、UA 等值计算出唯一值作为用户的唯一标识，因为每次生成的标识都是唯一的，为了保持同一个用户 ID 的唯一且相同性，会把生成的用户 ID 存在 cookie 里，后续 JS SDK 直接从 cookie 里读取这段值用以标识用户。

而设置 httponly 会导致神策的 JS SDK 无法从前端读取 cookie，进而无法获取存储在 cookie 中的用户 ID，这将导致同一用户的每次访问行为都会生成一个新的用户 ID，从而错误的标识用户。

针对设置 httponly 属性防护 XSS 攻击，开发人员可通过其他方式，如：对用户输入过滤、引用 ESAPI 等方式去避免。

3.1.3 安全建议

1. 如不考虑 httponly 属性对 SDK 获取 Cookie 功能的影响，则建议引入 SDK 的用户通过服务端设置 httponly，防护 xss 攻击。

四. 审计结果及建议

4.1 源代码安全审计结果

通过本次源代码安全审计活动，我们发现 JS SDK 设计了较好的安全防护策略和技术防控功能，但是因为种种原因，在开发过程中由于业务需求，会存在一些风险点。我们建议开发商在保证业务功能正常使用的基础上有针对性的通过其他方式去避免风险，切实保障系统业务稳定运行以及用户个人信息安全。

本次对 JS SDK 的源代码安全审计，已发现安全问题的开发建议如下：

表 4.1 源代码安全审计结果及建议

发现问题	威胁级别	安全建议
Cookie 未设置 httponly 属性	风险提示	结合业务因素分析，仅作风险提示，可不做代码修改

4.2 整体安全改进建议

针对本次源代码安全审计发现的安全问题，以及持续保证应用系统的安全防护能力，我们给出如下安全编码建议：

1. 梳理应用系统接收参数及数据的格式，通过过滤器/拦截器的方式，对用户输入数据中可能包含的攻击利用字符进行整体处理，如<、>、’、”、*、|、,、.、\等；
2. 数据库在设计过程中对于主键设计为 UUID 格式，应用程序配合数据库设计实现，降低因存在水平越权所导致的危害；
3. 应用程序实现文件上传功能时，应通过白名单的方式限定上传文件的类型，保存在服务器中时，应对保存的文件重新命名，并保存于中间件可解析的目录中，如独立的文件服务器。
4. 对于文件下载功能，应判断请求下载的文件是否处于指定目录范围之内。
5. 设计实现系统容错机制，自定义友好的异常错误显示页面，当发生 404、500 错误时，跳转到指定页面，防范服务器端敏感信息泄露。
6. 涉及短信验证码以及发送邮件相关功能时，应在服务器端代码中限定发送的频率，防范接口被滥用对其他用户的正常生活造成影响。
7. 应用系统发布时，对前端 JavaScript 代码进行 YUI Compressor 压缩处理，删除相关注释，并且通过压缩降低代码的可阅读性。
8. 保证业务系统的运行环境安全，安装中间件当前版本的最新版，应用程序所引用的组件为最新版本。
9. 定制安全编码规范，提高开发人员的安全编码意识；
10. 在系统设计和编码过程中，关注行业标准相关文件的要求，提高系统的合规性。

五. 感谢

感谢项目实施过程中神策数据相关部门协调工作，感谢相关开发人员的积极配合，也感谢绿盟科技项目组成员付出的努力，通过大家有效的沟通和积极协作，使得本次源代码安全审计工作顺利完成。

附录 A漏洞风险定级

风险值计算方式说明如下：

评定维度	说明
影响范围系数 F	大（3）：能够获取大量数据、敏感数据或影响大量用户。 中（2）：能够获取部分数据、一般敏感数据或影响部分用户 小（1）：能够获取少量数据、非敏感数据或影响少量用户
漏洞系数 Y	权限获取类（20）:可获取服务器权限的漏洞 数据获取类（8）：可导致结构化数据批量获取的漏洞 非法访问类（6）：以非正常方式访问资源的漏洞 业务缺陷类（5）：业务逻辑不完善导致的漏洞 暴力破解类（4）：与登录或认证凭据相关的暴力破解 客户端攻击类（3）：攻击其他客户端的漏洞 信息泄露类（3）：泄漏敏感的技术类信息及非结构化数据的漏洞 辅助攻击类（1）：单独存在不能形成攻击，但可以增加其他攻击手段效果的漏洞 其中权限获取类、暴力破解类、辅助攻击类影响范围系数 F 固定为 1。
系统重要性系数 I	核心应用系统（4）：主站、网银、商城等盈利站点 普通应用系统（2）：客服、OA 等辅助类站点 边缘应用系统（1）：单一功能、单一接口类站点
漏洞风险值	漏洞风险值= F×Y×I

漏洞风险评定说明

威胁级别	评定说明
严重问题	风险值>18，直接导致系统被入侵或数据被破坏，一旦发生，就是严重的安全事件。建议紧急修复。
中等问题	6 ≤风险值≤18，可能导致重要信息的泄漏、系统拒绝服务或有较高可能导致系统被入侵控制。
轻度问题	2 ≤风险值≤6，可导致敏感信息泄漏或存在轻微安全问题，一般不会产生严重的安全事件。
风险提示	风险值小于 2，不合规编码且利用难度较大的安全问题，一般不会产生严重的安全事件

漏洞可利用性评价说明

可利用性定级	定级标准（如具备如下任一条件即可）
容易	有公开的利用工具或攻击代码
	无需登录系统即可利用
	无需和被攻击用户交互即可利用
	本地修改数据包或客户端程序即可利用
一般	需要和目标用户交互才能够利用
	通过网络远程抓取目标用户数据包才可利用

困难	需要控制被攻击用户的终端设备才可利用
	需要知道被攻击用户的账号口令才可利用