神策科技 SDK 源代码安全审计报告

(IOS SDK)

■ 文档编号		■ 密级	商业机密
■ 版本编号	V1.0	■ 日期	2020 年 08 月 16 日

■ **版本变更记录**

时间	版本	说明	修改人
2020-8-16	V1.0	文档创建	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”）在神策科技 IOS SDK 源代码安全审计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

表格索引

插图索引

一. 执行摘要

经神策科技授权，绿盟科技对其神策科技 IOS SDK 进行源代码安全审计。

通过本次代码审计活动，我们发现神策科技 IOS 安全防护策略和技术防控功能设计较好，结合各个安全风险审计项进行审计，未见安全风险。

如下，为本此审计工作中重点审计风险点及对应审计结果：

表 1.1 审计风险点

审计项名称	审计目的	审计结果
日志信息安全	开发者习惯使用日志辅助程序调试，则可能泄露访问链接、收发的数据包甚至身份凭证等内容。	安全
程序数据存储安全	检设备中测程序对应的数据文件夹是否包含敏感数据。由于设备可以进行越狱，且越狱之后数据不会丢失，则导致本地存储敏感信息将造成较大风险。	安全
符号表信息泄露	符号表是逆向工程中的兵家必争之地。有效的符号表能够极大地方便反汇编和逆向工作的进行，在发布应用的时候未清除掉相关的符号则会辅助逆向分析。	安全
越狱环境检测	越狱的设备是不安全环境，各种恶意程序都容易运行在越狱环境之中。越狱环境中运行则难以保证程序使用过程中数据的安全性。检测设备是否越狱是一个比较有用的方法。	安全
界面切换保护	用户在登录界面写入账号密码之后，若界面切换时未清除用户输入的密码，可导致手机丢失后使用已输入的信息进行登录。	安全
安全退出/注销	程序在进行正常退出之后如果后台不消除相关的用户身份凭证，则在身份凭证被窃取之后容易造成数据泄露。	安全
缓冲区溢出	系统向缓冲区空间中填入超过缓冲区容量大小的数据，可破坏内存中的的相关数据结构从而形成漏洞。	安全
通信协议安全	使用 HTTP 协议进行通信过程中可能会遭受中间人攻击，威胁数据安全。	安全
通信数据加密	程序提交数据给服务端时，密码等关键字段是否进行了加密和校验，防止恶意嗅探和修改用户数据包中的密码等敏感信息。	安全
证书强制性校验	可能存在忽略服务端证书校验的安全漏洞，对服务器没有校验或者没有在校验错误时候进行错误提示等。导致攻击者可通过伪造证书等手法进行攻击获取。	安全
软件升级缺陷	程序在进行版本检查的时候，通常由服务器返回对应的升级地址以方便用户访问升级，可能跳转到 APP Store，也可能跳转到某个页面进行下载安装。	安全

二. 服务综述

2.1 审计概述

在对神策科技 IOS 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。对 SDK 相关代码，除常规的安全漏洞如日志信息安全性、越狱环境检测、通信协议安全等漏洞外，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
- OSSTMM OSSTMM_Web_App_Alpha

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
神策科技	神策科技 IOS SDK	神策科技 IOS SDK 源代码

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	2020 年 8 月 12 日
结束时间	2020 年 8 月 16 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	武立鑫	**所属部门**	绿盟科技安全服务部	**联系方式**	wulixin@nsfocus.com
姓名	李昊	**所属部门**	绿盟科技安全服务部	**联系方式**	lihao5@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 2.1 源代码安全审计工具

工具名称	工具版本	简要介绍
Vs Code		第三方提供的代码/文本编辑器

三. IOS SDK 源代码安全审计重点场景

在本项目中，我们会重点关注如下的功能，对其实现代码进行人工重点审计。

3.1 数据安全

检测程序中是否使用日志辅助调试程序、是否包含程序对应数据文件夹及敏感信息，如果存在，则可导致敏感信息泄露，从而导致安全风险。

3.2 静态安全

检测程序中对于自身完整性的校验、二进制保护及符号信息是否泄露，如果相关静态设置不规范，可导致篡改程序、钓鱼等安全风险。

3.3 运行时安全

检测程序中对越狱、动态调试跨域请求的相关约束，如果程序中没有对相关功能做出限制，可导致获取逻辑代码、恶意程序攻击等安全风险。

3.4 安全策略设置

检测程序中对用户登陆、界面交互、登出等功能安全策略，防止不完备的安全策略导致数据泄露、用户信息泄露等安全风险。

3.5 网络通信安全

检测程序中通信协议安全性、通信数据加密完备性、证书强校验等相关网络通信安全，防止恶意嗅探和修改用户数据包中的密码等敏感信息、证书伪造等安全风险。

四. 源代码安全审计结果

4.1 日志信息安全

4.1.1 漏洞概述

漏洞描述	开发者习惯使用日志辅助程序调试，在日志中打印信息，则可能泄露访问链接、收发的数据包甚至身份凭证等内容。
利用场景	攻击者通过泄露的访问链接、收发的数据包甚至身份凭证等内容有针对的对应用进行攻击
风险等级	高风险，泄露用户身份凭证（如 token、cookie）、用户或其他用户隐私信息。 中风险，泄露关键函数（加解密、校验函数）调用和逻辑信息或者部分隐私信息
测试结果	安全

4.1.2 缺陷代码分析

重点审查此类代码，审查发现无敏感内容直接打印

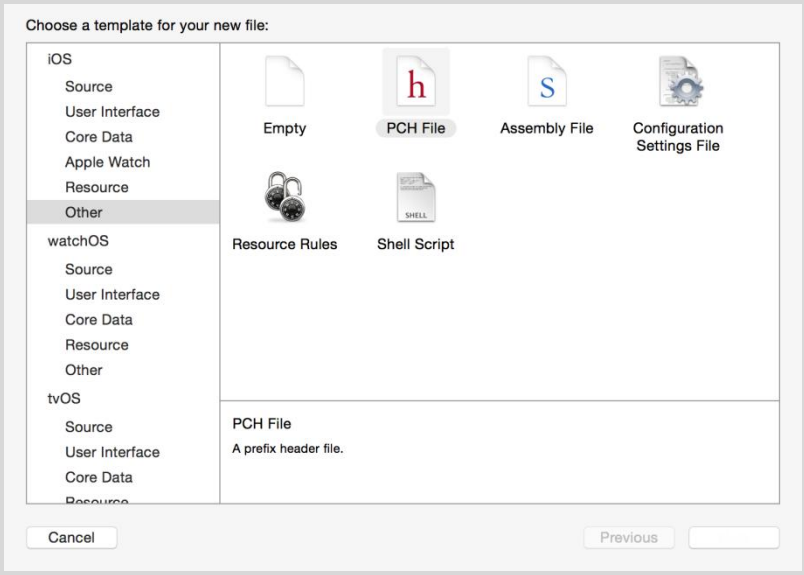
视为无风险：

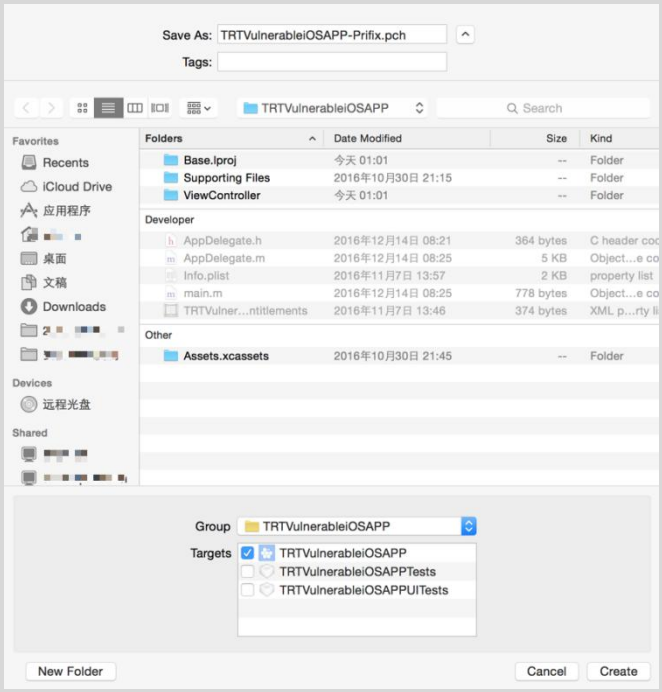
```
90          line:(NSUInteger)line {
91      @try {
92          NSString *logMessage = [[NSString alloc] initWithFormat:@"[SALog][%@]  %s [line %lu]    %s %@", [self descriptionForLevel:level], function, (u
93          NSLog(@"%@",logMessage);
94      } @catch(NSException *e) {
95
96      }
97  }
```

4.1.3 修复建议

若存在，或后续功能扩展需要进行日志打印，可通过如下方式进行，以保证代码安全性。

- 安全建议一，将所有的 NSLog 调用去掉再重新编译打包程序
- 安全建议二，重新封装 NSLog，新建<AppName>-Prefix.pch 文件





文件中宏定义代码如下：

```
1    #ifdef DEBUG
2    #define NSLog(FORMAT,...) fprintf(stderr,"[%s]:[line %
   d] %s %s \n",[[[NSString stringWithUTF8String:__FILE__] la
   stPathComponent] UTF8String], __LINE__, [NSString stringWi
   thUTF8String:__PRETTY_FUNCTION__], [[NSString stringWithF
   ormat:FORMAT, ##__VA_ARGS__] UTF8String]);
3    #else
4    #define NSLog(...) (void)0
```

```
5       #endif
```

➢ **说明**

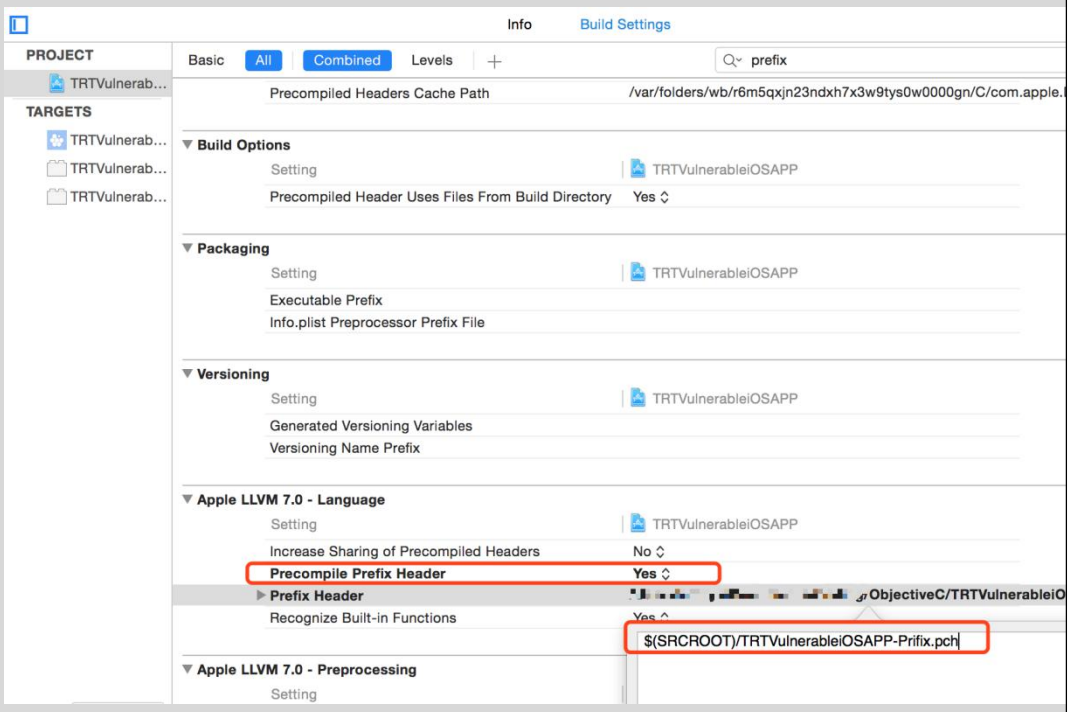
__FILE__ ： 预编译时候会替换成当前源文件名

__LINE__ ： 预编译的时候会替换成当前行号

__PRETTY_FUNCTION__ ： 预编译的时候会替换成<return-type> <class-name>::<member-function-name>(<parameters-list>)

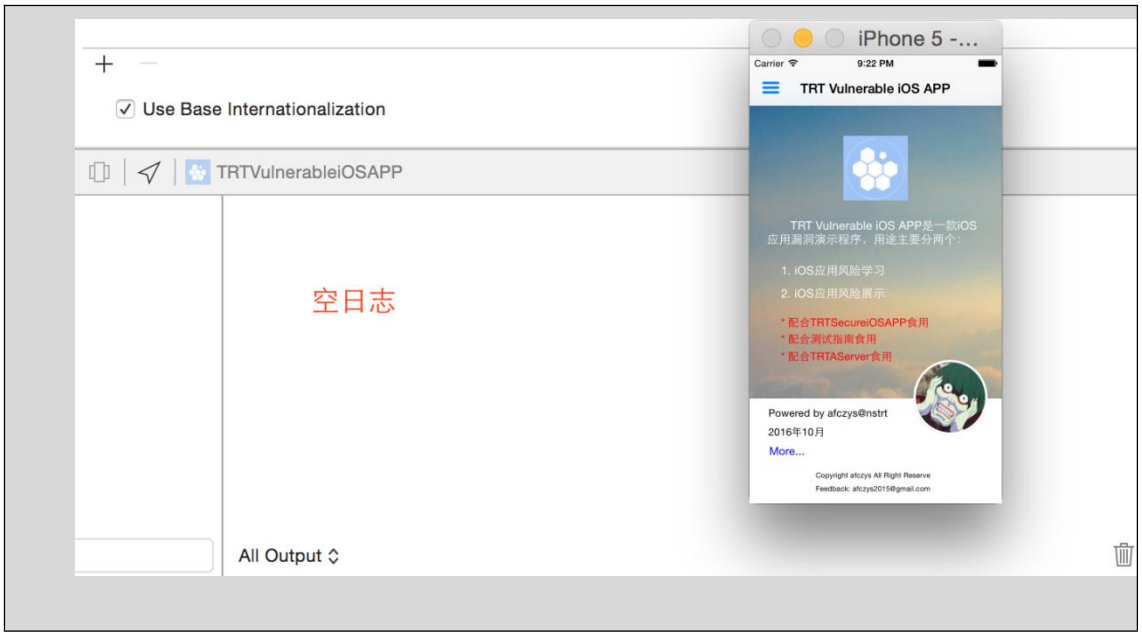
__VA_ARGS__ ： C99 规范新增可变长度宏参数，主要是用来处理 NSLog 传输过来的内容，##将如果是没有参数的情况下让预处理器忽略前面的逗号

在 build settings 中（xCode7）配置编译相对应的宏：



在 debug 版本中：

在 release 版本中：



4.2 程序数据存储安全

4.2.1 漏洞概述

漏洞描述	设备中测程对应的数据文件夹是否包含敏感数据。由于设备可以进行越狱，且越狱之后数据不会丢失，则导致本地存储敏感信息将造成较大风险。
利用场景	攻击者通过本地存储的身份认证信息发起攻击。
风险等级	高风险，用户信息、登录密码、支付密码等高敏感度信息泄露。
测试结果	安全

4.2.2 缺陷代码分析

重点审查此类代码，审查发现不包含敏感数据

```
1002
1003  - (void)autoTrackAppStart {
1004      // 是否首次启动
1005      BOOL isFirstStart = NO;
1006      if (![[NSUserDefaults standardUserDefaults] boolForKey:SA_HAS_LAUNCHED_ONCE]) {
1007          isFirstStart = YES;
1008          [[NSUserDefaults standardUserDefaults] setBool:YES forKey:SA_HAS_LAUNCHED_ONCE];
1009          [[NSUserDefaults standardUserDefaults] synchronize];
1010      }
1011
```

审查发现不包含敏感数据库入口等信息

```
@implementation MessageQueueBySqlite {
    sqlite3 *_database;
    JSONUtil *_jsonUtil;
    NSInteger _messageCount;
    CFMutableDictionaryRef _dbStmtCache;
}

- (void)closeDatabase {
    if (_dbStmtCache) CFRelease(_dbStmtCache);
    _dbStmtCache = NULL;

    sqlite3_close(_database);
    sqlite3_shutdown();
    SADebug(@"%@ close database", self);
}

- (void)dealloc {
    [self closeDatabase];
}

- (id)initWithFilePath:(NSString *)filePath {
    self = [super init];
    _jsonUtil = [[JSONUtil alloc] init];
    if (sqlite3_initialize() != SQLITE_OK) {
        SAError(@"failed to initialize SQLite.");
        return nil;
    }
    if (sqlite3_open_v2([filePath UTF8String], &_database, SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, NULL) == SQLITE_OK) {
        // 创建一个缓存表
```

同时还会审查代码中是否包含明文存储的敏感数据，以及是否包含后台、测试地址等信息

4.2.3 修复建议

若存在，或后续功能扩展需要涉及到敏感数据，可通过如下方式进行，以保证代码安全性。

- 针对 UserDefault，建议加密存储，而且建议使用用户输入数据作为加密秘钥（但是可能会影响到自动登录）；
- 对于其他文件，进行独立加密，并解密读取即可；
- 关于加密秘钥，其实也可以从服务器获取，但是数据库就需要存储多一个字段用于存储每个用户的加密秘钥，且保证用户秘钥都不相同。

4.3 Webview 跨域请求

4.3.1 漏洞概述

漏洞描述	检测是否存在相关的 Webview 跨域请求漏洞。该漏洞可以通过 file://伪协议进行本地文件读取并显示。该漏洞可能被攻击者利用获取本地数据文件。
利用场景	攻击者获取本地数据文件

风险等级	低风险，需要在本地 HTML 中进行利用，所以风险低
测试结果	安全

4.3.2 缺陷代码分析

重点审查是否加载 HTML 程序，审查发现未加载，无风险。

4.3.3 修复建议

若存在，或后续功能扩展需要涉及到加载 html，可通过如下方式进行，以保证代码安全性。

如果非本地加载 html，使用 WKWebView（默认设置）替换 UIWebView。WKWebView 默认 allowFileAccessFromFileURLs 和 allowUniversalAccessFromFileURLs 属性为 false。

如果本地加载 html 需要开启属性，可以先进行本地文件加密（秘钥需要根据设备变化）；关键信息存储 keychain，防止读取。

4.4 缓冲区溢出

4.4.1 漏洞概述

漏洞描述	系统向缓冲区空间中填入超过缓冲区容量大小的数据，从而破坏内存中的的相关数据结构从而形成漏洞。
利用场景	攻击者可以利用此类漏洞执行任意代码，或者进行拒绝服务攻击
风险等级	高风险，包含危险函数，且未进行长度校验
测试结果	安全

4.4.2 缺陷代码分析

重点审查是否包含危险函数，审查发现危险函数，无风险。
危险函数有 strcpy、strcat 等。

```
122
123     int hoisted_method_count = !directOriginalMethod && !directAlternateMethod ? 2 : 1;
124     struct objc_method_list *hoisted_method_list = malloc(sizeof(struct objc_method_list) + (sizeof(struct ob
125     hoisted_method_list->obsolete = NULL;    // sooth valgrind - apparently ObjC runtime accesses this value
126     hoisted_method_list->method_count = hoisted_method_count;
127     Method hoisted_method = hoisted_method_list->method_list;
```

4.4.3 修复建议

避免使用 strcpy 和 strcat 方法，改用 strlcpy 和 strlcat 方法，如果必须使用 strcpy 时需要进行 strnlen 验证。

4.5 安全策略设置

4.5.1 漏洞概述

漏洞描述	密码策略、界面切换保护以及安全退出等方面未作安全策略。
利用场景	攻击者安全策略未作，进行恶意登录。
风险等级	中风险
测试结果	安全

4.5.2 缺陷代码分析

SDK 代码，未见相关应用代码。

4.5.3 修复建议

若存在，或后续功能扩展需要涉及到扩展界面访问功能，可通过如下方式进行，以保证代码安全性。

1. 验证原密码才允许设置新密码。
2. 针对弱口令进行重复数字、连续数字、连续字符、关键字符串 (admin、test、hahaha)、常见口令（1qaz2wdc、123qwe!@#）之类的密码进行过滤。
3. 切换界面，需清除前期已输入的登录信息。
4. 退出时消除其身份凭证，等。

4.6 网络安全通信

4.6.1 漏洞概述

漏洞描述	1、使用 HTTP 协议进行通信过程中可能会遭受中间人攻击，威胁数据安全。 2、存在忽略服务端证书校验的安全漏洞，对服务器没有校验或者没有在校验错误时候进行错误提示等。导致攻击者可通过伪造证书等手法进行攻击获取
利用场景	攻击者通过中间人攻击，获取敏感信息。
风险等级	中风险
测试结果	安全

4.6.2 缺陷代码分析

重点审查此类代码，审查发现未明确规定协议、证书，不存在风险代码

```objc
/**
 SASecurityPolicy 是参考 AFSecurityPolicy 实现
 使用方法与 AFSecurityPolicy 相同
 感谢 AFNetworking: https://github.com/AFNetworking/AFNetworking
 */
@interface SASecurityPolicy : NSObject <NSSecureCoding, NSCopying>

/// 证书验证类型，默认为：SASSLPinningModeNone
@property (readonly, nonatomic, assign) SASSLPinningMode SSLPinningMode;

/// 证书数据
@property (nonatomic, strong, nullable) NSSet <NSData *> *pinnedCertificates;

/// 是否信任无效或者过期证书，默认为：NO
@property (nonatomic, assign) BOOL allowInvalidCertificates;

/// 是否验证 domain
@property (nonatomic, assign) BOOL validatesDomainName;

/**
 从一个 Bundle 中获取证书

 @param bundle 目标 Bundle
 @return 证书数据
 */
+ (NSSet <NSData *> *)certificatesInBundle:(NSBundle *)bundle;

/**
 创建一个默认的验证对象
```

SASecurityPolicy.h ● | NSString+HashCode.m | *SASecurityPolicy.m* × | SALogger.h | heatmap.min.js

```objc
        if (certificates) {
            CFRelease(certificates);
        }

        continue;
    }
    CFRelease(policy);

    return [NSArray arrayWithArray:trustChain];
}

#pragma mark -

@interface SASecurityPolicy()
@property (readwrite, nonatomic, assign) SASSLPinningMode SSLPinningMode;
@property (readwrite, nonatomic, strong) NSSet *pinnedPublicKeys;
@end

@implementation SASecurityPolicy

+ (NSSet *)certificatesInBundle:(NSBundle *)bundle {
    NSArray *paths = [bundle pathsForResourcesOfType:@"cer" inDirectory:@"."];

    NSMutableSet *certificates = [NSMutableSet setWithCapacity:[paths count]];
    for (NSString *path in paths) {
        NSData *certificateData = [NSData dataWithContentsOfFile:path];
        [certificates addObject:certificateData];
```

4.6.3 修复建议

若存在，或后续功能扩展需要涉及到扩展通信相关功能，可通过如下方式进行，以保证代码安全性。

建议客户端同服务器进行通信时信道使用 SSL 加密信道进行传输，同时保证加密信道的本身安全（SSLV2，SSLV3 已被证明存在漏洞，建议至少使用 TLSV1.1 以上的算法），或是在通信过程中自实现类似 TLS 协议的算法，同时也要保证自实现算法的安全性针对弱口令进行重复数字、连续数字、连续字符、关键字符串 (admin、test、hahaha)、常见口令 (1qaz2wdc、123qwe!@#) 之类的密码进行过滤。

单向证书校验（AFNetworking 为例）

```
//  before AFNetworking 2.6
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];

NSString *cerPath = [[NSBundle mainBundle] pathForResource:@"https" ofType:@"cer"];
NSData *certData = [NSData dataWithContentsOfFile:cerPath];
AFSecurityPolicy *securityPolicy = [[AFSecurityPolicy alloc] init];
[securityPolicy setAllowInvalidCertificates:NO];// YES if self sign
[securityPolicy setPinnedCertificates:@[certData]];
[securityPolicy setSSLPinningMode:AFSSLPinningModeCertificate];
[securityPolicy setValidatesDomainName:YES];
[securityPolicy setValidatesCertificateChain:NO];

manager.securityPolicy = securityPolicy;
```

对于 AFNetworking3.0 及之后：

```
+(AFSecurityPolicy *)customSecurityPolicy
{
    NSString *path = [[NSBundle mainBundle]pathForResource:certificate ofType:@"cer"];
    NSLog(@"cerPath-%@",path);
    NSData *certData = [NSData dataWithContentsOfFile:path];

    AFSecurityPolicy *securityPolicy = [AFSecurityPolicy policyWithPinningMode:
        AFSSLPinningModeCertificate];
    [securityPolicy setAllowInvalidCertificates:YES];
    [securityPolicy setValidatesDomainName:NO];
    NSSet *set = [NSSet setWithArray:@[certData]];
    [securityPolicy setPinnedCertificates:set];

    return securityPolicy;

}
```

需要在请求方法中进行调用：

```
[manager setSecurityPolicy:[self customSecurityPolicy]];
```

五. 审计结果及建议

通过本次源代码安全审计活动，我们发现神策科技 IOS SDK 的安全防护策略和技术防控功能优秀，未见安全漏洞代码。

六. 感谢

感谢项目实施过程中神策科技相关部门协调工作，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