APP WALLET AUDIT REPORT

for

BHOP CONSULTANTING PTE. LTD

Prepared By: Shuxiao Wang

Jan. 06, 2021

文档摘要

客户信息	BHOP Consultanting Pte. Ltd
报告标题	App Wallet Audit Report
审计标的	HBTC Wallet
报告版本	1.0
报告作者	Huaguo Shi
审计人员	Huaguo Shi, Xin Li
报告复核	Chiachih Wu, Shuxiao Wang
报告终审	Xuxian Jiang
报告状态	Confidential

版本信息

版本	日期	审计负责人	描述
1.0	Jan. 06, 2021	Huaguo Shi	Final Release
1.0-rc1	Dec. 28, 2020	Huaguo Shi	Release Candidate #1
0.3	Dec. 25, 2020	Huaguo Shi	Additional Findings #2
0.2	Dec. 21, 2020	Huaguo Shi	Additional Findings #1
0.1	Dec. 20, 2020	Huaguo Shi	Initial Draft

联系方式

关于本审计报告的更多详细信息，请联系PeckShield [10] 。

联系人	Shuxiao Wang
电话	+86 173 6454 5338
电子邮件	contact@peckshield.com

Contents

1 介绍

我们(PeckShield [10]) 受客户委托对 **HBTC Wallet** 的HBTC钱包代码进行安全审计，本次审计范围包括三类客户端：Android/iOS/Web。我们在报告中概述了我们的系统方法 评估App 整体实施中的潜在安全问题，公开 钱包代码和设计文档之间可能存在语义上的不一致，并提供其他建议或改进建议。 分析结果表明， 该 **HBTC Wallet** 特定版本的程序代码存在若干安全性隐患存在一定的改进空间。修复的方式请参考第3章 3检测结果详情部分的内容。本文档对审计结果作了分析和阐述。

1.1 关于 HBTC Wallet

HBTC Chain 提供了基于区块链的新一代的去中心化资产托管和清算技术。HBTC Wallet 是基于区块链HBTC Chain 开发的数字托管单元，包括其他为方便用户使用区块链系统而开发的辅助工具。 HBTC Wallet的基本信息如下：

Table 1.1: HBTC Wallet的基本信息

条目	描述
发行方	BHOP Consultanting Pte. Ltd
网站	https://hbtcchain.io/
类型	Wallet App
平台	Android/iOS/Web
开发语言	C/Java/Type Script
审计方法	白盒
审计完成时间	Jan. 06, 2021

以下是审计对象（即钱包代码）相关信息：

- Android: https://github.com/hbtc-chain/wallet-android (f4e75f8)
- iOS: https://github.com/hbtc-chain/wallet-ios (9a28e11)
- Web: https://github.com/hbtc-chain/wallet-web (0d06d72)

下面是修复所有审计问题后提交的代码信息：

- Android: https://github.com/hbtc-chain/wallet-android (e7f4e87)
- iOS: https://github.com/hbtc-chain/wallet-ios (9052e84)
- Web: https://github.com/hbtc-chain/wallet-web (832614a)

1.2 关于PeckShield

PeckShield (派盾) 是面向全球的业内顶尖区块链安全团队，以提升区块链生态整体的安全性、隐私性以及可用性为己任，通过发布行业趋势报告、实时监测生态安全风险，负责任曝光0day漏洞，以及提供相关的安全解决方案和服务等方式帮助社区抵御新兴的安全威胁。可以通过下列联系方式联络我们： Telegram (https://t.me/peckshield), Twitter (twitter), or Email (contact@peckshield.com).

Table 1.2: 危害性定义

影响力 \ 可能性	高	中	低
高	致命	高危	中危
中	高危	中危	低危
低	中危	低危	低危

1.3 评估方法和模型

为了检测评估的标准化，我们根据OWASP Risk Rating Methodology [9]定义下列术语：

- 可能性： 表示某个特定的漏洞被发现和利用的可能性；
- 影响力： 度量了（利用该漏洞的）一次成功的攻击行动造成的损失；
- 危害性： 显示该漏洞的危害的严重程度；

可能性和影响力各自被分为三个等级：高、中和低。危害性由可能性和影响力确定，分为四个等级：致命、高危、中危、低危，如表 1.2所示。

为了评估风险，我们详细按类别列出可能存在风险项列表，我们仔细核查项目列表的风险点并做安全评估。 对于检查项目，如果我们的工具或分析未发现任何问题，该项目则是安全的。 对于发现的任何问题我们会进一步进行测试以确认结果。 如有必要我们还将另外做 PoC 来测试风险存在的可能性。 该审核涵盖了多个平台上的应用程序，覆盖每个平台的特性和应用场景。 检查项目的具体列表在 Table 1.3 中显示。

我们根据以下审核步骤执行审核:

- 自动静态分析：首先以自研的自动化静态检测工具侦测常见代码和系统应用级别漏洞。在整个漏洞扫描测试中，我们将根据漏扫工具生成的错误日志文件来重现每个问题。 对于每个漏洞案例，我们将进一步分析原因并检查它是否确实是漏洞。 一旦确认存在风险，我们将对其进行进一步分析，这是作为白盒审核的一部分。

- 业务逻辑分析: 我们接下来将了解业务逻辑，审查系统范围的操作，并检查不同组件之间的交互， 仔细检查与钱包相关的逻辑以发现可能存在的漏洞和异常错误。 PeckShield 审核人员会对程序的源代码进行全面检查和理解，然后创建特定的测试用例，执行并分析结果。这方法的目的是要检查软件中是否有内部安全漏洞、意外输出、路径损坏或结构不良等问题。

- 其他建议: 我们还提供有关编码的其他建议，针对各个应用开发平台的开发代码规范、代码优化以及第三方库的使用等方面提出建议。

为了更好地描述我们发现的每个问题，我们还基于常见弱点对发现进行了分类 Common Weakness Enumeration (CWE-699) [8]， 这是社区开发的软件脆弱性类型列表，可以更好地围绕软件开发中经常遇到的软件开发问题进行分类和组织。 我们使用 Table 1.4 中的CWE类别对发现的问题进行分类。

Table 1.3: The Full Audit Checklist

Platform	Category	Check Item
Android	System Components	Export of Component Activity
		Export of Component Service
		Export of Component Broadcast Receiver
		Export of Component Content Provider
	Android Secure Development	Un-obfuscated Java Code
		Arbitrarily Debugging
		Intent Scheme URLs Attack
		Dynamically and Safely Loading DEX File
		Executing Command Function in DLL
	System Security Detection	Signature Validation
		Detection on root/hook
		App Self-protection
iOS	Known Vulnerability	URL Schema Vulnerability Detection
		AFNetworking SSL Vulnerability Detection
		XcodeGhost Virus Detection
		"Youmi" Malicious SDK Detection
		iBackDoor Backdoor Detection
	System Security Detection	Detection on Jailbreak
Android/ iOS/Web	Third-party Plug-in	Security of using third-party plugins
	Common Secure Development	Correct Random Number
		Sensitive Information Printed by Log
	Communication Security	Network Communication Security
		Network Proxy Security
	Data Protection	Program Data Arbitrarily Backup
		Global File Security
		Configuration File Security
		Clipboard Security
		Anti-screenshot
	Business Logic	Private Key Generating
		Encryption Algorithm
		Password Strength
		Mnemonic Words Generating
		Private Key and Mnemonic Words Storage
		Local Sensitive Data Storage
		Mnemonic Words Import
		Private Key Import
		User Input Security
		Transaction Logic
		Wallet Communication
		Password Strength
		Password Updating
		Server Interaction Logic
		Functionality Integrity

Table 1.4: Common Weakness Enumeration (CWE) Classifications Used in This Audit

Category	Summary
Configuration	Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.
Resource Management	Weaknesses in this category are related to improper management of system resources.
Behavioral Issues	Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logics	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	Weaknesses in this category are related to improper use of arguments or parameters within function calls.
Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an exploitable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

1.4 免责声明

请注意该审计报告并不保证能够发现钱包软件中存在的一切安全问题，即评估结果并不能保证发现所有可能存在的区块链软件安全性问题。我们一向认为单次审计结果可能并不全面，因而推荐采取多个独立审计团队进行审计和公开的漏洞奖赏计划相结合的方式来确保项目的安全性。最后必须要强调的是，审计结果仅针对钱包软件代码的安全性，不构成任何投资建议。

2 检测结果

2.1 总结

这是在分析 HBTC Wallet 实现之后我们发现的问题摘要。 在审核的第一阶段，我们学习并分析钱包源代码，通过内部静态代码分析工具运行分析。 此处的目的是静态地识别已知的编码错误，然后通过人工分析的方式验证我们工具所发的问题。 第二阶段我们进一步人工检查业务逻辑，检查操作系统、分析私钥存储、签名验证的安全性问题，并仔细检查各个方面可能存在的安全漏洞和风险隐患。

Severity	# of Findings	
严重	0	
高危	0	
中危	6	■■■■■■
低危	2	■■
参考	3	■■■
总计	11	

到目前为止，我们已经确定了一系列潜在安全风险：其中一些从钱包开发者的角度会忽略的问题，例如钱包的系统平台的安全性问题。还有一些其他安全风险如私钥存储和开发Keystore泄露等问题。 因此，对于每个未确认的问题，我们自己开发一些测试用例来进行推理、复现和验证。 经过进一步的分析和内部讨论，我们确定了需要提出并引起更多重视的几个问题，这些问题在上表中进行了分类。 详细信息在下一章中都有做具体描述。

2.2 主要发现

我们在本次审计中我们发现了11个安全隐患的存在，包括 0 个高危漏洞, 6 个中危漏洞, 2 个低危漏洞, 和 3 个修改建议. 主要问题已经修复，如下表 2.1所示:

Table 2.1: 主要发现

编号	严重性	平台	名称	分类	状态
PVE-001	参考	Android	插件存在AES弱加密漏洞	Coding Practices	Confirmed
PVE-002	中危	Android	xuexiangjys 插件存在调用Shell命令风险	Coding Practices	Fixed
PVE-003	参考	Android	xuexiangjys 插件存在Zipper Down漏洞	Coding Practices	Fixed
PVE-004	参考	Android	xuexiangjys 插件存在隐式意图调用漏洞	Coding Practices	Fixed
PVE-005	低危	Android	程序数据可任意备份	Business Logic	Fixed
PVE-006	中危	Android/iOS/Web	剪贴板复制私钥	Business Logic	Confirmed
PVE-007	中危	Android/iOS	root/hook系统安全性检测	Business Logic	Fixed
PVE-008	中危	iOS	私钥/助记词存储存在安全风险	Security Features	Fixed
PVE-009	中危	Web	resolve-url-loader 插件低版本漏洞	Coding Practices	Fixed
PVE-010	低危	Android/iOS/Web	设置密码强度无校验	Business Logic	Confirmed
PVE-011	中危	Android/iOS	缺少网络代理检测	Business Logic	Fixed

除了已发现的问题外，我们强烈建议对于任何面向用户的应用程序和服务，需要搭建必要的风险控制和制定应急机制， 并确保在产品发布前先启动风险控制和应急方案。 具体发现细节请参考第 3 章。

3 检测结果详情

3.1 bitcoinj 插件存在AES弱加密漏洞

- ID: PVE-001
- 危害性: 参考
- 可能性: N/A
- 影响力: N/A
- 平台: Android
- Target: `org.bitcoinj.crypto`
- Category: Coding Practices [6]
- CWE subcategory: CWE-1104 [1]

漏洞描述

钱包中使用 第三方插件 `bitcoinj-core-0.15.8` 实现钱包核心功能， 我们通过审计分析现有代码中所使用的 `bitcoinj` API功能是没问题的， 但是我们再进一步审计的时候发现插件的 `org.bitcoinj.crypto.BIP38PrivateKey` 的 `encryptNoEC()/decryptNoEC()`实现中， 存在一个算法安全风险：在AES加密时使用的是 `AES/ECB/NoPadding` 的模式，ECB是将文件分块后对文件块做同一加密，破解加密只需要针对一个文件块进行解密，降低了破解难度和文件安全性。 虽然当前代码并没有使用,但仍然提醒开发者需要注意此API风险，不要使用此接口。

```
128     private ECKey decryptNoEC(String normalizedPassphrase) {
129         try {
130             byte[] derived = SCrypt.generate(normalizedPassphrase.getBytes(
                    StandardCharsets.UTF_8), addressHash, 16384, 8, 8, 64);
131             byte[] key = Arrays.copyOfRange(derived, 32, 64);
132             SecretKeySpec keyspec = new SecretKeySpec(key, "AES");

134             DRMWorkaround.maybeDisableExportControls();
135             Cipher cipher = Cipher.getInstance("AES/ECB/NoPadding");

137             cipher.init(Cipher.DECRYPT_MODE, keyspec);
138             byte[] decrypted = cipher.doFinal(content, 0, 32);
139             for (int i = 0; i < 32; i++)
140                 decrypted[i] ^= derived[i];
141             return ECKey.fromPrivate(decrypted, compressed);
142         } catch (GeneralSecurityException x) {
143             throw new RuntimeException(x);
144         }
145     }
```

Listing 3.1: org.bitcoinj.crypto.BIP38PrivateKey

修改建议 可以在开发文档中做相应的API风险说明，或者从github上下载bitcoinj插件源码（https://github.com/bitcoinj/bitcoinj）编译修复后使用。

状态 该问题已确认， bitcoinj-core-0.15.8 是钱包开发所必需的插件之一， 因为 bitcoinj-core-0.15.8 插件没有再更新版本，除非开发者自己用开源代码修复后重新编译并导入才能解决。 但由于当前的实现中未使用有缺陷的API，因此团队决定忽略该问题。

3.2 xuexiangjys 插件存在调用Shell命令风险

- ID: PVE-002
- 危害性: 中危
- 可能性: 低危
- 影响力: 高危
- 平台: Android
- Target: `com.xuexiang.xutil.common.ShellUtils`
- Category: Coding Practices [6]
- CWE subcategory: CWE-1104 [1]

漏洞描述

钱包中二维码扫描功能中引用了第三方插件 `com.github.xuexiangjys.XUtil:xutil-core v1.1.5` ，在审计的时候我们发现此插件的 `com.xuexiang.xutil.common.ShellUtils.execCommand` 中 运行调用了 `Runtime` 的 `exec` 方法执行命令，并有尝试调用 `su` 。虽然分析未发现实际危害，但仍建议慎用此插件。

```
128     /**
129      * execute shell commands
130      *
131      * @param commands          command array
132      * @param isRoot            whether need to run with root
133      * @param isNeedResultMsg whether need result msg
134      * @return <ul>
135      * <li>if isNeedResultMsg is false, {@link CommandResult#successMsg}
136      * is null and {@link CommandResult#errorMsg} is null.</li>
137      * <li>if {@link CommandResult#result} is -1, there maybe some
138      * excepiton.</li>
139      * </ul>
140      */
141     public static CommandResult execCommand(String[] commands, boolean isRoot,
142                                             boolean isNeedResultMsg) {
143         int result = -1;
144         if (commands == null  commands.length == 0) {
145             return new CommandResult(result, null, null);
146         }
147
148         Process process = null;
149         BufferedReader successResult = null;
150         BufferedReader errorResult = null;
151         StringBuilder successMsg = null;
152         StringBuilder errorMsg = null;
153
154         DataOutputStream os = null;
155         try {
156             process = Runtime.getRuntime().exec(
157                     isRoot ? COMMAND_SU : COMMAND_SH);
158             os = new DataOutputStream(process.getOutputStream());
159             for (String command : commands) {
160                 if (command == null) {
161                     continue;
162                 }
163
164                 // donnot use os.writeBytes(commmand), avoid chinese charset
165                 // error
166                 os.write(command.getBytes());
167                 os.writeBytes(COMMAND_LINE_END);
168                 os.flush();
169             }
```

Listing 3.2: com.xuexiang.xutil.common.ShellUtils

修改建议 移除 `com.github.xuexiangjys.XUtil:xutil-core v1.1.5` 插件，如果必须使用此功能建议重新修改并替换此版本（插件链接: `https://github.com/xuexiangjys/XUtil`）。

状态 这个问题已经被确认并在 `e7f4e87` 中被修复。

3.3 xuexiangjys 插件存在 ZipperDown 漏洞

- ID: PVE-003
- 危害性: 参考
- 可能性: N/A
- 影响力: N/A
- 平台: Android
- Target: `com.xuexiang.xutil.file.ZipUtils`
- Category: Coding Practices [6]
- CWE subcategory: CWE-1104 [1]

漏洞描述

钱包中二维码扫描功能中引用了第三方插件 `com.github.xuexiangjys.XUtil:xutil-core v1.1.5` ，在审计的时候我们发现此插件的 `com.xuexiang.xutil.common.ShellUtils.execCommand`中存在ZipperDown漏洞： 在解压 zip 文件，使用 `getName()` (line 318 和 line 324)获取压缩文件名后未对名称进行校验。攻击者可构造恶意zip文件，被解压的文件将会进行目录跳转被解压到其他目录，覆盖相应文件导致任意代码执行。 虽然当前代码并没有使用,但仍然提醒开发者需要注意此API风险，由于此项目代码开源，更应尽量避免此风险。

```
307     public static List<File> unzipFileByKeyword(final File zipFile,
308                                                 final File destDir,
309                                                 final String keyword)
310             throws IOException {
311         if (zipFile == null  destDir == null) return null;
312         List<File> files = new ArrayList<>();
313         ZipFile zf = new ZipFile(zipFile);
314         Enumeration<?> entries = zf.entries();
315         if (isSpace(keyword)) {
316             while (entries.hasMoreElements()) {
317                 ZipEntry entry = ((ZipEntry) entries.nextElement());
318                 String entryName = entry.getName();
319                 if (!unzipChildFile(destDir, files, zf, entry, entryName)) return files;
320             }
321         } else {
322             while (entries.hasMoreElements()) {
323                 ZipEntry entry = ((ZipEntry) entries.nextElement());
324                 String entryName = entry.getName();
325                 if (entryName.contains(keyword)) {
326                     if (!unzipChildFile(destDir, files, zf, entry, entryName)) return
                            files;
327                 }
328             }
329         }
330         return files;
331     }
```

Listing 3.3: com.xuexiang.xutil.file.ZipUtils

修改建议 移除 com.github.xuexiangjys.XUtil:xutil-core v1.1.5 插件，如果必须使用此功能建议重新修改并替换此版本（插件链接 https://github.com/xuexiangjys/XUtil）。

状态 这个问题已经被确认并在 e7f4e87 中被修复。

3.4 xuexiangjys 插件存在隐式意图调用漏洞

- ID: PVE-004
- 危害性: 参考
- 可能性: N/A
- 影响力: N/A
- 平台: Android
- Target: com.xuexiang.xutil
- Category: Coding Practices [6]
- CWE subcategory: CWE-1104 [1]

漏洞描述

钱包中二维码扫描功能中引用了第三方插件 com.github.xuexiangjys.XUtil:xutil-core v1.1.5 ，在审计的时候我们发现此插件有多处存在隐式意图调用，即调用封装Intent时采用隐式设置，只设定action而未限定具体的接收对象，导致Intent可被其他应用获取并读取其中数据。Intent隐式调用发送的意图可能被第三方劫持，可能导致内部隐私数据泄露。有隐式调用风险代码如下：

```
356     public static void shutdown() {
357         ShellUtils.execCommand("reboot -p", true);
358         Intent intent = new Intent("android.intent.action.ACTION_REQUEST_SHUTDOWN");
359         intent.putExtra("android.intent.extra.KEY_CONFIRM", false);
360         XUtil.getContext().startActivity(intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK))
                ;
361     }
```

Listing 3.4: com.xuexiang.xutil.system.DeviceUtils

```
84      public static void openWirelessSettings() {
85          XUtil.getContext().startActivity(new Intent(android.provider.Settings.
                ACTION_WIRELESS_SETTINGS)
86                  .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK));
87      }
```

Listing 3.5: com.xuexiang.xutil.net.NetworkUtils

```
126     public static void openAppSettings() {
127         Intent intent = new Intent("android.settings.APPLICATION_DETAILS_SETTINGS");
128         intent.setData(Uri.parse("package:" + XUtil.getContext().getPackageName()));
129         XUtil.getContext().startActivity(intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK))
                ;
130     }
```

Listing 3.6: com.xuexiang.xutil.system.PermissionUtils

修改建议 移除 com.github.xuexiangjys.XUtil:xutil-core v1.1.5 插件，如果必须使用此功能建议重新修改并替换此版本（插件链接 https://github.com/xuexiangjys/XUtil）。

状态 这个问题已经被确认并在 e7f4e87 中被修复。

3.5 程序数据可任意备份

- ID: PVE-005
- 危害性: 低危
- 可能性: 低危
- 影响力: 中危
- 平台: Android
- Target: AndroidManifest.xml
- Category: Coding Practices [6]
- CWE subcategory: CWE-281 [3]

漏洞描述

在Android系统中，AndroidManifest.xml 文件中 android: allowBackup 为 true 时 App数据可以被备份导出的，由于钱包软件数据安全的特殊性，不应允许在用户未授权的前提下导出用户的任何数据，以免带来风险。

修改建议 将 AndroidManifest.xml 配置文件中设置为android:allowBackup="false"。

状态 这个问题已经被确认并在 e7f4e87 中被修复。

3.6 通过剪贴板复制私钥

- ID: PVE-006
- 危害性: 中危
- 可能性: 低危
- 影响力: 高危
- 平台: Android/iOS/Web
- Target: N/A
- Category: Business Logic [7]
- CWE subcategory: CWE-841 [4]

漏洞描述

在审计私钥存储导出功能时发现一个可能会导致私钥泄露的一个安全隐患，钱包允许用户通过将私钥复制到剪贴板上来方便用户保存到其他应用程序，这种操作是钱包类 App 出于产品的易用性考虑而采用这样设计方案。然而，当前流行一种黑客的攻击手段是通过黑客App在后台运行，会自动识别并复制剪贴板中的私钥和密码数据并上传到黑客服务器，导致私钥泄露。

修改建议 最好的解决方案是禁止通过剪贴板复制私钥功能。但是如果考虑对用户体验的影响，建议可以在选择剪贴板备份私钥前，明确提示用户复制私钥存在的风险，并在复制备份私钥返回私钥备份界面时提供清除剪贴板功能，这样也可以极大降低剪贴板私钥泄露的风险。

状态 这个问题在 iOS 平台钱包版本已修复 9052e84，但在移动平台(Android/iOS)中仍保留此问题未修复。

3.7 系统安全性检测

- ID: PVE-007
- 危害性: 中危
- 可能性: 中危
- 影响力: 中危
- 平台: Android/iOS
- Target: N/A
- Category: Business Logic [7]
- CWE subcategory: CWE-841 [4]

漏洞描述

移动平台操作系统一旦被 root/hook 后,攻击者可以直接获取客户端中的本地所有信息,比如私钥或者密码等用户信息,或者通过注入代码的方式盗取用户临时存储在内存中的私钥数据。

修改建议 建议在平台中检测用户手机是否存在 root/hook 风险并提示用户，由用户自行选择是否继续使用应用。

检测方法建议 Android 平台的root检测项较多，具体可参考链接 https://github.com/hamada147/AndroidRootChecker/blob/master/RootChecker.java，iOS平台可参考下面代码进行检查:

```
+ (BOOL) isJeilBreakingByPath {
    NSArray *jailbreak_tool_paths = @[@"/Applications/Cydia.app",@"/Library/
        MobileSubstrate/MobileSubstrate.dylib",@"/bin/bash",@"/usr/sbin/sshd",@"/etc
        /apt"];
    for (int i=0; i<jailbreak_tool_paths.count; i++) {
        if ([[NSFileManager defaultManager] fileExistsAtPath:jailbreak_tool_paths[i
            ]]) {
            NSLog(@"The device is jail broken by path: %@!", jailbreak_tool_paths[i
                ]);
            return YES;
        }
    }
    return NO;
}
```

Listing 3.7: iOS越狱检测代码

状态 这个问题已经被确认并在 e7f4e87 和 9052e84 中被修复。

3.8 私钥和助记词存储存在安全风险

- ID: PVE-008
- 危害性: 中危
- 可能性: 低危
- 影响力: 高危
- 平台: iOS
- Target: N/A
- Category: Security Features [5]
- CWE subcategory: CWE-260 [2]

漏洞描述

钱包中私钥和助记词是最重要的数据信息，拥有私钥或助记词就是拥有全部钱包的资产。我们对iOS代码中私钥存储部分做了仔细分析，创建私钥函数的 `createAction()` 中调用了 `encryptSecretStorageJSON()`（line 80），privatekey的加密是用标准KeyStore存储方式存储钱包数据，并使用椭圆形算法生成慢哈希加密私钥，基本杜绝爆破私钥的可能。

```
63 - (void)createAction {
64     if (![self.textFieldView.textField.text isEqualToString:KUser.localPassword]) {
65         Alert *alert = [[Alert alloc] initWithTitle:LocalizedString(@"
               TwoPasswordInconsistent") duration:kAlertDuration completion:^{
66         }];
67         [alert showAlert];
68         return;
69     }
70     [MBProgressHUD showActivityMessageInView:nil];
71     NSLog(@"%@   %@",KUser.localPrivateKey,KUser.localPhraseString);
72     if (!IsEmpty(KUser.localPhraseString)) {
73         self.account = [Account accountWithMnemonicPhrase:KUser.localPhraseString];
74     } else if (!IsEmpty(KUser.localPrivateKey)) {
75         SecureData * data = [SecureData secureDataWithHexString:KUser.localPrivateKey];
76         self.account = [Account accountWithPrivateKey:data.data];
77     }  else {
78         self.account = [Account randomMnemonicAccount];
79     }
80     [self.account encryptSecretStorageJSON:KUser.localPassword callback:^(NSString *json
           ) {
81         [self backupKeystore:json];
82     }];
83 }
```

Listing 3.8: wallet-ios/Bluehelix/Bluehelix/Class/CreateWallet/XXRepeatPasswordVC.m

然而当我们进一步分析调用的 `backupKeystore()`，函数功能为将model对象数据保存到sqlite数据库中，但我们发现在代码中，model对象不仅保存了keystore数据，还保存了密码的md5值(line 91)、私钥和助记词密文。我们继续分析代码中私钥和助记词的密文数据，他们都采用 `AES(privatekey/mnemonicPhrase, password)` 的方式保存的。如果密码设置的不是很复杂，攻击者只要活得sqlite数据库数据，通过彩虹表攻击就可以很容易爆破私钥。

```
85  - (void)backupKeystore:(NSString *)json {
86      XXAccountModel *model = [[XXAccountModel alloc] init];
87      model.privateKey = [AESCrypt encrypt:self.account.privateKeyString password:KUser.
            localPassword];
88      model.publicKey = self.account.pubKey;
89      model.address = self.account.BHAddress;
90      model.userName = KUser.localUserName;
91      model.password = [NSString md5:KUser.localPassword];
92      model.keystore = json;
93      if (self.account.mnemonicPhrase && IsEmpty(KUser.localPhraseString)) {
94          NSString *mnemonicPhrase = [AESCrypt encrypt:self.account.mnemonicPhrase
                password:KUser.localPassword];
95          model.mnemonicPhrase = mnemonicPhrase;
96          model.backupFlag = NO;
97      } else {
98          model.mnemonicPhrase = @"";
99          model.backupFlag = YES;
100     }
101     model.symbols = [NSString stringWithFormat:@"btc,eth,usdt,%@",kMainToken];

103     if (KUser.accounts) {
104         for (XXAccountModel *a in KUser.accounts) {
105             if ([a.address isEqualToString:model.address]) {
106                 Alert *alert = [[Alert alloc] initWithTitle:LocalizedString(@"
                        PrivateKeyRepetition") duration:kAlertDuration completion:^{
107                 }];
108                 [alert showAlert];
109                 [[XXSqliteManager sharedSqlite] deleteAccountByAddress:model.address];
110             }
111         }
112     }
113     [[XXSqliteManager sharedSqlite] insertAccount:model];
114     KUser.address = model.address;
115     [MBProgressHUD hideHUD];
116     if (model.backupFlag) {
117         Alert *alert = [[Alert alloc] initWithTitle:LocalizedString(@"ImportSuccess")
                duration:kAlertDuration completion:^{
118             KWindow.rootViewController = [[XXTabBarController alloc] init];
119             [self showBiometricAlert];
120         }];
121         [alert showAlert];
122     } else {
123         XXCreateWalletSuccessVC *successVC = [[XXCreateWalletSuccessVC alloc] init];
124         successVC.text = KUser.localPassword;
125         [self.navigationController pushViewController:successVC animated:YES];
126         [self showBiometricAlert];
127     }
128     KUser.localPassword = @"";
129     KUser.localUserName = @"";
130     KUser.localPhraseString = @"";
131     KUser.localPrivateKey = @"";
```

```
132 }
```

Listing 3.9: wallet –ios/ Bluehelix / Bluehelix / Class/CreateWallet/XXRepeatPasswordVC.m

修改建议 删除此多余数据，仅保留原有 `Keystore` 即可(`KeyStore`中已经保存了密文私钥)。另如需备份助记词，也请按同样的方式将助记词加密存储到 `Keystore` 中。

状态 这个问题已经被确认并在 `9052e84` 中被修复。

3.9 resolve-url-loader 插件低版本漏洞

- ID: PVE-009
- 危害性: 中危
- 可能性: 低危
- 影响力: 高危
- 平台: Web
- Target: N/A
- Category: Coding Practices [6]
- CWE subcategory: CWE-1104 [1]

漏洞描述

Web 开发时会引入许多依赖的三方库或者官方库，我们在审计代码过程中，首先先检测依赖库的安全性，发现代码引用的库文件 `resolve-url-loader v3.1.1` 已知存在几个安全风险，其中严重的一个风险可以导致DOS攻击（https://npmjs.com/advisories/1556）

修改建议 将`package.json` 中 依赖的 `resolve-url-loader v3.1.1` 升级到 `resolve-url-loader v3.1.2` 以上，另外建议编译时如有依赖库有变化，请运行 `npm audit fix` 进行检查,确保所使用的库没有0day风险。

状态 这个问题已经被确认并在 `832614a` 中被修复。

3.10 设置密码强度无校验

- ID: PVE-010
- 危害性: 低危
- 可能性: 低危
- 影响力: Medium
- 平台: `Android/iOS/Web`
- Target:
- Category: Business Logic [7]
- CWE subcategory: CWE-841 [4]

漏洞描述

HBTC 钱包 采用的 `KeyStore` 来保存私钥密文的存储，只有输入正确的密码才能解密出私钥明文。我们分析时发现钱包中密码设置没有校验密码强度, 用户在没有软件限制的前提下，很容易将密码设置成简单的、可命中彩虹表的密码信息， 但由于文件存储使用的是标准 `KeyStore` 存储，极大降低爆破的风险，因此我们评估为低危。

修改建议 增加设置密码的强度限制，需要限定最小位数和大小写字母以及数字。(可以包含一些特殊字符)

状态 这个问题在Web平台钱包版本已修复 `832614a`, 但在移动平台(Android/iOS)中，考虑到用户的产品体验，仍保留此问题不做修复。

3.11 缺少网络代理检测

- ID: PVE-011
- 危害性: 中危
- 可能性: 低危
- 影响力: 高危
- 平台: Android/iOS
- Target: N/A
- Category: Business Logic [7]
- CWE subcategory: CWE-841 [4]

漏洞描述

在移动终端App中，iOS/Android 中都使用 https 协议进行网络通讯，这样可以有效的避免中间人攻击。 不过由于目前有很多网络抓包工具（如 Fiddler，Charles等），通过在移动端安装证书，可以实现嗅探、捕获程序 https 网络数据包，并可以篡改消息报文。 因此通常钱包类的App是不允许有网络代理存在的，发现网络代理应停止通讯。 但我们看到钱包移动端App并没有做相关保护。

修改建议 Android 平台 现有代码逻辑中支持用户证书，将其修改为系统证书验证失败后直接抛异常(line 222)。

```
214     @Override
215     public void checkServerTrusted(X509Certificate[] chain, String authType) throws
            CertificateException
216     {
217         try
218         {
219             defaultTrustManager.checkServerTrusted(chain, authType);
220         } catch (CertificateException ce)
221         {
222             throw new CertificateException("error in validating certificate" , ce);
223         }
224     }
```

Listing 3.10: wallet–android/Lib/Lib_Network/src/main/java/com/bhex/network/utils

iOS平台检测代理的代码片段仅供参考:

```
1   (BOOL)isVPNOn{
2       BOOL flag = NO;
3       CFDictionaryRef dicRef = CFNetworkCopySystemProxySettings();
4       const CFStringRef proxyCFstr = (const CFStringRef) CFDictionaryGetValue(dicRef,
            (const void*)kCFNetworkProxiesHTTPProxy);
5       NSString* proxy = (__bridge NSString *) proxyCFstr;
6       if (proxy != NULL){
7           flag = YES;
8       }
9       return flag;
10  }
```

状态 这个问题已经被确认并在 e7f4e87 和 9052e84 中被修复。

4 结论

我们对 HBTC Wallet 的三个平台的App（Android/iOS/Web)进行了安全审计，整体来看，HBTC Wallet 的代码整体逻辑以及架构设计是很严谨和专业的。但正如免责声明 1.4所述，该审计报告的结果并不意味着该钱包不存在其它安全问题，亦不构成任何投资建议。我们再次强调，即便钱包的安全性很高并不能保证您免受社会工程，物理威胁或人为错误的攻击，用户需要使用常识并应用基本安全方法来保护自身数字资产安全。

参考文献

[1] MITRE. CWE-1104: Use of Unmaintained Third Party Components. https://cwe.mitre.org/data/definitions/1104.html.

[2] MITRE. CWE-260: Password in Configuration File. https://cwe.mitre.org/data/definitions/260.html.

[3] MITRE. CWE-281: Improper Preservation of Permissions. https://cwe.mitre.org/data/definitions/281.html.

[4] MITRE. CWE-841: Improper Enforcement of Behavioral Workflow. https://cwe.mitre.org/data/definitions/841.html.

[5] MITRE. CWE CATEGORY: 7PK - Security Features. https://cwe.mitre.org/data/definitions/254.html.

[6] MITRE. CWE CATEGORY: Bad Coding Practices. https://cwe.mitre.org/data/definitions/1006.html.

[7] MITRE. CWE CATEGORY: Business Logic Errors. https://cwe.mitre.org/data/definitions/840.html.

[8] MITRE. CWE VIEW: Development Concepts. https://cwe.mitre.org/data/definitions/699.html.

[9] OWASP. Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

[10] PeckShield. PeckShield Inc. https://www.peckshield.com.