Tezos Foundation

Pentest tzBTC Android App

Document Name:	report_89395_Pentest_tzBTC_Android_App_v1.0.docx
Version:	v1.0
Project Number:	89395
Date of Delivery:	March 27th, 2024
Time of Test:	March 18th, 2024 - March 22nd, 2024
Author:	Lukasz Dykcik, Compass Security Schweiz AG
Classification:	STRICTLY CONFIDENTIAL

Compass Security Schweiz AG, Werkstrasse 20, CH-8645 Jona
T +41 58 510 36 00, team.csch@compass-security.com, www.compass-security.com

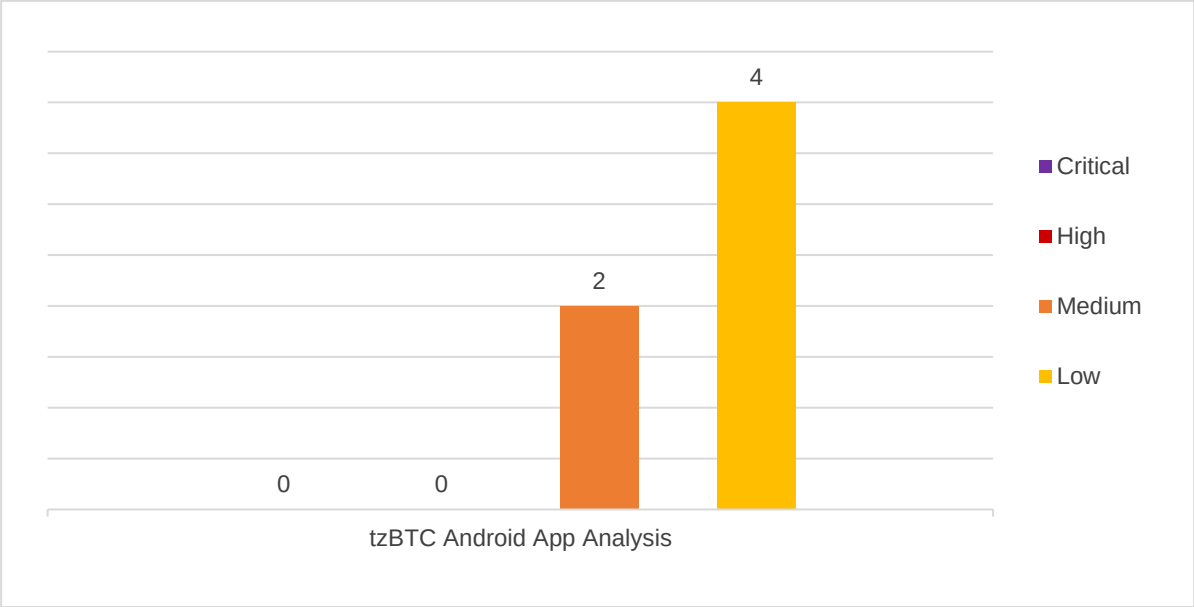
Executive Summary

Compass Security Schweiz AG conducted a security assessment of the tzBTC acurast processor app. This section briefly highlights the most important results and provides recommendations for future steps. Technical details are provided in subsequent chapters.

Results

The performed review of the mobile app did not result in a discovery of any high-rated vulnerabilities. The concept of using mobile devices with a hardware security element to provide decentralized validation used by tzBTC, as well as its implementation, was rated as mature. Nevertheless, a few security deficiencies have been identified and a number of possible attacks that would eradicate the trust put in the tzBTC processors have been found.

The following diagram gives an overview of the identified vulnerabilities and their severity:



General Recommendations

Compass Security recommends that critical vulnerabilities be immediately reviewed and addressed.

All remaining issues listed in the vulnerability table in chapter 3 as well as the presented possible attack vectors described in chapter 2 should be discussed internally. The risk of all the vulnerabilities should be assessed and addressed based on the company-internal risk management process and the technical rating proposed by Compass Security.

Table of Contents

1 Overview

This document is intended for project teams, development personnel, and other individuals concerned with the security of the tested app. The purpose of this document is to summarize the results of the security assessment.

1.1 Document Structure

Chapter	Content
-	Executive summary
1	Document overview, scope, and procedure
2	A list of the possible attack vectors as well as suggestions for their mitigation
3	A list of the identified weaknesses as well as suggestions for improvement
4	Protocol of the performed security tests
5	Appendix

1.2 Scope and Procedures

The security assessment covered the following:

Target	Module	Effort
tzBTC Acurast Processor App	M1 Android App Review	4 PD

2 Possible Attack Vectors

This chapter summarizes the attack vectors that are not immediately possible to be performed but their execution would require exploiting an additional vulnerability. A definition for each table column is given here:

No.	Reference	Attack Scenario	Requirements	Remediation	Difficulty	Comment
Each attack is numbered consecutively.	Reference to the corresponding test case in the following chapters, if applicable.	Explains the attack scenario identified during the assessment.	Explains the preconditions of the attack scenario to be exploited.	Explanation how the attack can be avoided or mitigated.	Compass rating of the difficulty performing the attack: ▪ **Low** ▪ **Medium** ▪ **High**.	Comment and information provided by the customer.

2.1 tzBTC Android App Analysis

No.	Reference	Attack Scenario	Requirements	Remediation	Difficulty	Comment
1.	-	**Faking Attestation** An attacker could obtain the valid key attestation by installing the app on an older non-rooted device. Then, by exploiting a privilege escalation vulnerability, the attacker will be able to undermine the app's integrity during runtime.	The attacker would need to be able to obtain root privileges on any of the allowed Android OS versions without unlocking the boot loader. For this an arbitrary local privilege elevation exploit would suffice. Note that there are already publicly available proof-of-concept exploits for older Android OS.	The OS version and OS patch level values are present in the key attestation in the reliable teeEnforced sequence. These values should be considered when labelling the device as attested. Ideally, only the latest patch level should be required to pass attestation and re-attestation should be performed after each patch release.	**Low**	
2.	-	**Job Escape** An attacker could deploy a malicious job on processors used for tzBTC. Then, exploit a weakness in the job processing component of the app and obtain code execution in the context of the Android app, not restricted by the JavaScript execution environment. Once the attacker is able to execute code within the app, using the cryptographic keys stored in the Keystore is not restricted.	This attack requires the ability to run attacker-controlled malicious jobs on processors used for the tzBTC. Furthermore, as just executing a job would not lead to the compromise of the tzBTC since each job uses its own keys, the attacker would need to escape from the job context into the app context.	For the tzBTC, it is already intended that the used processors only accept jobs from one specified account. It needs to be ensured that this account is appropriately protected from unauthorized access. Alternatively, multiple accounts could be used. A subset of processors used for tzBTC could accept jobs only from one account and another subset only from the other account. In this way, even if the attacker compromised one of the accounts and exploited some processors with a malicious job, other processors would remain unaffected.	**Medium**	

No.	Reference	Attack Scenario	Requirements	Remediation	Difficulty	Comment
3.	-	**App Installation on a Compromised Device** The attack would work as follows: an attacker compromises an older Android device and establishes persistence on the device that allows it to be updated to one of the versions of the Android OS that are considered acceptable in the attestation, without losing root access.	A way of rooting a device and establishing persistence that survives factory reset required for app provisioning is needed. Also, the way of maintaining root privileges on the device should not rely on unlocking the bootloader or modifying the boot partition since those changes would affect the key attestation. Furthermore, if the Android OS version that could be used to get the required persistent root access is not considered acceptable in the attestation, a way of maintaining root access after updating the device will be required.	It should be ensured that only latest patch levels of the Android OS are accepted for the attestation. Furthermore, regular Play Integrity checks should be performed to take advantage of the Google API providing device integrity checks.	**Medium**	
4.	-	**Compromising Locked Down Device** This attack could be performed after the app was provisioned on a not compromised device and attestation was successfully completed. The attacker would then attempt to exploit the locked down device to manipulate the integrity of the app running on it.	The attacker would need to exploit one of the exposed interfaces of the locked down device. Currently devices expose mostly the Wi-Fi and the USB mass storage interfaces. The exploit would either need to provide code execution on the Android device with root privileges or another privilege escalation exploit would need to be used afterwards. Note that the attacker is not restricted to a particular device hardware model as plenty of Android devices would be accepted as processors and exploitation of the exposed interfaces may be noticeably easier on some of the devices.	The locked down device should expose as few interfaces as possible. It is already planned to disable USB mass storage as it will not be needed in further releases. Furthermore, it should be considered that only devices from established manufacturers such as Google are accepted for the attestation of tzBTC processors.	**High**	

3 Vulnerabilities and Remediation

This chapter summarizes the security issues found during the security review. A definition for each table column is given here:

No.	Reference	Weakness	Threat	Remediation	Rating	Comment
Each issue is numbered consecutively.	Reference to the corresponding test case in the following chapters.	Explains the weakness identified during the assessment.	Explains the impact of the weakness if it were to be exploited.	Recommendation on how to address the weakness.	Compass rating of the weakness and the corresponding threat: ▪ **Critical** ▪ **High** ▪ **Medium** ▪ **Low** ▪ **Info** *See chapter 5 for a detailed rating description and color code definition.*	Comment and information provided by the customer.

3.1 tzBTC Android App Analysis

No.	Reference	Weakness	Threat	Remediation	Rating	Comment
1.	4.7 #1	**Spoofed Attestation** Untrustworthy devices with unlocked bootloader and modified boot partition are shown as attested in the acurast console.	Although in the key attestation present on the acurast parachain, it is shown that the root of trust for the key is not appropriate, relying on the values shown in the acurast console leads to the false conclusion.	It should be ensured that the values shown in the acurast console are trustworthy. In particular, all relevant elements of the key attestation, in particular the root of trust, should be correctly evaluated before the device is labeled as attested.	**Medium**	
2.	4.3.1 #1	**Unsupported Android Versions** The minimum API level of the app is set to 30. This API level corresponds to an Android version that is no longer supported.	Security updates are no longer available for devices running this specific version of Android. An Attacker could exploit known vulnerabilities to gain root privileges on the device and compromise the integrity of the app and the data it processes.	Only Android versions that can receive security patches should be supported by the app.	**Medium**	

No.	Reference	Weakness	Threat	Remediation	Rating	Comment
3.	4.7 #5,6	**Vague Chain of Trust** Users of the tzBTC can barely verify the trust assumptions on their own. Although the original, unparsed key attestations are present on the blockchain, to find them, particular transactions would need to be found. Parsed key attestations are put in the on-chain storage and are easily browsable, however, the validity of the attestation cannot be inducted from this parsed data. Furthermore, the integrity of the app cannot be externally verified as no hash of it is present in the attestation.	During the initial deployment of the app, a malicious version signed with the same certificate could be deployed and users of tzBTC would not be able to determine whether the legitimate or malicious app has access to the attested private key. Also, verification of the key attestation of the processor is cumbersome.	The issues should be mitigated to provide an easy way for users of the tzBTC to verify that only the legitimate app can use the attested private key. Note that it is already planned to provide a user-friendly view of the processors' key attestations and the problem of trusting the initial app installation is also planned to be addressed.	**Low**	
4.	4.3.2 #5	**No Stack Canaries** The app uses libraries that were not compiled with stack canaries: ▪ libsweet_b.so.	Stack canaries, also called stack cookies, are placed after buffers in a program to guard against buffer overflow vulnerabilities. If this feature is not enabled, the executable does not make use of this additional layer of protection.	Only libraries with enabled stack canaries should be used.	**Low**	
5.	4.3.2 #6	**No Relocation Read-Only (RELRO)** The libj2v8.so library for arm64 is not compiled with the Relocation Read-Only (RELRO) enabled.	When RELRO is not enabled in Android binary libraries, it leaves the door open for attackers to exploit memory-related vulnerabilities more easily. Without RELRO, the Global Offset Table (GOT) remains writable, and an attacker could potentially manipulate function pointers and control the program's execution flow.	Only libraries with RELRO enabled should be used.	**Low**	
6.	4.4.1 #1 4.4.2 #1	**Play Integrity Not Used** The Play Integrity API provided by Google is not used to detect whether the device where the app runs is rooted.	Without using the Play Integrity API, the trustworthiness of the job's execution on the processor may be diminished by attacks on the device's integrity that occurred after the initial key attestation and are detectable by Play Integrity.	The app should call the Play Integrity API at important moments to get a proof that job results are coming from the unmodified app binary running on a genuine Android device. More information: ▪ https://developer.android.com/google/play/integrity	**Low**	

No.	Reference	Weakness	Threat	Remediation	Rating	Comment
7.	4.8 #4 4.5.2 #1	**Job for tzBTC Still in Development** During the tests, the actual job code of the tzBTC was not yet ready. The security of the tzBTC was reviewed based on jobs with similar functionalities and the not-yet-finished tzBTC job.	-	It should be reviewed whether the final version of the tzBTC job differs significantly from the analyzed jobs and possibly introduces vulnerabilities into the tzBTC that could not be discovered during the performed security audit.	**Info**	

4 tzBTC Android App Analysis

4.1 Overview

4.1.1 Backend Environment / URIs

URI	Description
acurast-canarynet-node.prod.gke.acurast.com	CANARY environment

4.1.2 Devices

Device Model	Android Version	Root Status
Pixel 4a	13	Rooted.
Pixel 4a	11	Rooted.
Pixel 7a	13	Rooted.

4.2 Basic Information

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	App Name	-	Acurast Processor (CANARY)	**N/A**
2.	Version Name	-	1.4.0-canary	**N/A**
3.	Version Code	-	26	**N/A**
4.	Package Name	-	com.acurast.attested.executor.canary	**N/A**
5.	SHA256 Checksum of the app	-	5569b4db6e759fc8c827542e125c6ed1cd7933a25d5cecedc5c0c9ea44733d2a	**N/A**
6.	Contents of AndroidManifest.xml	-	See details.	**N/A**
7.	App Screenshot	-	See details.	**N/A**

Details #6

Android Manifest:

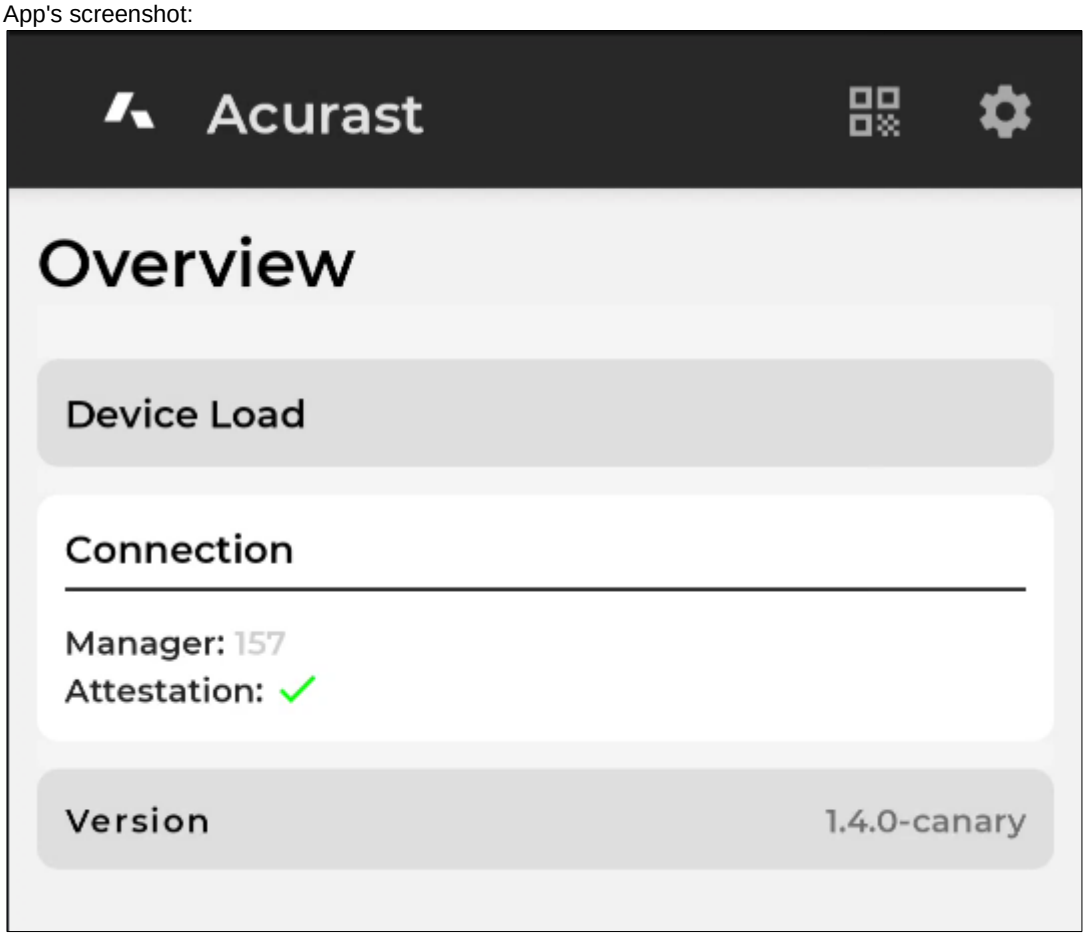
```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="26" android:versionName="1.4.0-canary"
android:compileSdkVersion="33" android:compileSdkVersionCodename="13"
package="com.acurast.attested.executor.canary" platformBuildVersionCode="33"
platformBuildVersionName="13"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="30" android:targetSdkVersion="33" />
    <uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"
/>
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.QUERY_ALL_PACKAGES" />
```

```
    <permission
android:name="com.acurast.attested.executor.canary.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"
android:protectionLevel="signature" />
    <uses-permission
android:name="com.acurast.attested.executor.canary.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"
/>
    <application android:theme="@style/Theme.Acurast" android:label="@string/app_name"
android:icon="@mipmap/ic_launcher" android:name="com.acurast.attested.executor.App"
android:testOnly="false" android:supportsRtl="true" android:extractNativeLibs="false"
android:roundIcon="@mipmap/ic_launcher_round"
android:appComponentFactory="androidx.core.app.CoreComponentFactory">
        <service android:name="com.acurast.attested.executor.services.V8ExecutorService"
android:enabled="true" android:exported="true" />
        <service android:name="com.acurast.attested.executor.services.JobFetcherService"
android:enabled="true" android:exported="true" />
        <service android:name="com.acurast.attested.executor.services.HeartbeatService"
android:enabled="true" android:exported="true" />
        <service android:name="com.acurast.attested.executor.services.OTAUpdateService"
android:enabled="true" android:exported="true" />
        <receiver android:name="com.acurast.attested.executor.InstallReceiver" />
        <receiver android:name="com.acurast.attested.executor.JobFetcherBroadcastReceiver"
android:enabled="true" android:exported="true">
            <intent-filter android:directBootAware="true">
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="android.intent.action.LOCKED_BOOT_COMPLETED" />
                <action android:name="android.intent.action.QUICKBOOT_POWERON" />
                <action android:name="android.intent.action.REBOOT" />
            </intent-filter>
        </receiver>
        <receiver android:name="com.acurast.attested.executor.HeartbeatBroadcastReceiver"
android:enabled="true" android:exported="true" />
        <receiver android:name="com.acurast.attested.executor.OTAUpdateBroadcastReceiver"
android:enabled="true" android:exported="true" />
        <receiver android:name="com.acurast.attested.executor.V8ExecutorBroadcastReceiver"
android:enabled="true" android:exported="true" />
        <receiver
android:name="com.acurast.attested.executor.AlarmPermissionBroadcastReceiver"
android:enabled="true" android:exported="true">
            <intent-filter>
                <action
android:name="android.app.action.SCHEDULE_EXACT_ALARM_PERMISSION_STATE_CHANGED" />
            </intent-filter>
        </receiver>
        <activity android:theme="@style/Theme.Acurast"
android:name="com.acurast.attested.executor.ui.MainActivity" android:exported="true"
android:excludeFromRecents="true" android:launchMode="singleInstance">
            <intent-filter>
                <category android:name="android.intent.category.HOME" />
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="http" android:host="executor.acurast.com" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.app.action.PROVISIONING_SUCCESSFUL" />
                <action android:name="android.app.action.PROFILE_PROVISIONING_COMPLETE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <receiver android:label="@string/app_name"
android:name="com.acurast.attested.executor.lockdown.LockdownDeviceAdminReceiver"
android:permission="android.permission.BIND_DEVICE_ADMIN" android:exported="true"
android:description="@string/app_name">
            <meta-data android:name="android.app.device_admin"
android:resource="@xml/device_admin_receiver" />
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
```

```
                <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
                <action android:name="android.app.action.PROFILE_OWNER_CHANGED" />
                <action android:name="android.app.action.DEVICE_OWNER_CHANGED" />
            </intent-filter>
        </receiver>
        <activity
android:name="com.acurast.attested.executor.lockdown.AdminPolicyComplianceActivity"
android:permission="android.permission.BIND_DEVICE_ADMIN" android:exported="true"
android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.app.action.ADMIN_POLICY_COMPLIANCE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
android:name="com.acurast.attested.executor.lockdown.ProvisioningModeActivity"
android:permission="android.permission.BIND_DEVICE_ADMIN" android:exported="true"
android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.app.action.GET_PROVISIONING_MODE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <meta-data android:name="io.sentry.auto-init" android:value="false" />
        <provider android:name="androidx.startup.InitializationProvider"
android:exported="false" android:authorities="com.acurast.attested.executor.canary.androidx-
startup">
            <meta-data android:name="androidx.emoji2.text.EmojiCompatInitializer"
android:value="androidx.startup" />
            <meta-data android:name="androidx.lifecycle.ProcessLifecycleInitializer"
android:value="androidx.startup" />
            <meta-data android:name="androidx.profileinstaller.ProfileInstallerInitializer"
android:value="androidx.startup" />
        </provider>
        <meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version" />
        <receiver android:name="androidx.profileinstaller.ProfileInstallReceiver"
android:permission="android.permission.DUMP" android:enabled="true" android:exported="true"
android:directBootAware="false">
            <intent-filter>
                <action android:name="androidx.profileinstaller.action.INSTALL_PROFILE" />
            </intent-filter>
            <intent-filter>
                <action android:name="androidx.profileinstaller.action.SKIP_FILE" />
            </intent-filter>
            <intent-filter>
                <action android:name="androidx.profileinstaller.action.SAVE_PROFILE" />
            </intent-filter>
            <intent-filter>
                <action android:name="androidx.profileinstaller.action.BENCHMARK_OPERATION"
/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

Details #7

App's screenshot:



4.3 App Development & Release

4.3.1 Distribution

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Can the app be installed on an unsupported version of Android?	No, the minimum API level prevents installation on unsupported versions.	The minimum required API level is 30. This corresponds to Android 11 that is no longer supported. Taking into account how relevant the integrity of the device is for trustworthiness of its computations, supporting devices with old, unpatched OSes poses a risk.	**FAIL**
2.	What signature versions does the app use?	The present signatures have versions consistent with the targeted and minimum API levels.	As expected, v3 and earlier are used.	**PASS**
3.	What is the key length of the app's signing certificate?	Minimum 2048-bit RSA key or equivalent.	2048-bit RSA key.	**PASS**
4.	Is the `debuggable` attribute set to `true` in the manifest?	No.	As expected.	**PASS**

4.3.2 Third-Party Components & Libraries

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Are any third-party libraries used?	-	Yes.	**N/A**
2.	Are any of the used libraries outdated?	No.	No, the important j2v8 library is used in the latest version although it was not updated since 2021.	**PASS**
3.	Are there any known vulnerabilities in the used libraries?	No.	As expected.	**PASS**
4.	Do binary libraries have NX bit set?	Yes.	As expected.	**PASS**
5.	Do binary libraries use stack canaries?	Yes.	No, libsweet_b.so library does not use stack canaries.	**FAIL**
6.	Do binary libraries have full RELRO enabled?	Yes.	No, libj2v8.so library for arm64 does not have full RELRO.	**FAIL**

Details #2

The used version of j2v8 was released in 2021 (https://mvnrepository.com/artifact/com.eclipsesource.j2v8/j2v8), anyway the latest version is used. From build.gradle:

```
com.eclipsesource.j2v8:j2v8:6.2.1@aar
```

Details #4-6

Using `checksec` (https://github.com/slimm609/checksec.sh), the following properties of app's binaries were retrieved:

```
filename                                                nx    canary  relro
processor-1.4.0-canary/lib/arm64-v8a/libj2v8.so         yes   yes     no
processor-1.4.0-canary/lib/arm64-v8a/libsweet_b.so      yes   no      full
processor-1.4.0-canary/lib/armeabi-v7a/libj2v8.so       yes   yes     full
processor-1.4.0-canary/lib/armeabi-v7a/libsweet_b.so    yes   no      full
```

4.3.3 Embedded Information

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Are any hardcoded passwords present in the app?	No.	As expected.	**PASS**
2.	Are any API keys present in the app?	Only the keys that need to be stored in the app are there.	As expected.	**PASS**
3.	Are any cryptographic keys stored in the app?	No if they are used for important cryptographic operations.	As expected, only test private keys in `com.google.api.client.testing.json.webtoken.TestCertificates` were found.	**PASS**
4.	Are there any hints present in the decompiled code that indicate presence of unfinished functionalities or debug features?	No, the code released in the app is free from immature code.	Nothing found.	**PASS**

4.4 Device Security

4.4.1 Root Detection

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Is a root detection mechanism implemented?	Yes, for apps handling sensitive information	No explicit mechanism is present.	**INFO**
2.	Does the app still work normally on a rooted device?	No, or at least critical functionalities are disabled.	If the device was rooted by unlocking its bootloader and modifying the boot partition, this would be reflected in the key attestation.	**PASS**

4.4.2 Play Integrity

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Does the app use the Play Integrity API to check device integrity?	Yes.	No.	**FAIL**
2.	If standard API requests are used, does `requestHash` include a digest of all relevant values from the app's request?	Yes, all values that need to be protected are included.	Not applicable.	**N/A**
3.	If classic API requests are used, does `nonce` contain a unique value generated on the server side?	Yes, a unique value is used to protect against replay attacks.	Not applicable.	**N/A**
4.	If classic API requests are used, does `nonce` include a digest of all relevant values from the app's request?	Yes, to protect the contents of the app's request against tampering.	Not applicable.	**N/A**
5.	Is it possible to reuse the integrity token obtained from a classic API request?	No, the backend server validates the uniqueness and freshness of the nonce.	Not applicable.	**N/A**
6.	Do the `nonce` or `requestHash` values used contain any cleartext sensitive data?	No, the value of `requestHash` in a standard API request and the value of `nonce` in a classic API request are collected by Google Play.	Not applicable.	**N/A**
7.	Is the integrity token obtained from a classic API request only verified on the device?	No, the response token is decrypted and validated on the server side.	Not applicable.	**N/A**
8.	Is the `MEETS_DEVICE_INTEGRITY` integrity label required to pass the integrity check?	Yes.	Not applicable.	**N/A**
9.	Is the `MEETS_STRONG_INTEGRITY` integrity label required to pass the integrity check?	Yes, for apps that require a high level of security.	Not applicable.	**N/A**
10.	What happens if the Play Integrity API call fails?	If errors persist after a few retries, all integrity checks should be considered failed.	Not applicable.	**N/A**

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
11.	Can Magisk (https://github.com/topjohnwu/Magisk) with Play Integrity Fix module (https://github.com/chiteroman/PlayIntegrityFix) be used to bypass the integrity checks?	No, either the `MEETS_STRONG_INTEGRITY` integrity label should be required, or other mechanisms should be implemented to prevent this easy bypass.	Not applicable.	**N/A**

4.4.3 Device Lockdown Policy

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Is it possible to use the app on a not locked down device?	No.	No unless the attacker has root privileges on the device.	**PASS**
2.	Is it possible to install other apps on the locked device?	No.	No. Although the tested app can install apps, only apps with a whitelisted hash can be installed. Other methods of app installation are blocked.	**PASS**
3.	Is it possible to use Bluetooth on the locked device?	No unless needed.	No.	**PASS**
4.	Is it possible to modify wireless networking settings on the locked device?	No unless needed.	Yes, but connecting to a wireless network to get Internet access is required.	**PASS**
5.	Is it possible to attach an external storage drive to the locked device?	No unless needed.	Yes, `no_usb_file_transfer` and `no_physical_media` are not set, however using an external storage for updating the app is necessary.	**PASS**
6.	Is it possible to activate ADB on the locked device?	No.	As expected.	**PASS**

Details #2-5

The following restrictions are applied:

```
INSTALL_RESTRICTIONS = CollectionsKt.listOf((Object[]) new String[]{"no_install_apps",
"no_install_unknown_sources", "no_install_unknown_sources_globally"});
RESTRICTIONS = CollectionsKt.arrayListOf("no_add_user", "no_adjust_volume",
"no_ambient_display", "no_control_apps", "no_autofill", "no_bluetooth",
"no_bluetooth_sharing", "no_config_bluetooth", "no_config_brightness",
"no_config_cell_broadcasts", "no_config_credentials", "no_config_date_time",
"no_config_locale", "no_config_location", "no_config_mobile_networks",
"disallow_config_private_dns", "no_config_screen_timeout", "no_config_tethering",
"no_config_vpn", "no_content_capture", "no_create_windows", "no_cross_profile_copy_paste",
"no_data_roaming", "no_debugging_features", "no_fun", "no_install_apps",
"no_install_unknown_sources", "no_install_unknown_sources_globally", "no_set_wallpaper",
"no_printing", "no_modify_accounts", "no_system_error_dialogs", "no_sms",
"no_network_reset", "no_outgoing_beam", "no_outgoing_calls", "no_remove_user",
"no_unified_password", "no_uninstall_apps", "no_unmute_microphone", "no_user_switch");
```

Details #6

com.acurast.attested.executor.lockdown.Lockdown.java:

```
devicePolicyManager.setGlobalSetting(componentName, "adb_enabled", "0");
```

4.5 Network Communication

4.5.1 Communication Partners

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	What servers does the app communicate with?	-	acurast-canarynet-node.prod.gke.acurast.com and other servers as specified in the job.	**N/A**
2.	Is data received from or transmitted to third-party servers?	It needs to be ensured that no sensitive information is sent to third-party servers and all data obtained from third-party servers is handled securely.	Only requests specified in the job will be sent to third-party servers. No direct issues of handling those requests were found.	**PASS**
3.	Does the used network security configuration allow cleartext traffic?	No.	No network security configuration was defined.	**N/A**
4.	Does the used network security configuration modify the set of trust anchors?	The app should not weaken the default settings.	No network security configuration was defined.	**N/A**

4.5.2 TLS Settings

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Is the communication between the app and all servers secured with TLS?	Yes.	No, the tzBTC job in the version available during the security audit intended to access an IP address via HTTP.	**INFO**
2.	Does the app verify the server certificate?	Yes, it is not possible to perform a Man-in-the-Middle attack using a self-signed certificate.	As expected.	**PASS**
3.	Does the app verify whether the hostname from the certificate matches the server it communicates with?	Yes, a valid certificate for one hostname cannot be used to intercept traffic to another hostname.	As expected.	**PASS**
4.	Does the app perform certificate pinning?	Yes, the app does not trust the CA list of Android but performs additional certificate checks, e.g.: ▪ Cert issuer ▪ Cert hash	No, but as the app is meant to execute arbitrary jobs, it cannot restrict what certificates should be considered trusted. The extended certificate validation could nevertheless happen in a job or the job's results could contain the certificate chain obtained from the server.	**PASS**

Details #1

The tzBTC job (https://gitlab.papers.tech/papers/customer-tezos-foundation/tzbtc-v2/tzbtc-acurast-jobs/-/blob/61b5917a7831f8dc7e456a0d532a422051df0864/src/tzbtc_jobs.ts) that was not in the final version yet contained http requests to an IP address:

```
import ECPairFactory from "ecpair";
import * as ecc from "tiny-secp256k1";
import { testnet, regtest } from "bitcoinjs-lib/src/networks";
```

```
import https from "https";
import http from "http";
import { sha256 } from "bitcoinjs-lib/src/crypto";

console.log("starting")
[CUT BY COMPASS]
const bitcoinRPCPostCall = function (method: string, params: any[]): Promise<any> {
  return new Promise((resolve, reject) => {
    const payload = JSON.stringify({
      "jsonrpc": "1.0",
      "id": "curltest",
      "method": method,
      "params": params
    })
    const options = {
      method: 'POST',
      hostname: "172.17.0.1",
      port: 18443,
      headers: {
        "User-Agent": "Mozilla/5.0",
        "Content-Type": 'application/json',
        "Content-Length": payload.length,
        //"x-api-key": "[CUT BY COMPASS]",
        'Authorization': 'Basic ' + Buffer.from('a:b').toString('base64')
      },
    }
    const req = http
      .request(
        options,
        (res) => {
          const data: Buffer[] = [];

          res.on("data", (chunk: Buffer) => {
            data.push(chunk);
          })

          res.on("end", () => {
            resolve(JSON.parse(Buffer.concat(data).toString()));
          })
        }
      )
    req.write(payload)
    req.on("error", (err) => {
      reject("Error: " + err.message);
    })
  })
}
```

4.6 Android KeyStore

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Does the app make sure that the keys are bound to the hardware?	Yes, `isInsideSecureHardware()` is called, or the properties of the keys are verified with `getSecurityLevel()`.	Keys are required to be in the StrongBox, if StrongBox is available.	**PASS**
2.	Have all the keys been generated in the Android KeyStore?	Keys generated in a hardware-backed Android KeyStore were never exposed outside the secure hardware. Imported keys were available to the app process.	Yes, all keys were generated in the keystore.	**PASS**

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
3.	Are there any other properties of the keys that are incorrectly set?	No, the purposes of the keys, allowed usage modes, paddings, etc. were set correctly.	As expected.	PASS
4.	Is user authentication required to use the keys?	Depending on the use case, some keys may need to be used without prior user authentication.	Not needed for the tested app.	PASS

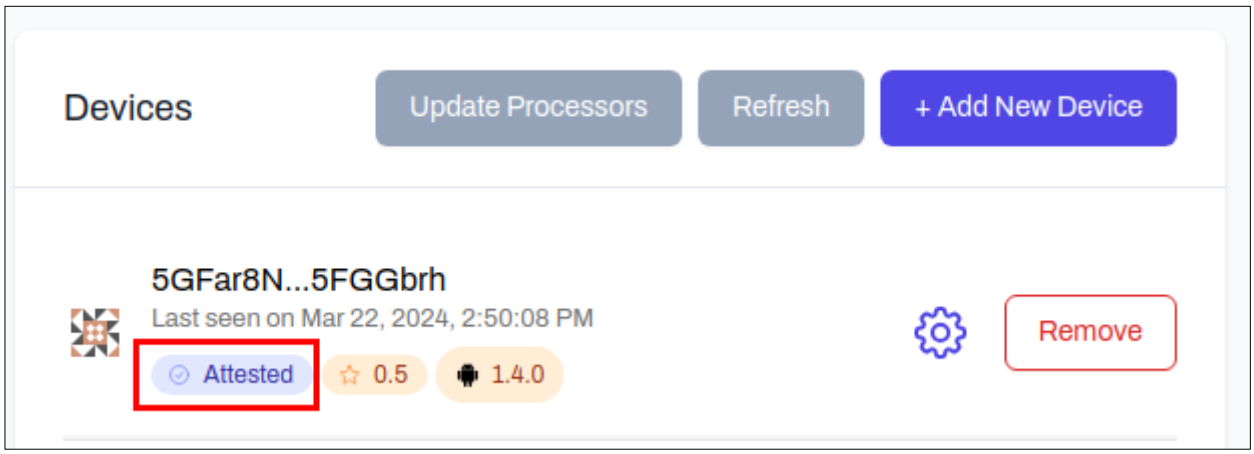
4.7 Device Attestation

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	Is it possible to obtain attestation on a compromised device?	No.	It was possible to obtain attestation on a device with unlocked bootloader and modified boot partition. Then modify the code during the runtime of the app using Frida hooking framework. Nevertheless, the key attestation submitted to the acurast parachain showed that the device should not be considered attested as the root of trust was invalid.	FAIL
2.	How cat users of the tzBTC be sure that only correctly signed job answers are accepted?	-	On-chain signatures for Bitcoin and Tezos transactions are created by the processors, so the validity of signatures is verified on-chain. Note that this part of the tzBTC ecosystem was not explicitly tested.	PASS
3.	How cat users of the tzBTC be sure that the keys used to sign job answers are trustworthy?	-	The confirmation of the job sent from the device includes public keys that can be used to verify the signatures of job answers. That confirmation is signed with the processor's key.	PASS
4.	How can users of the tzBTC be sure that the job being executed on the processor is the expected script?	-	In the confirmation of the job sent from the device, there is the job creator and job's id. It is possible to see on the acurast parachain what IPFA link the job with the particular id has.	PASS

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
5.	How can users of the tzBTC be sure that the processor's key is trustworthy?	-	During the processor registration, key attestation of the processor's key is published on the acurast parachain. Currently it is possible to view the parsed attestation: https://polkadot.js.org/apps/?rpc=wss%3A%2F%2Facurast-canarynet-ws.prod.gke.papers.tech#/chainstate. However, to get the original message with the actual attestation statement searching through the blockchain would be needed.	FAIL
6.	How can users of the tzBTC be sure that the usage of the processor's key is trustworthy?	-	In the key attestation statement, there are values present that represent the app's package name and the digest of the certificate used to sign the app. However, no information about what the code of the app is is present. Although to update the app on the processor, the apk's hash needs to be present in a whitelist – this is checked by the already installed processor app, no verification of the initial apk can be performed by users of the tzBTC.	FAIL

Details #1

For 5GFar8NXmx6CVGiN64mYtiuwNphYXn8JHKNmReosv5FGGbrh, attestation could be obtained and in the acurast console the device was shown as attested:



However, in the parsed attestation statement stored on the acurast parachain, it is shown that the root of trust for this particular device is invalid:

```
  [
    [
      5GFar8NXmx6CVGiN64mYtiuwNphYXn8JHKNmReosv5FGGbrh
    ]
    {
      certIds: [
```

```
        [
          0x301b311930170603550405131066393230303965383533623662303435
          0x00e8fa196314d2fa18
        ]
        [
          0x301b311930170603550405131066393230303965383533623662303435
          0x060d896bdc60a576a5947be0895f5989
        ]
        [

0x303f31123010060355040c0c095374726f6e67426f7831293027060355040513206633646631393762313431633933343763376461663033373565633066393439
          0x569a2401ba9238309bdac006c2ac251d
        ]
        [

0x303f31123010060355040c0c095374726f6e67426f7831293027060355040513203036383432663834626362616462643139363430356266643661363334396562
          0x01
        ]
      ]
      keyDescription: {
        attestationSecurityLevel: StrongBox
        keyMintSecurityLevel: StrongBox
        softwareEnforced: {
          purpose: null
          algorithm: null
          keySize: null
          digest: null
          padding: null
          ecCurve: null
          rsaPublicExponent: null
          mgfDigest: null
          rollbackResistance: false
          earlyBootOnly: false
          activeDateTime: null
          originationExpireDateTime: null
          usageExpireDateTime: null
          usageCountLimit: null
          noAuthRequired: false
          userAuthType: null
          authTimeout: null
          allowWhileOnBody: false
          trustedUserPresenceRequired: false
          trustedConfirmationRequired: false
          unlockedDeviceRequired: false
          allApplications: null
          applicationId: null
          creationDateTime: 1,710,956,123,346
          origin: null
          rootOfTrust: null
          osVersion: null
          osPatchLevel: null
          attestationApplicationId: {
            packageInfos: [
              {
                packageName: com.acurast.attested.executor.canary
                version: 26
              }
            ]
            signatureDigests: [
              0xec70c2a4e072a0f586552a68357b23697c9d45f1e1257a8c4d29a25ac4982433
            ]
          }
          attestationIdBrand: null
          attestationIdDevice: null
          attestationIdProduct: null
          attestationIdSerial: null
          attestationIdImei: null
          attestationIdMeid: null
          attestationIdManufacturer: null
          attestationIdModel: null
```

```
          vendorPatchLevel: null
          bootPatchLevel: null
          deviceUniqueAttestation: false
        }
        teeEnforced: {
          purpose: 0x020306
          algorithm: 3
          keySize: 256
          digest: 0x00
          padding: null
          ecCurve: 1
          rsaPublicExponent: null
          mgfDigest: null
          rollbackResistance: false
          earlyBootOnly: false
          activeDateTime: null
          originationExpireDateTime: null
          usageExpireDateTime: null
          usageCountLimit: null
          noAuthRequired: true
          userAuthType: null
          authTimeout: null
          allowWhileOnBody: false
          trustedUserPresenceRequired: false
          trustedConfirmationRequired: false
          unlockedDeviceRequired: false
          allApplications: null
          applicationId: null
          creationDateTime: null
          origin: 0
          rootOfTrust: {
            verifiedBootKey:
0x0000000000000000000000000000000000000000000000000000000000000000
            deviceLocked: false
            verifiedBootState: Unverified
            verifiedBootHash:
0x3ed2237b2aeed1b0a856f7227746f29ee6b9a19f6b34d553dd9c93e312a0bbc1
          }
          osVersion: 130,000
          osPatchLevel: 202,309
          attestationApplicationId: null
          attestationIdBrand: null
          attestationIdDevice: null
          attestationIdProduct: null
          attestationIdSerial: null
          attestationIdImei: null
          attestationIdMeid: null
          attestationIdManufacturer: null
          attestationIdModel: null
          vendorPatchLevel: 20,230,901
          bootPatchLevel: 20,230,901
          deviceUniqueAttestation: false
        }
      }
      validity: {
        notBefore: 0
        notAfter: 2,461,449,600,000
      }
    }
  ]
```

Details #5

Signature digest from the attestation:

```
          attestationApplicationId: {
            packageInfos: [
              {
                packageName: com.acurast.attested.executor.canary
                version: 26
              }
            ]
```

```
            signatureDigests: [
              0xec70c2a4e072a0f586552a68357b23697c9d45f1e1257a8c4d29a25ac4982433
            ]
          }
```

This is the same value as the signature of the certificate used to sign the app:

```
$ apksigner verify -v --print-certs processor-1.4.0-canary.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Verifies
Verified using v1 scheme (JAR signing): false
Verified using v2 scheme (APK Signature Scheme v2): false
Verified using v3 scheme (APK Signature Scheme v3): true
Verified using v4 scheme (APK Signature Scheme v4): false
Verified for SourceStamp: false
Number of signers: 1
Signer #1 certificate DN: O=Acurast, L=Zug, ST=Zug, C=CH
Signer #1 certificate SHA-256 digest:
ec70c2a4e072a0f586552a68357b23697c9d45f1e1257a8c4d29a25ac4982433
Signer #1 certificate SHA-1 digest: 2cbe58b09ce8f794c34bf63c5bd2076cad131c79
Signer #1 certificate MD5 digest: 4f243e72eaf337d63a943f5d161f5206
Signer #1 key algorithm: RSA
Signer #1 key size (bits): 2048
Signer #1 public key SHA-256 digest:
2fc2adb3f006fd242f8df7fda8eba2503c08214632a2752f26d6c0dca68d39ec
Signer #1 public key SHA-1 digest: c7a636a94285a9b2591736ceaeba105b2b0d176b
Signer #1 public key MD5 digest: 93ec720bccb49967a7a85b716cea99d0
```

4.8 Job Execution

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
1.	What jobs are supposed to be run on the tzBTC processor apps?	Only the required jobs.	A single tzBTC job will be run.	**PASS**
2.	How can the integrity of the expected job be assured?	Strong, externally verifiable integrity assurance should be performed.	Job code is hosted on IPFS. During the generation of the job confirmation by the processor, a public key is registered, signatures depend on the hash of the script.	**PASS**
3.	Who is allowed to create jobs for processors used in tzBTC?	Processors should not accept arbitrary jobs.	Only jobs created by a specified account should be executed.	**PASS**
4.	Are there any security issues with the tzBTC job?	No.	During the timeframe of the security audit, the code of the job was still in the development phase (https://gitlab.papers.tech/papers/customer-tezos-foundation/tzbtc-v2/tzbtc-acurast-jobs/-/blob/61b5917a7831f8dc7e456a0d532a422051df0864/src/tzbtc_jobs.ts). Security posture of the job was analyzed based on similar jobs and interviews with developers. No general issues were found, however, it will need to be checked whether the final version works as expected.	**INFO**

No.	Description of Test	Expected Result	Actual Result	PASS FAIL
5.	Can anyone run jobs of for the tzBTC?	Depends on the use case. If yes, the discrepancies of the job answers need to be securely handled.	No, only 10 processors shared among 5 entities are intended to run the tzBTC jobs.	PASS

5 Appendix

5.1 Compass Weaknesses Rating

Compass rates weaknesses based on their intrinsic technical properties. It should be noted that it is *not* a risk rating. Neither a threat actor's motivation (e.g., financial gain, fame, etc.) nor financial loss incurred by a successful exploitation of a weakness are taken into consideration.

All weaknesses are usually rated in isolation without considering the environment or any additional security controls that might be in place (i.e., vulnerabilities are rated in the context in which they are discovered).

A general description of each rating is given in the table below:

Rating	Description
Critical	▪ Exploitation requires little effort, no user interaction, no elevated privileges, or combination with other issues ▪ Successful exploitation has an immediate critical impact on the application/system: disclosure, manipulation or loss of sensitive data, elevation of privileges, disruption of service availability, etc.
High	▪ Exploitation typically requires additional resources: user permissions, user interaction, large amounts of time/computing power, etc. ▪ Central security features & controls are turned off/not used ▪ Security-relevant processes or concepts are neither defined, nor implemented
Medium	▪ Exploitation typically requires significant effort and/or combination with other issues ▪ Security features & controls are implemented, but not configured according to best practices ▪ Security-relevant processes or concepts are defined, but important aspects are missing, unclear, or do not follow best practices
Low	▪ Exploitation typically leads to disclosure of information that is not sensitive but might be used in broader attack efforts (e.g., version information, user account names, etc.) ▪ Security features & controls are implemented with minor deviations from best practices ▪ Security-relevant processes or concepts are defined, but minor aspects are missing, unclear, or do not follow best practices
Info	▪ The entry is purely informational and has no security impact

The customer should review the individual weaknesses and their ratings and assign a risk score based on the company's risk management processes. Based on these risk scores, the customer should decide how and when the risk is handled (e.g., mitigate, accept, transfer, avoid).

5.2 Recheck Coloring

The following color code is used in rechecks to show the current state of a weakness:

Fixed	Partially Fixed	Not Fixed	New	Not Rechecked