



**Circulo Test Targets:**

- Mobile apps*
- Threat Model*
- Supply Chain*
- Backend Services*
- Server Configuration*
- AWS Infrastructure*

**Pentest Report**

---

**Client:**  
*Circulo Team @ Guardian Project*

- 7ASecurity Test Team:**
- Abraham Aranguren, MSc.
  - Daniel Ortiz, MSc.
  - Dariusz Jastrzębski
  - Jesus Arturo Espinoza Soto
  - Miroslav Štampar, PhD.
  - Szymon Grzybowski, MSc.

**7ASecurity**  
*Protect Your Site & Apps  
From Attackers*  
sales@7asecurity.com  
[7asecurity.com](http://7asecurity.com)

## INDEX

<b>Introduction</b>	<b>4</b>
<b>Scope</b>	<b>5</b>
<b>Identified Vulnerabilities</b>	<b>7</b>
CIR-01-003 WP1: Possible Phishing via StrandHogg 2.0 on Android (Medium)	7
CIR-01-005 WP1: Leaks via Missing Security Screen on Android & iOS (Low)	9
CIR-01-006 WP1: Multiple DoS via Exported Activities (Medium)	11
CIR-01-007 WP1: Circulo Chat History Access via Memory Leaks (Medium)	14
CIR-01-008 WP1: Circulo Room ID Access via Log Leaks (Medium)	15
CIR-01-009 WP1: User DoS via DNS Spoofing on iOS & Android (High)	17
CIR-01-013 WP1: Auth Token Access via inadequate Keychain Usage (Medium)	18
CIR-01-021 WP4: Synapse Admin API Exposed to the Internet (Medium)	19
CIR-01-022 WP4/6: Data Leaks in Nginx and CloudWatch Logs (Medium)	21
<b>Hardening Recommendations</b>	<b>24</b>
CIR-01-001 WP1: Missing Jailbreak/Root Detection on Android & iOS (Info)	24
CIR-01-002 WP1: Support of Insecure v1 Signature on Android (Info)	25
CIR-01-004 WP1: Android Config Hardening Recommendations (Info)	25
CIR-01-010 WP1: Usage of Insecure Crypto Functions (Low)	28
CIR-01-011 WP1: iOS Binary Hardening Recommendations (Info)	29
CIR-01-012 WP1: Weak PIN Policy on iOS & Android (Low)	30
CIR-01-014 WP1: Excessive Fingerprint Permissions on Android (Info)	31
CIR-01-015 WP6: Weaknesses in Vuln Management Processes (Medium)	32
CIR-01-016 WP6: Insecure Default Settings (Low)	34
CIR-01-017 WP6: Lack of KMS Encryption & Hardening for S3 Buckets (Medium)	37
CIR-01-018 WP6: Insufficient Network Restrictions (Low)	39
CIR-01-019 WP6: ELB Hardening Recommendations (Low)	41
CIR-01-020 WP6: Insufficient AWS Logging & Monitoring (Medium)	42
CIR-01-023 WP5: Missing SSH MFA & Auth Hardening (Low)	46
CIR-01-024 WP5: Vulnerable Nginx in Use (Low)	47
CIR-01-025 WP5: Insecure Configuration of Third-party Software in Use (Low)	48
CIR-01-026 WP5: Boot Loader Password Not Set (Low)	50
CIR-01-027 WP5: Insufficient Logging and Monitoring (Medium)	51
CIR-01-028 WP5: Lack of Full Disk Encryption (Medium)	52
CIR-01-029 WP5: Insecure Network Stack Configuration (Low)	52
<b>WP2: Circulo Lightweight Threat Model Review</b>	<b>55</b>
Introduction	55
Relevant assets and threat actors	55
Attack surface	56



Threat 01: Identity or Data Spoofing	56
Threat 02: Data Tampering or Unauthorized Modifications	58
Threat 03: Repudiation Attacks	59
Threat 04: Information disclosure	60
Threat 05: Temporary or Permanent Denial of Service	61
Threat 06: Privilege Escalation	63
<b>WP3: Circulo Supply Chain Implementation</b>	<b>65</b>
Introduction and General Analysis	65
SLSA v1.0 Analysis and Recommendations	66
SLSA v1.0 Producer	66
SLSA v1.0 Build platform	68
SLSA v0.1 Analysis and Recommendations	69
SLSA v0.1 Conclusion	71
<b>Conclusion</b>	<b>72</b>



## Introduction

*“Safety starts with community*

*Circulo helps connect you to a reliable network of six peers. Establish protocols, send alerts and keep those in your circle informed.”*

From <https://encirculo.org/en/>

This document outlines the results of a penetration test and *whitebox* security review conducted against the Circulo platform. The project was solicited by Circulo, funded by the Open Technology Fund (OTF), and executed by 7ASecurity in September and October 2024. The audit team dedicated 35 working days to complete this assignment. Please note that this is the first penetration test for this project. Consequently, the identification of security weaknesses was expected to be easier during this engagement, as more vulnerabilities are identified and resolved after each testing cycle.

During this iteration the goal was to review the solution as thoroughly as possible, to ensure Circulo users can be provided with the best possible security. The methodology implemented was *whitebox*: 7ASecurity was provided with access to a staging environment, documentation, test users, and source code. A team of 6 senior auditors carried out all tasks required for this engagement, including preparation, delivery, documentation of findings and communication.

A number of necessary arrangements were in place by September 2024, to facilitate a straightforward commencement for 7ASecurity. In order to enable effective collaboration, information to coordinate the test was relayed through email, as well as a shared Signal chat group. The Circulo team was helpful and responsive throughout the audit, which ensured that 7ASecurity was provided with the necessary access and information at all times, thus avoiding unnecessary delays. 7ASecurity provided regular updates regarding the audit status and its interim findings during the engagement.

This audit split the scope items into the following work packages, which are referenced in the ticket headlines as applicable:

- WP1: Mobile Security tests against Circulo Android & iOS apps
- WP2: Circulo Lightweight Threat Model documentation
- WP3: Whitebox Tests against Circulo Supply Chain Implementation
- WP4: Whitebox Tests against Circulo Implementation on Backend Services
- WP5: Whitebox Tests against Servers, Infrastructure & Config via SSH
- WP6: White-box Tests against Circulo AWS Infrastructure

The findings of the security audit (WP1 & WP4-6) can be summarized as follows:

<i>Identified Vulnerabilities</i>	<i>Hardening Recommendations</i>	<i>Total Issues</i>
9	20	29

Please note that the analysis of the remaining work packages (WP2, WP3) is provided separately, in the following section of this report:

- [WP2: Circulo Lightweight Threat Model Review](#)
- [WP3: Circulo Supply Chain Implementation](#)

Moving forward, the scope section elaborates on the items under review, while the findings section documents the identified vulnerabilities followed by hardening recommendations with lower exploitation potential. Each finding includes a technical description, a proof-of-concept (PoC) and/or steps to reproduce if required, plus mitigation or fix advice for follow-up actions by the development team.

Finally, the report culminates with a conclusion providing detailed commentary, analysis, and guidance relating to the context, preparation, and general impressions gained throughout this test, as well as a summary of the perceived security posture of the Circulo applications.

## Scope

The following list outlines the items in scope for this project:

- **WP1: Mobile Security tests against Circulo Implementation on Android & iOS apps**
  - Android:
    - <https://gitlab.com/circuloapp/circulo-keanu-android/-/tags/2.3-RC-5>
    - <https://guardianproject.info/releases/Circulo-2.3-RC-5...apk>
    - <https://play.google.com/.../details?id=org.article19.circulo.next>
  - iOS:
    - <https://gitlab.com/circuloapp/circulo-for-ios/-/commits/1.2.0>
    - <https://apps.apple.com/us/app/c%C3%ADrculo/id1571749642>
- **WP2: Circulo Lightweight Threat Model documentation**
  - Circulo Mobile Applications as above.
  - AWS infrastructure hosting Matrix server documentation:
    - <https://docs.keanu.im/backend/>
  - Infrastructure as Code based on:
    - <https://gitlab.com/guardianproject-ops/...>

- Project Documentation:
  - [https://encirculo.org/en/assets/docs/A19CirculoReport2021\\_en.pdf](https://encirculo.org/en/assets/docs/A19CirculoReport2021_en.pdf)
  - <https://encirculo.org/en/community-resources/>
- Deployments
  - <https://casa.encirculo.org>
  - <https://matrix.unready.im>
- **WP3: Whitebox Tests against Circulo Supply Chain Implementation**
  - As above
- **WP4: Whitebox Tests against Circulo Implementation on Backend Services**
  - <https://casa.encirculo.org>
- **WP5: Whitebox Tests against Circulo Servers, Infrastructure & Configuration via SSH**
  - SSH access via AWS SSM to *gp-unreadyim-prod* server was provided to 7ASecurity
- **WP6: White-box Tests against Circulo AWS Infrastructure**
  - AWS Account:
    - 384647892761

## Identified Vulnerabilities

This area of the report enumerates findings that were deemed to exhibit greater risk potential. Please note these are offered sequentially as they were uncovered, they are not sorted by significance or impact. Each finding has a unique ID (i.e. *CIR-01-001*) for ease of reference, and offers an estimated severity in brackets alongside the title.

### CIR-01-003 WP1: Possible Phishing via StrandHogg 2.0 on Android (*Medium*)

**Retest Notes:** The Circulo team fixed this issue<sup>1</sup> and 7ASecurity confirmed that the fix is valid.

Testing confirmed that the Circulo Android app is currently vulnerable to a number of Task Hijacking attacks. The *launchMode* for the app-launcher activity is currently not set and hence defaults to *standard*<sup>2</sup>, which mitigates Task Hijacking via *StrandHogg*<sup>3</sup> and other older techniques documented since 2015<sup>4</sup>, while leaving the app vulnerable to *StrandHogg 2.0*<sup>5</sup>. This vulnerability affects Android versions 3-9.x<sup>6</sup> but was only patched by Google on Android 8-9<sup>7</sup>. Since the app supports devices from Android 5 (API level 21), this leaves all users running Android 5-7.x vulnerable, as well as users running unpatched Android 8-9.x devices (common).

A malicious app could leverage this weakness to manipulate the way in which users interact with the app. More specifically, this would be instigated by relocating a malicious attacker-controlled activity in the screen flow of the user, which may be useful to perform Phishing, Denial-of-Service or capturing user-credentials. This issue has been exploited by banking malware trojans in the past<sup>8</sup>.

In *StrandHogg* and regular Task Hijacking, malicious applications typically use one or more of the following techniques:

- **Task Affinity Manipulation:** The malicious application has two activities M1 and M2 wherein *M2.taskAffinity = com.victim.app* and *M2.allowTaskReparenting = true*. If the malicious app is opened on M2, once the victim application has initiated, M2 is relocated to the front and the user will interact with the malicious application.

---

<sup>1</sup> <https://gitlab.com/circuloapp/circulo-keanu-android/-/issues/159>

<sup>2</sup> <https://developer.android.com/guide/topics/manifest/activity-element#mode>

<sup>3</sup> <https://www.helpnetsecurity.com/2019/12/03/strandhogg-vulnerability/>

<sup>4</sup> <https://s2.ist.psu.edu/paper/usenix15-final-ren.pdf>

<sup>5</sup> <https://www.helpnetsecurity.com/2020/05/28/cve-2020-0096/>

<sup>6</sup> <https://www.xda-developers.com/strandhogg-2-0-android-vulnerability-explained-developer-mitigation/>

<sup>7</sup> <https://source.android.com/security/bulletin/2020-05-01>

<sup>8</sup> <https://arstechnica.com/.../...fully-patched-android-phones-under-active-attack-by-bank-thieves/>

- **Single Task Mode:** If the victim application has set *launchMode* to *singleTask*, malicious applications can use *M2.taskAffinity = com.victim.app* to hijack the victim application task stack.
- **Task Reparenting:** If the victim application has set *taskReparenting* to *true*, malicious applications can move the victim application task to the malicious application stack.

However, in the case of StrandHogg 2.0, all exported activities **without** a *launchMode* of *singleTask* or *singleInstance* are affected on vulnerable Android versions<sup>9</sup>.

This issue can be confirmed by reviewing the *AndroidManifest* of the Android application.

### Affected File:

*AndroidManifest.xml*

### Affected Code:

```
<activity
android:theme="@style/InviteDialog"
android:name="info.guardianproject.keanu.core.RouterActivity"
android:enabled="true"
android:exported="true">
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
```

As can be seen above, the *launchMode* is not set and hence defaults to *standard*.

To ease the understanding of this problem, an example malicious app was created to demonstrate the exploitability of this weakness.

### PoC Demo:

[https://7as.es/CIR-01\\_Circulo\\_m6HBfjCK4tdP9/PoC/StrandHogg2.0\\_PoC.mp4](https://7as.es/CIR-01_Circulo_m6HBfjCK4tdP9/PoC/StrandHogg2.0_PoC.mp4)

It is recommended to implement as many of the following countermeasures as deemed feasible by the development team:

- The task affinity should be set to an empty string. This is best implemented in the Android manifest **at the application level**, which will protect all activities and ensure the fix works even if the launcher activity changes. The application should use a randomly generated task affinity instead of the package name to prevent Task Hijacking, as malicious apps will not have a predictable task affinity to

<sup>9</sup> <https://www.xda-developers.com/strandhogg-2-0.../>



target.

- The *launchMode* should then be changed to *singleInstance* (instead of *singleTask*). This will ensure continuous mitigation in *StrandHogg 2.0*<sup>10</sup> while improving security strength against older Task Hijacking techniques<sup>11</sup>.
- A custom *onBackPressed()* function could be implemented to override the default behavior.
- The *FLAG\_ACTIVITY\_NEW\_TASK* should not be set in *activity launch* intents. If deemed required, one should use the aforementioned in combination with the *FLAG\_ACTIVITY\_CLEAR\_TASK* flag<sup>12</sup>.

#### Affected File:

*AndroidManifest.xml*

#### Proposed Fix:

```
<application android:theme="@style/AppTheme" android:label="@string/app_name"
android:icon="@drawable/ic_launcher_foreground"
android:name="org.article19.circulo.next" [...] android:taskAffinity="" >
[...]
<activity android:label="@string/app_name" android:name="org.article19.circulo.next"
android:launchMode="singleInstance"
android:configChanges="keyboard|keyboardHidden|orientation|screenSize|uiMode"
android:windowSoftInputMode="adjustPan" >
[...]
```

### CIR-01-005 WP1: Leaks via Missing Security Screen on Android & iOS (Low)

**Retest Notes:** The Circulo team fixed this issue<sup>131415</sup> and 7ASecurity confirmed that the fix is valid.

It was found that the Circulo Android and iOS apps fail to render a security screen when they are backgrounded. This allows attackers with physical access to an unlocked device to see data displayed by the apps before they disappeared into the background. A malicious app or an attacker with physical access to the device could leverage these weaknesses to gain access to user-information, such as sensitive or compromising data related to PII.

To replicate this issue in Android or iOS, simply navigate to some sensitive screen and then send the application to the background. After that, show the open apps and

<sup>10</sup> <https://www.xda-developers.com/strandhogg-2-0-android-vulnerability-explained.../>

<sup>11</sup> <http://blog.takemyhand.xyz/2021/02/android-task-hijacking-with.html>

<sup>12</sup> <https://www.slideshare.net/phdays/android-task-hijacking>

<sup>13</sup> <https://gitlab.com/circuloapp/circulo-for-ios/-/issues/146>

<sup>14</sup> <https://gitlab.com/circuloapp/circulo-keanu-android/-/issues/160>

<sup>15</sup> <https://gitlab.com/circuloapp/circulo-keanu-android/-/commit/67...e6>

observe how the input text can be read by the user. This text will be readable even after a phone reboot.

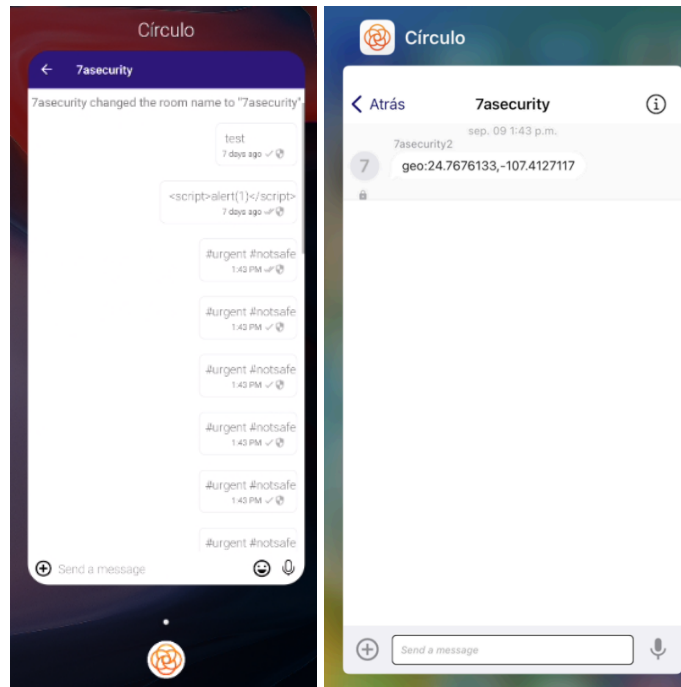


Fig.: Leaks via missing security screen on Android (left) and iOS (right)

It is recommended to render a security screen on top when the app is going to be sent to the background:

For iOS apps, the application being sent into the background can be detected in *Swift*<sup>16</sup> and *Objective-C*<sup>17</sup>. After that, a different screen, namely the security screen without user-data, can be shown. A revised approach prevents leakage of sensitive information via iOS screenshots. This is typically accomplished in the *AppDelegate* file, using the *applicationWillResignActive* or *applicationDidEnterBackground* methods.

For Android apps, it is recommended to implement a security screen by capturing the relevant backgrounding events, typically *onActivityPause*<sup>18</sup> or the *ON\_PAUSE* Lifecycle event<sup>19</sup> are used for such purposes. After that, if possible, ensure that all views have the Android *FLAG\_SECURE* flag<sup>20</sup> set. This will guarantee that even apps running with root privileges are unable to directly capture information displayed by the app on screen. Alternatively, an activity that all other activities inherit could be amended to always set this flag, regardless of the focus.

<sup>16</sup> <https://www.hackingwithswift.com/example-code/system/how-to-detect-when-your-app-mo...ackground>

<sup>17</sup> <https://developer.apple.com/...-applicationwillresignactive?language=objc>

<sup>18</sup> <https://developer.android.com/.../Application.ActivityLifecycleCallbacks#onActivityPaused...>

<sup>19</sup> <https://developer.android.com/reference/androidx/lifecycle/Lifecycle.Event>

<sup>20</sup> [http://developer.android.com/reference/android/view/Display.html#FLAG\\_SECURE](http://developer.android.com/reference/android/view/Display.html#FLAG_SECURE)

## CIR-01-006 WP1: Multiple DoS via Exported Activities (*Medium*)

**Retest Notes:** The Circulo team fixed this issue<sup>21</sup> and 7ASecurity confirmed that the fix is valid.

The Circulo Android app can be crashed by invoking activities with specially-crafted intents. Malicious apps on the same device can exploit this to repeatedly crash the app, discouraging use. This is primarily a risk on API level 28 and below, as newer Android versions restrict background apps from sending intents unless the app is in the foreground<sup>22</sup>.

### Affected exported activities:

- *org.article19.circulo.next/.main.JoinCircleActivity*
- *info.guardianproject.keanuapp.ui.PanicSetupActivity*
- *org.article19.circulo.next.main.updatestatus.UpdateStatusActivity*
- *info.guardianproject.keanu.core.RouterActivity*

### Steps to reproduce:

1. Open the Circulo app and push it to the background whilst running.
2. Record the Android logs locally via:  
`adb logcat > log.txt`
3. Open the IntentFuzzer<sup>23</sup> app and select NonSystemApps > *org.article19.circulo.next*
4. Scroll down in the activities and long-press one of the exported activities until a serializable intent is sent.
5. Confirm in the logcat output that a serializable intent caused a fatal crash in *org.article19.circulo.next/.main.JoinCircleActivity*

### Resulting Crash Output in Logcat:

```
09-18 20:33:39.898 1700 1932 I ActivityManager: Start proc 09-18 20:33:41.420 1700
3819 I ActivityManager: Start proc 12270:org.article19.circulo.next/u0a170 for activity
org.article19.circulo.next/.main.JoinCircleActivity caller=com.android.intentfuzzer
[...]
09-18 20:33:43.786 12270 12270 D AndroidRuntime: Shutting down VM
09-18 20:33:43.787 12270 12270 E AndroidRuntime: FATAL EXCEPTION: main
09-18 20:33:43.787 12270 12270 E AndroidRuntime: Process: org.article19.circulo.next,
PID: 12270
09-18 20:33:43.787 12270 12270 E AndroidRuntime: java.lang.RuntimeException: Parcelable
encountered ClassNotFoundException reading a Serializable object (name =
com.android.intentfuzzer.util.SerializableTest)
```

<sup>21</sup> <https://gitlab.com/circuloapp/circulo-keanu-android/-/issues/161>

<sup>22</sup> <https://developer.android.com/guide/components/activities/background-starts>

<sup>23</sup> <https://github.com/MindMac/IntentFuzzer>

```
09-18 20:33:43.787 12270 12270 E AndroidRuntime: at android.os.Parcel.  
readSerializable(Parcel.java:2941)  
09-18 20:33:43.787 12270 12270 E AndroidRuntime: at android.os.Parcel.  
readValue(Parcel.java:2722)  
[...]  
09-18 20:33:43.787 12270 12270 E AndroidRuntime: at android.os.BaseBundle.  
containsKey(BaseBundle.java:513)  
09-18 20:33:43.787 12270 12270 E AndroidRuntime: at android.content.Intent.  
hasExtra(Intent.java:7283)  
09-18 20:33:43.787 12270 12270 E AndroidRuntime: at android.app.Activity.  
finish(Activity.java:5684)  
[...]
```

The application crash looks as follows on the user interface:

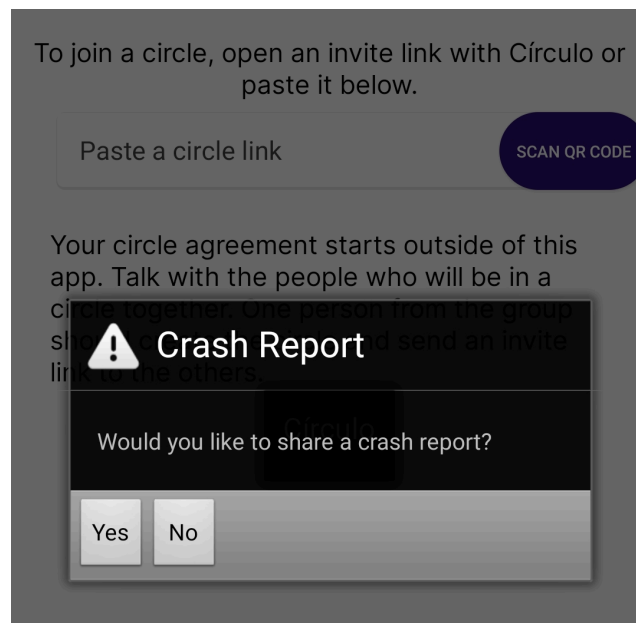


Fig.: Crash caused by the Serializable fuzzing attempt

The crash occurs when the internal Android mechanisms trigger deserialization as the onCreate activity event finishes and processes objects in intent extras. Specifically, if an unsupported Serializable class is included in the intent, the `android.os.Parcel.readSerializable()` method is implicitly invoked, causing a `ClassNotFoundException`. This affects activities that do not properly validate or handle deserialization failures when transitioning out of their lifecycle by calling `finish()`.

The problematic code is found in activities like `JoinCircleActivity`, where implicit deserialization occurs during the lifecycle transition triggered by `finish()`, without proper type-checking or error handling for intent extras:

**Affected Code Example:**

```
class JoinCircleActivity : BaseActivity() {
    [...]
    override fun onCreate(savedInstanceState: Bundle?) {
        [...]
        if (intent?.dataString?.isNotEmpty() == true)
            editTextCircleLink.setText(intent?.dataString)
        if (intent?.data != null)
            editTextCircleLink.setText(intent?.data?.toString())

        mViewBinding.btnNext.setOnClickListener {
            Config.ENABLE_EMPTY_STATE = false
            joinRoom()
        }

        queryCircles()
    }
    [...]
    private fun joinRoom () {
        var roomId = mViewBinding.editTextCircleLink.text.toString()
        if (roomId.isNotEmpty()) {
            mCoroutineScope.launch (mCoroutineExceptionHandler) {
                roomId = roomId.split("#")[1]
                if (mCurrentRoom == null) {
                    [...]
                    finish()
                }
            }
        }
    }
}
```

When `finish()` is called, Android attempts to deserialize any *Serializable* objects in the intent extras via the *Parcel* mechanism. If the intent contains an unrecognized class (e.g., `com.android.intentfuzzer.util.SerializableTest`), this triggers a *ClassNotFoundException* during deserialization in `Parcel.readSerializable()`, causing a crash. This issue arises due to deserialization handling, particularly when passing *Parcelable* or *Serializable* objects between activities<sup>24</sup>.

To mitigate this issue, it is recommended to implement the following measures:

1. **Wrap all deserialization calls** (e.g., `getSerializableExtra`, `getParcelableExtra`) in *try-catch* blocks to gracefully handle any exceptions, such as *ClassNotFoundException* or *ClassCastException*.
2. **Validate all incoming intents** to ensure that only expected data types are processed. Reject or sanitize any unexpected data types before processing.
3. **Log unrecognized or malformed data** in intents and ignore it, rather than allowing the app to crash.

More broadly, it is advised to export the minimum possible number of activities for the application to work. Once this is done, it may be possible to protect some of the

<sup>24</sup> <https://stackoverflow.com/questions/46728857/kotlin-parcelable-class-throwing-classnotfoundexception>

remaining exported activities with appropriate Android permissions. For additional mitigation guidance and background regarding the protection of Android activities with permissions, please see the slides of *An In-Depth Introduction to the Android Permission Model*<sup>25</sup>, as well as the stackoverflow discussion regarding *How to use custom permissions in Android*<sup>26</sup>. For activities that must be exported and cannot be protected, adequate input validation and exception handling should be in place to eliminate this attack vector.

### CIR-01-007 WP1: Circulo Chat History Access via Memory Leaks (*Medium*)

It was found that the Android app keeps chat history in memory. This approach is insecure because that information could be accessed by a malicious attacker with physical access or memory access. Furthermore, given the large volume of publicly known Android kernel vulnerabilities<sup>27</sup> and the high likelihood of users on unpatched Android devices, it should be assumed that malicious apps may be able to gain such access via privilege escalation vulnerabilities.

To confirm this issue the app process memory was dumped and the contents were examined for possible leaks. In particular, a search for the chat history discovered occurrences in memory:

#### Command:

```
grep -E "payload\":".*" strings.txt --text
```

#### Output:

```
{"payload":{"type":"m.room.message","content":{"msgtype":"m.text","body":"I am in a meeting room","format":"org.matrix.custom.html"}
[...]
```

The possible root cause for this issue appears to be in the Matrix SDK which uses a *toString()*<sup>28</sup> method to obtain a String representation of the object:

#### Affected File:

```
org.matrix.android.sdk.api.session.room.model.message.MessageEmoteContent
```

#### Affected Code:

```
public String toString() {
    return "MessageEmoteContent(msgType=" + getMsgType() + ", body=" + getBody() +
    ", format=" + getFormat() + ", formattedBody=" + getFormattedBody() + ", relatesTo=" +
    getRelatesTo() + ", newContent=" + getNewContent() + ")";
}
```

<sup>25</sup> [https://www.owasp.org/...How\\_to\\_Secure\\_MultiComponent\\_Applications.pdf](https://www.owasp.org/...How_to_Secure_MultiComponent_Applications.pdf)

<sup>26</sup> <https://stackoverflow.com/questions/8816623/how-to-use-custom-permissions-in-android>

<sup>27</sup> [https://www.cvedetails.com/vulnerability-list.php?vendor\\_id=1224&product\\_id=19997...](https://www.cvedetails.com/vulnerability-list.php?vendor_id=1224&product_id=19997...)

<sup>28</sup> [https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Object.html#toString\(\)](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Object.html#toString())

```
}

```

To resolve this issue, at a minimum, chat information should be regularly wiped from memory to avoid potential leaks. Additionally, sensitive data like encryption keys, should not be retained in RAM longer than necessary. Variables storing keys or messages ought to be nullified after use. Immutable objects like *java.lang.String* should be avoided for sensitive information, opting for char arrays instead. Even after removing or nullifying references to immutable objects, they might persist in memory until garbage collection, which apps are unable to enforce. For additional mitigation guidance, please see the *Testing Memory for Sensitive Data* section of the *Mobile Application Security Testing Guide (MASTG)*<sup>29</sup>.

### CIR-01-008 WP1: Circulo Room ID Access via Log Leaks (*Medium*)

**Retest Notes:** The Circulo team fixed this issue<sup>30,31</sup> and 7ASecurity confirmed that the fix is valid.

Both the Android and iOS apps were found to leak the Room ID via log messages during multiple operations. A malicious attacker with access to an unlocked device could exploit this by enabling USB debugging and retrieving the Room ID from the *logcat buffer*<sup>32</sup> or iOS logs, potentially gaining unauthorized access to a Circle.

To confirm the issue, log messages on both platforms were inspected for leaked Room IDs, with iOS logs reviewed as shown below.

#### Affected File:

[https://github.com/matrix-org/matrix-ios-sdk/\[...\]/MatrixSDK/Data/MXRoom.m#L232](https://github.com/matrix-org/matrix-ios-sdk/[...]/MatrixSDK/Data/MXRoom.m#L232)

#### Affected Code:

```
- (MXHTTPOperation*)members:(void (^)(MXRoomMembers *members))success
    lazyLoadedMembers:(void (^)(MXRoomMembers
*lazyLoadedMembers))lazyLoadedMembers
    failure:(void (^)(NSError *error))failure
{
    NSLog(@"[MXRoom] members: roomId: %@", _roomId);

```

#### Corresponding iOS Log message:

```
11:43:24.931 [MXRoom] members: roomId: !QnAHspzlwKuJMaECMF:encirculo.org
[...]
```

<sup>29</sup> <https://mas.owasp.org/MASTG/tests/android/MASVS-STORAGE/MASTG-TEST-0011/>

<sup>30</sup> <https://gitlab.com/circuloapp/circulo-for-ios/-/issues/147>

<sup>31</sup> <https://gitlab.com/circuloapp/circulo-keanu-android/-/issues/163>

<sup>32</sup> <https://developer.android.com/studio/command-line/logcat>

In the case of the Android app, the *OkHttp*<sup>33</sup> package is currently configured in a way that leaks at least certain HTTP requests like the following:

**Affected File:**

*org.matrix.android.sdk.BuildConfig*

**Affected Code:**

```
public final class BuildConfig {  
    public static final String BUILD_TYPE = "release";  
    public static final boolean DEBUG = false;  
    [...]  
    public static final HttpLoggingInterceptor.Level OKHTTP_LOGGING_LEVEL =  
    HttpLoggingInterceptor.Level.BASIC;
```

**Corresponding Android Log message:**

```
09-18 17:21:23.706 24084 24835 V FormattedJsonHttpLogger: --> POST  
https://casa.encirculo.org/_matrix/client/r0/join/!qhXhKVeuzUUhcyYeeZ:encirculo.org  
(2-byte body)  
[...]
```

To prevent the leakage of sensitive information from the Software Development Kit (SDK) in the mobile application logs, even when the debug level is disabled, the following should be considered:

- Disable or minimize SDK logging, especially debug or verbose levels. Check SDK documentation for log-level settings.
- Redact sensitive data from SDK logs using custom hooks or by extending existing functions.
- Contact the SDK vendor for support if sensitive information is still logged despite configuration.

More broadly, for Android apps, it is further advised to avoid logging sensitive information, especially when the *debug* flag is not set in the APK. Common approaches to implement this are:

- To create a *log wrapper*, check if the build is a *debug* build there, only log debug and verbose messages for a *debug* build<sup>34</sup>.
- To create ProGuard rules so that *Log.d* and *Log.v* are removed when the build is for production<sup>35</sup>.
- Avoid the usage of intent filters that may leak sensitive information to *logcat*.

Regarding iOS leaks, where possible, it is further recommended to replace all *print* and

<sup>33</sup> <https://square.github.io/okhttp/3.x/.../okhttp3/logging/HttpLoggingInterceptor.Level.html#BASIC>

<sup>34</sup> <https://stackoverflow.com/a/4592958>

<sup>35</sup> <https://stackoverflow.com/a/2466662>



*NSLog* debugging calls with equivalent *os\_log*<sup>36</sup> calls. Apart from the user privacy improvements, this comes with a number of other advantages including logging levels<sup>37</sup> and better log filtering in *Console.app*<sup>38</sup>. Please note that starting with iOS 14, a new logger API is available which wraps around *os\_log* with additional improvements<sup>39</sup>. However, this may not be an option while the app keeps supporting lower versions such as iOS 13.

## CIR-01-009 WP1: User DoS via DNS Spoofing on iOS & Android (High)

**Retest Notes:** The Circulo team fixed this issue<sup>4041</sup> and 7A Security confirmed that the fix is valid.

It was found that the Android and iOS apps are vulnerable to DoS attacks via spoofing of the DNS domains used for establishing backend communications. Malicious attackers, able to modify clear-text network communications (i.e. open Wi-Fi without guest isolation, BGP hijacking, ISP MitM, DNS rebinding, etc.), could leverage this weakness to prevent legitimate app users from accessing the system. In a worst case scenario, a malicious attacker may leverage DNS rebinding to prevent all Circulo users from notifying their peers and trusted authorities of danger.

This issue was confirmed by changing the DNS settings on the test iOS and Android devices (i.e. to simulate DNS spoofing), so that they point to an attacker-controlled DNS server using *dnschef*<sup>42</sup>, spoofing the domains as follows:

### Command:

```
./dnschef.py -i 192.168.1.6 --fakeip 0.0.0.0 --fakedomains "casa.encirculo.org"
```

### Corresponding Logcat Output:

```
09-20 14:10:58.643 1132 1200 E SYNC/SyncThread: Caused by: java.net.ConnectException:
Failed to connect to casa.encirculo.org/0.0.0.0:443
09-20 14:11:08.727 1132 1207 V FormattedJsonHttpLogger: --> GET
https://casa.encirculo.org/_matrix/client/r0/sync?filter=0&set_presence=online&timeout=
0&since=s64557_11405184_90_9265_15469_5411_4931_33734_0_1
09-20 14:11:08.763 1132 1207 V FormattedJsonHttpLogger: <-- HTTP FAILED:
java.net.ConnectException: Failed to connect to casa.encirculo.org/0.0.0.0:443
```

This issue occurs due to all backend communication being built on top of the insecure

<sup>36</sup> [https://developer.apple.com/documentation/os/os\\_log](https://developer.apple.com/documentation/os/os_log)

<sup>37</sup> <https://www.lordcodes.com/articles/clear-and-searchable-logging-in-swift-with-oslog>

<sup>38</sup> <https://www.avanderlee.com/workflow/oslog-unified-logging/>

<sup>39</sup> <https://betterprogramming.pub/ios-14s-new-logger-api-vs-oslog-ef88bb2ec237?gi=87ad8c30fb48>

<sup>40</sup> <https://gitlab.com/circuloapp/circulo-for-ios/-/issues/148>

<sup>41</sup> <https://gitlab.com/circuloapp/circulo-keanu-android/-/issues/164>

<sup>42</sup> <https://github.com/iphelix/dnschef>

DNS protocol. An example of this can be seen in the output of the command below, which shows a failure to leverage the latest iOS protections for iOS 14 (explained in the mitigation guidance below):

**Command:**

```
egrep -Ir '(NWParameters|PrivacyContext)' * | wc -l
```

**Output:**

```
0
```

It is recommended to switch over to *DNS over HTTPS (DoH)*<sup>43</sup> to mitigate these types of attacks. This will automatically remove all clear-text DNS resolution with its associated privacy and security problems and instead encrypt all DNS traffic over HTTPS, this ensures DNS traffic will have the confidentiality and integrity protections offered by the HTTPS protocol thereafter. On iOS, since iOS 14 Apple allows developers to specify DoH connection parameters via *NWParameters.PrivacyContext*<sup>44,45</sup>, however, it is also possible to implement DoH at the app level in a compatible way with older iOS versions<sup>46</sup>. On Android, DoH can be easily deployed via the *okhttp-dns-over-https* module<sup>47</sup>, which has a Kotlin implementation available<sup>48</sup> that could alternatively be used as a reference.

### CIR-01-013 WP1: Auth Token Access via inadequate Keychain Usage (Medium)

It was found that the iOS app fails to make adequate use of the iOS Keychain. This hardware-backed security enclave is the most secure location to store app secrets on the device. A malicious attacker with access to the memory of the iOS device could gain access to user credentials and take over user accounts.

To confirm this issue the app process memory was dumped and the contents were reviewed for username, password, and token leaks.

In particular, a search for the user authentication token revealed 1 occurrence in memory:

**Commands:**

```
grep Bearer ios_mem.dump.strings | wc -l
```

<sup>43</sup> [https://en.wikipedia.org/wiki/DNS\\_over\\_HTTPS](https://en.wikipedia.org/wiki/DNS_over_HTTPS)

<sup>44</sup> <https://developer.apple.com/documentation/network/nwparameters/privacycontext>

<sup>45</sup> <https://gist.github.com/sschizas/ed03571ed129a227947f482b51ffabc5>

<sup>46</sup> <https://www.wwdcnotes.com/notes/wwdc20/10047/>

<sup>47</sup> <https://github.com/square/okhttp/tree/master/okhttp-dns-over-https>

<sup>48</sup> <https://github.com/square/okhttp/blob/master/okhttp-dns-over-https/.../dns-over-https/DnsOverHttps.kt>

## Output:

1

In order to solve this problem it is recommended to:

1. Only store app secrets in the iOS Keychain
2. Clear sensitive information such as authentication tokens from memory
3. Restrict the level of access for each stored Keychain item as much as possible

For keychain items that are not required by processes running in the background, it is recommended to use a more restricted level of access. The best options for approaching this are noted below, ordered by the protection level they provide (i.e. ideal option first):

### **Option 1: *AccessibleWhenPasscodeSetThisDeviceOnly***<sup>49</sup>

This is the absolute best option, it requires users to have a passcode set in the device and makes keychain items only available while the device is unlocked. Data will not be exported to backups and credentials will not be restored on another device when backups are restored.

Please note this option can be further secured by requiring the user to authenticate via *Face* or *Touch ID* prior to the application being able to access the relevant keychain item<sup>50</sup>.

### **Option 2: *AccessibleWhenUnlockedThisDeviceOnly***<sup>51</sup>

This is the best option if the data should not be exported to backups. Credentials will not be restored on another device when the backup is restored.

### **Option 3: *AccessibleWhenUnlocked***<sup>52</sup>

This is the best option if the data should be exported to backups. Credentials will be restored on another device when the backup is restored.

---

<sup>49</sup> <https://developer.apple.com/documentation/security/ksecattraccessiblewhenpasscodesetthisdeviceonly>

<sup>50</sup> [https://developer.apple.com/.../accessing\\_keychain\\_items\\_with\\_face\\_id\\_or\\_touch\\_id](https://developer.apple.com/.../accessing_keychain_items_with_face_id_or_touch_id)

<sup>51</sup> <https://developer.apple.com/documentation/security/ksecattraccessiblewhenunlockedthisdeviceonly>

<sup>52</sup> <https://developer.apple.com/documentation/security/ksecattraccessiblewhenunlocked>

## CIR-01-021 WP4: Synapse Admin API Exposed to the Internet (*Medium*)

The Circulo backend uses *Synapse*<sup>53</sup> as a Matrix HomeServer, connecting all clients to a single port that exposes both the regular and Admin API. The Admin API is publicly accessible, allowing an attacker who compromises an admin user to perform administrative actions remotely. Synapse also permits admin user registration via a script or the Admin API with a shared registration secret, and existing users can be upgraded to admin<sup>54</sup> status. This creates a risk where, if admin credentials or a session are compromised, attackers could execute admin actions, including deleting history, listing rooms, and accessing user data, including email addresses.

### Affected Resources:

*casa.encirculo.org*  
*matrix.unready.im*

To verify the Admin API is exposed the following steps can be performed:

### Command:

```
curl -X GET "https://casa.encirculo.org/_synapse/admin/v2/users"
```

### Output:

```
{"errcode": "M_MISSING_TOKEN", "error": "Missing access token"}
```

A valid administrator user, created for testing purposes on the development environment, can utilize the Admin API to for instance delete rooms or fetch information such as a complete list of users:

### Command:

```
curl -X GET "https://matrix.unready.im/_synapse/admin/v2/users" -H "Authorization: Bearer syt_c2dfN2FzZW[...]"
```

### Output:

```
{
  "users": [
    [...]
    {
      "name": "@weblite-gfpp33eeo4dl:matrix.unready.im",
      "user_type": null,
      "is_guest": false,
      "admin": false,
      "deactivated": false,
      "shadow_banned": false,
      "displayname": "Guest Butterfly",
    }
  ]
}
```

<sup>53</sup> <https://github.com/element-hq/synapse>

<sup>54</sup> [https://element-hq.github.io/synapse/latest/usage/administration/admin\\_api/index.html](https://element-hq.github.io/synapse/latest/usage/administration/admin_api/index.html)

```
"avatar_url": "mxc://matrix.unready.im/UYeGDpKwFQ0iyvJKVxXsYItJ",
"creation_ts": 1701270906000,
"approved": true,
"erased": false,
"last_seen_ts": null,
"locked": false
}
],
"total": 259,
"next_token": "100"
}
```

It is recommended to review Synapse documentation<sup>55</sup> and limit Admin API access to internal, well-monitored networks. Administrative actions, such as creating new admins or promoting users, should also be logged and monitored to detect potential compromises early.

### CIR-01-022 WP4/6: Data Leaks in Nginx and CloudWatch Logs (*Medium*)

CloudWatch logs were found to store unmasked sensitive data, including session access tokens and RoomIDs. An attacker, with access to logs, could potentially extract sensitive data and create a link to join a room or steal a bearer token.

#### **Affected Resources:**

*AWS Account 384647892761*  
*matrix.unready.im*

#### **Example 1: RoomID not redacted from logs**

Multiple actions involving a room utilize a URL containing the room identifier. If this identifier is extracted from logs, an attacker can craft a link to join a non-public room, if the join rule is set to public, without the room creator consent.

This can be confirmed by reviewing the CloudWatch log group:

1. Open the *AWS Management Console*
2. Navigate to the following CloudWatch Log Group URL.

#### **PoC URL:**

[https://eu-central-1.console.aws.amazon.com/cloudwatch/home?region=eu-central-1#logsV2:log-groups/log-group/\\$252Fgp\\$252Funreadyim\\$252Fprod\\$252Fmain/log-events/i-00340570a840becd5](https://eu-central-1.console.aws.amazon.com/cloudwatch/home?region=eu-central-1#logsV2:log-groups/log-group/$252Fgp$252Funreadyim$252Fprod$252Fmain/log-events/i-00340570a840becd5)

3. Search for sample join room events using %<SOMEROOMID>%.

#### **Output (SYSTEMD\_USER\_SLICE event):**

<sup>55</sup> [https://element-hq.github.io/synapse/latest/reverse\\_proxy.html#synapse-administration-endpoints](https://element-hq.github.io/synapse/latest/reverse_proxy.html#synapse-administration-endpoints)

```
{
  "@timestamp": "2024-10-04T07:57:00.477997Z",
  "_SYSTEMD_USER_SLICE": "app.slice",
  "hostname": "gp-unreadyim-prod",
  "instance_name": "gp-unreadyim-prod-main",
  "level": "INFO",
  "message": "88.156.202.205 - 18000 - {@sg_7asec:matrix.unready.im} Processed
request: 0.260sec/0.001sec (0.015sec, 0.001sec) (0.006sec/0.058sec/20) 51B 200 \"POST
/_matrix/client/r0/rooms/!vSnwYBiXeksnSXvPhc:matrix.unready.im/join HTTP/1.1\"
\\\"curl/8.8.0\\\" [4 dbevts]\",
  "namespace": "gp",
  "stage": "prod",
  "synapse": {
    "authenticated_entity": "@sg_7asec:matrix.unready.im",
    "instance": "matrix-synapse-worker-generic-0",
    "method": "POST",
    "namespace": "synapse.access.http.18000",
    "protocol": "HTTP/1.1",
    "request": "POST-76736dd74e0a68db2ff5d9d52e71ed9b",
    "requester": "@sg_7asec:matrix.unready.im",
    "server_name": "matrix.unready.im",
    "site_tag": "18000",
    "time": 1728028620.48,
    "url": "/_matrix/client/r0/rooms/!vSnwYBiXeksnSXvPhc:matrix.unready.im/join",
    "user_agent": "curl/8.8.0"
  },
},
[...]
```

### Output (NGINX access.log event):

```
{
  "@timestamp": "2024-10-04T07:57:00+00:00",
  "file": "/var/log/nginx/access.log",
  "host": "gp-unreadyim-prod",
  "http": {
    "bytes_request": 614,
    "bytes_response": 62,
    "host": "matrix.unready.im",
    "method": "POST",
    "proxy_port": "25266",
    "request_id": "76736dd74e0a68db2ff5d9d52e71ed9b",
    "request_uri":
"/_matrix/client/r0/rooms/!vSnwYBiXeksnSXvPhc:matrix.unready.im/join",
    "scheme": "http",
    "server_port": "80",
    "server_protocol": "HTTP/1.1",
    "status": 200,
    "synapse_backend": "synapse_worker_generic",
    "synapse_uri_group": "synapse_client_transaction",
    "upstream_addr": "127.0.0.1:18000",
    "uri": "/_matrix/client/r0/rooms/!vSnwYBiXeksnSXvPhc:matrix.unready.im/join",
  },
}
```

```
    "user_agent": "curl/8.8.0"  
  },  
  [...]
```

### Example 2: Bearer token not redacted if used in query parameters

The Synapse HomeServer accepts bearer tokens in query strings. While Circulo mobile applications do not use this method, non-standard clients or tools, such as those used to query the Admin API, may. In such cases, the unredacted token could be logged.

This can be confirmed by reviewing the same CloudWatch log group as in the previous example, but searching for *access\_token* values *%access\_token=syt%*:

#### Output:

```
[...]  
"http": {  
  "bytes_request": 496,  
  "bytes_response": 7785,  
  "cf_connecting_ip": "[...]",  
  "host": "matrix.unready.im",  
  "method": "GET",  
  "proxy_port": "2080",  
  "request_id": "3f8b8a53f67a17f2408beafff3b32102",  
  "request_uri":  
  "/_matrix/client/r0/sync?access_token=syt_c2dfN2FzZWMtbn0ybWFs_dumgpHr[...]",  
  "scheme": "http",  
  "server_port": "80",  
  "server_protocol": "HTTP/1.1",  
  "status": 200,  
  "synapse_backend": "synapse_worker_initial_sync",  
  "synapse_uri_group": "synapse_initial_sync",  
  "time_request": 0.28,  
  "time_upstream_connect": 0,  
  "time_upstream_header": 0.28,  
  "time_upstream_response": 0.28,  
  "upstream_addr": "127.0.0.1:18100",  
  "uri": "/_matrix/client/r0/sync",  
  "user_agent": "curl/8.8.0",  
  "user_http_referer": "",  
  },  
  [...]
```

In some cases, the *access\_token* value was replaced with *<redacted>*, but not in all log entries, indicating inconsistent redacting rules.



It is recommended to periodically review logs and create patterns to alert on detected sensitive data. Sensitive data in logs should be removed or masked to prevent data leakage and potential session hijacking.



## Hardening Recommendations

This area of the report provides insight into less significant weaknesses that might assist adversaries in certain situations. Issues listed in this section often require another vulnerability to be exploited, need an uncommon level of access, exhibit minor risk potential on their own, and/or fail to follow information security best practices. Nevertheless, it is recommended to resolve as many of these items as possible to improve the overall security posture and protect users in edge-case scenarios.

### CIR-01-001 WP1: Missing Jailbreak/Root Detection on Android & iOS (*Info*)

It was found that the Circulo Android and iOS apps do not implement any form of root or Jailbreak detection features at the time of writing. Hence, the applications fail to alert users about the security implications of running the app in such an environment<sup>56</sup>. This issue can be confirmed by installing each application on a jailbroken/rooted device and validating the complete lack of application warnings.

It is recommended to implement a comprehensive Jailbreak and root detection solution to address this problem. Please note that, since the user has root access and the application does not, the application is always at a disadvantage. **Mechanisms like these should always be considered bypassable** when enough dedication and skill characterize the attacker.

Some freely available libraries for iOS are *IOSSecuritySuite*<sup>57</sup> and *DTTJailbreakDetection*<sup>58</sup>, although custom checks are also possible in Swift applications<sup>59</sup>. Such solutions should be considered bypassable but sufficient to warn users about the dangers of running the application on a jailbroken device. For best results, it is recommended to test some commercial and open source<sup>60,61</sup> solutions against well-known *Cydia tweaks* like *LibertyLite*<sup>62</sup>, *Shadow*<sup>63</sup>, *tsProtector 8+*<sup>64</sup> or *A-Bypass*<sup>65</sup>. Based on this, the development team could determine the most solid approach.

---

<sup>56</sup> <https://www.bankinfosecurity.com/jailbreaking-ios-devices-risks-to-users-enterprises-a-8515>

<sup>57</sup> <https://cocoapods.org/pods/IOSSecuritySuite>

<sup>58</sup> <https://github.com/thii/DTTJailbreakDetection>

<sup>59</sup> <https://sabatsachin.medium.com/detect-jailbreak-device-in-swift-5-ios-programatically-da467028242d>

<sup>60</sup> <https://github.com/thii/DTTJailbreakDetection>

<sup>61</sup> <https://github.com/securing/IOSSecuritySuite>

<sup>62</sup> <http://ryleyangus.com/repo/>

<sup>63</sup> <https://ios.ijolano.me/>

<sup>64</sup> <http://apt.thebigboss.org/repofiles/cydia/>

<sup>65</sup> <https://repo.rpgfarm.com/>

The freely available *rootbeer* library<sup>66</sup> for Android could be considered for the purpose of alerting users on rooted devices, while bypassable, this would be sufficient for alerting users of the dangers of running the app on rooted devices.

## CIR-01-002 WP1: Support of Insecure v1 Signature on Android (Info)

It was found that the Circulo Android build currently in production is signed with an insecure *v1 APK signature*. Using the *v1 signature* makes the app prone to the known Janus<sup>67</sup> vulnerability on devices running Android < 7. The problem lets attackers smuggle malicious code into the APK without breaking the signature. At the time of writing, the app supports a minimum SDK of 21 (Android 5), which also uses the *v1 signature*, hence being vulnerable to this attack. Furthermore, Android 5 devices no longer receive updates and are vulnerable to many security issues, it can be assumed that any installed malicious app may trivially gain root privileges on those devices using public exploits<sup>686970</sup>.

The existence of this flaw means that attackers could trick users into installing a malicious attacker-controlled APK that matches the *v1 APK signature* of the legitimate Android application. As a result, a transparent update would be possible without warnings appearing in Android, effectively taking over the existing application and all of its data.

It is recommended to increase the minimum supported SDK level to at least 24 (Android 7) to ensure that this known vulnerability cannot be exploited on devices running older Android versions. In addition, future production builds should only be signed with *v2* and greater APK signatures.

---

<sup>66</sup> <https://github.com/scottyab/rootbeer>

<sup>67</sup> <https://www.guardsquare.com/en/blog/new-android-vulnerability-allows-atta....affecting-their-signatures>

<sup>68</sup> <https://www.exploit-db.com/exploits/35711>

<sup>69</sup> <https://github.com/davidqphan/DirtyCow>

<sup>70</sup> [https://en.wikipedia.org/wiki/Dirty\\_COW](https://en.wikipedia.org/wiki/Dirty_COW)

## CIR-01-004 WP1: Android Config Hardening Recommendations (Info)

It was found that the Circulo Android app fails to leverage optimal values for a number of security-related settings. This unnecessarily weakens the overall security posture of the application. These weaknesses are documented in more detail next.

### Issue 1: Missing Tapjacking Protection

The Android app accepts user taps while other apps render anything on top of it. Malicious attackers might leverage this weakness to impersonate users using a crafted app, which launches the victim app in the background while something else is rendered on top. Please note that this attack vector is mitigated in Android 12<sup>71</sup>. Since the app supports Android 5.0 (Lollipop), this leaves users on Android 5-11 vulnerable to this attack. The following command confirms that Tapjacking protections are missing in the source code provided and the decompiled app:

#### Command:

```
egrep -r  
'(filterTouchesWhenObscured|FLAG_WINDOW_IS_OBSCURED|FLAG_WINDOW_IS_PARTIALLY_OBSCURED)'  
* | wc -l
```

#### Output:

```
0
```

Please note this was also validated at runtime using the *FSecure tapjacking-poc*<sup>72</sup>.

Since Android API level 9 (Android 2.3), it is possible to mitigate Tapjacking attacks utilizing at least one of the following approaches:

**Approach 1:** The *filterTouchesWhenObscured*<sup>73,74</sup> attribute could be set at the Android root view level<sup>75</sup>. This will ensure that taps will be ignored when the Android app is not displayed on top.

**Approach 2:** Alternatively, *MotionEvent* could be checked against the following flags to present a protection screen on top:

1. *FLAG\_WINDOW\_IS\_OBSCURED*<sup>76</sup> (since Android 2.3)
2. *FLAG\_WINDOW\_IS\_PARTIALLY\_OBSCURED*<sup>77</sup> (since Android 10)

<sup>71</sup> <https://developer.android.com/topic/security/risks/tapjacking#mitigations>

<sup>72</sup> <https://github.com/FSecureLABS/tapjacking-poc>

<sup>73</sup> [http://developer.android.com/reference/\[...\]/View.html#setFilterTouchesWhenObscured\(boolean\)](http://developer.android.com/reference/[...]/View.html#setFilterTouchesWhenObscured(boolean))

<sup>74</sup> [http://developer.android.com/reference/\[...\]/View.html#attr\\_android:filterTouchesWhenObscured](http://developer.android.com/reference/[...]/View.html#attr_android:filterTouchesWhenObscured)

<sup>75</sup> <https://developer.android.com/reference/android/view/View#security>

<sup>76</sup> [https://developer.android.com/reference/android/view/MotionEvent#FLAG\\_WINDOW\\_IS\\_OBSCURED](https://developer.android.com/reference/android/view/MotionEvent#FLAG_WINDOW_IS_OBSCURED)

<sup>77</sup> [https://developer.android.com/reference/android/view/MotionEvent#FLAG\\_WINDOW\\_IS\\_PARTIALLY...](https://developer.android.com/reference/android/view/MotionEvent#FLAG_WINDOW_IS_PARTIALLY...)

## Issue 2: Missing *FLAG\_SECURE* for screenshot protection

The Android app allows applications to capture what is being displayed on the screen. Malicious apps without any special permissions may accomplish this by simply prompting the user for screen capture access, which is common in Android for screenshot and video recording apps. Malicious apps with root privileges can accomplish this without any user warnings or prompts. Please note that malicious apps could gain root privileges simply prompting the user for them on a rooted phone or exploiting a number of publicly known Android vulnerabilities<sup>78</sup> on unpatched devices (common). In the paper *Security Metrics for the Android Ecosystem*<sup>79</sup> researchers from the *University of Cambridge* showed that root privileges can in fact be gained on 87.7% of Android phones through a security vulnerability.

This issue can be verified on a physical device or emulator with the following commands, which using a non-root adb session will capture what is displayed on the screen while the Android app is open, and then download it to the computer:

### Commands:

```
adb shell screencap -p /sdcard/screenshot1.png
adb pull /sdcard/screenshot1.png
```

It is recommended to ensure that all Webviews have the Android *FLAG\_SECURE* flag<sup>80</sup> set. This will guarantee that even apps running with root privileges cannot directly capture the information displayed by the app. This is best accomplished in a centralized security control, such as the *onCreate* event of a base activity that all other activities inherit:

### Proposed Fix:

```
public class BaseActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(LayoutParams.FLAG_SECURE,
            LayoutParams.FLAG_SECURE);
    }
}
```

<sup>78</sup> [https://www.cvedetails.com/vulnerability-list.php?vendor\\_id=1224&product\\_id=19997&...](https://www.cvedetails.com/vulnerability-list.php?vendor_id=1224&product_id=19997&...)

<sup>79</sup> <https://www.cl.cam.ac.uk/~drt24/papers/spsm-scoring.pdf>

<sup>80</sup> [http://developer.android.com/reference/android/view/Display.html#FLAG\\_SECURE](http://developer.android.com/reference/android/view/Display.html#FLAG_SECURE)

### Issue 3: Undefined `android:hasFragileUserData`

Since Android 10, it is possible to specify whether application data should survive when apps are uninstalled with the `android:hasFragileUserData` attribute. When set to `true`, the user will be prompted to keep the app information despite uninstallation.

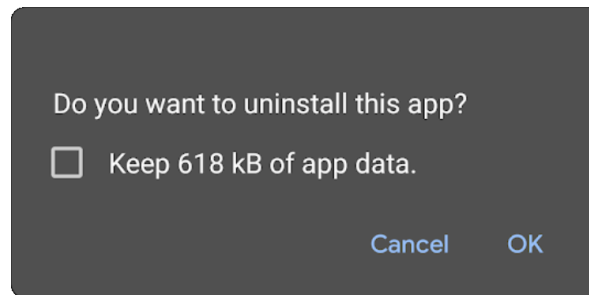


Fig.: Uninstall prompt with check box for keeping the app data

Since the default value is `false`, there is no security risk in failing to set this attribute. However, it is still recommended to explicitly set this setting to `false` to define the intention of the app to protect user information and ensure all data is deleted when the app is uninstalled. It should be noted that this option is only usable if the user tries to uninstall the app from the native settings. Otherwise, if the user uninstalls the app from Google Play, there will be no prompts asking whether data should be preserved or not.

### CIR-01-010 WP1: Usage of Insecure Crypto Functions (Low)

The Android application was found to employ the MD5 hashing algorithm, which has known security vulnerabilities. This hashing technique is no longer regarded as collision-resistant, allowing for the production of distinct inputs that result in the same hash value. Consequently, it is inappropriate for applications where data integrity or security are essential considerations<sup>81</sup>. This can be confirmed by reviewing the following code snippet:

#### Affected File:

[https://gitlab.com/circuloapp/circulo-keanu-android/\[...\]/core/util/AES\\_256\\_CBC.java](https://gitlab.com/circuloapp/circulo-keanu-android/[...]/core/util/AES_256_CBC.java)

#### Affected Code:

```
public static String decrypt(byte[] headerSaltAndCipherText, String password) {
    try {
        // --- extract salt & encrypted ---

        // header is "Salted__", ASCII encoded, if salt is being used (the
        // default)
```

<sup>81</sup> [https://en.wikipedia.org/wiki/MD5#Overview\\_of\\_security\\_issues](https://en.wikipedia.org/wiki/MD5#Overview_of_security_issues)

```
byte[] salt = copyOfRange(headerSaltAndCipherText, SALT_OFFSET, SALT_OFFSET +
SALT_SIZE);
byte[] encrypted = copyOfRange(headerSaltAndCipherText, CIPHERTEXT_OFFSET,
headerSaltAndCipherText.length);

// --- specify cipher and digest for EVP_BytesToKey method ---

Cipher aesCBC = Cipher.getInstance("AES/CBC/PKCS5Padding");
MessageDigest md5 = MessageDigest.getInstance("MD5");
```

It is recommended to replace *MD5* with adequate replacements without cryptographic weaknesses<sup>82</sup>. It has to be noted that certain secrets should be stored in a deliberately slow manner to avoid brute force attacks, these require a different set of hashing algorithms for secure storage as explained in the *OWASP Password Storage Cheat Sheet*<sup>83</sup>.

## CIR-01-011 WP1: iOS Binary Hardening Recommendations (Info)

It was found that the iOS binary fails to leverage some code injection, privilege escalation, and anti-tampering flags available. While this is not a serious issue on its own, it could facilitate code injection, privilege escalation, and application tampering attacks in edge-case scenarios.

### Issue 1: Missing `__RESTRICTED` segment

The binary does not have a restricted segment that prevents dynamic loading of *DYLIB* for arbitrary code injection.

#### Command:

```
size -x -l -m Circulo | grep __RESTRICT | wc -l
```

#### Output:

```
0
```

It is recommended to use the following compiler options to enable the restricted segment feature.

#### Proposed fix (compiler options):

```
-Wl,-sectcreate,__RESTRICT,__restrict,/dev/null
```

<sup>82</sup> [https://en.wikipedia.org/wiki/Secure\\_Hash\\_Algorithms](https://en.wikipedia.org/wiki/Secure_Hash_Algorithms)

<sup>83</sup> [https://cheatsheetseries.owasp.org/cheatsheets/Password\\_Storage\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)

## Issue 2: Binary not encrypted

Binary encryption refers to the process of securing the binary code (machine code) of an iOS application to protect it from unauthorized access, reverse engineering, and tampering. This is crucial for safeguarding the intellectual property and sensitive information contained within the app. This was confirmed using the following command on the Mach-O file.

### Command:

```
otool -l Circulo | egrep "LC_ENCRYPTION.*" -A 5
```

### Output:

```
cmd LC_ENCRYPTION_INFO_64
  cmdsize 24
  cryptoff 245760
  cryptsize 4096
  cryptid 0
  pad 0
```

In general, it is recommended to ensure all the security flags are applied correctly to all application binaries, specifically:

- The following should be enabled: *PIE*, *ARC*, *Canary*.
- These flags should be disabled: *Stack Exec*, *RootSafe*.

This will leverage all security features available in iOS, increasing the scope of protections in terms of memory corruption attacks.

## CIR-01-012 WP1: Weak PIN Policy on iOS & Android (Low)

The iOS and Android apps fail to enforce strong PIN validation, permitting weak, guessable PINs like 123456 or 000000. This creates a security risk, as attackers can use brute force techniques with automated tools like *Flipper Zero*<sup>84</sup> or *BadUSB*<sup>85</sup> to try multiple PIN combinations without triggering locks or restrictions, compromising user security. This allows attackers to eventually guess the correct PIN undetected, compromising user data confidentiality and integrity. This issue can be confirmed as follows:

### PoC Steps:

1. Access the user profile.
2. Navigate to the *Physical Security* section.
3. Activate the PIN lock option.
4. Set a weak PIN, such as 123456 or 000000.

<sup>84</sup> <https://flipperzero.one/>

<sup>85</sup> <https://shop.hak5.org/products/usb-rubber-duddy>

**Result:**

The weak PIN is accepted

This was further validated with the following *Flipper Zero*<sup>86</sup> script:

**Script:**

```
DELAY 500
STRING 012419
ENTER
DELAY 500
STRING 590055
ENTER
DELAY 500
STRING 686651
ENTER
DELAY 500
STRING 115864
ENTER
DELAY 500
STRING 264273
ENTER
DELAY 500
STRING 581121
ENTER
STRING 051793
[...]
```

**Output:**

---

<sup>86</sup> <https://flipperzero.one/>



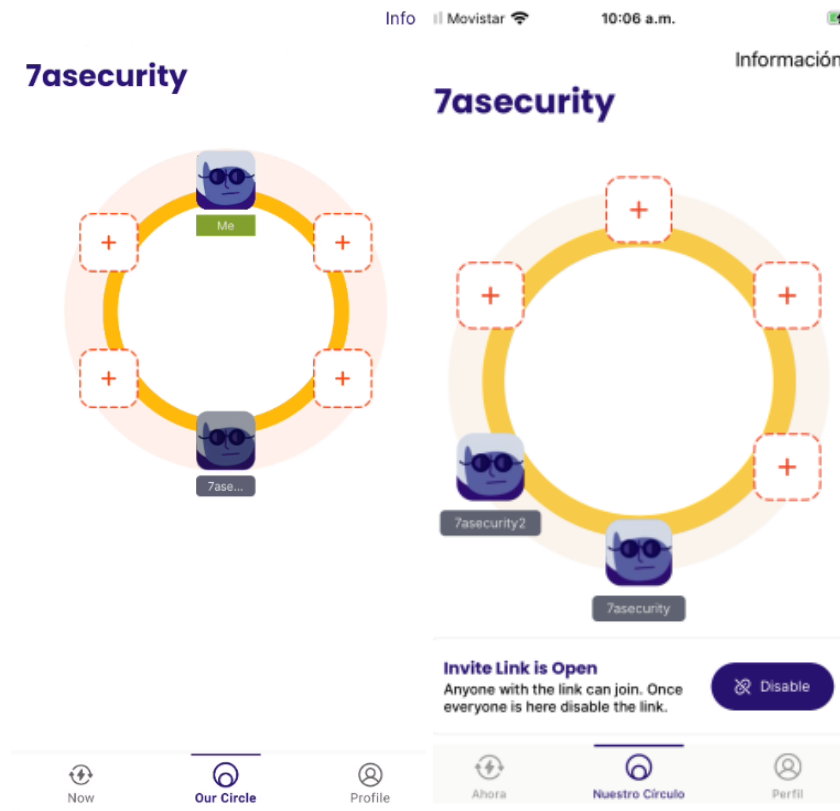


Fig.: Bypass Weak PIN on Android (left) and iOS (right)

A stricter PIN validation policy<sup>87</sup> should be implemented that requires a minimum of complexity, such as minimum length requirements, not allowing common or repeated numeric sequences, and limiting the number of failed PIN entry attempts. In addition, a temporary or permanent lockout mechanism should be implemented after multiple failed attempts to mitigate brute force attacks. Reinforce these measures with multi-factor authentication (MFA) to improve overall system security.

<sup>87</sup> <https://owasp.org/www-project-mobile-top-10/2016-risks/m4-insecure-authentication>

## CIR-01-014 WP1: Excessive Fingerprint Permissions on Android (Info)

The Android app includes the `android.permission.USE_BIOMETRIC` and `android.permission.USE_FINGERPRINT` permissions, despite fingerprint authentication being removed from the user interface due to concerns about physical security risks (e.g., coercion). These permissions remain in the configuration, which is unnecessary and might be exploited or cause security inconsistencies. Unnecessary permissions increase the attack surface, potentially allowing malicious components to access biometric data or unintended functions. This can be verified as follows:

**Affected File:**

*AndroidManifest.xml*

**Affected Code:**

```
<uses-permission android:name="android.permission.USE_BIOMETRIC"/>
```

Given that biometric authentication is no longer in use, it is recommended to remove the aforementioned permissions from the *AndroidManifest.xml* file to resolve this issue.

## CIR-01-015 WP6: Weaknesses in Vuln Management Processes (Medium)

The AWS configuration audit revealed that several critical security services are not enabled, leaving the infrastructure vulnerable due to insufficient hardening. Integrating native or third-party security tools into the environment is essential for early detection of anomalies, PII exposure, and lateral movement during a data breach.

**Affected Resources:**

*AWS Account 384647892761*

Please note that, as most of the AWS services are region-based, it is important to determine which regions are used first, to focus the analysis on the regions that are actually in use.

**Issue 1: Security Hub is not enabled**

*Security Hub* is a region-based service that provides a comprehensive view of security issues from regions where it is enabled<sup>88</sup>. The following command describes the status of *Security Hub* for AWS regions:

**Command:**

```
for r in $REGIONS; do aws securityhub describe-hub --region $r; done
```

<sup>88</sup> <https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-get-started.html>

**Output:**

[...]

**region: eu-central-1**

An error occurred (InvalidAccessException) when calling the DescribeHub operation:

**Account 384647892761 is not subscribed to AWS Security Hub**

[...]

**Issue 2: AWS GuardDuty is not enabled**

*Guard Duty*<sup>89</sup> is a service that provides insight into anomalies detected by native AWS breach detection solutions. The service is able to spot potentially malicious activities which can be used to notify administrators in case of a compromise. The disabled status of *Guard Duty* for the main *eu-central-1* region can be confirmed using the following command:

**Command:**

```
aws guardduty list-detectors --region eu-central-1
```

**Output:**

```
{ "DetectorIds": [] }
```

It is recommended to implement as many AWS Security related services as possible within the budget and business needs. It is advised to start enabling *Security Hub*<sup>90</sup> and *Guard Duty* later optionally *Macie*<sup>91</sup> and *Inspector*<sup>92</sup>. Please note that, even though those services provide great coverage and auditability, it might still be necessary to use 3rd-party services to detect anomalies and analyze cloud logs to reach the highest security standards.

Furthermore, consideration should be given to disable unused regions completely, this will ensure that it is not possible to create resources in unsupported regions. Alternatively, all region-based services should be enabled in all used regions. An additional security control that may help in this area would be to auto-enable services, such as *Security Hub*, for newly created accounts under the organization.

---

<sup>89</sup> <https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html>

<sup>90</sup> <https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-benefits.html>

<sup>91</sup> <https://docs.aws.amazon.com/maciek/latest/user/monitoring-s3.html>

<sup>92</sup> <https://docs.aws.amazon.com/inspector/latest/user/what-is-inspector.html>

## CIR-01-016 WP6: Insecure Default Settings (Low)

AWS provides default configurations to prevent accidental resource exposure if proper settings are missed. In dynamic infrastructures managed by CI/CD pipelines, relying on secure defaults is beneficial due to the short time between committing changes and live deployment. This rapid pace often leaves little time to review options, especially defaults, which are often assumed secure.

### Affected Resources:

AWS Account 384647892761

### Issue 1: Account-level S3 Public Access not blocked by default

AWS accounts do not restrict public access for S3 buckets<sup>93</sup>. Without such settings, administrators or users creating S3 buckets in the account can modify the visibility and accidentally expose a bucket publicly.

This can be confirmed by reviewing the S3 settings like so:

1. Open the *AWS Management Console*
2. Navigate to the following S3 section URL.

#### PoC URL:

<https://eu-central-1.console.aws.amazon.com/s3/settings?region=eu-central-1>

3. Review the *Block Public Access settings for this account* option.

### Block Public Access settings for this account Info

Use Amazon S3 Block public access settings to control the settings that allow public access to your data.

---

#### Block Public Access settings for this account

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies or all. In order to ensure security for all current and future buckets and access points, AWS recommends that you turn on Block all public access, but before applying any of these settings to your buckets or objects, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access**

⚠ Off

- Block public access to buckets and objects granted through *new* access control lists (ACLs)
  - ⚠ Off
- Block public access to buckets and objects granted through *any* access control lists (ACLs)
  - ⚠ Off
- Block public access to buckets and objects granted through *new* public bucket or access point policies
  - ⚠ Off
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
  - ⚠ Off

<sup>93</sup> <https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-control-block-public-access.html>

Fig.: Account-level S3 Block Public Access

## Issue 2: Default EBS encryption is disabled and public access is not blocked.

Default EBS encryption<sup>94</sup> was found to be disabled, for the account and used regions, allowing new EBS volumes and snapshots to be created unencrypted.

This can be confirmed by reviewing the EBS settings inside EC2 section settings like so:

1. Open the *AWS Management Console*
2. Navigate to the following *AWS EC2 EBS* section URL.

**PoC URL:**

<https://eu-central-1.console.aws.amazon.com/ec2/home?region=eu-central-1#Settings:tab=dataProtectionAndSecurity>

3. Review the *Always encrypt new EBS volumes* option.

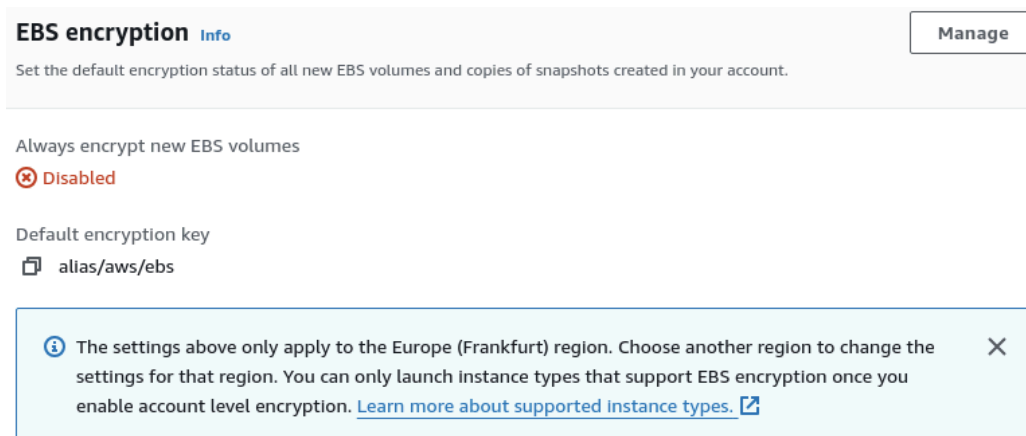


Fig.: Default EBS encryption

The EBS Public Access option fails to prevent accidental snapshot exposure. This can be confirmed by reviewing the EBS inside EC2 section settings from the *Block public access for EBS snapshots* option:

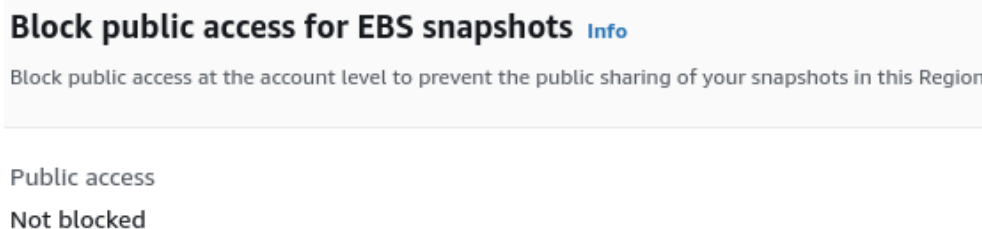


Fig.: EBS Snapshots block public access

<sup>94</sup> <https://docs.aws.amazon.com/ebs/latest/userguide/work-with-ebs-encr.html#encryption-by-default>

### Issue 3: IMDSv2 metadata endpoint not enabled by default

The IMDSv2<sup>95</sup> metadata service provides access to environment-specific information, including IAM role tokens, making it a valuable target for attackers. IMDSv2 requires multiple requests, making it harder to exploit with basic SSRF attacks. Without IMDSv2 enabled by default, the less secure v1 is accessible, making it easier for attackers to reach this service and obtain AWS IAM tokens. Even though the instances created in the environment do have IMDSv2 enabled in infrastructure code, it is still worth enabling it by default for the whole AWS account.

This can be confirmed by reviewing the IMDS settings inside the EC2 section like so:

1. Open the *AWS Management Console*
2. Navigate to the following *AWS Data Protection* section URL.

**PoC URL:**

<https://eu-central-1.console.aws.amazon.com/ec2/home?region=eu-central-1#ModifyInstanceMetadataDefaults>:

3. Review the *IMDS defaults* option.

#### IMDS defaults [Info](#)

Set the IMDS defaults at the account level for new instance launches in this Region.

Instance metadata service

No preference

Metadata version

No preference

Access to tags in metadata

No preference

Metadata response hop limit

No preference

Fig.: Default IMDS settings

<sup>95</sup> <https://aws.amazon.com/blogs/security/get-the-full-benefits-of-imdsv2-...->

Stricter defaults should be enabled in AWS environments to prevent unencrypted resources and public exposure. IMDSv2 should be the only metadata service enabled by default, rather than selectively enabled for defined instances.

## CIR-01-017 WP6: Lack of KMS Encryption & Hardening for S3 Buckets (*Medium*)

Sensitive data is stored in S3 buckets with default encryption. Attackers with read access could exploit this to download sensitive data, including terraform states potentially containing tokens and credentials.

### Affected Resources:

*AWS Account 384647892761*

This issue can be confirmed navigating to the sample S3 bucket view on the *AWS Management Console*:

### URL:

<https://eu-central-1.console.aws.amazon.com/s3/buckets/keanu-unready-ss0-terraform-state?region=eu-central-1&bucketType=general&tab=properties>

Selecting the *keanu-unready-ss0-terraform-state* bucket properties reveals a lack of hardening using a default AWS managed encryption key instead of an AWS-KMS key:

### Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

### Encryption type [Info](#)

Server-side encryption with Amazon S3 managed keys (SSE-S3)

*Fig.: Default SSE-S3 encryption*

The only bucket using Server-side encryption with AWS KMS is *gp-unreadyim-prod-synapse-media*, making the setting inconsistent. Both ALB logs, the config bucket, and the bucket containing Terraform state should be as protected as the media bucket.

MFA delete was found disabled for all buckets, allowing attackers, in case of compromise, to delete current and previous versions of sensitive files:

Multi-factor authentication (MFA) delete

An additional layer of security that requires multi-fa

[Learn more](#) 

Disabled


*Fig.: MFA delete option disabled*

Bucket versioning was found to be set inconsistently for existing buckets:

Bucket Versioning	Buckets
Disabled	<i>gp-unreadyim-prod-synapse-media</i>
Suspended	<i>gp-unreadyim-prod-main-config</i> (empty bucket)
Enabled	<i>keanu-unready-ss0-terraform-state</i> <i>gp-unreadyim-prod-alb-access-logs.</i>

**Example:**

**Bucket Versioning**

Versioning is a means of keeping multiple variants of an object in the same bucket from both unintended user actions and application failures. [Learn more](#) 

Bucket Versioning

Disabled

*Fig.: Versioning suspended for gp-unreadyim-prod-synapse-media bucket*

It is recommended to implement the *AWS Key Management Service (AWS KMS)*<sup>96</sup> for bucket encryption, as it allows for more fine-tuned settings and better access control. This is particularly important for the S3 buckets storing sensitive data, such as terraform state files. Tight access control should then be defined for AWS KMS keys, cloud-managed or customer-managed depending on business requirements, to prevent unauthorized access to files. Additionally, for better control over the data it is recommended to enable MFA delete as well as versioning to protect against unauthorized data deletion, especially for the media bucket which currently has the setting disabled.

<sup>96</sup> <https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html>



## CIR-01-018 WP6: Insufficient Network Restrictions (Low)

It was found that the cloud environment insufficiently implements *network-level* restrictions. In fact, *outbound traffic* in most of the *Security Groups* is unrestricted and *Network Access Control Lists* are set to default permissive values. When implemented correctly, both Security Groups and Network ACLs serve as valuable network-level isolation mechanisms. These weaknesses might allow malicious attackers to attack cloud resources from the Internet in case some resources are accidentally exposed, and/or move laterally after an initial compromise.

### Affected Resource:

AWS Account 384647892761

### Issue 1: Unused and unmanaged launch-wizard security group:

The following security groups have a default launch-wizard-\* name, indicating they were created ad-hoc rather than managed through an infrastructure-as-code pipeline, or they are default groups automatically created during AWS account setup.

Security Group in *eu-central-1* in 384647892761:

- *launch-wizard-1 (sg-0a59d1256dcd8f633)*
- *default (sg-0ec113906022899fe)*

The unused default security group allows unrestricted traffic between instances with this group assigned.

### Issue 2: Unrestricted outbound traffic in security groups:

All security groups, except for *gp-unreadyim-prod-vpc-default*, including those assigned to EC2 instances and RDS, do not restrict outbound traffic. This lack of restriction allows attackers, if they gain a foothold on the virtual machine, to easily initiate reverse outbound connections.

Security Groups in *eu-central-1* in 384647892761:

- *sg-0c00dd6a4bc278cd3*
- *sg-0a59d1256dcd8f633*
- *sg-05b35561b727a565d*
- *sg-0ec113906022899fe*
- *sg-0f7ec7d3c5c80577a*
- *sg-0c00dd6a4bc278cd3*

**Command:**

```
aws ec2 describe-security-groups
```

**Output:**

```
[...]{
[
  {
    "IpProtocol": "-1",
    "IpRanges": [
      {
        "CidrIp": "0.0.0.0/0",
        "Description": "Allow all egress"
      }
    ],
    "Ipv6Ranges": [
      {
        "CidrIpv6": "::/0",
        "Description": "Allow all egress"
      }
    ],
    "PrefixListIds": [],
    "UserIdGroupPairs": []
  }
]
[...]
```

**Issue 3: Unrestricted Network ACLs**

The lack of Network ACL configuration can be confirmed as follows:

1. Open the *AWS Management Console*
1. Navigate to the following *AWS VPC* section URL.

**PoC URL:**

<https://eu-central-1.console.aws.amazon.com/vpcconsole/home?region=eu-central-1#acls>:

2. Select some Network ACL.
3. Review *Inbound* or *Outbound* rules.

## acl-05c165b6b6bf5ad73 / gp-unreadyim-prod-vpc-subnet-ec2-private

Details	Inbound rules	Outbound rules	Subnet associations	Tags
<b>Inbound rules (2)</b>				
<input type="text" value="Filter inbound rules"/>				
Port range	Source	Allow/Deny		
All	0.0.0.0/0	✔ Allow		
All	0.0.0.0/0	✘ Deny		

Fig.: Unrestricted Network ACL

Fine-grained network restrictions should be applied using Network ACLs, Firewalls, and Security Groups where appropriate. The development team should carefully review the architecture and only permit the minimum necessary connections for the solution to function. Network-level isolation should be enforced in cloud environments just as it is for regular servers, including proper host-based firewall rules. Any identified unused or unmanaged resources should be removed from the environment.

### CIR-01-019 WP6: ELB Hardening Recommendations (Low)

The AWS infrastructure makes use of an *Elastic Load Balancer* (ELB) to expose services. It was found that minor hardening improvements can be applied to improve its configuration. Specifically, the load balancer does not remove invalid headers, fails to leverage the AWS WAF, and has no or an insufficient logging configuration.

#### Affected Resources:

AWS Account 384647892761

#### Issue: Missing Drop Invalid Headers and WAF<sup>97</sup>

This can be confirmed by reviewing the ELB attributes inside EC2 ELB section settings:

1. Open the *AWS Management Console*
2. Navigate to the following *AWS EC2 ELB* section URL.

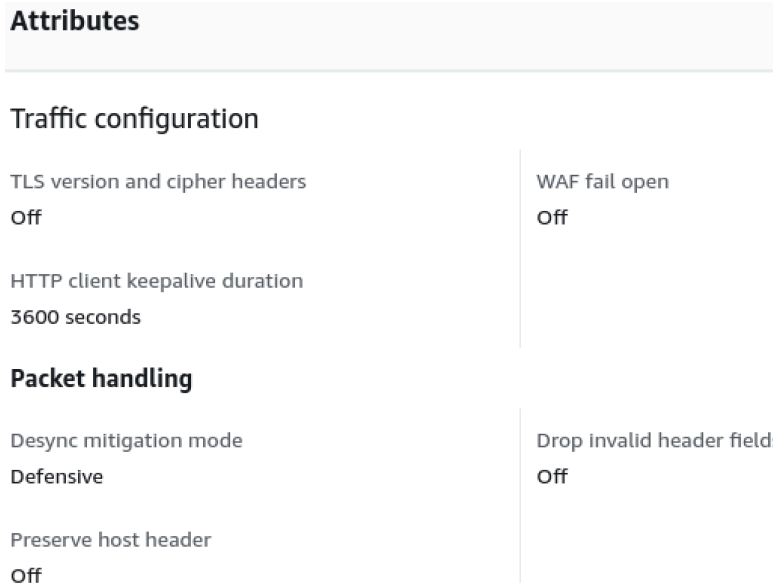
#### PoC URL:

<https://eu-central-1.console.aws.amazon.com/ec2/home?region=eu-central-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:eu-central-1:3846478>

<sup>97</sup> <https://docs.aws.amazon.com/elasticloadbalancing/latest/.../application-load-balancers.html#...-waf>

[92761:loadbalancer/app/gp-unreadyim-prod-alb/3acdb5d33ab6415b:tab=attributes](https://92761:loadbalancer/app/gp-unreadyim-prod-alb/3acdb5d33ab6415b:tab=attributes)

3. Switch to *Attributes* tab.



The screenshot shows the 'Attributes' tab of an ELB configuration. It is divided into two main sections: 'Traffic configuration' and 'Packet handling'. Under 'Traffic configuration', there are three settings: 'TLS version and cipher headers' (Off), 'HTTP client keepalive duration' (3600 seconds), and 'WAF fail open' (Off). Under 'Packet handling', there are three settings: 'Desync mitigation mode' (Defensive), 'Drop invalid header fields' (Off), and 'Preserve host header' (Off).

Section	Setting	Value
Traffic configuration	TLS version and cipher headers	Off
	HTTP client keepalive duration	3600 seconds
	WAF fail open	Off
Packet handling	Desync mitigation mode	Defensive
	Drop invalid header fields	Off
	Preserve host header	Off

*Fig.: ELB attributes*

4. Review the *Drop invalid header fields* option.
5. Review *Desync mitigation mode*.

It is recommended to enable the *Drop invalid headers* option<sup>98</sup>, *strictest desync mitigation mode*<sup>99</sup> and consider deployment of the AWS WAF<sup>100</sup>. If more comprehensive insight into ELB-level access logs is needed, logs should be enabled. Please note that implementing the *Drop invalid headers* and *AWS WAF* options will provide some protection against *HTTP Smuggling* and other web-based attacks.

<sup>98</sup> [https://docs.aws.amazon.com/config/latest/developerguide/alb-http-drop-invalid-header\(...\).html](https://docs.aws.amazon.com/config/latest/developerguide/alb-http-drop-invalid-header(...).html)

<sup>99</sup> <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/config-desync-mitigation-mode.html>

<sup>100</sup> <https://docs.aws.amazon.com/waf/>

## CIR-01-020 WP6: Insufficient AWS Logging & Monitoring (*Medium*)

It was found that the environment implements insufficient logging and monitoring for compromise detection. Only basic logs are collected, primarily aiding the operations team in identifying service outages or significant resource consumption, rather than events indicating a compromise.

### Affected Resources:

AWS Account 384647892761

### Issue 1: No security-oriented *CloudWatch Alarms*

Multiple alarms are defined by the environment, but none are associated with actions that can be performed by an attacker in post-exploitation phases for early compromise detection. Defining appropriate alarms in the event crucial environment settings like security groups, VPCs, IAM, or EC2 instances are modified can help the security team identify the threat early in the kill chain<sup>101</sup>.

### Command:

```
aws cloudwatch describe-alarms
```

### Output:

```
[  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-anomalousConnectionCount",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-highCPUUtilization",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-highDiskQueueDepth",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-highSwapUsage",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-lowCPUCreditBalance",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-lowEBSBurstBalance",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-lowFreeStorageSpace",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-lowFreeableMemory",  
  "gp-unreadyim-prod-db-rds-db-KVGJ70VJHMQHPGCKNFFRZRXI04-maximumUsedTransactionIDs",  
  "gp-unreadyim-prod-ingress-client-alb-5xx-count-high",  
  "gp-unreadyim-prod-ingress-client-alb-elb-5xx-count-high",  
  "gp-unreadyim-prod-ingress-client-alb-target-response-high",  
  "gp-unreadyim-prod-ingress-federation-alb-5xx-count-high",  
  "gp-unreadyim-prod-ingress-federation-alb-elb-5xx-count-high",  
  "gp-unreadyim-prod-ingress-federation-alb-target-response-high"  
]
```

<sup>101</sup> <https://docs.aws.amazon.com/whitepapers/latest/.../what-is-an-intrusion-method.html>

## Issue 2: No VPC flow logs defined for the main VPC

No VPC flow logs are defined for the main VPC (*gp-unreadyim-prod-vpc*) where all resources are configured. The second VPC created by AWS ControlTower (*aws-controltower-VPC*) has flow logs enabled, but it is not in use.

This can be confirmed by reviewing the VPC flow logs:

1. Open the *AWS Management Console*
2. Navigate to the *VPC Settings* and select a VPC to check.

**PoC URL:**

<https://eu-central-1.console.aws.amazon.com/vpcconsole/home?region=eu-central-1#vpcs:vpclid=vpc-0e9bd280da4c11165>

3. Review the *Flow Logs* tab.

The following command confirms there are no flow logs defined in the regions for the AWS accounts provided during this assignment:

**Command:**

```
aws ec2 describe-flow-logs --filter "Name=resource-id,Values=vpc-0e9bd280da4c11165"
```

**Output:**

```
{ "FlowLogs": [] }
```

## Issue 3: No CloudWatch Log Groups Exports defined

The environment does not define any exports for log groups and has short retention policies for all defined log groups. Without a proper logs archive it might be difficult to perform forensic analysis in case of a compromise. The retention policy for each log group in days can be confirmed using the following command:

**Command:**

```
aws logs describe-log-groups
```

**Output:**

```
14 /aws/lambda/aws-controltower-NotificationForwarder
14 /aws/lambda/us-east-1.gp-unr-test
90 /aws/rds/instance/gp-unreadyim-prod-db/postgresql
90 /aws/rds/instance/gp-unreadyim-prod-db/upgrade
30 /gp/unreadyim/prod/main
30 RDSOSMetrics
90 StackSet-AWSControlTowerBP-VPC-ACCOUNT-FACTORY-V1-[...]
```

Lack of S3 bucket logs export can be confirmed using the following command:

**Command:**

```
aws logs describe-export-tasks
```

**Output:**

```
{ "exportTasks": [] }
```

Logging should be improved for critical resources like S3 buckets<sup>102</sup> and VPC flow logs<sup>103</sup>. Once done, CloudWatch alerts should be implemented for security-related events based on logs tailored to the environment<sup>104</sup>. These should be reviewed and security-oriented AWS native services utilized to detect anomalies.

All logging and monitoring settings should be adjusted based on the threat model, compliance requirements, and data volume. Excessively verbose logs may increase infrastructure costs significantly. Lack of proper logging and monitoring decreases the chances of successful threat detection and analysis during a breach. It is advisable to review and improve logging and monitoring configuration in the context of a potential incident response case, rather than just regular daily operations<sup>105</sup>.

## CIR-01-023 WP5: Missing SSH MFA & Auth Hardening (Low)

Whitebox tests revealed that several SSH authentication and authorization settings use default values, making the operating system potentially vulnerable to brute force or resource exhaustion attacks.

**Affected Host:**

```
gp-unreadyim-prod (10.30.0.238)
```

**Issue 1: Missing MFA for SSH Access**

The reference host is missing *Multi Factor Authentication (MFA)* for SSH access.

It is recommended to implement MFA for SSH access. A possible way to accomplish this is by installing and configuring the *google-authenticator*<sup>106</sup> package.

---

<sup>102</sup> <https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-incident-response.html>

<sup>103</sup> <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html>

<sup>104</sup> <https://aws.amazon.com/blogs/mt/use-aws-cloudwatch-...-cis-aws-foundations-benchmark-controls/>

<sup>105</sup> <https://docs.aws.amazon.com/whitepapers/.../aws-security-incident-response...html>

<sup>106</sup> <https://ubuntu.com/tutorials/configure-ssh-2fa>

## Issue 2: *sudoers* Config allows root-level functions without a password

The *gp-unreadyim-prod* server employs a configuration that requires no password for running commands with root privileges via “*superuser do*” (*sudo*) for *admin* and *ssm-user* user accounts.

### Affected Files:

```
/etc/sudoers.d/admin  
/etc/sudoers.d/ssm-agent-users
```

### Command:

```
sudo grep -R --color "NOPASSWD" /etc/sudoers.d/
```

### Output:

```
/etc/sudoers.d/90-cloud-init-users: admin ALL=(ALL) NOPASSWD:ALL  
/etc/sudoers.d/ssm-agent-users: ssm-user ALL=(ALL) NOPASSWD:ALL
```

It is recommended to replace the *NOPASSWD* option with *ALL* to prevent automated processes from elevating privileges. If necessary, limit *NOPASSWD* to specific commands and parameters required by the user for their tasks.

One of the following actions can be taken to harden the *sudo* configuration:

- Replace the *NOPASSWD* option with *ALL* (more restrictive approach):

#### Proposed Fix:

```
admin ALL=(ALL:ALL) ALL
```

- Limit commands and/or parameters available being no password protected from running by the *ssm-user* user (less restrictive approach):

#### Proposed Fix:

```
ssm-user ALL=NOPASSWD: \  
/bin/chown -R
```



## CIR-01-024 WP5: Vulnerable Nginx in Use (Low)

White box tests on the Circulo server revealed that the *gp-unreadyim-prod* host is running *Nginx 1.22.1-9*, which is vulnerable to an over-read worker memory flaw that can cause the worker process to terminate<sup>107</sup>. This version of Nginx is built with the *http\_mp4\_module*, making it exploitable under specific conditions. For the vulnerability to be triggered, the *mp4* directive must be present in the Nginx configuration. This issue can be confirmed as follows:

### Affected Host:

gp-unreadyim-prod (10.30.0.238)

### Command:

```
apt list nginx
```

### Output:

```
nginx/stable,now 1.22.1-9 amd64 [installed]
```

### Command:

```
nginx -V
```

### Output:

```
nginx version: nginx/1.22.1
built with OpenSSL 3.0.8 7 Feb 2023 (running with OpenSSL 3.0.13 30 Jan 2024)
TLS SNI support enabled
configure arguments: --with-cc-opt='-g -O2
-fprofile-prefix-map=/build/nginx-AoTv4W/nginx-1.22.1=. -fstack-protector-strong -Wformat
-Werror=format-security -fPIC -Wdate-time -D_FORTIFY_SOURCE=2'
--with-ld-opt='-Wl,-z,relro -Wl,-z,now -fPIC' --prefix=/usr/share/nginx
--conf-path=/etc/nginx/nginx.conf --http-log-path=/var/log/nginx/access.log
--error-log-path=stderr --lock-path=/var/lock/nginx.lock --pid-path=/run/nginx.pid
--modules-path=/usr/lib/nginx/modules --http-client-body-temp-path=/var/lib/nginx/body
--http-fastcgi-temp-path=/var/lib/nginx/fastcgi
--http-proxy-temp-path=/var/lib/nginx/proxy --http-scgi-temp-path=/var/lib/nginx/scgi
--http-uwsgi-temp-path=/var/lib/nginx/uwsgi --with-compat --with-debug --with-pcre-jit
--with-http_ssl_module --with-http_stub_status_module --with-http_realip_module
--with-http_auth_request_module --with-http_v2_module --with-http_dav_module
--with-http_slice_module --with-threads --with-http_addition_module
--with-http_flv_module --with-http_gunzip_module --with-http_gzip_static_module
--with-http_mp4_module --with-http_random_index_module --with-http_secure_link_module
--with-http_sub_module --with-mail_ssl_module --with-stream_ssl_module
--with-stream_ssl_preread_module --with-stream_realip_module
--with-http_geoip_module=dynamic --with-http_image_filter_module=dynamic
--with-http_perl_module=dynamic --with-http_xslt_module=dynamic --with-mail=dynamic
--with-stream=dynamic --with-stream_geoip_module=dynamic
```

<sup>107</sup> <https://security-tracker.debian.org/tracker/CVE-2024-7347>

It is recommended to upgrade the *Nginx* server to the latest stable version.

## CIR-01-025 WP5: Insecure Configuration of Third-party Software in Use (*Low*)

Whitebox tests revealed that Circulo staff use Tailscale, a third-party VPN service, to access the *gp-unreadyim-prod* host. This approach bypasses AWS security controls, logging mechanisms, and alerts, reducing visibility into server activity and potentially introducing security risks. These risks are acknowledged in [WP2: Circulo Lightweight Threat Model Review](#).

### Affected Host:

gp-unreadyim-prod (10.30.0.238)

### Issue 1: Potentially insecure devices used to connect to Circulo environment

Tailscale allows Circulo staff to connect from various devices, including Apple iPads and Macs, to the Circulo server. While these devices must first be approved by the administrator, their use in a secure environment poses potential security risks, such as outdated or unpatched operating systems, malware, or vulnerabilities stemming from untrusted devices.

### Command:

```
tailscale status
```

### Output:

```
100.64.44.146      gp-unreadyim-prod-main  gp-unreadyim-prod-main.sable-tortoise.ts.net
linux -
100.89.180.25     abel-workstation        abel@      linux  offline
100.108.187.142  gp-monitor-prod         tagged-devices linux  active; relay "lhr", tx
115145059864 rx 13779314332
100.107.56.21    iains-mac-mini-2       irl@       macOS  offline
100.77.16.85     ipad                    abel@      iOS    offline
100.72.216.71    sigauth-unready-im     abel@      linux  -
```

It is recommended to use official connectivity via *AWS VPC peering*<sup>108</sup>. If Tailscale must be used, it should be included in the security assessment and be subject to strict access controls, auditability, and monitoring to prevent unauthorized access through this side-channel connection.

<sup>108</sup> <https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html>

## Issue 2: Prometheus metrics exposed on Tailscale network via HomeServer

The current configuration of the *Synapse Matrix HomeServer* allows internal attackers to access data from the Prometheus metrics endpoint, which is exposed within the Tailscale network. This Tailscale network, however, is only accessible from within the Circulo platform server and not from external sources.

### Command:

```
curl -v --silent 100.64.44.146:9100/metrics 2>&1 | grep -v ^#
```

### Output:

```
apt_autoremove_pending 0
apt_package_cache_timestamp_seconds 0
apt_upgrades_held{arch="all",origin="Debian:bookworm-security/stable-security"} 1
apt_upgrades_held{arch="all",origin="Debian:bookworm/stable"} 5
apt_upgrades_held{arch="all",origin="Debian:trixie/testing"} 1
apt_upgrades_held{arch="amd64",origin=":stable/"} 1
apt_upgrades_held{arch="amd64",origin="Debian:bookworm-security/stable-security"} 10
apt_upgrades_held{arch="amd64",origin="Debian:bookworm-security/stable-security,Debian:bookworm/stable"} 2
apt_upgrades_held{arch="amd64",origin="Debian:bookworm/stable"} 22
apt_upgrades_held{arch="amd64",origin="Debian:trixie/testing"} 2
apt_upgrades_held{arch="amd64",origin="Tailscale:bookworm/"} 1
go_gc_duration_seconds{quantile="0"} 2.8614e-05
go_gc_duration_seconds{quantile="0.25"} 3.2833e-05
go_gc_duration_seconds{quantile="0.5"} 3.5398e-05
go_gc_duration_seconds{quantile="0.75"} 4.7765e-05
go_gc_duration_seconds{quantile="1"} 0.00077025
go_gc_duration_seconds_sum 70.679403947
go_gc_duration_seconds_count 1.075502e+06
go_goroutines 8
go_info{version="go1.19.8"} 1
go_memstats_alloc_bytes 2.775208e+06
go_memstats_alloc_bytes_total 2.061351386896e+12
[...]
```

If this is not the intended behavior, it is recommended to review the Synapse configuration<sup>109</sup> materials to prevent unintentional endpoint exposure.

## Issue 3: Synapse HomeServer misconfiguration

The Synapse HomeServer has the *enable\_registration* option enabled, allowing new user account creation, which could be exploited by bots. Additionally, the *enable\_registration\_captcha* option is disabled. Enabling this feature would help prevent bot registrations by requiring *CAPTCHA* verification. Furthermore, the

<sup>109</sup> [https://element-hq.github.io/synapse/latest/usage/configuration/config\\_documentation.html](https://element-hq.github.io/synapse/latest/usage/configuration/config_documentation.html)

*enable\_registration\_without\_verification* option is active, allowing users to register accounts without any email or CAPTCHA verification, increasing the risk of abuse by unauthorized actors.

**Affected File:**

`/var/lib/matrix/synapse/config/homeserver.yaml`

**Affected Contents:**

```
enable_registration: true
enable_registration_captcha: false
enable_registration_without_verification: true
```

It is recommended to set the *enable\_registration\_captcha* enabled and disable the *enable\_registration\_without\_verification* option once the *enable\_registration* option is enabled. Then follow the instructions about Captcha configuration<sup>110</sup>.

## CIR-01-026 WP5: Boot Loader Password Not Set (Low)

No boot loader password is set on the backend *gp-unreadyim-prod* server (10.30.0.238). This allows unauthorized users with physical access to the server to set command line boot parameters, potentially compromising system security.

**Affected File:**

`/boot/grub/grub.cfg`

**Command:**

```
sudo grep "password" /boot/grub/grub.cfg
```

**Output:**

(empty)

It is recommended to create an encrypted password by using the *grub-mkpasswd-pbkdf2*<sup>111</sup> command, then update the `/boot/grub/grub.cfg` file to include this password. After updating, *update-grub* ought to be run to apply the changes.

<sup>110</sup> [https://element-hq.github.io/synapse/latest/CAPTCHA\\_SETUP.html](https://element-hq.github.io/synapse/latest/CAPTCHA_SETUP.html)

<sup>111</sup> <https://manpages.ubuntu.com/manpages/focal/en/man1/grub-mkpasswd-pbkdf2.1.html>

## CIR-01-027 WP5: Insufficient Logging and Monitoring (*Medium*)

The server does not follow standard logging and monitoring practices, making data breach detection difficult. Despite prioritizing privacy, the lack of local security logging rules and centralized logging makes the environment prone to undetected attacks.

### Affected Host:

gp-unreadyim-prod (10.30.0.238)

### Command:

```
dpkg-query -W -f='${binary:Package}\t${Status}\t${db:Status-Status}\n' auditd  
audispd-plugins
```

### Output:

```
dpkg-query: no packages found matching auditd  
dpkg-query: no packages found matching audispd-plugins
```

It is recommended to configure local auditd according to *CIS Debian* guidelines<sup>112</sup> and use rule sets from reputable threat detection sources<sup>113</sup>. Logs should be forwarded to an external server to prevent tampering during potential breaches. In a privacy-centric setup, consider implementing a minimal self-hosted logging solution, such as *ElasticStack*<sup>114</sup>, integrated with *Wazuh XDR*<sup>115</sup>, to enhance threat detection capabilities.

Once sufficient logging and monitoring are in place, all security tools may be fine-tuned based on threat modeling and MITRE ATT&CK scenarios<sup>116</sup>.

## CIR-01-028 WP5: Lack of Full Disk Encryption (*Medium*)

The server lacks full disk encryption, risking sensitive data extraction via physical access to hard drives. Attackers may dump server content or recover data from damaged hardware.

### Affected Host:

gp-unreadyim-prod (10.30.0.238)

### Command (listing no crypt type block devices):

```
lsblk /dev/nvme1n1 -o NAME,KNAME,FSTYPE,TYPE,MOUNTPOINT,SIZE
```

<sup>112</sup> [https://www.cisecurity.org/benchmark/debian\\_linux](https://www.cisecurity.org/benchmark/debian_linux)

<sup>113</sup> <https://github.com/Neo23x0/auditd>

<sup>114</sup> <https://www.elastic.co/elastic-stack>

<sup>115</sup> <https://wazuh.com/>

<sup>116</sup> <https://attack.mitre.org/>

**Output:**

NAME	KNAME	FSTYPE	TYPE	MOUNTPPOINT	SIZE
nvme1n1	nvme1n1	ext4	disk	/var/lib/matrix	100G

It is recommended to enable full disk encryption<sup>117</sup> using a configuration which supports automatic decryption<sup>118</sup> after each reboot (e.g. using *clevis*<sup>119</sup>).

**CIR-01-029 WP5: Insecure Network Stack Configuration (Low)**

During the assessment, the `log_martians` setting was not enabled on the `gp-unreadyim-prod` host. Additionally, `send_redirects`, `secure_redirects`, and `accept_redirects` were found to be enabled. Enabling `log_martians` allows administrators to log and investigate potentially spoofed packets. The `send_redirects` setting can be exploited by attackers to send invalid ICMP redirects, potentially corrupting routing and directing traffic to malicious systems. `secure_redirects` and `accept_redirects`, when enabled, allow ICMP redirect packet processing, which is unnecessary in well-configured networks and poses a security risk.

**Affected File:**

/etc/sysctl.conf

**Affected Host:**

gp-unreadyim-prod (10.30.0.238)

**Commands:**

```
sysctl net.ipv4.conf.all.log_martians
sysctl net.ipv4.conf.default.log_martians
sysctl net.ipv4.conf.ens5.log_martians

sysctl net.ipv4.conf.all.send_redirects
sysctl net.ipv4.conf.default.send_redirects
sysctl net.ipv4.conf.ens5.send_redirects

sysctl net.ipv4.conf.all.secure_redirects
sysctl net.ipv4.conf.default.secure_redirects
sysctl net.ipv4.conf.ens5.secure_redirects

sysctl net.ipv4.conf.all.accept_redirects
sysctl net.ipv4.conf.default.accept_redirects
sysctl net.ipv4.conf.ens5.accept_redirects
```

**Output:**

```
net.ipv4.conf.all.log_martians = 0
```

<sup>117</sup> <https://ubuntu.com/core/docs/full-disk-encryption>

<sup>118</sup> [https://wiki.archlinux.org/title/Trusted\\_Platform\\_Module#Data-at-rest\\_encryption\\_with\\_LUKS](https://wiki.archlinux.org/title/Trusted_Platform_Module#Data-at-rest_encryption_with_LUKS)

<sup>119</sup> <https://github.com/latchset/clevis>

```
net.ipv4.conf.default.log_martians = 0
net.ipv4.conf.ens5.log_martians = 0

net.ipv4.conf.all.send_redirects = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.ens5.send_redirects = 1

net.ipv4.conf.all.secure_redirects = 1
net.ipv4.conf.default.secure_redirects = 1
net.ipv4.conf.ens5.secure_redirects = 1

net.ipv4.conf.all.accept_redirects = 1
net.ipv4.conf.default.accept_redirects = 1
net.ipv4.conf.ens5.accept_redirects = 1
```

## Proposed Fix:

To set the runtime status of the aforementioned kernel parameters, it is recommended to run the following commands:

```
sysctl -w net.ipv4.conf.all.log_martians=1
sysctl -w net.ipv4.conf.default.log_martians=1
sysctl -w net.ipv4.conf.ens5.log_martians=1

sysctl -w net.ipv4.conf.all.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
sysctl -w net.ipv4.conf.ens5.send_redirects=0

sysctl -w net.ipv4.conf.all.secure_redirects=0
sysctl -w net.ipv4.conf.default.secure_redirects=0
sysctl -w net.ipv4.conf.ens5.secure_redirects=0

sysctl -w net.ipv4.conf.all.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.ens5.accept_redirects=0
```

## WP2: Circulo Lightweight Threat Model Review

### Introduction

Circulo is a safety app designed for individuals facing environmental threats, enabling them to share critical information like messages, mental state, and location with a trusted group. Originally developed to support female journalists in Latin America, it focuses on threats during high-risk work in hostile regions where the government may act as an oppressor. Users can establish a trusted circle via the app, which employs the Matrix protocol to secure messaging and location sharing, ensuring confidentiality, integrity, authenticity, and availability for safe communication and emergency aid.

Threat model analysis helps organizations identify potential security threats and vulnerabilities, allowing for effective mitigation strategies before attackers can exploit them, enhancing overall system security and resilience. Lightweight threat modeling simplifies this process by loosely following the *STRIDE*<sup>120</sup> methodology, focusing on system analysis, as performed by 7A Security, using documentation, specifications, source code, and existing threat models, with assistance from client representatives.

This section aims to identify potential security threats and vulnerabilities that could be exploited by adversaries in the form of categorized attack scenarios. It also suggests possible mitigations. The analysis targets mobile applications, infrastructure, and processes described in all resources delivered by the client and available during the engagement.

### Relevant assets and threat actors

The following key assets were identified as significant from a security perspective:

- Circulo user credentials.
- Encrypted messages and location data exchanged via the Circulo app.
- HomeServer configuration containing registration shared secrets, database credentials, and exposed user/admin API endpoints.
- Synapse Admin User credentials granting Admin API access.
- AWS credentials used by the backend team for cloud infrastructure access.
- SSH keys providing server access.
- Tailscale credentials granting access to a mesh VPN used for log collection and server access.
- Release artifacts and CI/CD pipelines containing source code and binaries.

The following threat actors are considered relevant for the analysis:

- Nation-State Attacker

---

<sup>120</sup> <https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats#stride-model>



- Internal Attacker (e.g. inside the infrastructure or Synapse admin)
- LAN Attacker targeting the user and the mobile application
- Malicious Mobile Application
- Physical Attacker threatening Circulo users
- Privileged Physical Attacker (e.g. boss at work, workplace administrator)
- Compromised user in trusted circle

## Attack surface

In threat modeling, the attack surface encompasses all potential entry points an attacker could exploit to compromise a system, including paths and interfaces for accessing, manipulating, or extracting sensitive data, or disrupting application availability. Identifying the attack surface helps pinpoint potential vulnerabilities and implement defenses to reduce risks.

By analyzing various threats and attack scenarios, organizations can better understand the techniques that might be used to compromise system security.

### Threat 01: Identity or Data Spoofing

#### Overview

A spoofing attack enables an adversary to impersonate a person or program by exploiting stolen credentials, authentication tokens, or bypassing authentication mechanisms. This can result in unauthorized data access or unintended actions being executed on behalf of the impersonated entity.

#### Countermeasures

Circulo implements several measures to restrict access to credentials on mobile devices and during transit, including application-level PIN, encrypted transport layer, and disallowed backups. However, the solution lacks proper DNS-spoofing protection and may have insufficient MiTM protections.

The reference infrastructure makes use of AWS Organization and recommended practices to prevent credentials theft and identity spoofing<sup>121</sup>.

#### Attack Scenarios

Despite the implementation of multiple countermeasures, room for improvement was identified in the following attack scenarios:

---

<sup>121</sup> <https://docs.keanu.im/backend/>

- Cloning the phone using known exploits or advanced nation-state spyware, when the device is stolen or confiscated, could result in two devices being authorized to query Synapse servers, allowing the actor to spy on or send messages within the circle. Both the user and circle participant devices could be targeted.
- ISP-level DNS spoofing and certificate forging by a nation-state actor to bypass potentially insufficient certificate pinning to MiTM traffic to extract bearer tokens and impersonate a user or participants.
- Work devices with installed MDM solutions may have CA certificates injected by administrators, allowing MiTM attacks on TLS communication.
- Stolen developer credentials could be used to commit unsigned code, planting backdoors due to the lack of enforced Git signatures.

## Recommendation

To improve security, monitoring should be enhanced to detect multiple devices using the same tokens, which could signal a compromise. It is also recommended to verify DNS data using *DNS over HTTPS (DoH)*<sup>122</sup> consistently, regardless of the mobile device or network configuration, to prevent manipulation by attackers.

Synapse should be configured to enforce<sup>123</sup> a single active device<sup>124</sup> per account or extended if necessary. This allows for the detection of stolen credentials and alerts for suspicious activity. Implementing anomaly detection based on IP addresses and device IDs would further mitigate risks.

To reduce supply chain attack risks, enforce multi-factor authentication (MFA), signed code<sup>125</sup>, mandatory code reviews, and multi-staged development and release cycles, aligning with higher SLSA levels.

Users should avoid using the app on corporate devices and be provided with a checklist for actions to take if their account or device is compromised.

---

<sup>122</sup> [https://en.wikipedia.org/wiki/DNS\\_over\\_HTTPS](https://en.wikipedia.org/wiki/DNS_over_HTTPS)

<sup>123</sup> <https://matrix.org/docs/matrix-concepts/end-to-end-encryption/#devices>

<sup>124</sup> <https://github.com/element-hq/synapse/issues/6616>

<sup>125</sup> [https://docs.gitlab.com/ee/user/project/repository/signed\\_commits/](https://docs.gitlab.com/ee/user/project/repository/signed_commits/)

## Threat 02: Data Tampering or Unauthorized Modifications

### Overview

Data tampering involves the unauthorized modification of information. It includes the manipulation of data on disk, in memory, or while in transit. This type of attack can result in the alteration of exchanged data, code changes, unauthorized system access, or even infrastructure compromise.

### Countermeasures and current state

The Matrix.org protocol and encrypted transport layers are used by Circulo to protect data integrity during transmission. Additionally, Matrix.org authentication mechanisms and room roles are employed to enforce authorization. On the backend, the documentation describes an AWS Organization-based setup that uses personal administrative accounts for auditability and robust cloud provider authorization. However, the environment also employs Tailscale services, creating a side channel that potentially bypasses cloud security controls by exposing metrics and administrative interfaces within a private Tailscale network.

### Attack Scenarios

Despite the implementation of multiple countermeasures, room for improvement was identified in the following attack scenarios:

- Data could be modified through the Synapse Admin API if administrative credentials are stolen or if authentication bypass vulnerabilities are discovered.
- Synapse vulnerabilities, such as CVE-2023-42453<sup>126</sup>, could be exploited to manipulate the database or storage system.
- Lack of enforced code reviews and missing code signatures could allow backdoors to be planted in the source code.
- Modifications to externally hosted scripts used during the build process<sup>127,128</sup> could compromise release artifacts or affect multiple environments.

### Recommendation

To mitigate supply chain attacks, reviewing and complying with higher SLSA levels is advised. Multiple security layers should be implemented to enhance visibility and response times, including host-based intrusion detection (e.g., OSSEC<sup>129</sup>), anomaly

---

<sup>126</sup> <https://security.snyk.io/vuln/SNYK-PYTHON-MATRIXSYNAPSE-5920321>

<sup>127</sup> [https://gitlab.com/guardianproject-ops/ansible-collection.../docker\\_host/Makefile?ref\\_type=heads](https://gitlab.com/guardianproject-ops/ansible-collection.../docker_host/Makefile?ref_type=heads)

<sup>128</sup> <https://gitlab.com/-/snippets/1957473>

<sup>129</sup> <https://www.ossec.net/>

detection, logging, and network-based intrusion detection (e.g., *Wazuh*<sup>130</sup>). Regular penetration tests and simulated attacks (*MITRE ATT&CK*<sup>131</sup>) should be conducted to ensure security measures are effective. Additionally, a robust CI/CD pipeline must be set up to detect known vulnerabilities and quickly deploy patches. The Admin API should only be accessible via a monitored internal channel.

## Threat 03: Repudiation Attacks

### Overview

Preventing attacks is as crucial as logging and monitoring. These provide essential insight into system actions and address repudiation threats. Often overlooked, logging and monitoring are key tools for forensic investigators to identify the initial attack vector. In modern security systems, they form the core of anomaly detection. If attackers erase their tracks or the company fails to trace the initial compromise, the same attack route may remain exploitable.

### Countermeasures

The Circulo system uses the Matrix.org protocol, which ensures cryptographic integrity of messages, making it possible to verify message authorship. However, in cases where credentials are leaked or a device is cloned, it becomes harder to differentiate between legitimate and attacker-generated messages.

On the backend, the infrastructure uses a mix of cloud-native and application-level logging to mitigate attribution risks. However, in practice, shared root/admin accounts are used, and proper host-based configurations are lacking. Additionally, a side-channel Tailscale VPN bypasses AWS logging mechanisms, and common security alerts for suspicious activities are not fully implemented.

### Attack Scenarios

The following attack scenarios are relevant and should be addressed with remediation strategies:

- A leaked SSH key for a shared Linux admin user is used via *Tailnet* to access the backend server.
- An attacker joins or compromises a node in *Tailnet*, bypassing AWS logging mechanisms and querying exposed endpoints, making suspicious activity harder to detect.

---

<sup>130</sup> <https://wazuh.com/platform/overview/>

<sup>131</sup> <https://attack.mitre.org/>

## Recommendation

The following measures should be considered to strengthen defenses against the listed attacks:

- Shared accounts should be avoided. Instead, unique accounts should be created for all staff members at both the OS and cloud levels, allowing centralized management for easier credential invalidation and rotation.
- Both hosts and networks should be closely monitored using host-level auditing tools (e.g., properly configured *auditd*) and log monitoring systems with alerting rules to detect suspicious activity.
- No side-channel connections should bypass security mechanisms to access infrastructure servers, log collection endpoints, or any internal systems. All traffic must pass through a single, well-monitored system.

## Threat 04: Information disclosure

### Overview

Information disclosure occurs when sensitive data is accessed by unauthorized entities. This typically happens due to weak security controls, insufficient encryption of data during transmission or storage, or design flaws that can be exploited to extract confidential information.

### Countermeasures

The Matrix protocol provides end-to-end encryption, making it difficult for unprivileged users to access rooms. Standard security measures like PIN protection and mobile application development practices are followed. Matrix rooms are private by default, and the user catalog is not searchable, reducing the risk of data harvesting by regular users.

In the AWS environment, the backend isolates the machine and restricts access to internal ports and protocols. However, security guarantees are compromised by the side-channel configuration using Tailscale.

### Attack Scenarios

The following attack scenarios require attention and would benefit from remediation strategies:

- Data exposure through memory leaks on mobile devices exploited by a threat actor via a malicious app or harvested from a confiscated device.
- Sensitive data, such as RoomID or access tokens, stored in mobile device log files and extracted by attackers using debugging features.

- Sensitive data, like RoomID or access tokens, not masked in backend logs (e.g., CloudWatch or Nginx), and accessed by a low-privileged backend user.
- Leaked RoomID could allow unauthorized users to craft an invitation to join non-public rooms.
- A compromised Synapse admin user could scan the server and extract data from all rooms and users.

## Recommendation

To enhance protection against the mentioned attacks, the following technical solutions are recommended:

- Mobile applications should be tested for data leaks in memory and local logs, with dedicated tests to identify sensitive data artifacts in memory or on disk.
- All sensitive data should be masked or removed from logs.
- Backends should be configured to scan logs regularly for sensitive data leaks and alert staff when detected.
- Procedures for handling sensitive data leaks ought to be established.
- Mechanisms to detect mass data extraction should be implemented to help identify potential compromises or data breaches.

## Threat 05: Temporary or Permanent Denial of Service

### Overview

Denial of service occurs when the system is unable to function as intended, either temporarily or permanently. Ensuring availability is critical for software exchanging sensitive messages. Users, especially those in insecure environments and facing advanced threats, may experience communication disruptions, which in extreme cases could lead to life-threatening situations. The inability to share location or request help must be prevented, and the system should be designed to mitigate these risks.

### Countermeasures

The backend documentation outlines a Cloudflare-based setup to protect against certain attacks, but this does not cover all scenarios. Additionally, the Synapse server in the backend appears to lack full implementation of protection settings.

### Attack Scenarios

The following attack scenarios should be considered relevant for users of Circulo and could benefit from remediation:

- GSM/LTE jammers disrupting mobile connectivity and location services.

- LAN attacks using DNS spoofing or IP blocking to hinder communication with Circulo servers.
- Large-scale traffic blackholing by ISPs using IP/DNS censorship against Circulo.
- Invitation spam on Matrix users, causing performance issues during server synchronization.
- Resource exhaustion due to a lack of anti-bot mechanisms, such as CAPTCHA<sup>132</sup>, leading to flood registrations.
- Delays in sending emergency messages due to the absence of a panic button feature that bypasses the mobile device and app unlock procedures. If the user is under threat and unable to use the regular features fast enough.
- Local malicious applications crashing Circulo by exploiting locally exposed services or endpoints (e.g. Android activities).
- Admin API<sup>133</sup> exploitation by a compromised admin to delete rooms or remove users.
- Physical attacks in low-signal areas prevent victims from sending help messages, with the app potentially losing unsent messages due to inactivity or device reboot after battery conservation attempts.

## Recommendation

To enhance defenses against the identified scenarios, consider the following measures:

- Implement *DNS over HTTPS (DoH)*<sup>134</sup> to prevent tampering, regardless of the mobile OS or network configuration.
- Utilize CloudFlare or a similar service to mitigate censorship and implement health checks to detect censorship attempts.
- Conduct focused mobile app testing targeting local denial-of-service (DoS) cases.
- Develop a panic button feature enabling users to send alerts to a predefined room without unlocking the device or app.
- Implement detection of sudden geolocation or connectivity losses, signaling potential jamming attempts.
- Research and integrate available spam-protection mechanisms on both mobile and Synapse servers<sup>135</sup><sup>136</sup><sup>137</sup>.
- Conduct resource consumption DoS testing, including mass media file uploads or request flooding to generate large log files.
- Implement a message queue and a low-power background service to archive unsent messages and send them when the device regains connectivity or powers back on.

<sup>132</sup> [https://element-hq.github.io/synapse/latest/CAPTCHA\\_SETUP.html](https://element-hq.github.io/synapse/latest/CAPTCHA_SETUP.html)

<sup>133</sup> [https://element-hq.github.io/synapse/latest/admin\\_api/rooms.html#delete-room-api](https://element-hq.github.io/synapse/latest/admin_api/rooms.html#delete-room-api)

<sup>134</sup> [https://en.wikipedia.org/wiki/DNS\\_over\\_HTTPS](https://en.wikipedia.org/wiki/DNS_over_HTTPS)

<sup>135</sup> [https://element-hq.github.io/synapse/latest/CAPTCHA\\_SETUP.html](https://element-hq.github.io/synapse/latest/CAPTCHA_SETUP.html)

<sup>136</sup> [https://matrix-org.github.io/synapse/v1.45/modules/spam\\_checker\\_callbacks.html](https://matrix-org.github.io/synapse/v1.45/modules/spam_checker_callbacks.html)

<sup>137</sup> <https://ubuntu.com/community/communications/matrix/spam>

## Threat 06: Privilege Escalation

### Overview

Privilege escalation occurs when an attacker obtains sufficient access to compromise critical assets in the environment. These threats are highly severe, indicating that most defensive mechanisms have already been bypassed. Common types of privilege escalation attacks include remote code execution, local privilege escalation, and supply chain compromises.

### Countermeasures

The mobile application adheres to good development practices, and OS-level isolation reduces the likelihood of local privilege escalation from other installed applications.

The backend documentation<sup>138</sup> recommends an AWS infrastructure setup utilizing native cloud security mechanisms, making privilege escalation more difficult when correctly implemented. Additionally, the local host configuration employs separate users for different Matrix services and containers for managing workers, further limiting common privilege escalation scenarios.

### Attack Scenarios

The following attack scenarios are relevant to the organization in the context of privilege escalation and would benefit from mitigation:

- A zero-day or one-day vulnerability in Synapse is exploited, allowing the host to be compromised.
- AWS credentials are compromised, leading to privilege escalation and lateral movement within the Circulo AWS account, potentially accessing virtual machines, databases, and S3 buckets storing sensitive data.
- Scripts<sup>139</sup> used in the build process are modified by an attacker to inject a backdoor or compromise the infrastructure.
- Container images or dependencies are infected, resulting in outbound connections to attacker-controlled systems.
- Admin privileges within a room are escalated beyond the original creator control.
- Debugging features or local root access on a mobile device are exploited to gain access to Circulo on a confiscated device.

---

<sup>138</sup> <https://docs.keanu.im/backend/>

<sup>139</sup> <https://gitlab.com/-/snippets/1957473/raw>



## Recommendation

To strengthen defenses against the identified scenarios, the following measures are recommended:

- Pin container images and dependencies to specific versions to prevent unintended updates to compromised binaries.
- Implement a thorough review process for new dependencies before integrating them into the environment.
- Use established security tools in CI/CD pipelines to scan dependencies and source code for known vulnerabilities before every deployment, halting the process if any issues are detected.
- Host deployment scripts internally, applying the same security measures as other organizational code, to avoid supply chain risks from hosted snippets.
- Enhance logging, monitoring, and anomaly detection using cloud-native, host-based intrusion detection, and network monitoring tools.
- Enforce strict access control in AWS, particularly over the main account, and ensure comprehensive monitoring and alerts for any signs of data breaches or malicious activity.
- Regularly perform attack simulations to validate the effectiveness of the security setup.

## WP3: Circulo Supply Chain Implementation

### Introduction and General Analysis

The *8th Annual State of the Software Supply Chain Report*, released in October 2022<sup>140</sup>, revealed a 742% average yearly increase in software supply chain attacks since 2019. Some notable compromise examples include *Okta*<sup>141</sup>, *Github*<sup>142</sup>, *Magento*<sup>143</sup>, *SolarWinds*<sup>144</sup>, and *Codecov*<sup>145</sup>, among many others. To mitigate this concerning trend, Google released an End-to-End Framework for *Supply Chain Integrity* in June 2021<sup>146</sup>, named *Supply-Chain Levels for Software Artifacts (SLSA)*<sup>147</sup>.

This area of the report elaborates on the current state of the supply chain integrity implementation of the Circulo project, as audited against versions 0.1 and 1.0 of the SLSA framework. SLSA assesses the security of software supply chains and aims to provide a consistent way to evaluate the security of software products and their dependencies.

The Circulo project utilizes a public GitLab repository<sup>148</sup> to efficiently manage and distribute source code while ensuring dependencies are well-defined. The scripted build process for debug builds enhances release consistency and speeds up deployment cycles. While the debug build process is fully automated, the actual release process is handled manually on a hardened build machine. This method improves development efficiency but also emphasizes the need for strict security and standardized environments to safeguard the integrity of the build.

While auditing the supply chain implementation, the Circulo project provided several positive impressions that must be acknowledged here:

1. TLS is enforced for network connections, with end-to-end encryption ensuring messaging confidentiality.
2. The server infrastructure uses automated scripted deployment within AWS, streamlining resource provisioning.
3. Dependencies are locked to major versions, with regular manual checks for updates to ensure timely integration.
4. Regular security audits are conducted on both client and server components to maintain adherence to optimal security standards.

<sup>140</sup> <https://www.sonatype.com/press-releases/2022-software-supply-chain-report>

<sup>141</sup> <https://www.okta.com/blog/2022/03/updated-okta-statement-on-lapsus/>

<sup>142</sup> <https://github.blog/2022-04-15-security-alert-stolen-oauth-user-tokens/>

<sup>143</sup> <https://sansec.io/research/rekoobe-fishpig-magento>

<sup>144</sup> <https://www.techtarget.com/searchsecurity/ehandbook/SolarWinds-supply-chain-attack...>

<sup>145</sup> <https://blog.gitguardian.com/codecov-supply-chain-breach/>

<sup>146</sup> <https://security.googleblog.com/2021/06/introducing-slsa-end-to-end-framework.html>

<sup>147</sup> <https://slsa.dev/spec/>

<sup>148</sup> <https://gitlab.com/circuloapp>

## SLSA v1.0 Analysis and Recommendations

SLSA v1.0 defines a set of four levels that describe the maturity of the software supply chain security practices implemented by a software project as follows:

- **Build L0: No guarantees** represent the lack of SLSA<sup>149</sup>.
- **Build L1: Provenance exists**. The package has provenance showing how it was built. This can be used to prevent mistakes but is trivial to bypass or forge<sup>150</sup>.
- **Build L2: Hosted build platform**. Builds run on a hosted platform that generates and signs the provenance<sup>151</sup>.
- **Build L3: Hardened builds**. Builds run on a hardened build platform that offers strong tamper protection<sup>152</sup>.

To produce artifacts with a specific SLSA level, the responsibility is split between the *Build* platform and the *Producer*. Broadly speaking, the *Build* platform must strengthen the security controls to achieve a specific level, while the *Producer* must choose and adopt a *Build* platform capable of achieving a desired SLSA level, implementing security controls as specified by the chosen platform.

The following sections summarize the results of the software supply chain security implementation audit, based on the SLSA v1.0 framework. Green check marks indicate that evidence of SLSA claims was found.

### SLSA v1.0 Producer

A package producer is the organization that owns and releases the software. It might be an open-source project, a company, a team within a company, or even an individual. The producer must select a build platform capable of reaching the desired SLSA Build Level.

In terms of the Producer, the following requirements must be met:

1. Choose an appropriate build platform.
2. Follow a consistent build process.
3. Distribute provenance.

Based on the documentation provided by the Circulo team, 7ASecurity conducted a SLSA v1.0 analysis, with the following results.

<sup>149</sup> <https://slsa.dev/spec/v1.0/levels#build-l0>

<sup>150</sup> <https://slsa.dev/spec/v1.0/levels#build-l1>

<sup>151</sup> <https://slsa.dev/spec/v1.0/levels#build-l2>

<sup>152</sup> <https://slsa.dev/spec/v1.0/levels#build-l3>

## Choose an appropriate build platform.

The Circulo build process for production uses a secure dedicated "build machine" compliant with SLSA L1 by generating build provenance. Achieving higher SLSA levels is restricted due to non-public provenance information. Ensuring reproducible and verifiable builds remains challenging as the environment lacks standardization, making it vulnerable to unintended or malicious modifications. Local machine builds do not generate sufficient provenance for auditing, which is essential for achieving higher SLSA compliance levels.

## Follow a consistent build process

The Circulo team leverages scripts<sup>153</sup> to create Android debug artifacts, ensuring a standardized build process. However, it is unclear whether the same approach applies to production builds, since the "build machine" functions as the build environment. This practice diverges from recommended SLSA framework guidelines.

## Distribute provenance

Distributed provenance requires each entity or system contributing to the build to generate and provide signed, verifiable provenance information. When code is built on a local machine, the information about its origin and history (provenance) may be lacking in completeness, consistency, or accuracy. This results in gaps in the provenance chain, leading to potential security and traceability issues.

The following table shows the results of Circulo according to Producer requirements described in the SLSA v1.0 Framework:

Implementer	Requirement	Degree	L1	L2	L3
Producer	Choose an appropriate build platform		✓	✗	✗
	Follow a consistent build process		✓	✗	✗
	Distribute provenance		✓	✗	✗

<sup>153</sup> [https://gitlab.com/circuloapp/circulo-keanu-android/-/blob/main/gitlab-ci.yml?ref\\_type=heads](https://gitlab.com/circuloapp/circulo-keanu-android/-/blob/main/gitlab-ci.yml?ref_type=heads)

## SLSA v1.0 Build platform

A package build platform is the infrastructure used to transform the software from source to package. The build platform is responsible for providing provenance generation and isolation between builds. In terms of the Circulo build platform, the following can be highlighted:

### Provenance Exists

Circulo builds are designed to align with SLSA L1 on the build machine. However, ensuring the reproducibility and verifiability of builds is challenging due to non-standardized environments that are vulnerable to tampering. Local builds lack provenance for auditing and tracking, essential in SLSA L2+.

### Provenance is Authentic

This requirement mandates validating provenance authenticity via a digital signature from a private key accessible only to the hosted build platform. Since Circulo builds occur on a build machine, not a hosted build platform, this requirement cannot be met.

### Provenance is Unforgeable

This requirement mandates that provenance be resistant to tenant forgery, achievable with a build platform producing Provenance L3, typically a hosting platform. The current Circulo build configuration cannot meet this requirement.

The following table shows the results of Circulo according to the Build platform requirements described in the SLSA v1.0 Framework:

Implementer	Requirement	Degree	L1	L2	L3
Build platform	Provenance generation	Exists	✓	✗	✗
		Authentic		✗	✗
		Unforgeable			✗
	Isolation strength	Hosted			✗
		Isolated			✗

To help enhance the project SLSA level, from L1 to L2, here is a well-structured recommendation for progression:

- **Automate Build Processes:** To ensure consistency and provenance tracking on production releases, utilize automated systems like *GitLab CI/CD*<sup>154</sup> not just for debug builds but across all builds.
- **Provenance & Build Metadata:** Capture build metadata (trigger, commit, dependencies) using *GitLab CI/CD* pipelines<sup>155</sup>. Use tools like *in-toto*<sup>156</sup> for linking code changes to builds for traceability.
- **Code Signing:** Implement code signing for all build artifacts using tools like *Sigstore*<sup>157</sup> to verify integrity and provenance.
- **Secret Management:** Use *GitLab CI/CD* Variables and integrate with tools like *HashiCorp Vault*<sup>158</sup> for secure secret management.
- **Log Retention:** Implement GitLab RBAC to secure and control access to build logs, and enable Audit Events to monitor and track log access and pipeline activities effectively.

## SLSA v0.1 Analysis and Recommendations

SLSA v0.1 defines a set of five levels<sup>159</sup> that describe the maturity of the software supply chain security practices implemented by a software project as follows:

- **L0: No guarantees.** This level represents the lack of any SLSA level.
- **L1:** The build process must be fully scripted/automated and generate provenance.
- **L2:** Requires using version control and a hosted build service that generates authenticated provenance.
- **L3:** The source and build platforms meet specific standards to guarantee the auditability of the source and the integrity of the provenance respectively.
- **L4:** Requires a two-person review of all changes and a hermetic, reproducible build process.

The following sections summarize the results of the software supply chain security implementation audit based on the SLSA v0.1 framework. Green check marks indicate that evidence of the noted requirement was found.

---

<sup>154</sup> <https://docs.gitlab.com/ee/ci/>

<sup>155</sup> <https://docs.gitlab.com/ee/ci/pipelines/>


<sup>156</sup> <https://in-toto.io/>

<sup>157</sup> <https://www.sigstore.dev/>

<sup>158</sup> [https://docs.gitlab.com/ee/ci/secrets/hashicorp\\_vault.html](https://docs.gitlab.com/ee/ci/secrets/hashicorp_vault.html)

<sup>159</sup> <https://slsa.dev/spec/v0.1/levels>

Requirement	L1	L2	L3	L4
Source - Version controlled	✓	✓	✓	✓
Source - Verified history			✓	✓
Source - Retained indefinitely			✓	✓
Source - Two-person reviewed				✗
Build - Scripted build	✓	✗	✗	✗
Build - Build service		✗	✗	✗
Build - Build as code			✗	✗
Build - Ephemeral environment			✗	✗
Build - Isolated			✗	✗
Build - Parameterless				✗
Build - Hermetic				✗
Build - Reproducible				✗
Provenance - Available	✓	✗	✗	✗
Provenance - Authenticated		✗	✗	✗
Provenance - Service generated		✗	✗	✗
Provenance - Non-falsifiable			✗	✗
Provenance - Dependencies complete				✗
Common - Security				✗
Common - Access				✗

Common - Superusers				
---------------------	--	--	--	---

## SLSA v0.1 Conclusion

After evaluating the Circulo software supply chain security practices, it was determined that the project achieves SLSA Level 1, reflecting basic security practices like source code version control and established build processes.

However, gaps prevent advancing to SLSA Levels 2 or 3. The main issue is reliance on build machines rather than a centralized, controlled environment, hindering build provenance generation and verification required for higher SLSA levels.

To reach SLSA Level 2, Circulo should use a hosted build system to generate authenticated provenance for all artifacts, enhancing software supply chain integrity and traceability. *FRSCA*<sup>160</sup> may be leveraged to offer a full pipeline and achieve SLSA level 2, as regular platform users are unable to inject or alter the contents of the provenance it generates. Alternatively, *GitLab CI/CD* is capable of producing SLSA L3 provenance.

In conclusion, while Circulo has made commendable progress in foundational security, investing in a build system infrastructure and automated provenance generation is essential for higher SLSA levels, bolstering supply chain security and resilience against evolving cybersecurity challenges.

---

<sup>160</sup> <https://github.com/buildsec/frsca>



## Conclusion

Despite the number and severity of findings encountered in this exercise, the Circulo solution defended itself well against a broad range of attack vectors. The platform will become increasingly difficult to attack as additional cycles of security testing and subsequent hardening continue.

7ASecurity would like to highlight several positive aspects of Circulo, as observed by the audit team:

- Multiple checks were found to be in place in all Circulo components to enhance resilience against potential exploits and unauthorized access.
- Most libraries and dependencies were found to be up-to-date, demonstrating a commitment to maintaining security hygiene. This approach mitigates vulnerabilities and guarantees a robust defense against potential threats.
- Overall, the Circulo backend components were found to be robust against many traditional web application security attack vectors. For example, no *Command Injection*, *SQL Injection (SQLi)*, *Cross-Site Request Forgery (CSRF)*, *Local File Inclusion (LFI)*, or *Remote Code Execution (RCE)* issues could be identified during this exercise.
- The source code of the solution is well-written, easy to read, and generally adheres to a number of security best practices. Importantly, no sensitive data was found to be exposed within the code.
- The Circulo infrastructure exhibits well-thought-out designs and configurations, prioritizing security best practices.
- The AWS infrastructure leverages modern DevOps tools, such as code automation, enabling straightforward maintenance for team members.
- Circulo demonstrates robust security measures across the iOS and Android platforms showcasing a comprehensive approach to safeguarding user data.

The security posture of the Circulo mobile applications will improve with a focus on:

- **Denial-of-Service (DoS):** It is recommended to implement appropriate fallback mechanisms to better protect users against DoS attacks. Both Android and iOS applications should incorporate safe DNS resolution mechanisms that ensure integrity and confidentiality ([CIR-01-009](#)).
- **Hijacking Attacks:** The Android application should mitigate well-known Task Hijacking attacks ([CIR-01-003](#)).
- **Input Validation:** The Android app should further reduce its attack surface by improving the validation of user input on all exported activities ([CIR-01-006](#)).
- **Information leakage:** The Android application should implement measures to mitigate risks associated with sensitive information leakage to prevent exposure of sensitive data in memory ([CIR-01-007](#)).

- **Avoidance of Logging Leaks:** Sensitive operational data, such as Circulo Room IDs, is currently being logged ([CIR-01-008](#)). This logging should be avoided or restricted to conditional debug builds to prevent unintentional data leaks.
- **Screenshot Leaks:** Both apps would benefit from implementing a security screen to avoid leaks via screenshots and app backgrounding ([CIR-01-005](#)).
- **Hardware-backed Security Enclave Usage:** The iOS application should leverage the hardware-backed security enclave available through the iOS Keychain for the best protection of secrets at rest ([CIR-01-013](#)).
- **Removal of Unsafe Crypto Functions:** Circulo should completely eliminate any presence of cryptographic algorithms with known security weaknesses in its entire codebase. The development team should instead leverage cryptographically-safe functions for adequate security of tokens, hashes, passwords, and any other application areas ([CIR-01-010](#)).
- **PIN Policy:** An adequate PIN policy should be implemented for the Android and iOS applications to prevent unauthorized access ([CIR-01-012](#)). This implementation will significantly enhance the overall security posture.
- **General Hardening:** Other less important hardening recommendations include implementing a root/jailbreak detection mechanism to alert users about security risks prior to using the application ([CIR-01-001](#)), improving settings to better protect users on older supported devices ([CIR-01-002](#)), enhancing iOS binary protections ([CIR-01-011](#)), removing excessive permissions ([CIR-01-014](#)) and adjusting several configuration options in the Android app ([CIR-01-004](#)).

Hardening of Circulo backend services & cloud infrastructure should be prioritized in the following areas:

- **Logging and Monitoring:** It is important to follow standard logging and monitoring practices at the OS level ([CIR-01-027](#)) and throughout the AWS cloud infrastructure ([CIR-01-020](#)), to preserve the integrity of captured security related events and enhance threat detection capabilities. It is advised that sensitive data be excluded from logs to prevent data leakage and potential session hijacking ([CIR-01-022](#)).
- **CI/CD Pipelines & Security Tool Usage:** The platform would benefit from implementing security tools in AWS and CI/CD pipelines. Multiple AWS tools should be enabled, utilized, and their results reviewed on a regular basis ([CIR-01-015](#)). Additionally, every change should go through CI/CD pipelines, and pipelines should be blocked when there are any reported issues.
- **S3 Bucket Hardening:** The security of S3 buckets storing sensitive data should be strengthened by implementing *AWS Key Management Service (AWS KMS)*<sup>161</sup>, providing fine-tuned encryption and improved access control to prevent unauthorized access to files ([CIR-01-017](#)).

<sup>161</sup> <https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html>

- **Attack Surface Reduction:** The cloud infrastructure would benefit from eliminating anything that is not strictly required to ensure adversaries have as little opportunity as possible. For example, outbound traffic ought to be disallowed ([CIR-01-018](#)), full disk encryption enabled ([CIR-01-028](#)), and encryption enabled by default for all the EBS volumes ([CIR-01-016](#)).
- **Software Patching:** All server components should adhere to appropriate software patching procedures, consistently applying security patches in a timely manner ([CIR-01-024](#)). In a day and age when a significant portion of code comes from underlying software dependencies, routine patching is crucial to prevent potential security vulnerabilities.
- **Network Configuration:** The network stack configuration should be improved to avoid a number of possible attacks ([CIR-01-029](#)). It is recommended to use secure defaults wherever possible ([CIR-01-025](#)).
- **Bootloader Password:** Backend hosts should set a bootloader password ([CIR-01-026](#)).
- **SSH Access Hardening:** The SSH configuration could be enhanced by implementing MFA and other options ([CIR-01-023](#)).
- **General Cloud Hardening** should be in place to further enhance all security measures, focusing on these key areas: A consistent approach should be implemented to ensure secure defaults are being used in all configurations ([CIR-01-016](#)), as well as the implementation of as many AWS security-related services as possible ([CIR-01-015](#)), and other hardening recommendations ([CIR-01-019](#)). Implementing these measures will significantly bolster the overall security of the Circulo cloud infrastructure.

It is advised to address all issues identified in this report, including informational and low severity tickets where possible. This will not just strengthen the security posture of the application significantly, but also reduce the number of tickets in future audits.

Once all issues in this report are addressed and verified, a more thorough review, ideally including another source code audit, is highly recommended to ensure adequate security coverage of the platform. This provides auditors with an edge over possible malicious adversaries that do not have significant time or budget constraints.

Please note that future audits should ideally allow for a greater budget so that test teams are able to deep dive into more complex attack scenarios. Some examples of this could be third party integrations, complex features that require to exercise all the application logic for full visibility, authentication flows, challenge-response mechanisms implemented, subtle vulnerabilities, logic bugs and complex vulnerabilities derived from the inner workings of dependencies in the context of the application. Additionally, the scope could perhaps be extended to include other internet-facing Circulo resources.



It is suggested to test the application regularly, at least once a year or when substantial changes are going to be deployed, to make sure new features do not introduce undesired security vulnerabilities. This proven strategy will reduce the number of security issues consistently and make the application highly resilient against online attacks over time.

7A Security would like to take this opportunity to sincerely thank Fabiola Maurice, Nathan Freitas and the rest of the Circulo team, for their exemplary assistance and support throughout this audit. Last but not least, appreciation must be extended to the Open Technology Fund (OTF) for sponsoring this project.