WARNING!

THIS DOCUMENT DESCRIBES A SECURITY VULNERABILITY AND CONTAINS SENSITIVE INFORMATION.
*DO **NOT** CIRCULATE.*

RESPONSIBLE PUBLIC DISCLOSURE NOTICE

March 19, 2025

This document details serious security vulnerabilities identified in a publicly accessible and downloadable application. Pursuant to established principles of responsible disclosure and in alignment with industry best practices, this report is provided in good faith to facilitate prompt remediation of a demonstrable security risk to users and infrastructure.
In accordance with bona fide security research protocols, the existence and nature of this vulnerability shall be publicly disclosed under one of the following conditions, whichever occurs first:

1. Upon formal confirmation that the vulnerability has been remediated, with written notification to the undersigned (including my inclusion in relevant correspondence confirming resolution).
   (In this case, retrospective disclosure will be undertaken when vulnerabilities have been removed.)

2. Fifteen (15) days from the date of this notice, should no substantive response or clear remediation plan be provided.

This timeline reflects standard security disclosure practices and is structured to afford the affected parties a reasonable opportunity to rectify the issue while ensuring that the broader security implications are not unduly prolonged.
If the affected entity elects to make an independent disclosure prior to the stated deadline, such an action will be regarded as fulfilling the disclosure requirement. However, absent such disclosure, or absent timely resolution, public disclosure shall proceed as specified.

~~~
~~~

Security Vulnerabilities in *Allen* App: Exposure of Sensitive Secrets in iOS Application Bundle

Sumukh Prasad
sumukh prasad (at) icloud (dot) com
https://sumukhprasad.github.io

Abstract

In the course of exploring the internal structure of the "Allen" iOS app (version `1.96`), several API keys, client IDs and miscellaneous secrets were discovered stored inside Property List (`.plist`) files stored within the application bundle. Such insecure storage provides malicious actors with unauthorised data access, and poses significant security and privacy risks, risk of service abuse, and potential financial costs to the organisation.
This report outlines the discovery process, potential implications, and recommendations for mitigating these risks.

I. Introduction

The security of sensitive credentials and secrets is essential to ensuring the integrity of an application and protecting user data and privacy.

During an exploration of the *Allen* iOS app's package contents, multiple sensitive credentials were found in plaintext within `.plist` files. This report highlights the risks associated with such exposure and provides recommendations for remediation.

Such credentials and secrets must, in no circumstance, be published to any broad public audience through any means, and specimens identified as having been leaked through such action as being published must be immediately revoked and/or deactivated, while measures must be taken to prevent a future leak of such kinds.

II. Discovery Process

Several keys and secrets were discovered through the process outlined in this section. It is important to note that no reverse engineering, decryption, or tampering was performed to obtain these credentials, as they were readily accessible in their original form.

- Download the *"Allen"* iOS app from the Apple App Store.
- View package contents and navigate to root directory of the app.
- Open `Info.plist` or `GoogleService-Info.plist` Property List files.

It is, again, important to note that these are highly sensitive credentials that were obtained without the use of reverse engineering, decryption, or tampering.

III. Data Recovered

The following data was obtained from this exercise:

- `GoogleService-Info.plist`
  - `CLIENT_ID`
  - `REVERSED_CLIENT_ID`
  - `ANDROID_CLIENT_ID`
  - `API_KEY`
  - `GCM_SENDER_ID`
  - `STORAGE_BUCKET`
  - `PROJECT_ID`
  - `BUNDLE_ID`
  - `GOOGLE_APP_ID`
- `Info.plist`
  - `CleverTapToken`
  - `APPSFLYER_KEY`
  - `DATADOG_KEY`
  - `MIXPANEL_API_KEY`
  - `CleverTapAccountID`

IV. Implications

The presence of hard-coded secrets in the application bundle exposes the application and its backend to several security threats:

- Unauthorised API access

  - Malicious actors may use the Google API keys to make requests to Google Cloud services, potentially leading to data leaks, access to storage buckets, and abuse of backend services.
  - If the API key has write permissions, an attacker could modify or delete data.
- Service Abuse
  - Exposed credentials such as the AppsFlyer API Key, CleverTap Token, GCM IDs, and Datadog Key allow for access to analytics, push notification services, and logging.
  - This can be exploited to manipulate app analytics, spam users with push notifications, or inject misleading data into logs.
- User Privacy Risks
  - If any of the exposed keys grant access to user-related data (such as Firebase Storage Buckets or Google Cloud services), attackers could potentially access private user information, leading to privacy breaches.

V. Recommendations

To mitigate security risks, it is recommended that the following steps be taken immediately:

- Remove secrets from application bundle
  - Sensitive secrets must never be stored in plaintext inside an application bundle, or ideally never stored on a client-facing downloadable or a client device.
- Rotate and revoke exposed credentials
  - Since the credentials have already been exposed, all affected API keys and tokens should be revoked immediately and new credentials should be issued.
- Implement remote storage for secrets
  - Sensitive APIs must be accessed only by servers that act as proxy APIs for client-facing services.
  - As a general rule, sensitive information must never be placed outside the purview of infrastructure the organisation has direct access to.
- Use encrypted local storage
  - If keys must be stored locally, consider using secret stores or services like Apple's Keychain Service to store credentials.
- Implement runtime security measures
  - Consider implementing certificate pinning and other app security techniques to prevent man-in-the-middle attacks
- Implement security audits as part of CI/CD pipelines
  - Automated security tools (e.g., GitHub Secret Scanning, Static Application Security Testing (SAST)) can help identify issues before deployment.

$\sim\sim\sim$

Listings

Info.plist (trimmed)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  ...
  <key>CleverTapToken</key>
  <string>[redacted]</string>
  ...
  <key>APPSFLYER_KEY</key>
  <string>[redacted]</string>
  ...
  <key>DATADOG_KEY</key>
  <string>[redacted]</string>
  ...
  <key>MIXPANEL_API_KEY</key>
  <string>[redacted]</string>
  ...
  <key>CleverTapAccountID</key>
  <string>[redacted]</string>
  ...
</dict>
</plist>
```

GoogleService-Info.plist (trimmed)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>CLIENT_ID</key>
  <string>[redacted]</string>
  <key>REVERSED_CLIENT_ID</key>
  <string>[redacted]</string>
  <key>ANDROID_CLIENT_ID</key>
  <string>[redacted]</string>
  <key>API_KEY</key>
  <string>[redacted]</string>
  <key>GCM_SENDER_ID</key>
  <string>[redacted]</string>
  ...
  <key>BUNDLE_ID</key>
  <string>[redacted]</string>
  <key>PROJECT_ID</key>
  <string>[redacted]</string>
  <key>STORAGE_BUCKET</key>
  <string>[redacted]</string>
  ...
  <key>GOOGLE_APP_ID</key>
  <string>[redacted]</string>
</dict>
</plist>
```

Figures

Figure 1: App Bundle contents

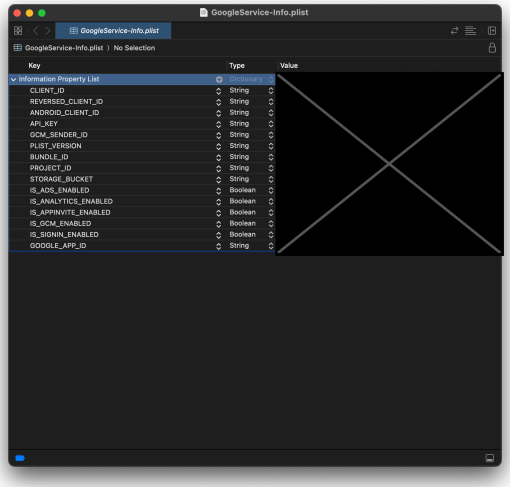

Figure 2: Contents of `GoogleService-Info.plist`

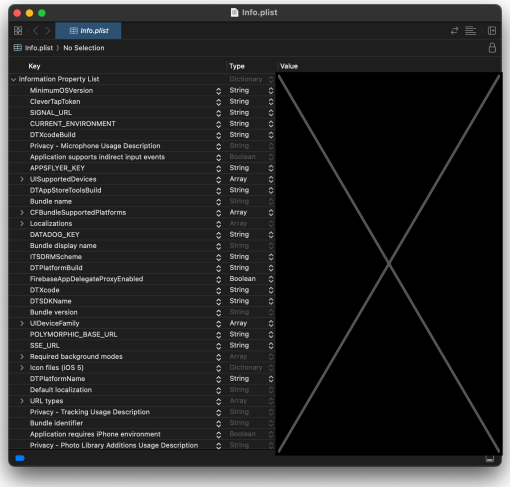

Figure 3: Contents of `Info.plist` (I)

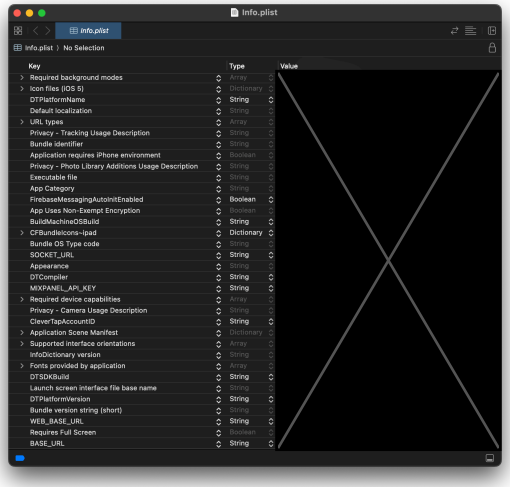

Figure 4: Contents of `Info.plist` (II)

~~~
~~~