INCLUDE SECURITY

Remediation Test of Least Authority's Gridsync application and Tahoe LAFS Android application

EXECUTIVE SUMMARY

Engagement Details

Client	Least Authority
Engagement Scope	Gridsync application and Tahoe LAFS Android application
Original Assessment Schedule	May 17th, 2021 - May 31st, 2021
Remediation Test Dates	March 25th, 2022

Remediation Test Update: Technical Findings Summary

The information below summarizes the observations of the Includesec team during the course of the remediation test intended to reproduce the findings as originally reported. The team attempted to bypass any added mitigations or protections put in place to hinder exploitation of the findings.

Finding	Risk Rating	Status
M1	Medium	Risk Accepted
L1	Low	Closed
L2	Low	Closed
L3	Low	Closed
I1	Informational	Closed
I2	Informational	Closed

MEDIUM-RISK FINDINGS

M1: [Wormhole] Gridsync Vulnerable to Denial of Service Attacks

Status: Risk Accepted

Notes:

Least Authority has accepted the risk of this finding as originally reported. The **Least Authority** team has provided the following statement regarding this finding:

"We consider this out-of-scope for the purposes of linking/sharing folders between mobile devices (which employ a different, QR code-based mechanism for exchanging cryptographic capabilities)."

LOW-RISK FINDINGS

L1: [Android] Application Configured to Support Unencrypted HTTP

Status: Closed

Notes:

This finding was retested and found to be remediated. The following snippet from **AndroidManifest.xml** shows that **usesCleartextTraffic** is set to **false**:

```
31              android:theme="@style/Theme.TahoeLafs"
32              android:usesCleartextTraffic="false"
33              tools:targetApi="m">
```

L2: [Android] Application Allowed Android Backups

Status: Closed

Notes:

This finding was retested and found to be remediated. The following snippet from **AndroidManifest.xml** shows that the **allowBackup** setting is set to **false**:

```
23          <application
24              android:name=".TahoeApplication"
25              android:allowBackup="false"
26              android:icon="@mipmap/ic_launcher"
```

L3: Client-Side Denial of Service via Malicious QR Code

Status: Closed

Notes:

This finding was retested and found to be remediated. The file **QRCodeContents.kt** implemented validation for illegal arguments, as shown in the following snippet:

```
    13        return if (parts.size <= 1) {
    14          Result.failure(IllegalArgumentException("Invalid QR code"))
    15        } else {
    16          val (url, token) = parts
    17          val validatedUrl = Result.success(url).mapCatching { URL(it) }.flatMap(HttpsURL::fromURL)
    18          val validatedToken =
    19              if (token.isNotEmpty()) {
    20                Result.success(token)
    21              } else {
```

```
22                Result.failure(IllegalArgumentException("Token has 0 length"))
23            }
```

The following screenshot shows that the application gracefully handled the error without crashing:

INFORMATIONAL FINDINGS

I1: Gridsync Application Denial of Service via Import Recovery Key Functionality

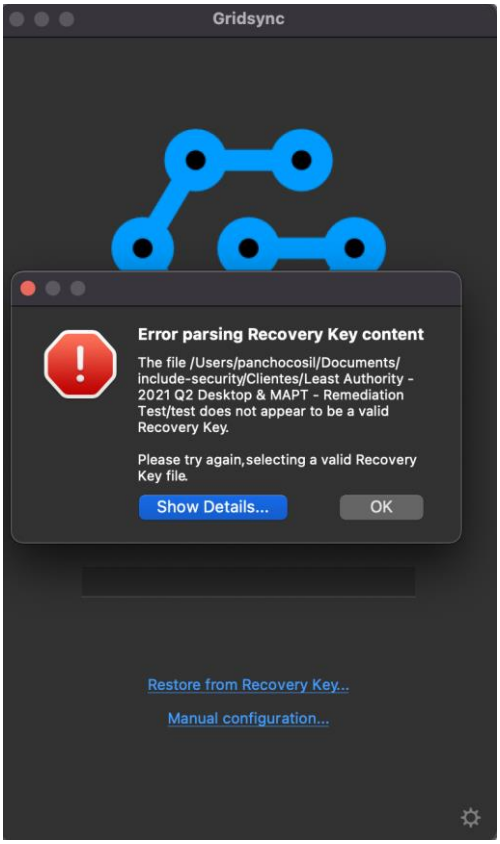
Status: Closed

Notes:

This finding was retested and both instances were found to be remediated.

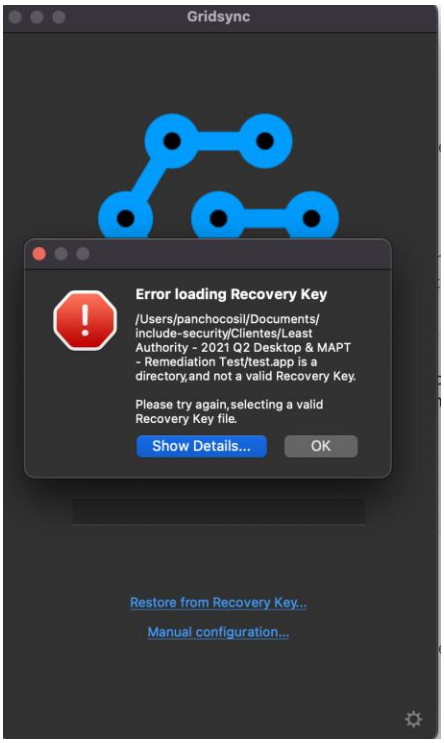
First Instance – DoS via Recovery Key Containing Only Numeric Characters

The following screenshot shows that the application gracefully handles this error without crashing:



Second Instance – DoS via Importing Directory (MacOS)

The following screenshot shows that the application gracefully handles this error without crashing:



I2: Gridsync Application Crash via Voucher Code Containing non-ASCII Characters

Status: Closed

Notes:

This finding was retested and found to be remediated. As shown in the file **/gridsync/voucher.py**, line 40, errors caused by non-ASCII characters were explicitly handled via **UnicodeEncodeError**:

```
36  def is_valid(code: str, checksum_length: int = 2) -> bool:
37      code = dehyphenate(code)
38      try:
39          decoded = base64.b32decode(code)
40      except (binascii.Error, UnicodeEncodeError, ValueError):
41          return False
42      b = decoded[:-checksum_length]
43      checksum = decoded[-checksum_length:]
44      if checksum == get_checksum(b):
45          return True
46      return False
```

The following screenshot shows that the application gracefully handled this error condition without crashing: