NGI REVIEW FACILITY.EU

Name: NGI Emergency Tech Review Facility High-level assessment ProteGO

Contract nr: LC-01499045

Date of signature: 15-01-2021

Name of the deliverable – Deliverable 3.3 – High-level assessment ProteGO

Authors: Abhinav Mishra, Marcus Bointon (Radically Open Security)

Level of distribution: Confidential, only for members of the facility (including the Commission Services)

Confidential: Yes

Radically Open Security

# Penetration Test Report

## Extraordinary Commissioner for the COVID-19 Emergency

V 1.0
Diemen, September 30th, 2020
Confidential

## Document Properties

| | |
|---|---|
| Client | Extraordinary Commissioner for the COVID-19 Emergency |
| Title | Penetration Test Report |
| Target | ProteGO Safe Android and iOS apps (version 4.2.4) |
| Version | 1.0 |
| Pentester | Abhinav Mishra |
| Authors | Abhinav Mishra, Marcus Bointon |
| Reviewed by | Marcus Bointon |
| Approved by | Melanie Rieback |

## Version control

| Version | Date | Author | Description |
|---|---|---|---|
| 0.1 | September 24th, 2020 | Abhinav Mishra | Initial draft |
| 0.2 | September 28th, 2020 | Marcus Bointon | Review |
| 1.0 | September 30th, 2020 | Marcus Bointon | Final version |

## Contact

For more information about this document and its contents please contact Radically Open Security B.V.

| | |
|---|---|
| Name | Melanie Rieback |
| Address | Overdiemerweg 28<br>1111 PP Diemen<br>The Netherlands |
| Phone | +31 (0)20 2621 255 |
| Email | info@radicallyopensecurity.com |

Radically Open Security B.V. is registered at the trade register of the Dutch chamber of commerce under number 60628081.

# Table of Contents

# 1 Executive Summary

## 1.1 Introduction

Between August 17, 2020 and September 21, 2020, Radically Open Security B.V. carried out a penetration test for Extraordinary Commissioner for the COVID-19 Emergency.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

## 1.2 Scope of work

The scope of the penetration test was limited to the following target:

- ProteGO Safe Android and iOS apps (version 4.2.4)

## 1.3 Project objectives

ROS performed a penetration test with the developers of the ProteGO Safe application(s). The test was intended to gain insight into the security of the apps. To do so, ROS accessed ProteGO Safe together with the developers, attempting to find vulnerabilities and gain further access and elevated privileges by exploiting any vulnerabilities found.

## 1.4 Timeline

The Security Audit took place between August 17, 2020 and September 21, 2020. The report was completed on September 30th.

## 1.5 Results In A Nutshell

During this crystal-box penetration test we found 1 Elevated issue and 5 Low-severity issue.

The elevated-severity issue is related to the JWT which is generated in response to supplying a valid OTP. This JWT can be used to send multiple requests with random/malicious TEK data, which might be abused to flood the service with malicious/fake data.

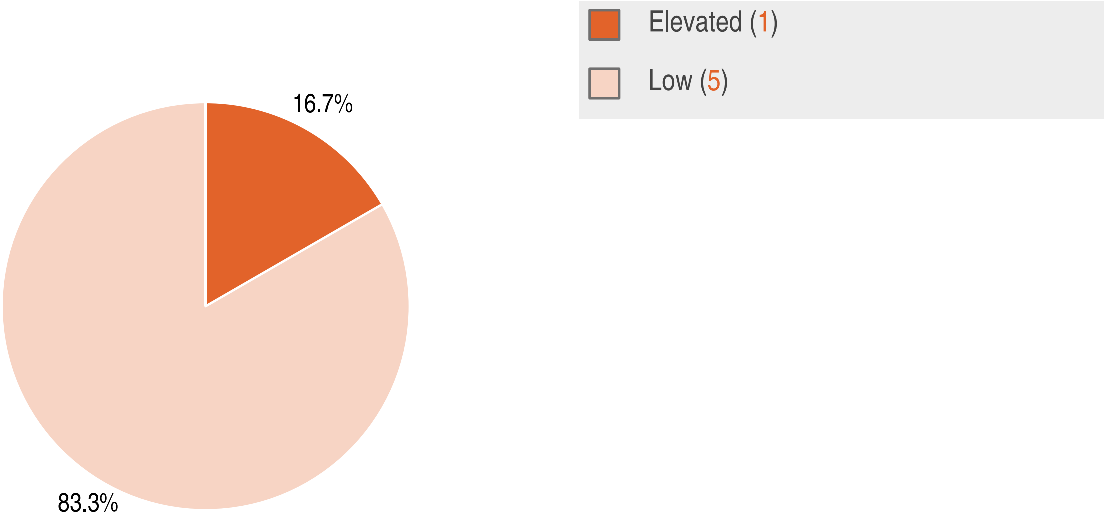
The low-severity findings are the lack of checks for jailbroken/rooted devices (which allow the app run without modification in a potentially hostile or compromised environment), insecure local storage, insecure webview implementation and NSUrl caching.

By exploiting these issues an attacker might be able to tamper with TEK uploads, and might modify the general behaviour of the application, undermining its trustworthiness.

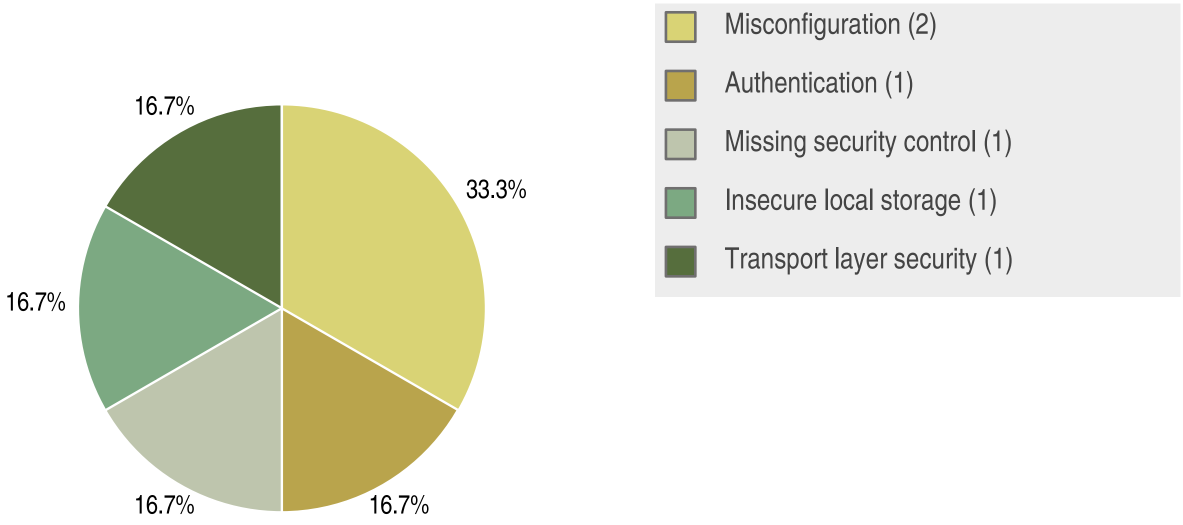
## 1.6 Summary of Findings

| ID | Type | Description | Threat level |
|---|---|---|---|
| PRO-001 | Authentication | The JWT which is generated by the OTP is considered valid for multiple upload requests until it expires. | Elevated |
| PRO-004 | Missing security control | The Android apps can be installed and run on a rooted device. | Low |
| PRO-005 | Insecure local storage | The Android app stores authentication tokens in the app's storage area in an unencrypted format. | Low |
| PRO-006 | Misconfiguration | The Android application implements a web view with Javascript enabled. | Low |
| PRO-007 | Misconfiguration | In the iOS app, the NSURLRequests are cached in the Cache.db file. | Low |
| PRO-008 | Transport layer security | Some back-end endpoints allow clients to choose weak cipher suites. | Low |

### 1.6.1 Findings by Threat Level



### 1.6.2 Findings by Type



## 1.7 Summary of Recommendations

| ID | Type | Recommendation |
|---|---|---|
| PRO-001 | Authentication | • Invalidate the JWT immediately a valid upload has been submitted; reject any re-use of the token. |
| PRO-004 | Missing security control | • Implement a root detection mechanism on Android app, and warn the user appropriately. |
| PRO-005 | Insecure local storage | • Consider encrypting the sensitive tokens/files or store them in the Android keystore. |
| PRO-006 | Misconfiguration | • Enable Javascript in web views only if strictly necessary. |
| PRO-007 | Misconfiguration | • Disable HTTP caching. |
| PRO-008 | Transport layer security | • Require a minimum of TLSv1.2<br>• Disable suites using RSA key exchange<br>• Disable CBC-mode cipher suites<br>• Test these changes on minimum supported iOS and Android OS versions |

# 2 Methodology

## 2.1 Planning

Our general approach during penetration tests is as follows:

1. **Reconnaissance**
   We attempt to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection, etc., afforded to the network. This usually involves trying to discover publicly available information by utilizing a web browser, visiting newsgroups, etc. An active form would be more intrusive and may show up in audit logs and may take the form of a social engineering type of attack.
2. **Enumeration**
   We use various fingerprinting tools to determine what hosts are visible on the target network and, more importantly, try to ascertain what services and operating systems they are running. Visible services are researched further to tailor subsequent tests to match.
3. **Scanning**
   Vulnerability scanners are used to scan all discovered hosts for known vulnerabilities or weaknesses. The results are analyzed to determine if there are any vulnerabilities that could be exploited to gain access or enhance privileges to target hosts.
4. **Obtaining Access**
   We use the results of the scans to assist in attempting to obtain access to target systems and services, or to escalate privileges where access has been obtained (either legitimately though provided credentials, or via vulnerabilities). This may be done surreptitiously (for example to try to evade intrusion detection systems or rate limits) or by more aggressive brute-force methods. This step also consist of manually testing the application against the latest (2017) list of OWASP Top 10 risks. The discovered vulnerabilities from scanning and manual testing are moreover used to further elevate access on the application.

## 2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.

- **High**

  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**

  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**

  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

# 3 Reconnaissance and Fingerprinting

Through automated scans we were able to gain the following information about the software and infrastructure. Detailed scan output can be found in the sections below.

## 3.1 Automated Scans

As part of our active reconnaissance we used the following automated scans:

- nmap – http://nmap.org
- Burp Suite Pro – https://portswigger.net/burp/pro
- MobSF – https://github.com/MobSF/Mobile-Security-Framework-MobSF
- Testssl – https://testssl.sh/

# 4 Findings

We have identified the following issues:

## 4.1 PRO-001 — JWT Generated by the OTP Is Valid for Multiple Uploads

**Vulnerability ID:** PRO-001

**Vulnerability type:** Authentication

**Threat level:** Elevated

### Description:

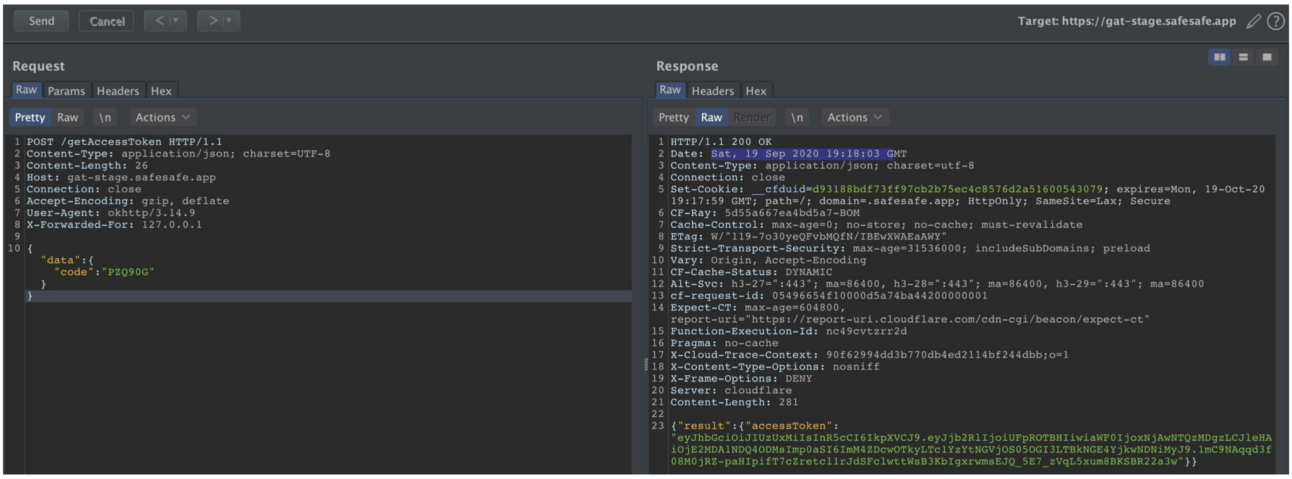
The JWT which is generated by the OTP is considered valid for multiple upload requests until it expires.

### Technical description:

When the user supplies a valid OTP to the endpoint `https://gat-stage.safesafe.app/getAccessToken`, the response contains a JWT. This JWT is then used in the next request to authenticate the TEK upload.

**JWT Generation**

JWT Generation time: 19:18; Expiration time: 19:48



```
POST /getAccessToken HTTP/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 26
Host: gat-stage.safesafe.app
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.14.9
X-Forwarded-For: 127.0.0.1

{
  "data":{
    "code":"PZQ90G"
  }
}
```

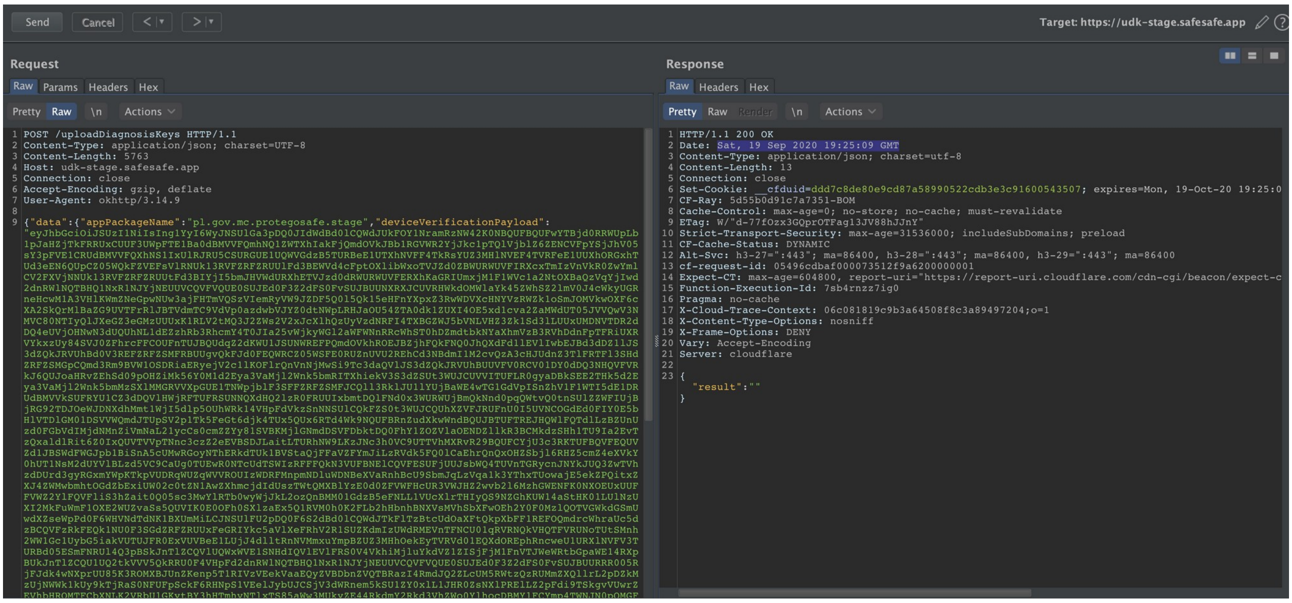
```
HTTP/1.1 200 OK
Date: Sat, 19 Sep 2020 19:18:03 GMT
Content-Type: application/json; charset=utf-8
Connection: close
Set-Cookie: __cfduid=d93188bdf73ff97cb2b75ec4c8576d2a51600543079; expires=Mon, 19-Oct-20
19:17:59 GMT; path=/; domain=.safesafe.app; HttpOnly; SameSite=Lax; Secure
CF-Ray: 5d55a667ea4bd5a7-BOM
Cache-Control: max-age=0; no-store; no-cache; must-revalidate
ETag: W/"119-7o30yeQFvbMQfN/IBEwXWAEaAWY"
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Vary: Origin, Accept-Encoding
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
cf-request-id: 05496654f10000d5a74ba44200000001
Expect-CT: max-age=604800,
report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Function-Execution-Id: nc49cvtzrr2d
Pragma: no-cache
X-Cloud-Trace-Context: 90f62994dd3b770db4ed2114bf244dbb;o=1
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Server: cloudflare
Content-Length: 281

{"result":{"accessToken":
"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJjb2RlIjoiUFpROTBHIiwiaWF0IjoxNjAwNTQzMDgzLCJleHA
iOjE2MDA1NDQ4ODMsImp0aSI6ImM4ZDcwOTkyLTc1YzYtNGVjOS05OGI3LTBkNGE4YjkwNDNiMyJ9.lmC9NAqqd3f
08M0jRZ-paHIpifT7cZretcl1rJdSFclwttWsB3KbIgxrwmsEJQ_5E7_zVqL5xum8BKSBR22a3w"}}
```

The expiration time of this JWT can be discovered by decoding the key as a Base64 string, and then converting the epoch expiration time; It was found to be 30 minutes.

During this period of 30 minutes, the same JWT can be used multiple times to upload the data. After the 30 min time, the server will respond with unauthenticated message.

**Uploading TEK as intended** (Request time: 19:25)



Target: https://udk-stage.safesafe.app

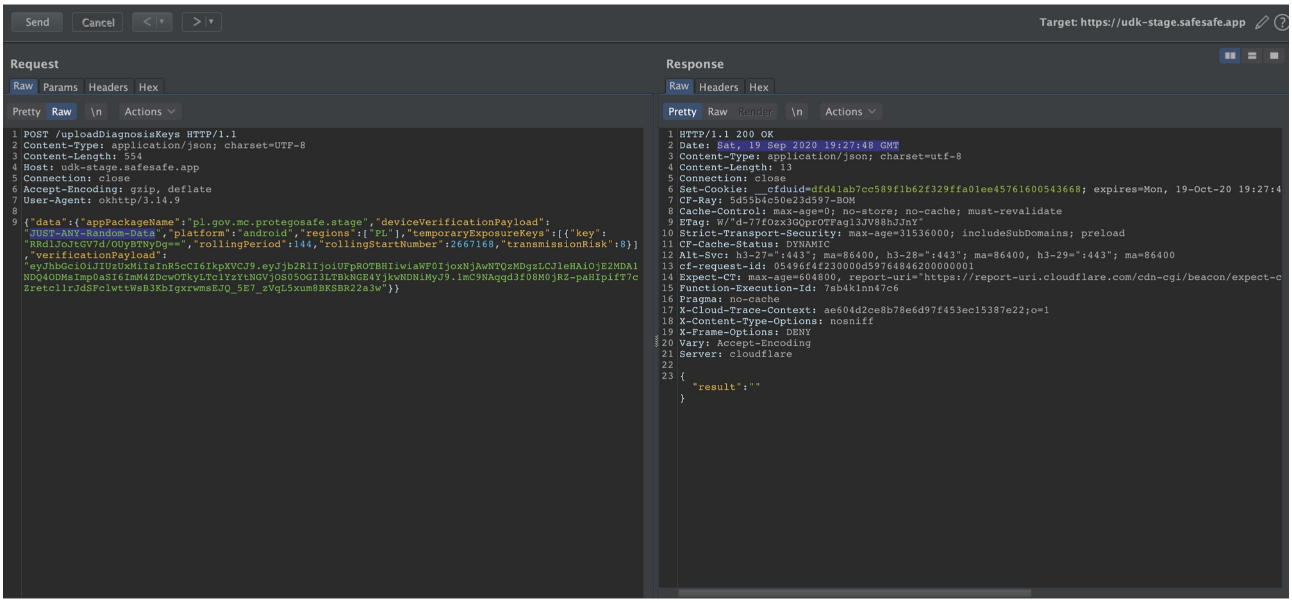
```
POST /uploadDiagnosisKeys HTTP/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 5763
Host: udk-stage.safesafe.app
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.14.9

{"data":{"appPackageName":"pl.gov.mc.protegosafe.stage","deviceVerificationPayload":
"eyJhbGciOiJSUzI1NiIsIng1YyI6WyJNSUlGa3pDQ0JIdWdBd0lCQWdJUkFOY1NramRzNW42K0NBQUFBQUFwYTBjd0RRWUpLb
1pJaHZjTkFRRUxCUUF3UWpFTE1Ba0dBMVVFQmhNQ1ZWTXhIakFjQmdOVkJBb1RGVWR2YjJkc1pTQlVjblZ6ZENCVFpYSjhV05
sY3pFVE1CRUdBMVVFQXhNS1IxUlRJRU5CSURGUE1UQWVGdzB5TURBeE1UTXhNVFF4TkRsYUZ3MHlNVEF4TVRFeE1UUXhORGxhT
Ud3eEN6QUpCZ05WQkFZVEFsVlRNUk13RVFZRFZRUUlFd3BEWVd4cFptOXlibWxoTVJZd0ZBWURWUVFIRXcxTmIzVnVkR0ZwYml
CV2FXVjNNUk13RVFZRFZRUUtFd3BIYjI5bmJHVWdURXhETVJzd0dRWURWUVFERXhKaGRIUmxjM1F1WVc1a2NtOXBaQzVqYjIwd
2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUUNXRXJCUVRHWkdOMWlaYk45ZWhSZ2lmV0J4cWkyUGR
neHcwM1A3VHlKWmZNeGpwNUw3ajFHTmVQSzVIemRyVW9JZDF5Q015Qk15eHFnYXpxZ3RwWDVXcHNYVzRWZk1oSmJOMVkwOXF6c
XA2SkQrMlBaZG9UVTFrRlJBTVdmTC9VdVp0azdwbVJYZ0dtNWpLRHJaOU54ZTA0dk1ZUXI4OE5xdlcva2ZaMWdUT05JVVQwV3N
MVC80NTIyQlJXeGZ3eGMzUUUxK1RLV2tMQ3J2ZWs2V2xJcXlhQzUyVzdNRFI4TXBGZWJ5bVNLVHZ3Zk1Sd3lLUUxUMDNVTDR2d
DQ4eUVjOHNwN3dUQUhNL1dEZzhRb3RhcmY4T0JIa25vWjkyWGl2aWFWNnRRcWhST0hDZmdtbkNYaXhmVzB3RVhDdnFpTFRiUXR
VYkxzUy84SVJ0ZFhrcFFCOUFnTUJBQUdqZ2dKWU1JSUNWREFPQmdOVkhROEJBZjhFQkFNQ0JhQXdFd1lEVlIwbEJBd3dDZ1lJS
3dZQkJRVUhBd0V3REFZRFZSMFRBUUgvQkFJd0FEQWRCZ05WSFE0RUZnUVU2REhCd3NBdmI1M2cvQzA3cHJUdnZ3TlFRTFl3SHd
ZRFZSMGpCQmd3Rm9BVW1OSDRiaERyejV2c1lKOFlrQnVnNjMwSi9Tc3daQVlJS3dZQkJRVUhBUUVFV0RCV01DY0dDQ3NHQVFVR
kJ6QUJoaHRvZEhSd09pOHZiMk56Y0M1d2Eya3Vaмjl2Wnk5bmRITXhiekV3S3dZSUt3WUJCUVVITUFLR0gyaDBkSEE2THk5d2E
ya3VaMjl2Wnk5bmMzSXlMMGRVVXpGUE1TNWpjblF3SFFZRFZSMFJCQll3RklJU1l1YUjBaWE4wTG1GdVpISnZhV1F1WTI5dE1DR
UdBMVVkSUFRYU1CZ3dDQVlHWjRFTUFRSUNNQXdHQ2lzR0FRUUIxbmtDQlFNd0x3WURWUjBmQkNnd0pqQWtvQ0tnSUlZZWFIUjB
jRG92TDJOeWJDNXdhMmt1WjI5dlp5OUhWRk14VHpFdVkzSnNNSUlCQkFZS0t3WUJCQUhXZVFJRUFnU0I5UVNCOGdEd0FIY0E5b
HlVTDlGM01DSVVWQmdJTUpSV2p1Tk5FeGt6djk4TUx5QUx6RTd4WlpNQUFBRnZudXdkXkwWndBQUJBTUFTREJHQWlFQTdlLzBZUnU
zd0FGbVdIMjdNMnZiVmNaL21ycCs0cmZZYy81SVBKMjlGNmdDSVFDbktDQ0FhY1ZOZVlaOENDZllkR3BCMkdzSHh1TU9Ia2EvT
zQxaldlRit6Z0IxQUVTVVpTNnc3czZ2eEVBSDJLaitLTURhNW9LZzJNc3h0VC9UTTVhMXRvR29BQUFCYjU3c3RKTUFBQVFEQUV
ZdlJBSWdFWGJpbEBiSnA5cUMwRGoyNThERkdTUklBVStaQjFFaVZFYmJiLzRVdk5FQ0lCaEhrQnQxOHZSbjl6RHZ5cmZ4eXVkY
0hUT1NsM2dUYVlBLzd5VC9CaUg0TUEwR0NTcUdTSUzRFFFQkN3VUFBNElCQVFFSUFjUUJsbWQ4TUVnTGRycnJNYkJUQ3ZwTVh
zdDUrd3gyRGxmYWpKTkpyVUDRqWUZqWVVROUIzWDRFMnpmNDluWDNBeXVaRnhBcU9SbmJqLzVqa1k3YThxTUowajE5ekZPQitxZ
XJ4ZWMwbmhtOGdZbExiUW02c0tZN1AwZXhmcjdIdUszTWtQMXB1YzE0d0ZFVWFHcUR3VWJHZ2wvb2l6MzhGWENFK0NXOEUxUUF
FVWZ2YlFQVFliS3hZait0Q05sc3MwYlRTb0wyWjJkL2ozQnBMM01GdzB5eFNLL1VUcXlrTHIyQS9NZGhKUW14aStHK01LUlNzU
XI2MkFuWmF1OXE2WUZvaSs5QUVIK0E0OFh0SXlzaEx5Q1RVM0h0K2FLb2hHbnhBNXVsMVhSbXFwOEh2Y0F0MzlQOTVGWkdSmU
wdXZseWpPd0F6WHVNdTdNK1BXUmMiLCJNSUlFU2pDQ0F6S2dBd0lCQWdJTkFlTzBtcUdOaXFtQkpXbFF1REFOQmdrcWhraUc5d
zBCQVFzRkFEQk1NU0F3SGdZRFZRUUxFeGRIYkc5aVlXeFRhV2R1SUZKdmIzUWdRMEVnTFNCU01qRVRNQkVHQTFVRUNoTUtSMnh
2WW1Gc1UybG5iakVUTUJFR0ExVUVBeE1LUjJ4dlltRnNVMmxuYmpBZUZ3MHhOekEyTVRVd01EQXdOREphRncweU1URXlNVFV3T
URBd05ESmFNRFl4Q3pBSkJnTlZCQVlUQWxWVE1SNHdIQVlEVlFRS0V4VkhiMjluYkdVZ1ZISjFjM1FnVTJWeWRtbGpaWE14RXp
BUkJnTlZCQU1UQ2tkVVV5QkRRU0F4VHpFd2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUURRR005R
jFJdk4wNXprUU85K3ROMXBJUnZKenp5T1RIVzVEekVaaEQyZVBDbnZVQTBRazI4RmdJQ2ZLcUM5RWtzQzRUMmZXQllrL2pDZkM
zUjNWWk1kUy9kTjRaS0NFUFpScrF6RHNpVEVlelJybUJCSjV3dWRnem5kSU1ZY0xlL1JHR0ZsNXFPRE1ZYzNVRlZPZi9TNkgvT
EVhbHROMTFCbXNLK2VRbU1GKytBY3hHTmhyNTlxTS85aWw3MUkyZE44RkdmY2Rkd3VhZWo0YlhocDBMY1FCYmp4TWNJN0pOMGF
```

```
HTTP/1.1 200 OK
Date: Sat, 19 Sep 2020 19:25:09 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 13
Connection: close
Set-Cookie: __cfduid=ddd7c8de80e9cd87a58990522cdb3e3c91600543507; expires=Mon, 19-Oct-20 19:25:0
CF-Ray: 5d55b0d91c7a7351-BOM
Cache-Control: max-age=0; no-store; no-cache; must-revalidate
ETag: W/"d-77fOzx3GQprOTFag13JV88hJJnY"
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
cf-request-id: 05496cdbaf000073512f9a6200000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-c
Function-Execution-Id: 7sb4rnzz7ig0
Pragma: no-cache
X-Cloud-Trace-Context: 06c081819c9b3a64508f8c3a89497204;o=1
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Vary: Accept-Encoding
Server: cloudflare

{
  "result":""
}
```

**Using the same JWT to upload different data within 30 min is accepted** (Request time: 19:27)



Target: https://udk-stage.safesafe.app

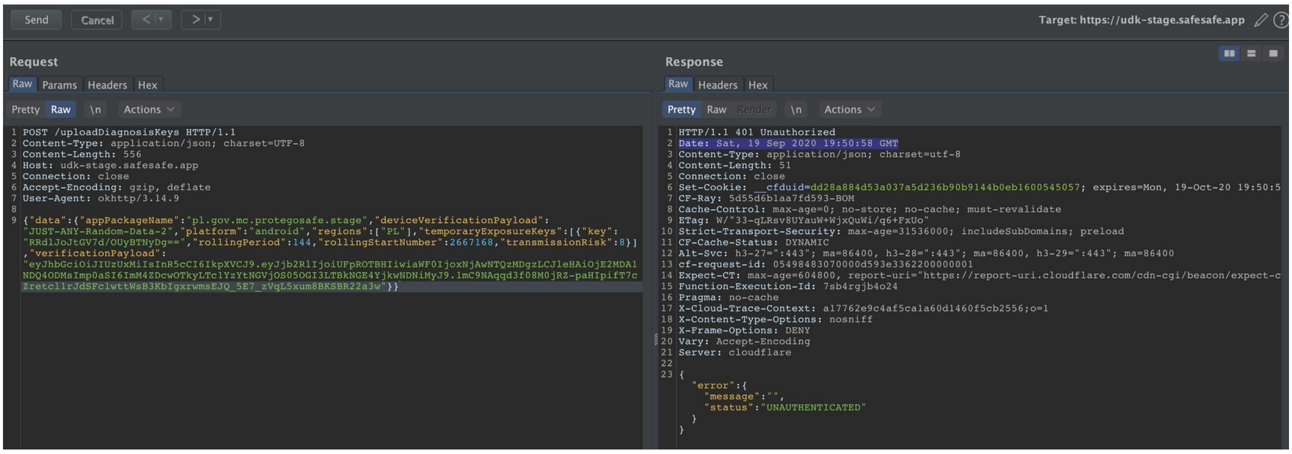
```
POST /uploadDiagnosisKeys HTTP/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 554
Host: udk-stage.safesafe.app
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.14.9

{"data":{"appPackageName":"pl.gov.mc.protegosafe.stage","deviceVerificationPayload":
"JUST-ANY-Random-Data","platform":"android","regions":["PL"],"temporaryExposureKeys":[{"key":
"RRdlJoJtGV7d/OUyBTNyDg==","rollingPeriod":144,"rollingStartNumber":2667168,"transmissionRisk":8}]
,"verificationPayload":
"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJjb2RlIjoiUFpROTBHIiwiaWF0IjoxNjAwNTQzMDgzLCJleHAiOjE2MDA1
NDQ4ODMsImp0aSI6ImM4ZDcwOTkyLTc1YzYtNGVjOS05OGI3LTBkNGE4YjkwNDNiMyJ9.1mC9NAqqd3f08M0jRZ-paHIpifT7c
Zretcl1rJdSFclwttWsB3KbIgxrwmsEJQ_5E7_zVqL5xum8BKSBR22a3w"}}
```

```
HTTP/1.1 200 OK
Date: Sat, 19 Sep 2020 19:27:48 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 13
Connection: close
Set-Cookie: __cfduid=dfd41ab7cc589f1b62f329ffa01ee45761600543668; expires=Mon, 19-Oct-20 19:27:4
CF-Ray: 5d55b4c50e23d597-BOM
Cache-Control: max-age=0; no-store; no-cache; must-revalidate
ETag: W/"d-77fOzx3GQprOTFag13JV88hJJnY"
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
cf-request-id: 05496f4f230000d5976484620000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-c
Function-Execution-Id: 7sb4k1nn47c6
Pragma: no-cache
X-Cloud-Trace-Context: ae604d2ce8b78e6d97f453ec15387e22;o=1
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Vary: Accept-Encoding
Server: cloudflare

{
  "result":""
}
```

**Sending an upload request with an expired JWT is rejected correctly** (Request time: 19:50)



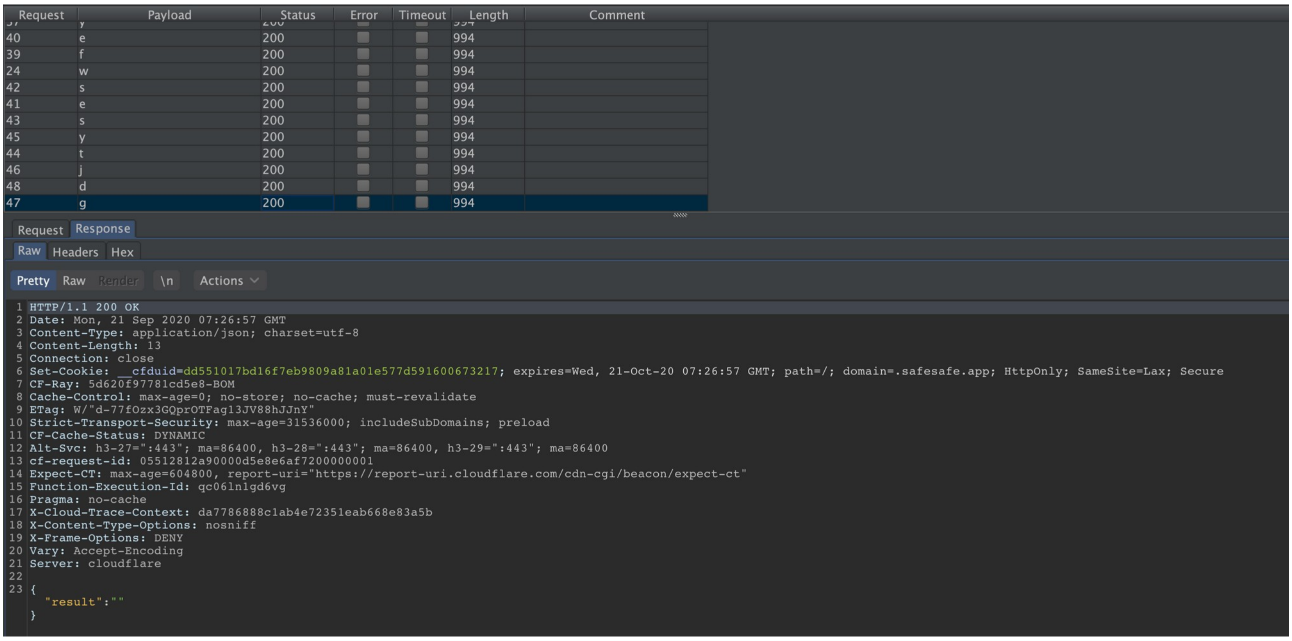
```
POST /uploadDiagnosisKeys HTTP/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 556
Host: udk-stage.safesafe.app
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.14.9

{"data":{"appPackageName":"pl.gov.mc.protegosafe.stage","deviceVerificationPayload":
"JUST-ANY-Random-Data-2","platform":"android","regions":["PL"],"temporaryExposureKeys":[{"key":
"RRdlJoJtGV7d/OUyBTNyDg==","rollingPeriod":144,"rollingStartNumber":2667168,"transmissionRisk":8}]
,"verificationPayload":
"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJjb2RlIjoiUFpROTBHIiwiaWF0IjoxNjAwNTQzMDgzLCJleHAiOjE2MDA1
NDQ4ODMsImp0aSI6ImM4ZDcwOTkyLTc1YzYtNGVjOS05OGI3LTBkNGE4YjkwNDNiMyJ9.1mC9NAqqd3f08M0jRZ-paHIpifT7c
ZretcllrJdSFclwttWsB3KbIgxrwmsEJQ_5E7_zVqL5xum8BKSBR22a3w"}}
```

```
HTTP/1.1 401 Unauthorized
Date: Sat, 19 Sep 2020 19:50:58 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 51
Connection: close
Set-Cookie: __cfduid=dd28a884d53a037a5d236b90b9144b0eb1600545057; expires=Mon, 19-Oct-20 19:50:5
CF-Ray: 5d55d6b1aa7fd593-BOM
Cache-Control: max-age=0; no-store; no-cache; must-revalidate
ETag: W/"33-qLRsv8UYauW+WjxQuWi/q6+FxUo"
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
cf-request-id: 05498483070000d593e3362200000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-c
Function-Execution-Id: 7sb4rgjb4o24
Pragma: no-cache
X-Cloud-Trace-Context: a17762e9c4af5ca1a60d1460f5cb2556;o=1
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Vary: Accept-Encoding
Server: cloudflare

{
  "error":{
    "message":"",
    "status":"UNAUTHENTICATED"
  }
}
```

### 50 upload requests using same JWT:



| Request | Payload | Status | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|
| 40 | e | 200 | □ | □ | 994 | |
| 39 | f | 200 | □ | □ | 994 | |
| 24 | w | 200 | □ | □ | 994 | |
| 42 | s | 200 | □ | □ | 994 | |
| 41 | e | 200 | □ | □ | 994 | |
| 43 | s | 200 | □ | □ | 994 | |
| 45 | y | 200 | □ | □ | 994 | |
| 44 | t | 200 | □ | □ | 994 | |
| 46 | j | 200 | □ | □ | 994 | |
| 48 | d | 200 | □ | □ | 994 | |
| 47 | g | 200 | □ | □ | 994 | |

```
HTTP/1.1 200 OK
Date: Mon, 21 Sep 2020 07:26:57 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 13
Connection: close
Set-Cookie: __cfduid=dd551017bd16f7eb9809a81a01e577d591600673217; expires=Wed, 21-Oct-20 07:26:57 GMT; path=/; domain=.safesafe.app; HttpOnly; SameSite=Lax; Secure
CF-Ray: 5d620f97781cd5e8-BOM
Cache-Control: max-age=0; no-store; no-cache; must-revalidate
ETag: W/"d-77fOzx3GQprOTFag13JV88hJJnY"
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
CF-Cache-Status: DYNAMIC
Alt-Svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
cf-request-id: 05512812a90000d5e8e6af7200000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Function-Execution-Id: qc06ln1gd6vg
Pragma: no-cache
X-Cloud-Trace-Context: da7786888c1ab4e72351eab668e83a5b
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Vary: Accept-Encoding
Server: cloudflare

{
  "result":""
}
```

## Impact:

As the upload feature of the app needs to be authenticated with a valid OTP, it is important to ensure that one OTP allows only a single upload. If the JWT can be used multiple times, it is possible to upload malicious/tampered/fake data multiple times, even when the attacker has one valid OTP.

### Recommendation:

- Invalidate the JWT immediately a valid upload has been submitted; reject any re-use of the token.

## 4.2 PRO-004 — (Android) Missing Root Detection

**Vulnerability ID:** PRO-004

**Vulnerability type:** Missing security control

**Threat level:** Low

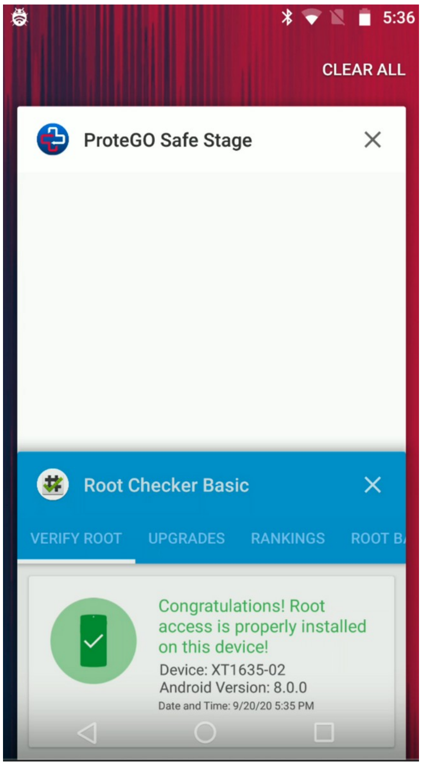
### Description:

The Android apps can be installed and run on a rooted device.

### Technical description:

There are no security controls to check whether the device has been rooted or not.

A rooting process compromises the security of the android device. The integrity of the operating system's controls over data that an application can access, is also lost. On a compromised device, a user/malicious app can disable key security features and may compromise the integrity of the data of all the applications. The Protego app (version 4.2.4, Stage) does not implement a check to verify whether the device on which it is running is rooted.



Congratulations! Root access is properly installed on this device!

Device: XT1635-02
Android Version: 8.0.0
Date and Time: 9/20/20 5:35 PM

## Impact:

When an application runs on a compromised device, the operating system and application's security controls, such as sandboxes, keystore access etc, might fail to prevent user data from being accessed by other malicious apps on the device. An application running on a compromised device also has a higher risk of exposure to malware, memory dumps, tampering, and other types of malicious activity that could possibly expose a user's credentials or other sensitive data.

## Recommendation:

Implement a root detection mechanism on Android app.

**Note:** It is not necessary to prevent the application from running on a compromised device, but users should be informed of the higher risk under which they are operating.

When attempting to detect if the application is running on a rooted device, checks such as the following could be used:

- Check for the presence of certain paths, for example:
  - `/sbin/su`
  - `/etc/apt`
  - `/system/bin/su`
  - `/system/app/Superuser.apk`

- Test for write access in the `/private` directory.
- Check for root access on an Android device
- Check for the presence of an su binary inside Android

Root detection is a security control which helps applications defend against being run on compromised devices. Though, like most other security controls, this can be bypassed on the attacker's device using several techniques.

## 4.3 PRO-005 — (Android) Insecure Storage of Tokens

**Vulnerability ID:** PRO-005

**Vulnerability type:** Insecure local storage

**Threat level:** Low

### Description:

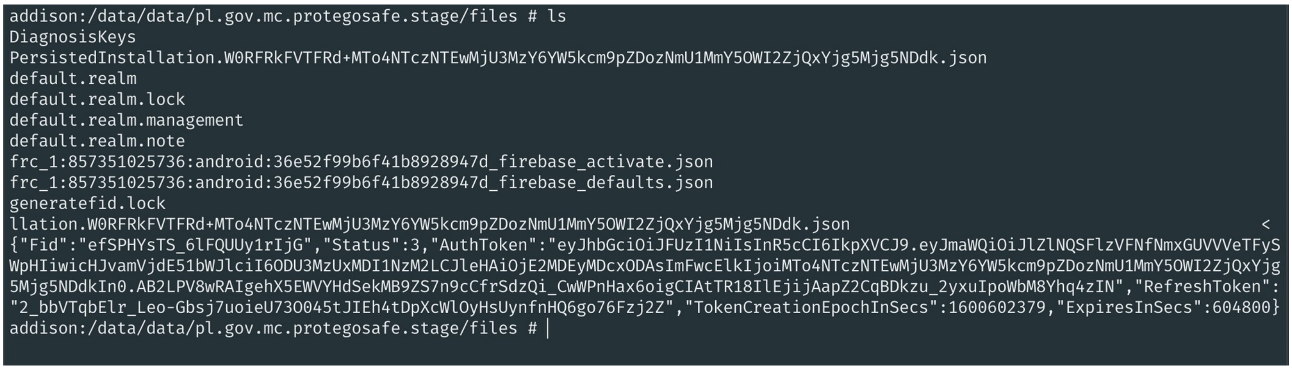
The Android app stores authentication tokens in the app's storage area in an unencrypted format.

### Technical description:

Android app stores the file `PersistedInstallation.W0RFR...[redacted]`, which includes the `AuthToken`, in the app's storage in an unencrypted format.

The following files contain a sensitive `AuthToken` in clear text format inside the app's storage in `/data/data/pl.gov.mc.protegosafe.stage/files/`

```
PersistedInstallation.W0RFR...[redacted].json
```



```
addison:/data/data/pl.gov.mc.protegosafe.stage/files # ls
DiagnosisKeys
PersistedInstallation.W0RFRkFVTFRd+MTo4NTczNTEwMjU3MzY6YW5kcm9pZDozNmU1MmY5OWI2ZjQxYjg5Mjg5NDdk.json
default.realm
default.realm.lock
default.realm.management
default.realm.note
frc_1:857351025736:android:36e52f99b6f41b8928947d_firebase_activate.json
frc_1:857351025736:android:36e52f99b6f41b8928947d_firebase_defaults.json
generatefid.lock
llation.W0RFRkFVTFRd+MTo4NTczNTEwMjU3MzY6YW5kcm9pZDozNmU1MmY5OWI2ZjQxYjg5Mjg5NDdk.json                              <
{"Fid":"efSPHYsTS_6lFQUUy1rIjG","Status":3,"AuthToken":"eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJmaWQiOiJlZlNQSFlzVFNfNmxGUVVVeTFyS
WpHIiwicHJvamVjdE51bWJlciI6ODU3MzUxMDI1NzM2LCJleHAiOjE2MDEyMDcxODAsImFwcElkIjoiMTo4NTczNTEwMjU3MzY6YW5kcm9pZDozNmU1MmY5OWI2ZjQxYjg
5Mjg5NDdkIn0.AB2LPV8wRAIgehX5EWVYHdSekMB9ZS7n9cCfrSdzQi_CwWPnHax6oigCIAtTR18IlEjijAapZ2CqBDkzu_2yxuIpoWbM8Yhq4zIN","RefreshToken":
"2_bbVTqbElr_Leo-Gbsj7uoieU73O045tJIEh4tDpXcWlOyHsUynfnHQ6go76Fzj2Z","TokenCreationEpochInSecs":1600602379,"ExpiresInSecs":604800}
addison:/data/data/pl.gov.mc.protegosafe.stage/files # |
```

### Impact:

It might be possible to use the token to access some of the Firebase services. In case of physical access to the device, or malware on the device, these tokens can be stolen and used maliciously to gain access to data/services.

### Recommendation:

- Consider encrypting the sensitive tokens/files or store them in the Android keystore.

## 4.4 PRO-006 — (Android) Web View Has Javascript Enabled

**Vulnerability ID:** PRO-006

**Vulnerability type:** Misconfiguration

**Threat level:** Low

### Description:

The Android application implements a web view with Javascript enabled.

### Technical description:

The Android application implements a web view that has `javaScriptEnabled = true` set. This allows the WebView to interpret JavaScript. JavaScript is disabled by default for WebViews and must be explicitly enabled. It should only be enabled only when strictly necessary, so as to reduce the attack surface of the app.

The web view also suppresses warnings with `@SuppressLint("SetJavaScriptEnabled")`.

**Source code snippet:**

File: `app/src/main/java/pl/gov/mc/protegosafe/ui/home/HomeFragment.kt`

```
@SuppressLint("SetJavaScriptEnabled")
private fun startPwaMigration(url: String) {
    binding.migrationLayout.isVisible = true
    binding.webView.apply {
        settings.javaScriptEnabled = true
        settings.domStorageEnabled = true
        webViewClient = object : WebViewClient() {
            override fun onPageFinished(view: WebView?, url: String?) {
                binding.webView.evaluateJavascript(DUMP_PWA_SCRIPT) { dump ->
                    pwaDump = dump
                    setUpWebView()
                }
            }
        }
        loadUrl(url)
    }
}

@SuppressLint("SetJavaScriptEnabled")
private fun setUpWebView() {
    binding.webView.apply {
        settings.javaScriptEnabled = true
        settings.domStorageEnabled = true
        webViewClient = ProteGoWebViewClient()
        addJavascriptInterface(
            NativeBridgeInterface(
                vm::setBridgeData,
                vm::getBridgeData
            ), NativeBridgeInterface.NATIVE_BRIDGE_NAME
        )
        loadUrl(urlProvider.getWebUrl())
        if (BuildConfig.DEBUG) {
            webChromeClient = object : WebChromeClient() {
                override fun onConsoleMessage(consoleMessage: ConsoleMessage): Boolean {
                    webViewTimber().d("webView console ${consoleMessage.message()}")
                    return true
                }
            }
        }
    }
```

## Impact:

Allowing Javascript to be run in a web view opens possible avenues for attacks, such as XSS.

## Recommendation:

As per OWASP, if JavaScript is necessary, the app should make sure:

- The communication to the endpoints consistently relies on HTTPS (or other protocols that allow encryption) to protect HTML and JavaScript from tampering during transmission.
- JavaScript and HTML are loaded locally, from within the app data directory or from trusted web servers only.
- The user cannot define which sources to load by means of loading different resources based on a user provided input.
- To remove all JavaScript source code and locally stored data, clear the WebView's cache with clearCache when the app closes.

## 4.5 PRO-007 — (iOS) NSURLRequests Are Being Cached

**Vulnerability ID:** PRO-007

**Vulnerability type:** Misconfiguration

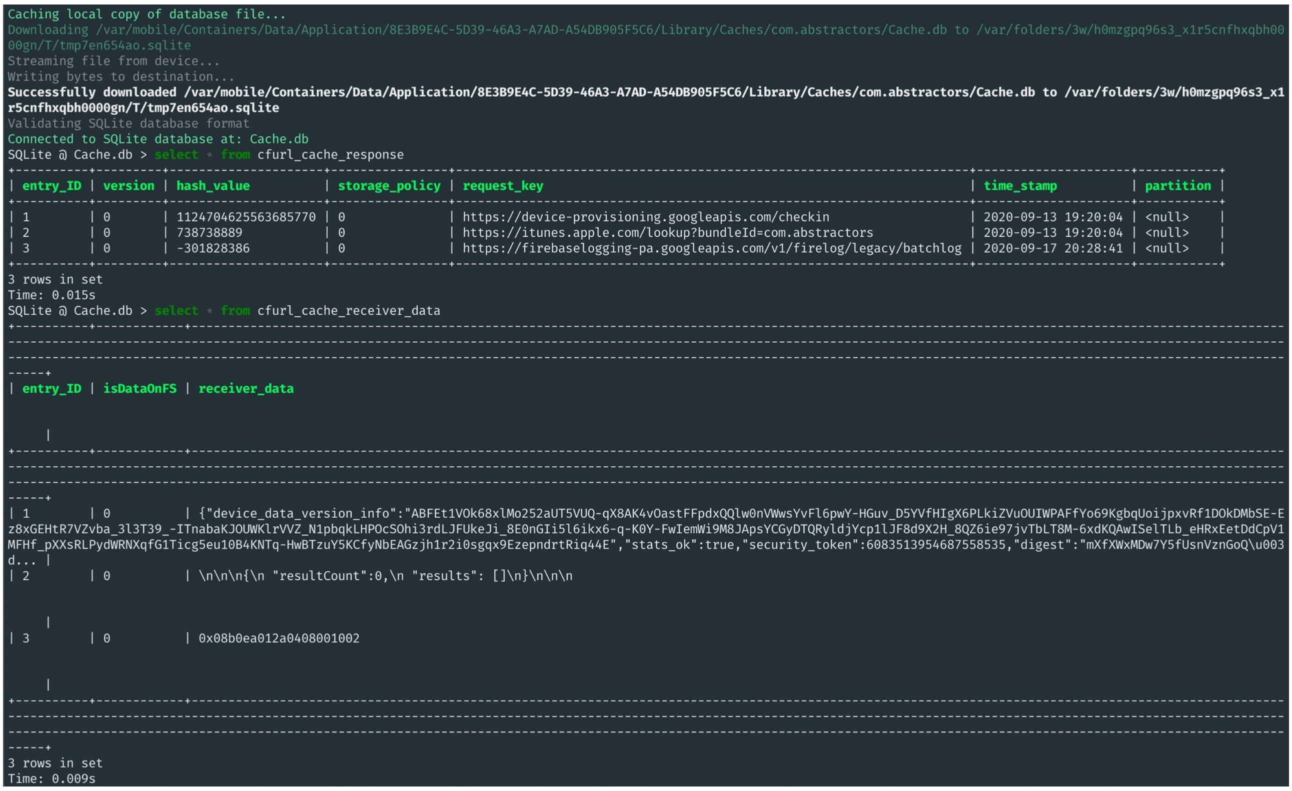
**Threat level:** Low

### Description:

In the iOS app, the NSURLRequests are cached in the `Cache.db` file.

### Technical description:

By default, iOS's NSURLRequest will cache responses in the `Cache.db` file, which is stored in `/Library/Caches/pl.gov.mc.protegosafe/Cache.db`.



```
Caching local copy of database file...
Downloading /var/mobile/Containers/Data/Application/8E3B9E4C-5D39-46A3-A7AD-A54DB905F5C6/Library/Caches/com.abstractors/Cache.db to /var/folders/3w/h0mzgpq96s3_x1r5cnfhxqbh00
00gn/T/tmp7en654ao.sqlite
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /var/mobile/Containers/Data/Application/8E3B9E4C-5D39-46A3-A7AD-A54DB905F5C6/Library/Caches/com.abstractors/Cache.db to /var/folders/3w/h0mzgpq96s3_x1
r5cnfhxqbh0000gn/T/tmp7en654ao.sqlite
Validating SQLite database format
Connected to SQLite database at: Cache.db
SQLite @ Cache.db > select * from cfurl_cache_response
+----------+---------+---------------------+----------------+------------------------------------------------------------------+---------------------+-----------+
| entry_ID | version | hash_value          | storage_policy | request_key                                                      | time_stamp          | partition |
+----------+---------+---------------------+----------------+------------------------------------------------------------------+---------------------+-----------+
| 1        | 0       | 1124704625563685770 | 0              | https://device-provisioning.googleapis.com/checkin               | 2020-09-13 19:20:04 | <null>    |
| 2        | 0       | 738738889           | 0              | https://itunes.apple.com/lookup?bundleId=com.abstractors         | 2020-09-13 19:20:04 | <null>    |
| 3        | 0       | -301828386          | 0              | https://firebaselogging-pa.googleapis.com/v1/firelog/legacy/batchlog | 2020-09-17 20:28:41 | <null>    |
+----------+---------+---------------------+----------------+------------------------------------------------------------------+---------------------+-----------+
3 rows in set
Time: 0.015s
SQLite @ Cache.db > select * from cfurl_cache_receiver_data
+----------+-------------+------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----+
| entry_ID | isDataOnFS | receiver_data


     |
+----------+-------------+------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----+
| 1        | 0           | {"device_data_version_info":"ABFEt1VOk68xlMo252aUT5VUQ-qX8AK4vOastFFpdxQQlw0nVWwsYvFl6pwY-HGuv_D5YVfHIgX6PLkiZVuOUIWPAFfYo69KgbqUoijpxvRf1DOkDMbSE-E
z8xGEHtR7VZvba_3l3T39_-ITnabaKJOUWKlrVVZ_N1pbqkLHPOcSOhi3rdLJFUkeJi_8E0nGIi5l6ikx6-q-K0Y-FwIemWi9M8JApsYCGyDTQRyldjYcp1lJF8d9X2H_8QZ6ie97jvTbLT8M-6xdKQAwISelTLb_eHRxEetDdCpV1
MFHf_pXXsRLPydWRNXqfG1Ticg5eu10B4KNTq-HwBTzuY5KCfyNbEAGzjh1r2i0sgqx9EzepndrtRiq44E","stats_ok":true,"security_token":6083513954687558535,"digest":"mXfXWxMDw7Y5fUsnVznGoQ\u003
d... |
| 2        | 0           | \n\n\n{\n "resultCount":0,\n "results": []\n}\n\n\n


     |
| 3        | 0           | 0x08b0ea012a0408001002


     |
+----------+-------------+------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----+
3 rows in set
Time: 0.009s
```

### Impact:

Storing sensitive data in unencrypted format in an SQLite database is not a secure practice. Such data might be leaked to an attacker/malicious app if the app is running on a compromised device (see PRO-004 (page 13)).

### Recommendation:

Set the cachePolicy property of the NSURLRequest to disable the caching of HTTP(S) requests and responses. For example:

```
(NSCachedURLResponse)connection:(NSURLConnection)connection willCacheResponse:
(NSCachedURLResponse *)cachedResponse { return nil;
```

For other methods of disabling the caching of HTTP(S) requests and responses, please refer to the Apple Developer article "Understanding Cache Access": https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/URLLoadingSystem/Concepts/CachePolicies.html

## 4.6 PRO-008 — Weak TLS Cipher Suites Allowed

**Vulnerability ID:** PRO-008

**Vulnerability type:** Transport layer security

**Threat level:** Low

### Description:

Some back-end endpoints allow clients to choose weak cipher suites.

### Technical description:

The following weak TLS cipher suites are allowed on `https://gat-stage.safesafe.app` and `https://udk-stage.safesafe.app` back-end services:

```
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
```

## Impact:

The `TLS_RSA` suites are considered to be weak because of the ROBOT attack; TLS 1.2 contains specific mitigations to prevent such attacks. The CBC mode ciphers are vulnerable to plain-text attacks in TLS 1.0 and lower. A fix is implemented with TLS 1.2 with GCM mode which is not vulnerable to the BEAST attack.

## Recommendation:

- Require a minimum of TLSv1.2 to eliminate issues in older TLS versions.
- Disable all CBC-mode cipher suites; prefer GCM-mode suites instead.
- Disable suites using RSA key exchange to mitigate the ROBOT attack, and ensure your server's private key is unique (i.e. do not use the same private key across multiple back-end servers).

None of these recommendations should present any issues for iOS, since these TLS settings are compatible with Apple's ATS requirements introduced in iOS 9.0 in 2015. GAEN requires iOS 13.5 (Apple docs https://developer.apple.com/documentation/exposurenotification), which is much more recent and has better TLS support. Android's inherently slow uptake of operating system updates means that far more outdated devices are found in the wild, so whether you are able to make all these changes depends on exactly which version of Android you want to support as a minimum. GAEN suggests a low limit of Android 6.0 using API level 23 (see Google's docs: https://support.google.com/googleplay/answer/9888358?hl=en and https://developers.google.com/android/exposure-notifications/exposure-notifications-api), with a possibility of supporting *some* 5.0 devices with API level 21. Android 6.0 still has usable cipher suites available when these weak ones are disabled, according to Qualys SSL Labs (https://www.ssllabs.com/ssltest/viewClient.html?name=Android&version=6.0&key=129), however, this will require testing with minimum OS versions to be certain of what will work.

# 5 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

## 5.1 NF-002 — Test Cases

The following test case have been performed on the Android & iOS apps:

- Tests to find sensitive information hardcoded in the application's source code
- Tests to find insecure webview implementation
- Tests against sensitive information leakage through device logs
- Tests against OTP brute force on the endpoint`https://gat-stage.safesafe.app/getAccessToken`
- Tests against the JSON Web Tokens, generated through the endpoint`https://gat-stage.safesafe.app/getAccessToken` 1. Sensitive data inside JWT payload 2. Brute forcing for JWT secret key (Wordlist brute force, Hashcat) 3. Signature validation check 4. Tests for alg:none implementation
- Tests against use of insecure libraries
- Tests for insecure local storage of data
- Tests for insecure transport layer security
- Tests for insecure application components eg; exported activities/broadcasts.
- Tests for unused/extraneous permissions
- Tests against unintentional data leakage
- Tests to check if keyboard cache is disabled on text inputs that process sensitive data
- Tests for sensitive data exposure via IPC mechanisms
- Tests to check insecure deeplink/URL schemes
- Tests to check if the app uses symmetric cryptography with hardcoded keys
- Tests to check if the app uses cryptographic protocols or algorithms that are widely considered depreciated for security purposes
- Test to check if the app's release built has appropriate settings for a release build (e.g. non-debuggable)
- Test to check if the app removes sensitive data from views when backgrounded
- Tests to check if the app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use
- Tests to check if any exceptions are listed in the ATS (iOS app)

# 6 Future Work

- **Implement Additional Security Controls**
  Beyond addressing the findings detailed in this report, we also recommend implementing two more security controls: dummy TEK upload sequences, and dummy analytics uploads. The Italian Immuni app (https://github.com/immuni-app) implements these features which make it difficult for an attacker to guess or find out whether a person is COVID-19-positive. See how the Immuni app implements them: https://github.com/immuni-app/immuni-documentation/blob/master/Traffic%20Analysis%20Mitigation.md.
- **Retest of findings**
  When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to confirm that they are effective and have not introduced other security problems.
- **Regular security assessments**
  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

# 7 Conclusion

We discovered 1 Elevated-severity issue and 5 Low-severity issues during this penetration test.

The impact of the elevated-severity issue considered high as it could lead to flooding of the service with malicious data (TEK uploads).

The low-severity issues are related to insecure local storage, insecure webview settings, and missing security controls. Taken together, these low-severity issues can adversely affect the application's integrity and trustworthiness, which may undermine public trust. We strongly recommend implementing the suggested security controls and fixing all the security vulnerabilities we found.

We want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Radically Open Security is the world's first not-for-profit computer security consultancy. We operate under an innovative new business model whereby we use a Dutch fiscal entity, called a “Fiscaal Fondswervende Instelling” (Fiscal Fund raising Institution), as a commercial front-end to send 90% of our profits, tax-free, to a not-for-profit foundation, Stichting NLnet. The NLnet Foundation has supported open-source, digital rights, and Internet research for almost 20 years.

In contrast to other organizations, our profits do not benefit shareholders, investors, or founders. Our profits benefit society. As an organization without a profit-motive, we recruit top-name, ethical security experts and find like-minded customers that want to use their IT security budget as a "vote" to support socially responsible entrepreneurship. The rapid pace of our current growth reflects the positive response the market has to our idealistic philosophy and innovative business model.

If you have any questions about the advice in this report, please contact us at info@radicallyopensecurity.com

For more information about Radically Open Security and its services please visit our website:
www.radicallyopensecurity.com.

# Appendix 1 Testing team

| | |
|---|---|
| Abhinav Mishra | Abhinav is a Senior Security Consultant with 9+ years of experience in hacking Web, Mobile apps and Infrastructure. He is an active speaker and trainer, at various security conferences/events. |
| Melanie Rieback | Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security. |

Front page image by Slava (https://secure.flickr.com/photos/slava/496607907/), "Mango HaX0ring",
Image styling by Patricia Piolon, https://creativecommons.org/licenses/by-sa/2.0/legalcode.