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 iOS App & Crypto 01.2020

Cure53, Dr.-Ing. M. Heiderich, BSc. C. Kean, Dr. N. Kobeissi

Index

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 documents the findings of a security review targeting the FlowCrypt application complex. Carried out by Cure53 in January 2020, this project investigated security aspects of the rather ‘young’ FlowCrypt compound, which includes a set of mobile applications and browser extensions. The broad aim of FlowCrypt is to facilitate the use of PGP for emails on mobile devices and desktop computers.

It should be clarified that this January 2020 marks the first of several auditing exercises that Cure53 is expected to conduct against the FlowCrypt. For this project, the FlowCrypt iOS application took center stage. While the assessment focused on application security and cryptography, the latter was inspected both in terms of integration and implementation.

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

As for the resources, three members of the Cure53 team completed this project over the course of six person-days in mid-to-late January 2020. The involved testers focused on areas best corresponding to their respective expertise. Expected level of coverage has been reached in the given budget.

A white-box methodology was employed as the most-suitable for the goals and operations performed by the FlowCrypt iOS complex. The testing team was granted access to all relevant sources, binaries of the application, as well as documentation detailing security architecture. A dedicated document created to shed light on security issues and threats known to the FlowCrypt iOS application team was also shared with Cure53. The in-house team informed the testers about specific areas and questions they considered as investigation-worthy from the maintainers' perspective.

The project started on time and involved close collaboration of Cure53 and the team at FlowCrypt. The communications during the test were done on a dedicated private Slack channel into which Cure53 invited relevant FlowCrypt personnel. At an early stage, it was necessary for the exchanges to be quite frequent and numerous, since the initially supplied binaries were not working as expected. Once the technical issues were resolved, the work progressed in a productive and efficient manner. Cure53 took advantage of live-reporting issues and tracked the bugs together with the maintainers. Besides status updates, Cure53 was also available to answer questions posed by the FlowCrypt team.

Given that the software is still in a rather 'young' phase of development, it is unsurprising to see a limited number of issues being spotted. This verdict should be read with the caveat that this situation may change when later and more mature stages are reached. Especially the future addition of security-relevant features can be expected to concurrently increase the attack surface exposed by FlowCrypt. During the January 2020 project, Cure53 managed to spot three findings. Two were given severity rankings of *High* and one should be viewed as *Medium.* With few issues, Cure53 enriches this report with a dedicated chapter on methodology, which illustrates what has been done in terms of testing and with what rationales. This section is also meant to help track the answers to the questions asked by the maintainers in a more structured way.

In the following sections, the report will first present the areas featured in the test's scope in more detail, including notes on the delineated work packages (WPs). Given the low number of findings, Cure53 extends this report with a navigational test coverage section, as explained above. The discussions then move on to three tickets which present technical aspects of the discoveries. The report closes with a conclusion in which Cure53 summarizes this early 2020 assessment of FlowCrypt. The testers issue a verdict about the security premise of the investigated scope. Besides general feedback,

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

Cure53 highlights some areas that could possibly be improved further with some advanced security approaches.

Scope

- **Cryptographic Review & Security Audit against FlowCrypt iOS App & PGP Crypto**
  - **WP1**: Crypto Review and Audit against Design & Implementation on iOS
    - Material was shared with Cure53 to illustrate the inner-workings of the FlowCrypt-utilized cryptography
  - **WP2**: Penetration Test & Source Code Audit against FlowCrypt iOS Mobile App
    - Sources have been shared with Cure53 and the build instructions were provided.
    - Pre-built binaries were shared with Cure53 as well to enable testing reminiscent of the production state of the project.

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.

WP1: Crypto Review and Audit against Design & Implementation on iOS

A list of items below seeks to detail the tasks completed during the cryptography-centered portion of 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 security items within FlowCrypt.

- The PGP parsing engine in *pgp-armor.ts* and *pgp-msg.ts* was tested for sanity. The focus was placed on the desired absence of attacks that could lead to messages being shown with misleading security properties or guarantees.
- FlowCrypt's PGP passphrase strength enforcement scheme was evaluated.
- FlowCrypt's bindings to *OpenPGP.js* from the mobile application was examined *(Swift <-> JS bridge*).
- FlowCrypt's sources for secure random/byte generation were evaluated.
- Handling of key material was inspected.
- Investigations zoomed in on transitions between *Swift - TypeScript - Swift - C*, and correctness of combining cryptographic functionality (*CoreHost* class) with *OpenPGP.js.*
- Exception handling for cryptographic functions was checked.

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

WP2: Penetration Test & Source Code Audit against FlowCrypt iOS Mobile 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 FlowCrypt iOS application successfully blocked the loading for all remote content. The *Cure53 HTTPLeaks*[2] payload was invoked as a means to collect information on IP address, operating system, browser version and the time at which the email was opened.
- The OpenPGP implementation of FlowCrypt was tested for the EFail bug which leaks the plaintext 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.
- Furthermore, the iOS app was tested for post-decryption XSS by sending encrypted XSS payloads to the FlowCrypt inbox. However, the attack has proven futile as the content of the email is not embedded in a webview.
- The local storage of the FlowCrypt iOS application was examined via an SSH connection[4] on a jailbroken device on version iOS 13 with the *checkra1n* exploit[5].
- The local storage was scanned for sensitive credentials in particular in binary files which led to the discovery of private keys and passphrases stored in plaintext, as documented in FLO-01-002 and FLO-01-003. It can be assumed that these files will be accessible for Apple when stored as part of an iCloud backup. Hence, it makes sense to employ an additional layer of cryptography to protect secrets in case of an iCloud compromise.
- The mobile app's network communications were also reviewed. It was found that plaintext HTTP communications are not in use. The team also attempted to intercept TLS traffic with invalid certificates, which the application correctly rejected. Under the premise of correctly implemented cryptography, it can be assumed that email providers cannot access or manipulate client files as long as end-to-end encryption and digital signatures are employed. However, this could not be verified as digital signatures were not yet supported at this stage of development.

[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/
[4] https://cydia.saurik.com/package/openssh/
[5] https://checkra.in/

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

- As iOS employs sandboxing to prevent apps from accessing other users' local storage, it can be assumed that the FlowCrypt local storage is secure when it comes to third-party app access. However, this countermeasure might vanish in a jailbroken or similarly altered iDevice.
- The iOS device logs were examined for potential information leaks from the app. However, no information leaks were discovered during or after usage of the application.

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-01-001*) for the purpose of facilitating any future follow-up correspondence.

FLO-01-001 OOS: FlowCrypt browser extension prone to HTTP leaks *(Medium)*

It was found that the FlowCrypt browser extension is vulnerable to HTTP leaks via a crafted email body in a default configuration. 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. This issue was confirmed by using the public HTTP leaks collection from Cure53[6] and sending them as an HTML email body to all clients. The path of the URL indicates which payload successfully leaked data.

The following list shows the identified leaking elements for the FlowCrypt browser extension.

Recorded leaks:

```
/video-source-src
/input-src
/svg-image-xlink-href
/img-src
/svg-image-href
/video-src
/image-src
/audio-source-src
/audio-src
/picture-img-srcset
/track-src
```

[6] https://raw.githubusercontent.com/cure53/HTTPLeaks/master/leak.html

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

```
/video-poster-2
/img-srcset
```

The HTTP leak results are summarized in the table next.

Client	Number of leaks
iOS	0
GMail Webmail FlowCrypt Integration	13
FlowCrypt Encrypted Inbox	13

It is recommended to disable loading external content by implementing a Content Security Policy (CSP). Best practice in this case would be to implement a CSP that sets all source directives to *none*. The CSP can be assembled with a CSP generator[7] and would provide a policy in the following format:

```
Content-Security-Policy: default-src 'none'; script-src 'none'; style-src
'none'; img-src 'none'; font-src 'none'; connect-src 'none'; media-src 'none';
object-src 'none'; child-src 'none'; frame-src 'none'; worker-src 'none'
```

This approach will further harden the FlowCrypt extension in a sense of providing defense-in-depth.

Note: *The FlowCrypt team fixed this issue during the audit by hardening the sanitization method for the HTML markup. It was noted that a CSP might be implemented in the future. The fix was reviewed, retested and confirmed by Cure53.*

FLO-01-002 iOS: Lack of filesystem protections exposes private key *(High)*

It was found that the iOS app does not take advantage of the native iOS filesystem protections and fails to fully protect some of its data files at rest. The affected files are only protected until the user authenticates for the first time after booting the phone. The problem is that the key to decrypt these files will remain readable in memory while the device is locked. The impact of this issue was evaluated as *High* because private keys and passphrases remain unprotected.

This issue requires physical access to an iDevice set to a locked screen and a method of accessing the local storage, for instance, via SSH connection established via a jailbreak.

[7] https://report-uri.com/home/generate

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

While being locked, the files below represented some of the data that remained unprotected.

Command:
```
tar cvfz files_locked.tar.gz *
```

Output:
```
Documents/default.realm
Library/Caches/com.flowcrypt.ios.testflight/Cache.db
Library/Caches/com.flowcrypt.ios.testflight/Cache.db-shm
Library/Caches/com.flowcrypt.ios.testflight/Cache.db-wal
[...]
```

Affected File:
/private/var/mobile/Containers/Data/Application/923422B0-8157-4D99-8D13-75D1BD09FE6F/Documents/default.realm

Affected Content:
```
-----BEGIN PGP PRIVATE KEY BLOCK-----
[...]
-----END PGP PRIVATE KEY BLOCK-----
-----BEGIN PGP PUBLIC KEY BLOCK-----
[...]
-----END PGP PUBLIC KEY BLOCK-----
[Passphrase]
```

In order to solve the problem related to file-access, it is recommended to implement the *NSFileProtection-Complete* entitlement at the application-level[8]. SQL cipher[9] could be considered to improve the SQLite database protections further. To prevent unintended information leaks, it is advised to only store sensitive data like private keys or passphrases in an encrypted form.

Note: *The FlowCrypt team fixed this issue during the audit by implementing the NSFileProtection-Complete entitlement for sensitive files. The fix was reviewed, retested and confirmed by Cure53.*

[8] https://developer.apple.com/library/ios/documentation/iP...App/StrategiesforImplementingYourApp.html
[9] https://www.zetetic.net/sqlcipher/ios-tutorial/

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

FLO-01-003 iOS: RealmDB drops encryption for uncaught KC condition *(High)*

During the discovery of FLO-01-002, it was found that the *RealmDB* file exposes the private key and passphrase. During consultations with the FlowCrypt development team, it was determined that this file is supposed to be encrypted. The development team discovered that the encryption of *RealmDB* is dropped when the keychain (KC) returns an uncatched *nil* value. The impact of this issue was evaluated as *High* because an unencrypted *RealmDB* would expose private key and passphrase in unencrypted device backups.

Affected File:
FlowCrypt/Common/Services/Encrypted Storage/EncryptedStorage.swift

Affected Code:

```
private var encryptedConfiguration: Realm.Configuration? {
        guard canHaveAccessToStorage else { return nil }
        let key = self.keychainService.getStorageEncryptionKey()
        return Realm.Configuration(encryptionKey: key)
    }
```

The applied fix is listed below. In case the keychain returns a *nil* value, the application will crash with a fatal error instead of dropping the encryption.

Affected File:
FlowCrypt/Common/Services/Encrypted Storage/KeyChainService.swift

Affected Code:

```
guard let validKey = keyFromKeychain as? Data
else { fatalError("KeyChainServiceType getStorageEncryptionKey keyFromKeychain
not usable as Data. Is nil?: \(keyFromKeychain == nil)")}
```

Note: *The FlowCrypt team fixed this issue during the audit by implementing a catch clause for unexpected keychain return values. The fix was reviewed, retested and confirmed by Cure53.*

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

Conclusions

The results of this Cure53 examination of the FlowCrypt iOS mobile applications and cryptographic premise are generally positive. After spending six days on the scope in January 2020, three members of the Cure53 team can conclude that the scope was well-protected against the majority of severe compromise attempts. However, it must be underscored that the examined compound is currently at its very early stages of development, so reaching a holistic and conclusive verdict on the non-final security premise would be ill-advised.

Despite reservations around maturity, Cure53 is quite optimistic about the FlowCrypt iOS app and its surroundings as regards security. 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 three discoveries have been resolved quickly and the lessons learned are likely to contribute to a solid foundation for future development.

Said foundation includes subtleties in the handling and implementation of the local storage, as demonstrated in FLO-01-002 and FLO-01-003, as well as the restriction of remote content noted in FLO-01-001. Beyond that, the testing methodology section in this report provides further attack vectors to look out for as the FlowCrypt complex grows. In essence, two issues with the severity of *High* were spotted on the scope, while the third finding represents a coincidental out-of-scope *Medium* threat in the browser extension.

It is clear to Cure53 that all features that will be introduced next, should be subject to particular scrutiny in future tests. This needs to include core features, like additions to the local storage in the form of backups, as well as the support for digital signatures and attachments. Cure53 positively noted that the FlowCrypt team addressed all discovered issues in a timely manner and enabled quick retest of the fixes. All of the verifications performed during this audit were conducted directly on the application and in the source code, so as to expand checks for possible bypasses.

Regarding cryptography, it was possible to test OpenPGP's integration into the *"TS Core"* layer as well as the basic PGP operations over the Swift bridge. Similarly, Cure53 could assess the cryptographic primitive implementations used. On the contrary, many of the core functionalities of a regular OpenPGP application were missing from the provided build, which significantly hindered the extent of testing. For example, no private key synchronization or backup method was supplied. In the same vein, both signing support and identity management were extremely limited overall.

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

Therefore, while the cryptographic assessment yielded no issues, it is paramount for the FlowCrypt iOS app to be re-tested at a later time. This will be necessitated by implementing additional functionality in the cryptographic layer. For now, Cure53 can only state that the furnished simple and incomplete application-layer implementation/integration of the OpenPGP stack can be seen as correct.

To conclude, Cure53 believes that this January 2020 provided evidence about the FlowCrypt iOS application and its cryptography being quite secure at its early developmental stage. Based on the test limitations, however, it would be irresponsible to issue a conclusive verdict. The inspected 'young' software seems promising, yet additional test iterations will add more clarity. It is hoped that future tests will be more effective in contributing to a better understanding of security's "bigger picture" at FlowCrypt.

Cure53 would like to thank Tomáš Holub from the FlowCrypt team for his excellent project coordination, support and assistance, both before and during this assignment.