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 Addon & Crypto 03.2020

Cure53, Dr.-Ing. M. Heiderich, BSc. T.-C. “Filedescriptor” Hong, Dr. N. Kobeissi

Index

Introduction

> *“FlowCrypt is an email solution that lets you work with end-to-end encrypted messages securely and easily.”*
>
> From https://flowcrypt.com/docs/

This report presents the findings of a security assessment targeting the FlowCrypt Add-on, as well as its cryptography. Cure53 conducted this project in March 2020 and documented six security-relevant findings on the scope.

It should be noted that this is a second iteration of security-centered collaboration between Cure53 and FlowCrypt. The first round of testing and auditing of the FlowCrypt complex took place just a few months prior, in January 2020. Back then, Cure53 focused on the FlowCrypt iOS application and the cryptographic code it relies on. The results can be found in the report labelled as *FLO-01.* As a next stage, in Calendar Week 11, Cure53 homed in on the browser extension of FlowCrypt, as well as the PGP cryptography and bindings.

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

As for the resources, the March 2020 assessment was completed by three members of the Cure53 team. They spent a total of six days on the scope and worked across two Work Packages formulated to best tackle all items in scope. Specifically, WP1 focused on the review and audit of FlowCrypt's cryptographic implementation found on the add-on. In WP2, the examination focused on penetration tests and source code auditing of the FlowCrypt browser add-on itself.

Zooming in on the approaches, Cure53 got access to sources, documentation and pre-built binaries, so as to obtain optimal coverage and benefit from ideal testing conditions. The chosen methodology was therefore white-box, just as in *FLO-01.* Communications were again done in a private and test-dedicated Slack channel into which Cure53 invited the FlowCrypt maintainers. Discussions were always productive and helpful.

As noted, six flaws were noted. Four were classified as security vulnerabilities and two belong to a wider-scoped category of general weaknesses, typically marked by lower exploitation potential. The highest severity level reached by a finding was *Medium*, which makes for a rather positive impression.

In the following sections, the report will first shed light on the scope and key test parameters, as well as test-data and materials available to Cure53 during this round of testing. Next, all findings will be discussed in a chronological order alongside technical descriptions, impact notes, as well as PoC and mitigation advice when applicable. Finally, the report will close with conclusions about this March 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 the FlowCrypt browser extension and broader cryptography are also incorporated into the final section.

Note: *Several issues spotted and reported in this audit were already addressed and the fixes were verified by Cure53 (*FLO-02-002*,* FLO-02-005*,* FLO-02-006*).*

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

Scope

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

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-02-001*) for the purpose of facilitating any future follow-up correspondence.

FLO-02-002 Extension: Clickjacking on web accessible resources *(Medium)*

It was found that several web accessible resources do not restrict which origins can embed them. This allows an attacker to embed them using an Iframe and overlay something on top of it. As a result, users may be tricked into clicking on something other than what they actually intended to click on. The resulting attack strategy is known as Clickjacking[1]. In this case, a malicious website can embed *compose.htm* and trick victims into sending an email.

PoC (Replace the logged-in email address in the highlighted area):

```
data:text/html,<iframe style=opacity:.1
src=chrome-extension://emghkmengffbgcnpcajidfdkffaikghk/chrome/elements/
compose.htm?frameId=&acctEmail=EMAIL_ADDRESS&parentTabId=>
```

[1] https://en.wikipedia.org/wiki/Clickjacking

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

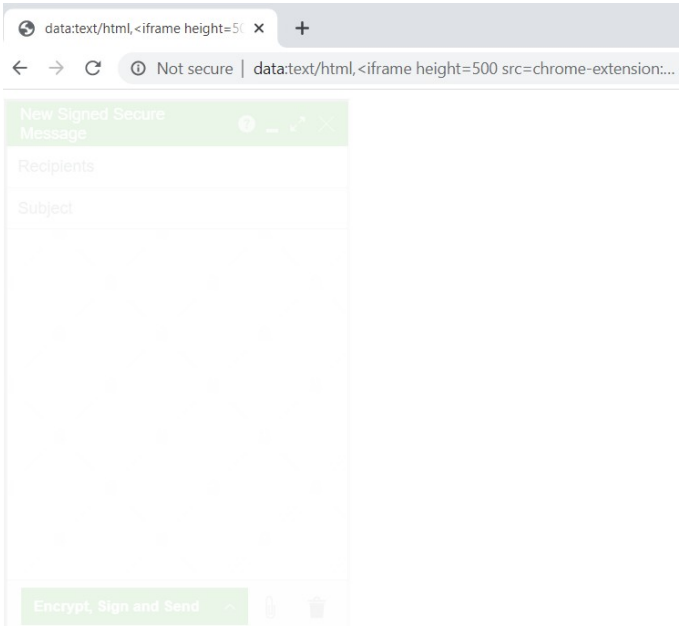

Fig.: External site embedding the compose page

It is recommended to restrict origins which can embed sensitive web accessible resources. This can be done by setting up *frame-ancestors* for CSP.

Note: *The FlowCrypt team fixed this issue during the audit by implementing the recommended changes. The fix was reviewed, retested and confirmed by Cure53.*

FLO-02-004 Extension: HTML Injection in error message on certain pages *(Low)*

A HTML injection was found on *pgp_block.htm* via the error message. This is because the returned message from the API call is rendered as HTML. However, this issue does not introduce XSS because the rendered content is sanitized and only allowed safe tags. Besides, CSP of WebExtension will block this attack nevertheless. At the same time, this could still be used for Phishing.

PoC (Replace the logged-in email address in the highlighted area):

```
chrome-extension://emghkmengffbgcnpcajidfdkffaikghk/chrome/elements/
pgp_block.htm?frameId=&hasPassword=___cu_false___&msgId=%3Cs
%3E&senderEmail=&isOutgoing=___cu_true___&acctEmail=EMAIL_ADDRESS&parentTabId=
```

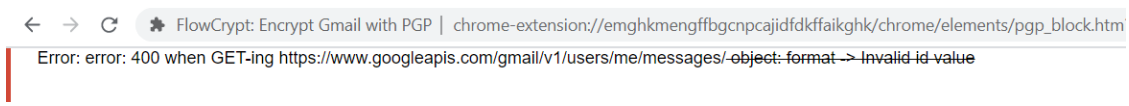

Fig.: Error message rendered as HTML

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

It is recommended not to render error messages as HTML, especially since the current handling is not necessary in this use-case.

FLO-02-005 Extension: Path traversal to Google API calls via *msgId* *(Low)*

Following the discovery of FLO-02-004, it was further found that the user-supplied *msgId* parameter was used in the API call to Google without validation of its value. Since Google API takes the parameter from the path, an attacker can supply a value that can be used for path traversal (../) and invoke other APIs instead of the intended one.

PoC (Replace the logged-in email address in the highlighted area):

```
chrome-extension://emghkmengffbgcnpcajidfdkffaikghk/chrome/elements/
pgp_block.htm?frameId=&hasPassword=___cu_false___&msgId=../../../../
&senderEmail=&isOutgoing=___cu_true___&acctEmail=EMAIL_ADDRESS&parentTabId=
```

Monitor the network traffic and note the API call is made to https://www.googleapis.com/gmail/ instead of https://www.googleapis.com/gmail/v1/users/me/messages/. It is recommended to validate the affected parameter before passing it to the API call.

Note: *The FlowCrypt team fixed this issue during the audit by implementing the recommended changes. The fix was reviewed, retested and confirmed by Cure53.*

FLO-02-006 Extension: CSS sanitization can be bypassed *(Low)*

FlowCrypt tries to prevent HTML leaks by sanitizing *style* attributes containing the keyword *url(,* which can then be used to initiate outbound requests. The code is flawed as it looks for the exact keyword but does not handle cases where the keyword is written in uppercase. As a result, using *Url(* will bypass the sanitization.

Affected File:
extension/js/common/platform/xss.ts

Affected Code:

```
if (style && (style.includes('url(') || style.includes('@import'))) {
        (node as Element).removeAttribute('style'); // don't want any leaks
through css url()
}
```

It is recommended to mitigate the issue by considering and accounting for alternative possibilities to circumvent the blacklist.

Note: *The FlowCrypt team fixed this issue during the audit by implementing the recommended changes. 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

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-02-001 Extension: Private key file backup stored directly in inbox *(Info)*

It was observed that FlowCrypt stores the user's PGP private key file directly as an attachment to an email, which is then saved within the user's inbox. The only protection for the private key file, therefore, is the standard PGP passphrase provided during the key setup process, with no further protection deployed.

This behavior is documented as a conscious design decision by the FlowCrypt team, owing to the difficulty of managing a UX scenario where users would potentially have to manage two different passphrases (i.e. their PGP passphrase and an additional passphrase used for further securing the backup). While this is understandable, the end result of the current implementation is such that Gmail (and therefore Google) obtains a copy of the user's private key file immediately upon the creation of an account.

Even though this key file is indeed still encrypted with its PGP passphrase, this may not be a sufficient safeguard for reasons named next.

- **PGP private keys are rarely, if ever, rotated:** It is the nature of PGP keys to function as long-term key pairs, rotated once in intervals many years apart, if at all. This means that once a PGP key file is given to Google or any other party with access to the user's email account, this entity may then conceivably have the ability to conduct key search attacks on the file with a generous time window.
- **No passphrase strength guarantees are enforced on imported keys:** It is often the case that PGP users specify private key passphrases without the intention of ever using those keys inside a web browser. When first installing FlowCrypt, these users may be prompted to import those keys, which may include passphrases that were chosen without FlowCrypt-specific web/Gmail attack surface taken into consideration. FlowCrypt does not impose any restrictions on the passphrases for the existing imported keys, rendering them perhaps unsuitable for encrypting key files that are immediately stored on third-party servers or email inboxes.

Considering explanations outlined above, it is recommended to adopt one of the following best-practice suggestions:

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

- **Prompt to change weak passphrase for the existing keys on import:** Simply check whether the passphrase for the existing keys is sufficiently strong on input. If it turns out not to be the case, force the user to make a change before rendering the key pair usable with FlowCrypt / backing it up inside the Gmail account.
- **Automatically upgrade PGP key passphrase:** Change the backup keys' passphrase by default to the result of the Scrypt hash of the same passphrase, importantly doing so with sufficiently strong password hashing parameters[2]. This allows users to keep the same passphrase while still strengthening it against backup email compromise.

FLO-02-003 Extension: Deprecated OAuth flow in Gmail integration *(Info)*

It was found that the Gmail integration relies on a deprecated OAuth flow for retrieving the *access* token. According to the documentation: *"This option is deprecated for OAuth 2.0 to Google. It was designed for embedded browsers, or web-views".*[3] Using this flow might allow other extensions to steal the *access* token intended for FlowCrypt.

Affected File:
extension/js/common/api/google-auth.ts

Affected Code:

```
GoogleAuth.OAUTH = {
    client_id: '717284730244-
ostjo2fdtr3ka4q9td69tdr9acmmru2p.apps.googleusercontent.com',
    url_code: `${GOOGLE_OAUTH_SCREEN_HOST}/o/oauth2/auth`,
    url_tokens: `${GOOGLE_API_HOST}/oauth2/v4/token`,
    url_redirect: 'urn:ietf:wg:oauth:2.0:oob:auto',
```

It is recommended to consider using the methods suggested by Google[4] and specifically designed for WebExtensions.

[2] https://blog.filippo.io/the-scrypt-parameters/
[3] https://developers.google.com/identity/protocols/oauth2/native-app
[4] https://developer.chrome.com/extensions/tut_oauth

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

Conclusions

The results of this Cure53 assessment demonstrate a stable situation of the FlowCrypt complex in terms of security. Three members of the Cure53 team, who examined the scope over six person-days in March 2020, can only confirm that the impression gained during the *FLO-01* project holds and the compound can be seen as moving towards an increasingly secure posture.

To comment on specific areas starting with cryptography, Cure53 performed close checks of the PGP parsing. This was requested by the customer and yielded no findings, although the potential for parser-level Denial-of-Service could not be ruled out, due to the nature of the data format found in PGP. This was discussed with the client further during a briefing over the phone.

Notably, all *OpenPGP.js* bindings were checked for sanity, as well as in relation to the usage of included cryptographic primitives. Further, Cure53 investigated the key import, email composition, signing and encryption functionalities. This was paired with an in-depth look into how FlowCrypt deals with key backup. On the latter, recommendations have been suggested in FLO-02-001 and can be considered as a response to inappropriate exposure of the users' private key files to Google/ Gmail servers. Finally, the *secure compose* feature used by FlowCrypt was briefly assessed from a cryptographic standpoint. In this realm, the majority of the attack surface appears to not be related to cryptography but rather to web stack attacks.

Moving on to security of the browser extension, one can start by saying that its requested permissions are checked to ensure no unnecessary items appear. This realm is handled well. Next, FlowCrypt extension checks content scripts to make sure they are only injected into designated pages. Potential for XSS was further examined to guarantee that crafted email or malicious websites cannot affect the extension through Gmail. No issues were identified in this regard.

Web accessible resources were investigated to ensure no sensitive pages become accessible to external parties. Due to no restrictions on embedding origins, Cure53 confirmed a Clickjacking flaw (FLO-02-002). Among other mistakes, HTML injection could be achieved via error messages on the extension's page (FLO-02-004). It was then found that the user-supplied parameter is not properly validated and can introduce path traversal to the Google API call (FLO-02-005).

The inspection centered on the overall XSS sanitization revealed a bypass in HTTP leaks protection, namely in the form of a lax blacklist on CSS keywords (FLO-02-006). It should be noted that the code quality should be improved in the aspects responsible for

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

XSS. This is because FlowCrypt exhibits many layers of sanitization/string concatenation, which additionally follow their own logic. This makes them hard to track and a perfect candidate for introducing bypasses.

The integration between Gmail API and FlowCrypt was also examined. A minor issue regarding a deprecated auth flow was identified (FLO-02-003). In some cases, this might allow a malicious WebExtension to steal the access token intended for FlowCrypt. Finally, the logic for organization rules/settings was briefly checked and no issues were found in this regard.

All in all, this March 2020 project has left Cure53 with a positive impression in terms of security noted in cryptography and browser extension of the FlowCrypt complex. It must be highlighted that the observed HTML sanitization can be characterized as quite adventurous and, as such, somewhat problematic. In Cure53's expert opinion, refactoring is paramount for this part of the scope. Otherwise, the browser extension appears sound and seems to be on the right track. The impressions gained during the previous project reported under *FLO-01* are generally mirrored across other scope components.

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.