Fine penetration tests for fine websites

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

Pentest-Report MyCrypto App 06.-07.2018

Cure53, Dr.-Ing. M. Heiderich, BSc. F. Fäßler, M. Kinugawa,
BSc. T.-C. “Filedescriptor” Hong, J. Larsson

Index

Introduction

> *“MyCrypto is an open-source, client-side tool for generating ether wallets, handling ERC-20 tokens, and interacting with the blockchain more easily.”*
>
> From https://github.com/MyCryptoHQ/MyCrypto

This report documents the findings of a security assessment targeting the MyCrypto compound. Carried out by Cure53 in June 2018, this project specifically targeted the MyCrypto open-source ether and the ERC-20 token wallet. The assessment has only yielded five security-relevant findings, which further have rather limited implications for the safety of the MyCrypto and its users. Overall the application made a solid impression.

The main objectives of this Cure53-MyCrypto security collaboration was to verify whether possible means for compromising the security of the users’ wallets existed. By auditing sources, Cure53 checked for implicit and more hidden ways for determined attackers seeking to break the MyCrypto compound’s integrity. A special emphasis was also placed on the integration of the Electron framework and its hardening, as well as the build scripts.

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

It should be noted that this Cure53 audit of MyCrypto was rooted in a white-box methodological premise. Under this approach, the testing team investigated and analyzed the code frozen in a dedicated Cure53 repository branch[1]. In accordance with the earlier brief, a total time budget of 11 days was allocated to the completion of this project, inclusive of core audit, communications and documentation process. A dedicated team of five Cure53 testers was tasked with conducting the assessment. With the white-box methods in use, Cure53 could stay in close contact with the MyCrypto development team throughout the assignment. The teams communicated via email and on a Slack channel devoted to the project.

Among the mere five security relevant findings spotted on the MyCrypto scope, only one actually constituted a vulnerability. It was determined that the problem stemmed from a pitfall in Electron. In addition, four miscellaneous issues have been documented. These are not directly exploitable but can rather offer additional hardening advice.

In the following sections, the report will first provide more details on the scope and then moves on to a case-by-case discussion of the discoveries made by the Cure53 team. In the final sections, broader conclusions will be offered about the security posture of the MyCrypto items in scope. The final verdict is evidence-based and draws on the observations made and the data collected by the involved Cure53 testers during this project.

Scope

- **MyCrypto Codebase / Electron App & Sources**
  - https://github.com/MyCryptoHQ/MyCrypto
- **MyCrypto Build Scripts**
- **Special Attention was given to several specific PRs**
  - https://github.com/MyCryptoHQ/MyCrypto/pull/871
  - https://github.com/MyCryptoHQ/MyCrypto/pull/1836
  - https://github.com/MyCryptoHQ/MyCrypto/pull/854
  - https://github.com/MyCryptoHQ/MyCrypto/pull/1777
  - https://github.com/MyCryptoHQ/MyCrypto/pull/1946

[1] https://github.com/MyCryptoHQ/MyCrypto/tree/cure53

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

MC-01-001 Desktop: UXSS/local file leak via drag and drop *(Medium)*

It was found that MyCrypto Electron desktop application allows dragging and dropping any local HTML files onto the window. Since the *file:* URL in Electron has the privilege to read any local files or websites, this leads to leakage of sensitive information if the user drags and drops a crafted local file.

In case the following HTML file is dropped onto the MyCrypto window, the contents of */etc/hosts* will be shown on an alert dialog.

PoC.html:

```
<!-- for Windows -->
<iframe src="file:///C:/Windows/System32/drivers/etc/hosts"
onload="alert(contentWindow.document.body.innerHTML)"></iframe>

<!-- for Mac -->
<iframe src="file:///etc/hosts"
onload="alert(contentWindow.document.body.innerHTML)"></iframe>
```

To abuse this, an attacker needs to guide victims to drag and drop a malicious file by social engineering. Because of the unusual user-interaction, the severity of this issue is considered “*Medium*”. It is recommended to disable the default behavior of the drop event by setting the event listener.

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.

MC-01-002 Web: CSP largely ineffective against XSS *(Medium)*

It was found that the CSP on *MyCrypto.com* uses unsafe values for the *script-src* directive. More specifically, the concerning values are *unsafe-inline and unsafe-eval*. The former allows the execution of unsafe in-page scripts and event handlers, while the latter allows the execution of code injected into the DOM APIs such as *eval()*. As a result, an attacker with the ability to inject JavaScript into the context of the MyCrypto's origin will not be hindered by the CSP in any notable way.

Current CSP configuration:

```
default-src 'none'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; worker-src
'self' blob:; style-src 'self' 'unsafe-inline'; manifest-src 'self'; font-src
'self'; img-src 'self' data: https://shapeshift.io; connect-src *;
```

In their current form, the CSP rules must be mostly considered useless. They only add weight to the overall footprint tied to every HTTP request. It is recommended to remove the permissions for *unsafe-eval* and *unsafe-inline*.

MC-01-003 App: Downloadable and Electron applications lack CSP *(Medium)*

MyCrypto makes use of CSP but it appears that CSP is only applied on applications backed by a server. This can be seen from it being served in the header. This means the standalone, downloadable and Electron applications are not protected by CSP.

It is recommended to also serve CSP using *<meta>* tag[2] so that it can be applied to non-server-based applications.

MC-01-004 Electron: Main window does not cancel navigation *(Low)*

It was noticed that the only prevention of external navigation on the main windows of the MyCrypto Electron desktop application is to ensure all *<a>* tags have the *target="_blank"* attribute. This is a little risky as it might happen that some *<a>* tags, or other means of navigation (e.g. JavaScript location change), navigate on the main window instead of opening on browsers. The impact of being able to navigate on the main window is considered "*High"* if not *"Critical"* as MyCrypto supports importing private key and the Electron application has no address bar. Thus, users have no way of telling if they are

[2] https://developers.google.com/web/fundamentals/security/csp/#the_meta_tag

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

still on MyCrypto or not, which means that they might give out their private keys to a third-party Phishing site.

It is recommended to listen on the *will-navigate* event[3] and cancel any navigation on the main window.

MC-01-005 Jenkins: Possibly unsecure use of environment variables *(Info)*

During the analysis of the *jenkins* build pipelines it was discovered that all AWS environment variables have been renamed to "*REDACTED*". This is of course not a security issue in itself but should be regarded as linked to good practices when sharing build scripts. However, depending on the configuration and implementation of docker, these variables could be stored in the running container in plain-text. If an attacker were to gain access to the docker container, all the stored sensitive environment variables would become readable. This can be leveraged throughout the AWS infrastructure.

Affected files:
Jenkinsfile.linux.develop
Jenkinsfile.linux.master

Affected configuration:
```
pipeline {
    agent {
        dockerfile {
            filename 'Dockerfile'
            dir 'jenkins/Docker'
            args '--env ETH_SIGNING_KEY=$ETH_SIGNING_KEY --env
S3_BUCKET_NAME=$S3_BUCKET_NAME --env AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID --env
AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY'
        }
    }
```

It is recommended to ensure that all sensitive environment variables are stored in a safe manner. Since the MyCrypto team uses AWS, it would be advisable to use "*Encrypt at rest"* implemented with KMS to ensure that all sensitive environment variables are stored safely.

[3] https://github.com/electron/electron/pull/931

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

Conclusions

The results of this June 2018 Cure53 security assessment of the MyCrypto entities are very impressive. Five Cure53 testers involved in this project, which focused on verifying the security promises of the MyCrypto open-source Ethereum and ERC-20 token wallet. The security posture of the MyCrypto project was judged as robust and the general security standards exhibited by various components attests to the development that happens with security in mind.

Even though Electron applications tend to be prone to "*Critical"*-level pitfalls, the MyCrypto app managed to avoid these and remains secure. Among the five unveiled findings, only one issue could lead to a serious compromise has been documented. Still, this vulnerability relies on the drag and drop operations on a malicious file and its exploitation would be quite difficult, requiring the user to carry out a number of steps. Therefore, the severity of this sole vulnerability was only set to "*Medium".*

Moving on to four miscellaneous issues identified during this test, it must be emphasized that they mostly suggest further hardening, both in connection with the app and as far as handling secrets during the build is concerned. Similarly, during the review of the MyCrypto's hardware wallet integration, no issues were found directly to be linked to the MyCrypto entity. However, through that process, a few minor issues have been identified in Trezor and these were responsibly disclosed to the relevant party.

In spite of substantial efforts, numerous approaches, thorough testing and the code review completed by several Cure53 testers, the MyCrypto project stood strong. It meets - and oftentimes exceeds - key security standards and should be seen as recommendable solution from a security standpoint going forward.

Cure53 would like to thank Daniel Ternyak & Taylor Monahan from the MyCrypto team for their excellent project coordination, support and assistance, both before and during this assignment.