Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Pentest-Report MetaMask 08.2017

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Inführ, N. Kobeissi, T.-C. Hong, M. Kinugawa

Index

Introduction

“MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today. It allows you to run Ethereum dApps right in your browser without running a full Ethereum node.

MetaMask includes a secure identity vault, providing a user interface to manage your identities on different sites and sign blockchain transactions.”

From https://metamask.io/

This report documents the findings of the Cure53 assessment of the MetaMask project. Specifically in scope was the MetaMask browser extension and its variation running from a ServiceWorker. Five members of the Cure53 team were involved in completing the assignment which took place over the course of six days in August 2017 and yielded six security-relevant findings.

With the premise of the scope encompassing both a browser extension and a considerable number of JavaScript files, the logical approach to follow was to rely on the white-box methodology. Consequently, the Cure53 testers were granted access to all relevant sources via GitHub repository. Notably, the tested software is open source, so the available public versions were investigated during this project. What is more, the

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

project benefit from ongoing communications between the Cure53 testers and the MetaMask team, with a development-centered dedicated Slack channel being joined by relevant parties. Though this was envisioned to help clarify any arising questions, the tests actually went very smoothly and no further discussions were actually necessary.

The discoveries made by this test regarding the MetaMask product were quite scarce and attest to the overall good security found on the items in scope. Among the aforementioned six results, four issues constituted security vulnerabilities, while the remaining two were deemed to be general weaknesses. What is crucial to underscore is that neither “Critical” nor even “High” level of severity could be attributed to any of the risks carried by the discoveries.

In the following sections, the report will provide a case-by-case discussion of findings, alongside mitigation advice and fix recommendations. The closing section will offer a brief conclusion and delivers a summary assessment of MetaMask from the security perspective.

Scope

- **MetaMask Browser Extension**
  - https://github.com/MetaMask/metamask-extension
- **MetaMask in Chrome Web Store**
  - https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn
- **MetaMask via Service Worker**
  - https://github.com/MetaMask/metamask-extension/tree/master/mascara

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 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. *MM-01-001*) for the purpose of facilitating any future follow-up correspondence.

MM-01-002 Extension: Phishing Detector can be bypassed *(Medium)*

It was found that the Phishing Detector, which works with a set of blacklisted domains, can be bypassed. Normally, if a blacklisted domain is accessed by the user, the MetaMask detector will try to redirect the user to the warning page. The logic is implemented inside the content script of the browser extension and can be found on the following line.

Affected Line:
https://github.com/MetaMask/metamask-extension/blob/0e6bc6647ee5c0554a4e149514c87ca4d8585d2c/app/scripts/contentscript.js#L97-L100

Redirect Code:

```
function redirectToPhishingWarning() {
  console.log('MetaMask - redirecting to phishing warning');
  window.location.href = 'https://metamask.io/phishing.html';
}
```

However, a malicious page can negate the redirect effect by aborting the page load with *window.stop.* This behavior is shown below.

PoC:

```
<script>
var i =50;
var si=setInterval(function(){
    if(--i){
        window.stop();
    }else{
        clearInterval(si);
        document.write("<h1>This is black-listed page</h1>");
    }

},10);
</script>
```

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

It is recommended to use the WebRequest API[1] to block the access to a blacklisted domain before users can visit it. This mechanism should be more robust rather than being prone to interruptions from the website's DOM.

MM-01-003 Extension: Unsafe Background Script Communications *(Info)*

The communication model implemented by the extension at present transfers any data sent by a web page to the background script via the content script. It was discovered that the content script does not enforce any checks with respect to the validity of the sent JSON structure. The following example of a JSON structure demonstrates that not only any value can be specified for known keys, but additional keys are also permitted.

The data sent via *postMessage* from a website is supplied next.

```
data = {"target":"contentscript","data":{"name":"provider","data":
{"jsonrpc":"1337.0","method":"cure53_test","whatever": "test","params":
[{"from":"123","test":"1"}],"cure53":"key"},"key":"key"}}

window.postMessage(data,"*");
```

Data received by the background script:

```
PortDuplexStream - saw message {"name":"provider","data":
{"jsonrpc":"1337.0","method":"cure53_test","whatever":"test","params":
[{"from":"123","test":"1"}],"cure53":"key"},"key":"key"}
```

Although this behavior did not introduce any security vulnerabilities, implementing a validation scheme in the content script should be taken into consideration. A revised approach would ensure that a hypothetical vulnerability in the background script cannot be reached by a malicious web page.

MM-01-005 Web: Missing Clickjacking protection on MetaMascara *(Info)*

It was found that *MetaMascara,* which is a ServiceWorker-based variation of the MetaMask extension, is framable. This allows an attacker to embed the site 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[2]. In the context of the tested product, however, performing a critical action would require multiple clicks and data input, which largely mitigates this issue.

PoC:

```
<iframe style="opacity:0.2" src="//localhost:9001"></iframe>
```

[1] https://developer.chrome.com/extensions/webRequest
[2] https://en.wikipedia.org/wiki/Clickjacking

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Fig.: MetaMascara being framed with low opacity

To observe the logic behind this issue, simply use the above code and save it as a HTML file. Consequently, the site will be rendered *framed* and enables Clickjacking attacks. Though there is currently no risk associated with this problem, it is nevertheless recommended to deploy proper Clickjacking protections by including the *X-Frame-Options: DENY* header. This approach signals the browser not to let any websites frame the site.

MM-01-006 Extension: Potentially buggy random ID assignment *(Low)*

The MetaMask application is required to assign a random ID to certain resources. It was found that the ID generation algorithm is exposed to certain potential bugs. These minor issues could be resolved, even if they are unlikely to occur in a regular daily use context. Following aspects can be considered:

1. The random ID generator starts by choosing an initial bound between 0 and *Number.MAX_SAFE_INTEGER*. The question remains: what happens if the chosen bound is close to the specified boundary?
2. Assuming that secure randomness is a requirement, *Math.Random()* does not constitute a good source for unpredictable random information.

Assuming that each resource for which an ID is being generated is unique, a better *createRandomId* function should be used. It could simply concatenate the full request into a string, hash it using a keyed hash construction (such as HMAC, with the key in this context acting as a kind of salt), and use the hash as the ID. This solution is superior to the one employed by MetaMask at present. There are several reasons for the proposed approach to be considered better:

- It offers a guaranteed-unique and guaranteed-uniform distribution of identifiers over a maximum identifier space of 2^{256} with no issue.
- There is no potential for integer overflow.
- No information regarding the ordering of the resources is exposed by their IDs.

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

To re-iterate, it is understood that the current method does not pose any serious or immediate risks to the design of MetaMask. However, it is still advised to implement the suggested alternative in order to strengthen the overall reliability of the application.

Affected Files:
./app/scripts/lib/inpage-provider.js
./app/scripts/lib/random-id.js

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.

MM-01-001 Extension: External Links not using HTTPS *(Info)*

It was found that certain external URLs rely on an unencrypted HTTP channel, which empowers an attacker with the ability to Man-in-the-Middle (MitM) the network. In this context, an adversary could use techniques like *sslstrip*[3] to proxy clear-text traffic to the victim-user.

Affected URLs:
https://github.com/MetaMask/metamask-extension/blob/master/ui/lib/account-link.js
https://github.com/MetaMask/metamask-extension/blob/master/ui/lib/explorer-link.js#L5
https://github.com/MetaMask/metamask-extension/blob/master/ui/app/info.js#L106
https://github.com/MetaMask/metamask-extension/blob/master/ui/app/info.js#L131

It is recommended to embed the links with a consistent use of HTTPS and potentially create a *commit* hook capable of checking for the use of HTTP links and resources. This would help avoid regressions in the described area.

MM-01-004 Extension: Improper MIME-type check in *Web3* injection *(Info)*

The MetaMask extension avoids the *Web3* script being injected into XML and PDF responses and documents. However, it was found that this check does not work properly. It currently takes the value of *location.href* and verifies whether it ends with *.xml* or *.pdf*. This means that if a URL has a query string, the check will simply fail and result in no action or benefit whatsoever.

[3] https://moxie.org/software/sslstrip/

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Affected Line:
https://github.com/MetaMask/metamask-extension/blob/0e6bc6647ee5c0554a4e149514c87ca4d8585d2c/app/scripts/contentscript.js#L84-L95

Affected Code:
```
function suffixCheck() {
  var prohibitedTypes = ['xml', 'pdf'];
  var currentUrl = window.location.href;
  var currentRegex;
  for (var i = 0; i < prohibitedTypes.length; i++) {
    currentRegex = new RegExp('\\.' + prohibitedTypes[i] + '$');
    if (currentRegex.test(currentUrl)) {
      return false;
    }
  }
  return true;
}
```

When the following URL is accessed, the injected script will be found in the XML tree.

Sample Affected URL:
https://html5sec.org/crossdomain.xml?aaa

Although this behavior is technically not a security issue, it might cause damage and evoke unpredictable problems for the affected document types. It is therefore recommended to check the *Content-Type* header instead of the file name's suffix.

Dr.-Ing. Mario Heiderich, Cure53
Rudolf Reusch Str. 33
D 10367 Berlin
cure53.de · mario@cure53.de

Conclusions

The results of this summer 2017 security assessment of MetaMask led the Cure53 testing team to believe that the MetaMask extension delivers on its security promises.

Tested over the course of six days by five members of the Cure53 team, the MetaMask project stood strong against the majority of the attempted compromise approaches. Overall, the MetaMask extension manages to bring the Ethereum to the browser and, in a more experimental manner, exhibits a capacity to use the designated ServiceWorker.

It should be noted that addition of the MetaMask extension does not dramatically expand the attack surface. This conclusion alone should be read as a positive indicator. Corroborating evidence of good developments included that the feature maintained the range of attacks already present in Web3 and Ethereum. In terms of severity, a noteworthy issue pertained to the possible bypass of the blacklist and the potentially ensuing Phishing described in MM-01-002. Other flaws warranting attention revolved around having a stronger focus on avoiding HTTP URLs (see MM-01-001) with the use of tests and *commit* hooks recommended for the purpose of enforcing better policies.

In sum, the MetaMask project should be considered safe and secure. Despite extensive testing and wide-spanning coverage of the provided code, no significant security-risks were unveiled by the Cure53 team completing this project.

Cure53 would like to thank Aaron Davis from the MetaMask team for his excellent project coordination, support and assistance, both before and during this assignment.