# Advanced Notification of Cyber Threats against jQuery Mobile Vulnerabilities

aeCERT
فريق الاستجابة لطوارئ الحاسب الآلي
Computer Emergency Response Team

| | | | | |
|---|---|---|---|---|
| **Security Advisory** | ADV-19-17 | **Criticality** | High | ○○○● |
| **Advisory Released On** | 12 May 2019 | | | |

### Impact

A vulnerability in jQuery Mobile (JQM) that could be exploited to a Document Object Model (DOM)-based Cross-Site Scripting (XSS).

### Solution

Adhere the advices written under the recommendations section.

### Affected Platforms

- All current versions of jQuery Mobile

## Summary

As the leading trusted secure cyber coordination center in the region, aeCERT has researched and found about a critical vulnerability in jQuery Mobile. All current versions of JQM are currently vulnerable to DOM-based XSS through crafted URLs, and comprise of a broken implementation of a URL parser that can lead to security issues in the impacted applications.

## Details

As of 4th of May 2019, all current versions of JQM are susceptible to DOM-based XSS through the use of crafted URLs. The requirement for this vulnerability to be exploited differs according to the version of JQM.

In JQM versions up to – and including – version 1.2.1, the sole requirement is that the library is included in a web application.

In JQM versions later than 1.2.1, in addition to the library being included in a web application, the application has to have a certain functionality: a server-side API that redirects back user input as part of an HTTP response; all web applications typically have at least one API that does this.

This vulnerability has been reported back in 2017; however, JQM maintainers left the vulnerability unpatched for the following:

- Exploiting said vulnerability required the usage of an already-existing vulnerability
- There was a risk of losing compatibility with existing applications should the issue be patched

Two methods have been discovered to exploit the vulnerability:

1. Missing content-type validation
2. Broken URL parsing.

In the former technique, an attacker takes advantage of JQM's lack of validation of content-type of XHR response. Typically, when a user performs a query in a REST search API, the format is as follows, where `<search_query>` contains the input of the user.

```
/search?q=<search_query>
```

The corresponding response would be as follows:

```
{"q":"<search_query>","results":["<search_results>"]}
```

However, in the case of a website containing both the API and JQM, an attacker would perform XSS as shown:

```
https://example.com/path/to/app/#/search?q=<iframe/src='javascript:alert(1)'><
/iframe>
```

Once the website is opened, the JQM application performs an XHR request to the search query, to which the API would respond with a blank result, as shown in the following:

```
{"q":"<iframe/src='javascript:alert(1)'></iframe>","results":[]}
```

Thus, jQuery Mobile would ignore the content-type and put the response into the Document Object Model as it is. As such, any HTML inside the structure is parsed by the browser and executed by JavaScript.

The second technique exploits the URL parser which is based on a Regular Expression (RegEx). Below is the exploitable RegEx based on urlParseRE, implemented in jQuery.mobile.path.parseUrl,

```
/^\s*(((([^:\/#\?]+:)?(?:(\/\/)((?:(([^:@\/#\?]+)(?:\:([^:@\/#\?]+))?)@)?(([^:
\/#\?\]\[]+|\[[^\/\]@#?]+\])(?:\:([0-
9]+))?))?)?)?((\/?(?:[^\/\?#]+\/+)*)([^\?#]*)))?(\?[^#]+)?)(#.*)?/
```

The parser compares two URLs and checks if they are of the same domain; the issue with the parser, however, is that it fails in this validation: it returns true regardless of whether the domains of the two URLs match or not.

Once the URL is parsed and is erroneously believed to be of the same source, JQM issues a request, and the malicious payload is thus loaded into the DOM.

## Recommendations

**In order to circumvent the exploit, we highly recommend the following:**

- Avoid submitting information in websites that make use of JQM framework.
- As JQM no longer seems to be maintained, it is recommended to use an alternative framework when programming an application.

## Disclaimer

Accessing third-party links in this advisory will direct you to an external website. Please note that aeCERT bears no responsibility for third-party website traffic. aeCERT will have no liability to the entities for the content or use of the content available through the hyperlinks that are referenced

## References

[Github - Vulnerabilities in jQuery Mobile]

## Contact Us

aeCERT
P.O. Box 116688
Dubai, United Arab Emirates

Tel (+971) 4 777 4003
Fax (+971) 4 777 4100
Email info[at]aeCERT.ae
Instagram @TheUAETRA
Twitter @TheUAETRA

For secure communications with aeCERT with regards to sensitive or vulnerability information please send your correspondences to aeCERT[at]aeCERT.ae