Bypass CSP by Abusing XSS Filter in Edge

Xiaoyin Liu Follow
Apr 15, 2018 · 3 min read

In this article, I will share a Content Security Policy (CSP) bypass vulnerability in Microsoft Edge, which I discovered in December 2016. The bypass was done by abusing the browser's XSS filter. This is quite ironical, because both XSS filter and CSP are designed to protect users from XSS attacks, but this vulnerability allows attackers to abuse one XSS protection mechanism to bypass another.

XSS filter was first introduced in IE 8. The purpose of a XSS filter is to mitigate reflective XSS attacks. The XSS filter of IE/Edge roughly works in this way: when the browser loads a URL, the XSS filter first checks if some parameters of the URL may contain XSS payloads, using a set of predefined regular expressions. For instance, `http://example.com/index.php?id=100` is clearly harmless, but for a URL like `http://example.com/index.php?id=<script>alert(1)</script>`,

the value of parameter `id` may be a XSS payload. Then, to determine whether it is a reflective XSS attack exactly, IE/Edge check if the returned HTML contains the substring `<script>alert(1)</script>`. If it does, IE/Edge assume it is reflected from the `id` parameter. Notice that this assumption may not be true: what if `<script>alert(1) </script>` is hardcoded in the page, and the `id` parameter actually doesn't have any effect?

There're two modes for XSS filter: default mode and block mode. Sites can enable the block mode by setting HTTP header: X-XSS-Protection: 1; mode=block". In the block mode, IE/Edge block the entire HTML from rendering. In the default mode, IE/Edge try to destroy XSS payloads by modifying the corresponding HTML tags. E.g. if Edge assumes `<script>alert(1)</script>` is a XSS, it changes this element to `<sc#ipt>alert(1)</script>`. Thus, the DOM parser won't parse this segment as a <script> element, so it won't execute.

Now let's see the vulnerability, CVE-2017–0135. There are two ways for a website to set a Content Security Policy: via Content-Security-Policy HTTP response header field, or via <meta> tag. An example of such meta element is:
`<meta http-equiv="Content-Security-Policy" content="script-src 'self'">`. Now suppose here is the HTML for URL http://example.com/xss.html:

```
<!DOCTYPE html>
```

```
<html>
 <head>
 <title>CSP Test</title>
  <meta http-equiv="Content-Security-Policy" content="script-
src 'self'">
 </head>
 <body>
 <script>alert(document.domain);</script>
 </body>
</html>
```

The CSP should block `alert(document.domain)` from executing. To bypass the CSP, let's ask users to visit `http://example.com/xss.html?<meta http-equiv="Content-Security-Policy" content="script-src 'self'">` (Just append meta element as a parameter to the URL.)

Then, by default, Edge's XSS filter simply modifies the meta tag to `<me#a http-equiv="Content-Security-Policy" content="script-src 'self'">`, which kills the CSP, and alert is executed.

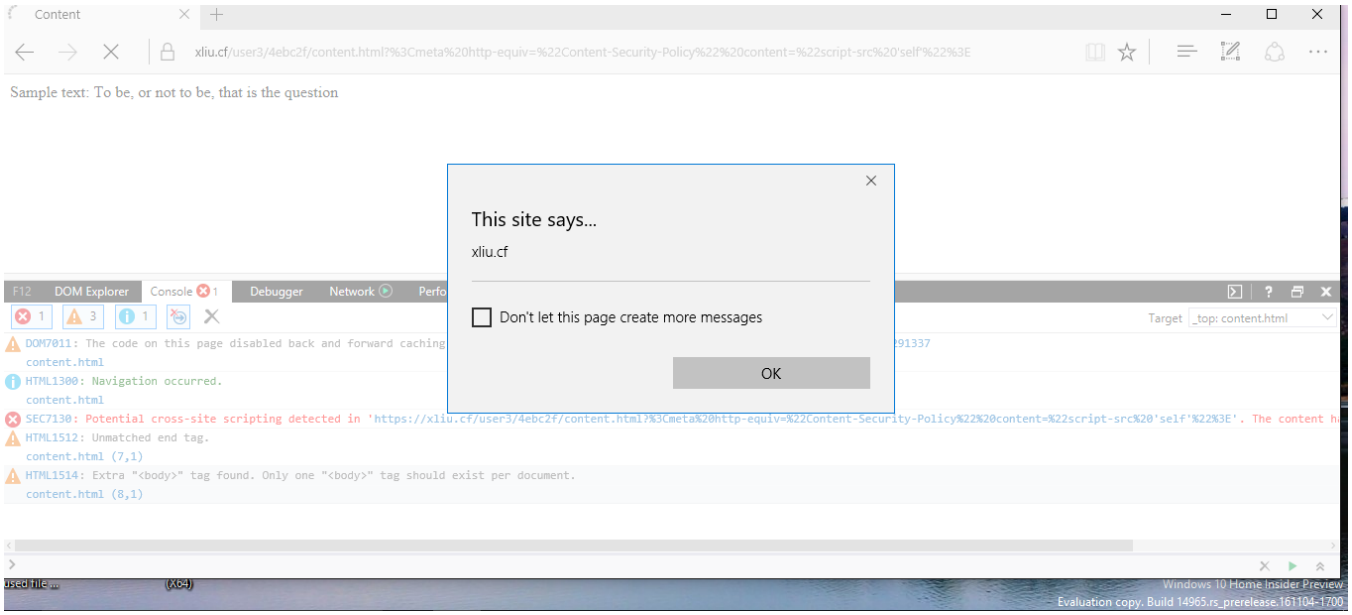

Figure 1 Screenshot of a successful CSP bypass

This vulnerability was fixed in MS17-007, released in March 2017. It was assigned CVE-2017-0135.

It's fixed by blocking the whole HTML page, when the XSS filter detects the query string matches any meta element, regardless of modes.

Although this vulnerability has been fixed, it's still a good idea to set

"X-XSS-Protection: 1; mode=block". Also CSP policies delivered via HTTP headers are not vulnerable to this bypass attack.

. . .

Timeline

- 12/2/2016: Vulnerability reported to MSRC
- 3/14/2017: Vulnerability fixed in MS17–007 (bug bounty: $1500)

. . .

References

- Nava, E. V. and Lindsay, D., "Abusing Internet Explorer 8's XSS Filters"
- W3C, "Content Security Policy Level 2"
- IE 8 XSS Filter Architecture / Implementation

. . .

Acknowledgements

- Many thanks to MSRC for fixing this bug and awarding me the

bounty.

- This article was first published on Chinese website FreeBuf, link: http://www.freebuf.com/articles/web/164871.html (in Chinese). I want to thank them for allowing me to post the English version on Medium.