Anti Cross-Site-Scripting revisited: Checksum-based integrity checks

Hauke Jan Lübbers

December 1, 2015

Abstract

This proposal describes a possible extension of the HTML standard which aims to secure websites against the injection of JavaScript on the client-side (Cross-Site-Scripting). To be executed by the browser each script tag must provide a checksum of the code it provides. The checksum is formed by a standard algorithm like SHA-256 applied to the code to be executed and salted with a value that is given in a meta element in the pages header. The value is randomly generated for each HTTP request. If a script tag does not provide a valid checksum the contained code is not executed. JavaScript embedded in other HTML elements is not executed.

1 Motivation

Cross-Site-Scripting (XSS) vulnerabilities remain one of the most wide-spread vulnerabilities of todays dynamic websites. [2] Even organizations widely respected for their web engineering are vulnerable for this type of attack: According to Google security researchers the largest share of vulnerabilities reported via their bug bounty program are XSS vulnerabilities. [1]

2 Proposal

I propose to extend the HTML standard to enable browsers to verify the integrity and authenticity of JavaScript code. The proposal does not interfere with the functionality of websites which do not make use of the proposed options and therefore is backwards compatible, which is important for acceptance with browser vendors. It is related to the W3C Candidate Recommendation on Subresource Integrity. [3]

2.1 Example

The following example shows the usage of the proposed functionality for both an external JavaScript file that is included in the head and an inline script block.

```
<!doctype HTML>
<html>
    <head>
        <meta http-equiv="script-integrity-salt" content="e3b0c9...952b855" />
        <script src="js/good.js"
                integrity="sha256-H4u...Gle37?script-integrity-salted">
    </head>
    <body>
        <img src="..." onclick="alert('embedded JS is disabled');" />
        <script integrity="sha256-Li9...EbzJr7?script-integrity-salted">
            // Inline JavaScript
        </script>
    </body>
</html>
```

2.2 Changes to the standard

The proposal requires the integration of a pragma extension ”script-integrity-salt” for the http-equiv attribute in meta elements. It furthermore builds up on the work of the W3C Candidate Recommendation on Subresource Integrity [3] using the integrity attribute to include the checksum followed by an option-expression ”script-integrity-salted” as defined in the candidate. The browsers behavior only changes if the ”script-integrity-salt” pragma is present. If the pragma is present and the browser supports the functionality it will check the integrity of every script section before it is executed. JavaScript embedded in other HTML element, for example via ”onclick” event listeners, is disabled.

2.3 Attack vector review

Because the script integrity salt is randomly generated on the server side for each request the integrity checksums change every time a user reloads the page. Therefore an attacker cannot extract or guess a valid checksum, in order to get their injected JavaScript code executed. It therefore protects against reflected and stored XSS - if the user communicates via a secure channel (HTTPS). Two XSS attack vectors remain: If the attacker can include code in a JavaScript section of a website that is generated on the server-side (stored XSS) this code is part of the script when the checksum is generated. Developers should not include user data directly in the JavaScript code, but access it via for example AJAX or from other DOM elements. The other remaining vector is the execution of code that can be manipulated by the user via `eval()` or other DOM-based XSS attacks (via vulnerabilities in the JavaScript code itself).

3 Implementation

A proof-of-concept could be implemented as an extension for a popular browser. The changes to the HTML standard were minimized - but it takes time to get them accepted and implemented. To gain acceptance in the developer community it is important to include the needed functionality in common web frameworks.

References

[1] L. Essers. (2012, May) Bug bounty hunters reveal eight vulnerabilities in google services. Visited December 1st 2015. [Online]. Available: http://www.pcworld.com/article/256151/bug_bounty_hunters_reveal_eight_vulnerabilities_in_google_services.html

[2] E. Kovacs. (2015, July) Invitation-only bug bounty programs becoming more popular: Bugcrowd. Visited December 1st 2015. [Online]. Available: http://www.securityweek.com/invitation-only-bug-bounty-programs-becoming-more-popular-bugcrowd

[3] F. M. J. W. Devdatta Akhawe, Frederik Braun. (2015, November) Subresource integrity. Visited December 1st 2015. [Online]. Available: https://w3c.github.io/webappsec-subresource-integrity/