Why Is CSP Failing? Trends and Challenges in CSP Adoption

Michael Weissbacher, Tobias Lauinger, and William Robertson

Northeastern University, Boston, USA
{mw,toby,wkr}@ccs.neu.edu

Abstract. Content Security Policy (CSP) has been proposed as a principled and robust browser security mechanism against content injection attacks such as XSS. When configured correctly, CSP renders malicious code injection and data exfiltration exceedingly difficult for attackers. However, despite the promise of these security benefits and being implemented in almost all major browsers, CSP adoption is minuscule—our measurements show that CSP is deployed in enforcement mode on only 1% of the Alexa Top 100.

In this paper, we present the results of a long-term study to determine challenges in CSP deployments that can prevent wide adoption. We performed weekly crawls of the Alexa Top 1M to measure adoption of web security headers, and find that CSP both significantly lags other security headers, and that the policies in use are often ineffective at actually preventing content injection. In addition, we evaluate the feasibility of deploying CSP from the perspective of a security-conscious website operator. We used an incremental deployment approach through CSP's report-only mode on four websites, collecting over 10M reports. Furthermore, we used semi-automated policy generation through web application crawling on a set of popular websites. We found both that automated methods do not suffice and that significant barriers exist to producing accurate results.

Finally, based on our observations, we suggest several improvements to CSP that could help to ease its adoption by the web community.

Keywords: Content Security Policy, Cross-Site Scripting, Web Security.

1 Introduction

The web as a platform for application development and distribution has evolved faster than it could be secured. Consequently, it has been plagued by numerous classes of security issues, but perhaps none are as serious as content injection attacks. Content injection, of which cross-site scripting (XSS) is the most well-known form, allows attackers to execute malicious code that appears to belong to trusted origins, to subvert the intended structure of documents, to exfiltrate sensitive user information, and to perform unauthorized actions on behalf of victims. In response, many client- and server-side defenses against content injection have been proposed, ranging from language-based auto-sanitization [17] to sandboxing of untrusted content [12] to whitelists of trusted content [11].

A. Stavrou et al. (Eds.): RAID 2014, LNCS 8688, pp. 212–233, 2014.
© Springer International Publishing Switzerland 2014

Content Security Policy (CSP) is an especially promising browser-based security framework for refining the same-origin policy (SOP), the basis of traditional web security. CSP allows developers or administrators to explicitly define, using a declarative policy language, the origins from which different classes of content can be included into a document. Policies are sent by the server in a special security header, and a browser supporting the standard is then responsible for enforcing the policy on the client. CSP provides a principled and robust mechanism for preventing the inclusion of malicious content in security-sensitive web applications. However, despite its promise and implementation in almost all major browsers, CSP is not widely used in practice—in fact, according to our measurements, it is deployed in enforcement mode by only 1% of the Alexa Top 100.

In this paper, we present the results of a long-term study to determine why this is the case. In particular, we repeatedly crawled the Alexa Top 1M to measure adoption of web security headers, and find that CSP significantly lags behind other, more narrowly-focused headers in adoption. We also find that for the small fraction of sites that have adopted CSP, it is often deployed in a manner that does not leverage the full defensive power of CSP.

In addition to our Internet-scale study, we also quantify the feasibility of incrementally deploying CSP from the perspective of a security-conscious administrator using its report-only mode at four websites. Although this is an oft-recommended practice, we find significant barriers to this approach in practice due to interactions with browser extensions and the evolution of web application structure over time.

Finally, we evaluate the feasibility of automatically generating CSP rules for web applications, again from the perspective of an administrator. We find that for websites that are well-structured and do not change significantly over time, rules can indeed by generated in a black-box fashion. However, for more complex sites such as those that make use of third-party advertising libraries in their proper site context, policy generation is significantly more difficult.

To summarize, the contributions of this paper are the following:

- We perform the first long-term analysis of CSP adoption in the wild, performing repeated crawls of the Alexa Top 1M over a 16 month period.
- We investigate challenges in adopting CSP, and why it is not deployed to its full extent even when it has been adopted.
- We evaluate the feasibility of both report-only incremental deployment and crawler-based rule generation, and show that each approach has fundamental problems.
- We suggest several avenues for enhancing CSP to ease its adoption.
- We release an open source CSP parsing and manipulation library. [1]

2 Content Security Policy

The goal of CSP is to mitigate content injection attacks against web applications directly within the browser [6, 19]. In the following, we describe CSP as it is

[1] `https://github.com/tlauinger/csp-utils`