PowerAmp: Amplifying Client-side Prototype Pollution and Gadget Detection

Samuel Kajava[1] SiKai Lu[2] Eric Cornelissen[2] Benjamin Eriksson[1]
Musard Balliu[2] Andrei Sabelfeld[1]

[1]Chalmers University of Technology and University of Gothenburg
[2]KTH Royal Institute of Technology

Abstract

Prototype pollution vulnerabilities in JavaScript enable attackers to tamper with object properties and affect application behavior. On the web, prototype pollution can lead to critical vulnerabilities like Cross-Site Scripting (XSS). While previous work shows that client-side prototype pollution occurs in today's web, a systematic attribution of its causes has yet to be established.

This paper is focused on attributing the causes of client-side prototype pollution and exploring their implications. We suggest that, in fact, a small number of URL-parsing libraries are responsible for the lion's share of vulnerabilities. This insight enables us with a powerful amplification methodology to identify vulnerabilities at scale.

We present PowerAmp, a novel approach that combines dynamic analysis for prototype pollution and pollution-based XSS gadgets with signature-based large-scale amplification to find vulnerable websites. Under PowerAmp, we use a lightweight dynamic scanner to efficiently detect prototype pollution vulnerabilities in a seed list of websites. From these, we analyze the vulnerable code, semi-automatically generating signatures for vulnerable libraries. We use these signatures to find additional vulnerable websites by identifying them on other domains using crawling datasets. Next, we perform a dynamic taint analysis on the prototype pollution vulnerable websites to find gadgets that may lead to XSS vulnerabilities. We manually verify that these gadgets result in XSS exploits and use the corresponding signatures to amplify XSS detection at scale. We evaluate PowerAmp on a seed of one million websites and extract signatures from 2 807 vulnerable websites. From these, we amplify and find a total of 14 248 additional websites vulnerable to prototype pollution.

Similarly for XSS gadgets, we find 4 libraries used on at least 2 websites that result in 293 exploitable XSS vulnerabilities. Based on our findings we suggest mitigation strategies. Our results confirm the prevalence of pollution-based XSS vulnerabilities on the web and suggest that secure URL-parsing is an effective short-term solution.