Is the Google Caja safe JavaScript project still relevant?

by Benny Bottema - Friday, January 09, 2015

http://www.bennybottema.com/2015/01/09/is-the-google-caja-safe-javascript-project-still-useful-and-being-maintained/

It doesn't look very good

I will focus on the client-sided javascript sanitation support. It is used by including html-sanitizer.js (in */src/com/google/caja/plugin*) from the project in your website and then call

```
html.sanitize()
```

on the strings you wish to make safe for inserting into DOM. The file uses two auxiliary script files (html4-defs.js and uri.js), which used to be included in the source code, but can now only be obtained by building the project using ant (allowing you to customize security policies in the form of json files). However, as the build uses UX specific scripts, it doesn't work in Windows and in May 24, 2013 it was not deemed a priority by the team.

The project however does include pre-build releases include pre-built releases, which include the aforementioned script files. However, the last release (r5127) was of November 6, 2012. The last SVN commit to the source file html-sanitizer.js was of August 28, 2013. So the source has not been updated very recently and the pre-build releases are even older. This means the only way to get a recent usable version of the html-santizer is to compile it yourself in a Linux environment.

But looks can be deceiving

All this being said, I still feel Google Caja is relevant in today's world. The Caja project as a whole is being maintained actively and the last commit was on December 15, 2014, which is less than a month ago at the date of writing this answer. Furthermore, it begs the question how quick and easily this project can be out of date, considering HTML has been a well established standard (or proposal) for a long time now and OWASP's XSS listings (one of several) have been developed into maturity for some time now. As Caja works by white-listing content, even changes in this space doesn't render Caja out of date per se. Only if existing white-listed elements behavior change in a new HTML version, say, an accepted attribute on an accepted element starts accepting scripting behavior, only then is Caja in trouble. What exactly is white-listed is an easy config change (unfortunately, which requires a custom build in Linux).

Personally, I used this in a major project recently to solve a known security issue. Our security department was happy with the solution including Caja.

Regarding AngularJS ngSanitize

A question that popped into my mind is: if Caja is maintained by Google and AngularJS developed by Google, why doesn't ngSanitize use Caja? It seems the AngularJS team reinvented the wheel. This does not bode well for Caja's relevancy, if Google itself doesn't use its own projects. That or they simply needed to be independent to be quick enough without depending on the Caja team. ngSantize however has been updated very recently as opposed to Caja's javascript sanitizer. On the other hand, ngSanitize exists on GitHub only since 2012 (and can't be older than Angular itself), while Caja has been developing it's sanitation procedures since 2006 and has had a lot of time to put the finishing touches while AngularJS was just getting born. I really can't tell which is more up-to-date.

[Is the Google Caja safe JavaScript project still useful and being maintained?]
