9-1-2013

Finding Information Leaks in JavaScript

Thomas Ashmore
Harvey Mudd College

Recommended Citation
http://scholarship.claremont.edu/interface/9

This Research is brought to you for free and open access by the HMC Student Scholarship at Scholarship @ Claremont. It has been accepted for inclusion in Interface Compendium of Student Work by an authorized administrator of Scholarship @ Claremont. For more information, please contact scholarship@cuc.claremont.edu.
Can browser add-ons leak your personal information?

Yes, some do!

How can we detect information leaks?

Currently, Mozilla volunteers manually inspect the source of popular addons for leaks. Our tool helps automate this process.

We create a graph of information flow between program statements. By tracing backwards from a call that sends data to the internet, we can determine whether sensitive information may be leaked.

The analysis requires us to approximate the program’s behavior. We use abstract interpretation to determine when two pointers could reference the same memory location.

Can the tool be improved?

Our analysis often reports potential leaks where none exist. We can help users identify false positives by classifying different types of data dependence.

Exceptions and goto statements alter the order of statement execution. It’s tricky to determine how they affect data flow. Our current method could be improved.

Acknowledgements & Citations

Thanks to Vineeth Kashyap and the rest of Ben Hardekopf’s team at UCSB!


All images belong to T. Ashmore and V. Kashyap, with the exception of company logos. The FireFox logo, Chrome logo, SurfCanyon logo, AdBlock Plus logo, and Ghostery logo belong to Mozilla Foundation, Google, Surf Canyon, Imprint, and Evidon, respectively.
x = new A()

w = x

z = x

if (w == z)

y = new B()

w.f = y

v = z.f