K.J. Grewal
01/08/2012
SE-4930 – Article 2

Verified Security for Browser Extensions

2011 IEEE Symposium on Security and Privacy - Session: Browsing Security and Privacy

Summary

This journal article discusses the security and reliability vulnerabilities associated with browser extensions due to the current methodologies implemented by browser vendors. The authors state that extensions are very different from JavaScript served on web pages since extensions can access cross-domain content, make arbitrary network requests, and can use local storage. A malicious or buggy extension can easily void many security guarantees that a browser tries to provide and can affect page load times and browser responsiveness as well. Browser vendors have put in place various processes to control how extensions are distributed, installed, and executed, but an empirical study of over 1,000 Chrome extensions has shown that current models are not very effective in limiting the privileges of extensions.

The authors of the article go into detail about not only what flaws currently exist, but they also provide a new framework called IBEX which they believe can be used for authoring, analyzing, verifying, and deploying secure browser extensions. A brief overview of IBEX can be found on the diagram on the next page.

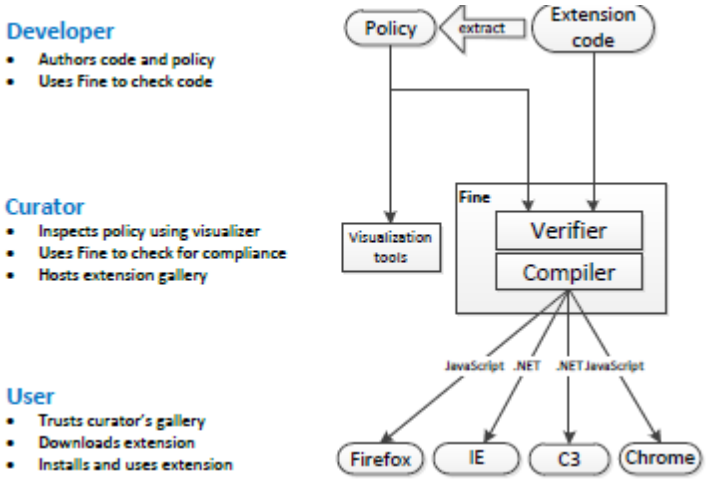

Figure 1: IBEX overview

This model applies to three main groups: extension developers, curators of extension hosting services, and end-users. The IBEX model depends on various features and policies including static checking of extension safety; this is done by employing a methodology which verifies that extensions are written in Fine, an ML dialect (shown in the figure above), and satisfies the safety condition. This static verification eliminates runtime security monitoring, and promotes robustness of the browser platform since extensions can never raise unexpected security exceptions. Additionally, the static verification is also supported by the fact that there will be a browser-agnostic API for extensions for high-level languages such as .NET and JavaScript.

The IBEX framework also stresses the importance of defining policies which will be done using a logic-based policy language. This language will allow for the specification of fine-grained authorization and data flow policies on web content and browser state accessible by extensions. Most importantly, the authors plan on employing multiple code generators implemented by the Fine compiler. This will allow the same extension source to be deployable

in multiple browsers. A key aspect of this feature is the use of the aforementioned browser-agnostic API, combined with the use of a standard ML-like source language.

What I learned

By reading this article, I understood the vast security vulnerabilities that accompany installing and running a browser extension. Prior to reading this article, I was not aware of all of the capabilities of a typical browser extension and how a user's system can become susceptible to a wide-array of security threats. I believe that this article will help me in my own web development and will also make me conscious of the risks of extensions before installing and using them. One question that I still have is whether this IBEX framework will gain widespread publicity in the near future and what preliminary steps will be necessary to allow this to be a web standard?