Most modern browsers have an extensible architecture that facilitates enriching the browser ecosystem with extensions developed by third-party developers. Such extensibility that leads to feature-rich browser thus essentially popularizes it among both developers and end-users. Towards the goal of enhancing the ecosystem with extensions characterized by useful and diverse functionality, browser vendors expose APIs that endow the developers with privileges to access various system resources. At the same time, in order to safeguard the browser from any security threats caused by these extensions which are often untrusted, architecture needs to restrict their authority. To that end, usually browser vendors recommend structuring extensions such that they follow certain programming and security principles, violation of which can result in exploitable vulnerabilities. However, the vendors transfers this burden to the third-party extension developers, and demands both their expertise and meticulous effort. Since these violations are hard to detect via manual inspection, extension developers need diligent skillfulness and security software tools necessary to identify them; today, even if they have the former, they lack the latter.

My research addresses the above issues with secure extension development process. In this talk, I will describe the use of program analysis and software engineering techniques to automate: (1) detecting extension vulnerability caused by developers’ failure to adhere to security and programming principles and (2) transforming legacy vulnerable extensions so that they conform to these principles resulting in enhanced security guarantee.