**After you, please: browser extensions order attacks and countermeasures:**

https://link.springer.com/article/10.1007/s10207-019-00481-8#Sec1

Browser extensions are small applications executed in the browser context that provide additional capabilities and enrich the user experience while surfing the web. The acceptance of extensions in current browsers is unquestionable. For instance, Chrome’s official extension repository has more than 63,000 extensions, with some of them having more than 10M users. When installed, extensions are pushed into an internal queue within the browser. The order in which each extension executes depends on a number of factors, including their relative installation times. In this paper, we demonstrate how this order can be exploited by an unprivileged malicious extension (i.e., one with no more permissions than those already assigned when accessing web content) to get access to any private information that other extensions have previously introduced. We propose a solution that does not require modifying the core browser engine, since it is implemented as another browser extension. We prove that our approach effectively protects the user against usual attackers (i.e., any other installed extension) as well as against strong attackers having access to the effects of all installed extensions (i.e., knowing who did what). We also prove soundness and robustness of our approach under reasonable assumptions.

Web browsers have become essential tools that are installed on nearly all computers. The most popular browsers as of this writing (April 2018) are Chrome (77.9%), Firefox (11.8%), Internet Explorer/Edge (4.1%), Safari (3.3%) and Opera (1.5%). Most browsers allow users to install small applications, generally developed by third parties, that provide additional functionality or enhance the user experience while browsing. Such plug-ins are known as browser extensions and they interact with the browser by sharing common resources such as tabs, cookies, HTML content or storage capabilities. As of May 2017, the Chrome Web Store (the official repository where all Chrome extensions are stored and distributed) contains more than 135,000 extensions, whereas for the case of the second most popular browser (Firefox), its extension store contains almost 70,000 items.

When an extension is installed, the browser often pops up a message showing the permissions this new extension requests and, upon user approval, the extension is then installed and integrated within the browser. Extensions run through the JavaScript event listener system. An extension can subscribe to a number of events associated with the browser (e.g., when a new tab is opened or a new bookmark is added) or with the content (e.g., when a user clicks on a HTML element or when the page is loaded). When a JavaScript event is triggered, the event is captured by the browser engine and all extensions subscribed to this event are executed.

In this paper, we focus on Chromium, which is an open source browser and the basis for Chrome, Opera, Brave, Edge Chromium or Yandex browsers. Extensions installed in Chromium can also run in all mentioned browsers. The execution engine is exactly the same in all the browsers and follows the same pipeline model that will be explained in some detail later. For this reason, we will refer to Chrome and Chromium interchangeably. Extensions in Chromium can be of three types: content scripts, background pages or both. In what follows, our main focus is on content scripts, which are JavaScript files that run in the context of the loaded web page. It is important to emphasize that the main aim of content scripts is to access and interact with the Document Object Model (DOM). This fact alone raises a fundamental privacy question, since it is explicitly assumed that extensions will have full access to any (sensitive or not) content that the user is accessing. Browsers (including Chromium) dodge this issue by assuming that the user should trust the extension before installing it. In this paper, we do not address this problem, which is essentially related to determining if an extension’s behavior is benign or malicious, but a related one described in what follows.

When analyzing the security and privacy implications of browser extensions, one question that has been largely overlooked is the potential leakage of information among extensions. In nearly all browsers, each content script uses its own wrapper of the DOM to read and make changes to the page loaded by the browser. They also run in a dedicated sandbox that the browser provides for security reasons. However, there is no isolation in terms of privacy, since all changes an extension performs in its own DOM are automatically synchronized with the main DOM. One straightforward—but nonetheless important—consequence of this is that a malicious extension could eavesdrop on other extensions (i.e., it can get access to the data they put on the DOM and observe their actions) and even manipulate their behavior by acting on their DOM elements (e.g., clicking on elements introduced by another extension). An attacker can exploit this using two different strategies:

1. Exploiting the order The way Chromium manages extensions introduces a default execution order among extensions with undesirable consequences. One key issue is that the nth extension in the pipeline can learn all contents introduced by the first n−1.
2. Order-independent attacks Some attacks enabled by the absence of effective isolation among extensions’ actions do not require exploiting the execution order (i.e., getting the malicious extension to be placed at the end of the execution pipeline). However, exploiting the order provides the attacker with a privileged position that facilitates such attacks, which will result in a simpler code for the malicious extension that will increase the chances of passing the analysis performed by official stores. Furthermore, not all attacks might involve adding event handlers, since access to information put in the DOM will only be possible once the attacked extension has executed.

We have experimentally verified the previous attacks and demonstrated, for instance, that an extension with no privileges can learn which pictures a user likes in Pinterest or change the picture a user wants to share; that it can tamper with the notes and events provided by the popular Evernote Web Clipper; or that it can profile the user’s video browsing preferences in YouTube. This lack of effective isolation is not only inherent to Chromium’s extension model, but also explicitly acknowledged. Browsers such as Chrome do not even attempt to guarantee some form of “non-interference” among extensions. On the contrary, developers are encouraged to implement appropriate mechanisms to protect any sensitive information that ends up in the DOM, since it is assumed that other extensions could simply read or manipulate it. Even if browsers do not factor this into their threat model, we believe that this is a serious vulnerability that has not been discussed before. More importantly, it can be easily exploited by a malicious extension, regardless of the fact that it is explicitly assumed in the browser’s extension model or not.