Security Analysis of Chrome Extensions

Amanda Li, David Rosales, Tiffany Yang

6.857 Spring 2019 Final Project

Abstract

In this paper we look at the security of Chrome extensions and what enables malicious attacks through Chrome extensions to happen. We take a look at past attacks and organize them into 2 categories - attacks from intentionally malicious code within the extension and attacks made possible by the insecurity of benign extensions. Through an interview with a Google Employee and studying documentation, we assess the measures the Chrome Team takes to ensure security. Finally, we recommended some improvements in order to curb the security risk for both users and developers.

Introduction

Background

Browser vendors like Chrome, Firefox, and Opera provide a way for people to access the internet in a user-friendly way. These browser vendors also have teams dedicated towards ensuring that their users' privacy is protected and that their software is not exploitable. In an effort to make a browser lightweight yet powerful, browser vendors tend to settle on key features that handle the majority of use cases. However, it remains a fact that users have different internet needs and wish to use their browsers differently. To fill this gap of personalization, Chrome and many other popular browsers support extensions. These extensions are community driven software programs made with the intention of customizing a browser. Extensions complicate security in the sense that, although they provide additional functionality and features, they increase the number of entry points an attacker can use. For the purposes of this project, we will focus on Chrome extensions.

Motivation

Although there are already security measures in place that prevent web applications from accessing data outside their domain and on a user's computer, Chrome extensions themselves have more permissions. Because Chrome extensions are more privileged, they potentially have access to sensitive user information. Web applications can exploit this and, through the Chrome Extension

API, can communicate with and learn this information [Ionut Arghire]. If the extension itself is vulnerable, an attacker can execute malicious code and even trigger downloads of malicious software.

Attackers can also directly use Chrome extensions to conduct an attack. A common type of attack is a man in the middle attack [Information Age]. A Chrome extension can directly manipulate contents on a page or read inputs to that page. If the extension itself is malicious, a user could accidentally supply an attacker with sensitive information, execute malicious code, and download malicious software.

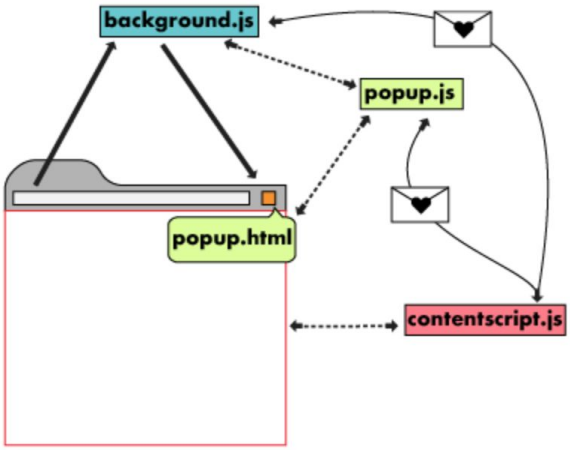
Extensions are powerful tools that can add features that cater to different users and different needs. However, browser extensions in general pose major security risks. There is potential for them to be exploited and used by an attacker, despite being made with good intentions. There is also the possibility that the extension itself was created by an adversary and is meant to do something malicious. We wish to research how much damage an adversary can do and how a user can be protected from their attacks.

Extension Architecture Overview

Architecture and complexity of extensions depend on the function of the extension. Chrome extensions are compressed files consisting mostly of HTML, CSS, JavaScript, images [Extension Overview]. Most extensions include:

- Manifest.json - contains extension file information and permissions
- Background Scripts - handles extension events
- UI Elements - lets users interact with the extension
- Content Scripts - lets the extension read or write on a webpage

An example system diagram is shown below. The popup is the UI of the extension. The content script passes messages about the webpage to the rest of the scripts.



Chrome Team Security Measures

Security Model:

The base structure of Google Chrome extensions allows for a certain amount of security in and of itself [Barth, Felt, Saxena]. The security model is based on three aspects: least privilege, privilege separation, and strong isolation. Least privilege restricts extensions from having certain privileges. This requires extensions to request their privileges explicitly on their manifest. Thus, since privileges must be declared at install time, a potential attacker would only have them at runtime. Privilege separation separates an extension's privileges into three separate components, making it harder for an attacker to take advantage of a user's machine. Finally, three different isolation mechanisms are used to separate the extension components from each other and from web content. This prevents attackers from accessing multiple parts of a system at the same time, as the components reside in different processes.

Additional Security Measures:

Having been around for nearly ten years, there has been additional work done to ensure the security of Google Chrome extensions for users [James Wagner]. One implementation to protect extensions from automatically accessing data from websites they are deployed on, is to require permission from the host, usually by the click of a button. This way there is more transparency on the side of the user such that they know what their extension is doing and reading. To improve transparency further,

Google has worked to remove inline installations of extensions, which are extensions installed from an outside website. As of mid-2019, inline installations have been completely disabled. Instead, all extensions will be downloaded straight from the Chrome Web Store, improving safety and reliability. There have also been harsher restrictions imposed in relation to the content of extensions, namely on obfuscated code. Obfuscated code has been found to be contained within more than 70% of malicious extensions that Chrome has blocked from its web store. It also makes the review process considerably harder as it mainly used to conceal code functionality. For those that wish to use it for concealment for their own benefit (protect proprietary code), it is not completely sufficient, as it doesn't provide complete protection from a truly motivated attacker and also results in hefty performance costs, such as slower execution time and larger memory footprints. Instead, to make the review process more straightforward, Google recommends minification. This make the code both faster and more straightforward to review, as code size is reduced (shortened variable and function names, removed white space, newlines, etc.).

Finally, to provide added security for the extension developers, a new required 2-step verification process has been implemented to prevent attackers from hijacking their accounts [James Wagner]. This provides an extra method of authentication, either from a phone or using a physical security key. It is especially helpful in the case of popular extensions, as oftentimes these are more likely to attract attackers who wish to hijack the developer account. Using a physical security key provides the same amount of security that Google would give for its own employees. On a more personal note, Google Chrome extensions also update automatically, meaning that users will constantly receive any security updates that are rolled through new patches, which could better serve to protect them from new vulnerabilities that could pop up.

Security Measures of Chromium (Interview)

A Google software engineer was interviewed regarding his experiences using and developing for Chromium, an open-source web browser that Chrome is based off of. We believe that Chromium was Google project that is useful to learn more about, as it supplies the large majority of code for Chrome. Therefore, vulnerabilities in Chromium could lead to vulnerabilities within extensions as

well. The breaches in security as a developer in Chromium could be also related to those in relation to extensions. The main ideas of the interview are summarized below:

As a developer, a number of security measures are put into place to help catch vulnerabilities. When implementing a feature, during the design review, the team determines whether or not they might need a security review. If so, they reach out to a privacy/security review team with the design proposal, who check it for security fallacies. In addition to that, there is also a general security team that constantly tries to find vulnerabilities in the system -- if one is found, the developers will be notified promptly. This constant checking of vulnerabilities serves to help find issues quickly, and not only just when a change is being made.

Past Attack Analysis

Despite Google's attempts to purge the extension store of malicious malware and fix known vulnerabilities, extensions have been and are still a common target of exploits in recent years. Through looking at at past attacks, we determined that most attacks through Chrome can be organized into 2 categories:

1. Widely used Chrome extensions were somehow exploited and turned malicious.
2. Malicious Chrome extensions were posed as harmless extensions and downloaded from the app store.

Within each category, we looked at what features of extensions presented possible threats to the security of its usage. In this threat model, we assume that the operating system, browser, and plugins are benign.

Harmless Extensions to Malicious

Harmless extensions can turn into malicious extensions most often due to human error. Extensions can leak data depending on the implementation. It is often up to the sensibility of the developer how much the extension leaks and how secure an extension is made. While the Chrome Team issues guidelines for users to create a secure extension [Developer Guide], it is not always guaranteed that the developers will follow the guidelines closely. Chrome recommends that the developers access as

little information as possible as leaky extensions will expose the user and the user's privacy. For example, in February 2018, it was revealed that the extension Grammarly grabbed authentication tokens, and could have allowed potential attackers access to information such as user history and documents [Swati Khandelwal].

Through obtaining the developer's account credentials, widely used Chrome extensions could turn into malicious extensions. The number of methods for obtaining such credential are not constrained to vulnerabilities pertaining to the browser or it's extensions. Between May and August of 2017, eight Chrome extensions were compromised in such a manner [Mark Maunder]. Developers of the extensions had their chrome developer account credentials stolen through a phishing attack. By injecting malicious JavaScript code, adversaries were able to steal user credentials and malvertise. All together, around 4.8 million users were targeted.

Existence of Malicious Extensions

Chrome extensions rely on user permissions to obtain abilities such as reading and modifying web pages as well as the ability to capture desktop information. There is a compromise between complexity and security when designing the permission model. While Chrome tries to separate permissions such that users understand what they are compromising [List of Permissions], separating permissions too finely can result in convoluted development process. In addition, users do not always read or understand what powers they are granting as not all users are technically apt. Around late 2017/early 2018, several malicious Chrome extensions were discovered by researchers at network security vendor ICEBRG [Christopher Kanaracus]. Chrome's execution of JavaScript code is contained within a Sandbox, but extensions are capable of retrieving and processing code from an externally-controlled server if given the permissions. This allows the extension author to inject and execute JavaScript code.

The existence of malicious extensions, despite Chrome Team's efforts to ensure user security and privacy, seems to be inevitable in the large inventory of the Chrome Web Store. Each extension goes

through a review process before being published in the webstore to ensure the exclusion of malicious extensions (this will be discussed in depth later).

In September of 2018, an attacker uploaded a malicious version of MEGA's Chrome extension [Lawrence Abrams]. MEGA is a cloud storage and file hosting service offered through web-based apps. When users updated to the newest version of the extension, the extension asked for additional permissions from the user. Malicious code in the trojaned version checked for login form submissions by checking for keywords such as "login", "user", and "username" and sent found credentials to a remote host. This attack was made possible due to the lack of security measures on both Chrome's extension management and the scope of information that extensions have access to when given permissions.

In addition, we have found that it take very few lines of code to turn an extension malicious. As a proof of concept, our group implemented several malicious activities in extensions assuming that the extensions already had permission. We implemented the following:

1. Denial of Service attack by closing all tabs
2. Redirecting from one site to another
3. Tracking of login credentials

Turning the extension malicious is rather trivial once an attacker has an extension ready to modify. These implemented attacks used rather common permissions such as "tabs" and "contextMenu". Chrome extension updates are pushed out automatically in order to ensure that users have the latest version of the extension. This means that once an updated extension is malicious, users will automatically receive the update.

Extension Distribution

In order to understand how attacks through Chrome extensions, can affect users, we need to know how a malicious extension can be distributed and installed into a user's browser. Each distribution option available to developers could potentially be an entry point for an attacker. For that reason,

Google Chrome has, over time, been limiting and restricting these installation entry points. There exists a difficult trade-off between availability of extensions and security.

External Installation

The most accessible vulnerability for attackers was the distribution method of external installation. Through any Google Chrome browser, a developer can package their extension source code into a CRX file (file extension .crx) using the developer tools. External installation allowed another user to take the CRX file and drag it into their browser to install it. The original purpose behind external installation was to allow developers to publish and distribute their own extensions. This allowed extensions to be included as part of a software package. However, external installations allowed attacks because of the lack of verification involved or checking done on the Chrome extension being installed. No third party needed to review the code in order to have it installed, meaning an attacker could distribute a malicious extension without anyone to stop them. Because of this, Chrome developers have removed this installation functionality entirely for both Windows and OSX operating systems. Although developers can still package their extensions into CRX files, a user can no longer install it to their browser [Hosting Changes]. Linux, however, still allows external installation to Chrome browsers.

Unpacked/Developer Mode

A major design goal of many platforms is to make it easy for third party developers to generate content. The Chrome browser does this through the Developer Mode setting in chrome://extensions. Once enabled, a developer can load the source code of an extension and run it. This improves the speed of testing code and makes developing Chrome extensions more accessible. The danger of Developer Mode is that any user can enable it and load the source code of an extension. Even though external installation was removed as an installation method, an attacker could still distribute an obfuscated version of their extension's source code to unsuspecting users. However, Chrome issues notifications if Developer Mode is enabled for an extended period of time. If a user was to enable Developer Mode, install an unpacked extension, and then disable Developer Mode, warnings about the danger of installing an unpacked and externally downloaded extension

would be given to a user appear sparsely and periodically. As such, an attacker could convince a user to follow these steps and install a compromised extension to their browser.

Chrome Web Store

The most secure and recommended option for Chrome extension distribution is through the Chrome Web Store. Hosted by Google itself, the Chrome Web Store is a website where users can browse for and install Chrome extensions. In order to have an extension listed on the Chrome Web Store, the developer must pay an account fee. Once the fee is paid and the extension is uploaded, the extension is subjected to an automated review that scans the code and checks for breaches in the security policy and extension guidelines. Once approved, other users can then find it in the Chrome Web Store and install it to their browsers. Chrome extensions can also be configured in such a way that updates are automatically pushed to users that have the extension installed. Should an attacker get past the automatic review, they could post a malicious extension to the Chrome Web Store without arousing suspicion. Further, an attacker could post updates to an existing extension through someone else's compromised developer account.

Chrome Web Store Review Process

Google has an automated review process for inspecting new and updated Chrome extensions [Approval Process]. If suspicious code is found at any point during the automated review, the extension is flagged for review by a human. Periodically once the extension is approved, this automated review will be conducted, taking into account suspicious activity, ratings, and reports [Pending Review]. If the extension is deemed to be breaching any of the guidelines or security principles, it can be removed from the Chrome Web Store and further disciplinary action might be taken upon the developer's account.

Although Google has made attempts to improve the security of Chrome extension distribution, some vulnerabilities still exist. Attackers could still convince users to install CRX files to their browsers on Linux. Users can also install potentially malicious extensions through the Developer Mode settings. If a developer's account is compromised, an attacker can post malicious updates that,

if not caught by Google's automated review, will be instantly pushed to all users who have it installed.

Chrome Recommendations

Developer Mode Warnings

Enabling Developer Mode allows users to install potentially malicious extensions that have not been reviewed by any third parties. A user that does not know the intended use of this setting might be misled and convinced by an attacker to use it as a way to install a compromised extension. Chrome should have more warnings and notifications regarding Developer Mode to educate users about the potential risks involved.

Disabling Developer Mode Installed Extensions

Once a Chrome extension is installed from source code via Developer Mode, the extension is treated the same as the rest of the extensions. Once Developer Mode is disabled, the extension is allowed to keep running. Periodically, a notification will appear giving users the option to disable the Developer Mode installed extensions. However, because this functionality exists solely to help developers create and test their own extensions, there is no need to keep these extensions running after Developer Mode has been disabled. After disabling Developer Mode, all unpacked extensions should also be disabled.

Manifest Modifications

As it is right now, the permissions can be more finely separated such that there is less of a need for overly-broad access in extensions. Instead of giving all parts of the extension the permissions listed, permissions should be differentiated amongst the various components of the extension. In this way, it would allow less leaks in the extension if it were to grab sensitive information from the users.

User Ability to Restrict Extensions

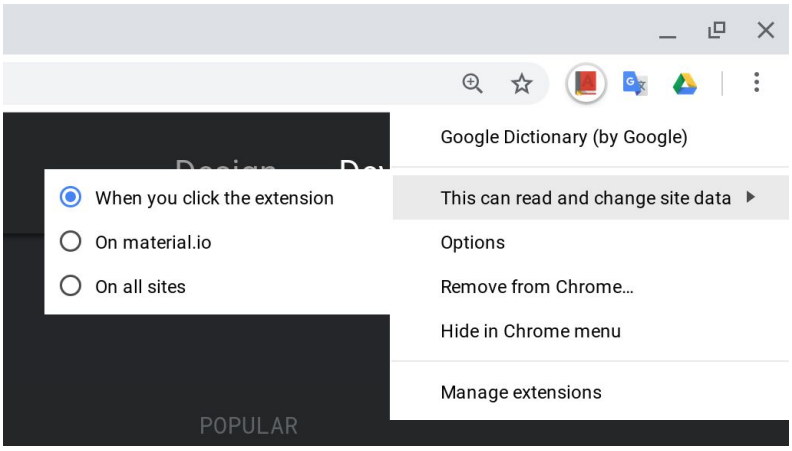
Users should be given readily available tools to modify the reach that extensions have. For example many extension permissions requests "https://*/*" permissions, which allows them to read and

send HTTPS requests from all sites. Users should have the power to modify the permission to more specific sites such as "http://*.facebook.com/*" instead, or be able to disable extension permissions on certain sites. This would protect websites such as banking websites or personal

User Recommendations

User Permission Awareness

Beyond reading reviews, users should read more closely what permissions they are giving to the extension, especially if the extension asks for additional permissions when being updated. Users should protect themselves by limiting instances of when extensions have control over their sites. For example, users can toggle when an extension can modify site data:



User Information Storage

Given the importance of passwords, we would recommend extension developers (and developers in general) to never store plaintext passwords directly onto a machine: everything should be encrypted to make it harder for an attacker to directly obtain them in the case of a hack. Furthermore, it should be advised to only store what you absolutely need. With everything localized on a computer, this makes it even easier for an attacker to obtain all private information.

Conclusion

Google Chrome extensions can be powerful tools that customize and enhance a user's web browsing experience. In the interest of giving developers the freedom to make useful tools, Chrome extensions offer a large subset of permissions over the browser. This can be a big security vulnerability if an attacker chose to create a malicious extension. Google has made attempts to remediate the problem by enforcing policies and guidelines as well as restricting the methods of distribution. However, vulnerabilities still exist through external installation on Linux, Developer Mode installations, and malicious Chrome Web Store installations that made it past an automated review. In order to combat this, we recommend that Google do more to educate both its users and its content developers on potential security risks and how to avoid them. Furthermore, giving users more control over and access to information about their installed extensions could help them to avoid attacks.