A Framework for Analysing the Security of Chrome Extensions

V. Aravind and M. Sethumadhavan

TIFAC Core in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore, India
aravind.venkitaraman@gmail.com, m_sethu@cb.amrita.edu

Abstract. Google Chrome, the most popular web browser today, allows users to extend its functionality by means of extensions available in its own store or any third party website. Users can also develop their own extensions easily and add them to their browser. Vulnerabilities in browser extensions could be exploited by malicious websites to gain access to sensitive user data or to attack another website. A browser extension can also turn malicious and attack a website or steal user data. This paper proposes a framework which can be used by users and developers to analyse Chrome extensions. The technique presented here uses the permissions feature of chrome extensions and the flow of data through the extension's JavaScript code to detect vulnerabilities in extensions and to check whether the extension could be malicious.

1 Introduction

Internet and web applications have become an essential part of our everyday life. We take the help of web applications to carry out a sizable number of our day to day activities.Most of the research activity in web security till now has been concentrating on securing the web applications. Web browsers and browser extensions may also have security vulnerabilities which could be exploited by an attacker.A browser extension is a third-party software module that extends the functionality of a web browser and lets users customize their browsing experience. Browser extensions modify the core browser user experience by changing the browser's user interface and interacting with websites. The two most popular web browsers, Mozilla Firefox and Google Chrome, support a number of extensions.Users can install extensions from the browser's extension module or from third party websites. In order to enhance the user experience the extensions are allowed to access various components like the current web page source, the browser history, bookmarks and if required, the end user's file system as well. Browser extensions are not always developed by security experts and hence are likelyto have vulnerabilities. A malicious website can exploit these vulnerabilities to gain access to the browser data or the file system. On the other hand, a malicious extension can use the same privileges to steal sensitive user data or to attack a website.

The Google Chrome Extension System follows an architecture which helps in reducing the security vulnerabilities to a certain extent. The principle of least

M.K. Kundu et al. (eds.), *Advanced Computing, Networking and Informatics - Volume 2*,
Smart Innovation, Systems and Technologies 28,
DOI: 10.1007/978-3-319-07350-7_30,

privilege, one of the mechanisms proposed in the architecture has to be enforced by the extension developer. This opens up vulnerability and an attack surface for a malicious website. Users do not always install extensions from the Chrome Web Store. If an attractive extension is available from a third-party website, an ordinary user may add the extension to his browser. If a user installs a malicious extension on his browser, the extension can perform malicious activities. Most of the research on extension security till now has assumed the extensions to be benign and the websites to be malicious. However with users installing a lot of third party extensions, a new class of malware is available to the not so benign developers. Hence, both cases need to be addressed while analysing the security of a browser extension. The remainder of the paper is structured as follows. Section 2 gives a glance of the Chrome Extension architecture and the previous research done in the field of browser extensions. Section 3 describes the framework being proposed and the expected results and section 4 concludes the paper.

2 Browser Extension Security: The Story So Far

Security of browser extensions has become an interesting research area for computer scientists. One of the most popular web browsers, Mozilla Firefox has an extension architecture which acts as a platform for a lot of developers to create extensions. A study about the security of these extensions revealed the vulnerabilities existing in many popular Firefox extensions and that the root cause of these vulnerabilities was in the underlying architecture. This led to the development of new extension architecture with focus on security, which is now known as the Chrome Extension Architecture[1].This section briefly describes the previous work done in the field of browser extension security.

2.1 Chrome Extension Architecture

The architecture of a Chrome extension divides every extension into three components: Content Script, Extension Core and Native Binary. Content Script, written in JavaScript, is the part of the extension which directly interacts with the web page. Content Scripts can access and modify the page DOM. Extension core, written in HTML and JavaScript is the part of the extension which has access to the APIs used to interact with the browser. Native binary is the part of the extension which can access the host machine with the user's full privileges [1]. From an implementation point of view, a chrome extension consists of a properties file, *manifest.json* where the metadata about the extension including the permissions are mentioned, one or more HTML pages which display the UI, one or more JavaScript source files which perform the underlying functionality and an image file which is used as the icon.

The Chrome extension system follows a security model based on three main concepts: least privilege, privilege separation and strong isolation. Least privilege is implemented by restricting the extensions permissions in the manifest file. An attacker who compromises the extensions is also limited to these privileges thereby reducing the intensity of attacks. Privilege separation is inherent in the architecture itself which separates the extension code into content scripts, extension core and

native binaries. Strong isolation is achieved by isolating various components of an extension as different processes and also by running the extension's content script in a different environment from that of the untrusted website. Thus Chrome extension system has inherent mechanisms to enhance the security of extensions. Still, there are attacks possible against these extensions due to improper implementation by the extension developers.

2.2 Chrome Extension Security Review

A study about chrome extension security analysed the effectiveness of the security mechanisms implemented in the Chrome extension architecture [2]. Extension code is written primarily using JavaScript and HTML. The isolated worlds and privilege separation mechanisms are effective in protecting the extension from Data as HTML vulnerabilities and the extension core from attacks on vulnerable content scripts [2]. The permissions system or the least privilege mechanism will be effective only if it is used properly by the developers. The extension system does not restrict the permissions that can be granted to an extension.

Browser extensions are not always on the receiving end of attacks. Extensions are increasingly being used as attack vectors against websites and user data.An experiment to create malicious chrome extensions which launch various attacks on websites and user data turned out to be successful. The group which conducted this experiment were successful in performing Email spamming, DDoS and password sniffing attacks on various websites using chrome extensions [3]. The attacks were as simple as injecting some content script on the web pages, steal the user data and pass on the information to a remote website though a HTTP request. Thus even the seemingly secure chrome extension system is found to be vulnerable to various attacks and it also act as agents in attacks on websites.

2.3 VEX – A Tool for Vulnerability Analysis of Firefox Extensions

VEX is a proof-of-concept tool used to detect potential security vulnerabilities in browser extensions using static analysis of explicit flows.VEX analyses Firefox extension source to find possibly vulnerable flows in JavaScript [4]. The tool tokenizes the JavaScript code and defines rules based on which it analyses the flow of information from a source to sink. VEX analyses flow of web page content to **eval** and **innerHTML** methods and the flow of data from browser's DOM API and XPCOM components as RDF to **innerHTML**. The concepts used in VEX can be extended to Chrome extensions also, as both use JavaScript as the primary development platform.

3 Proposed Framework for Chrome Extension Analysis

We now propose a new framework for analysing chrome extensions based on the principle of least privilege and the flow of information. The proposal is to develop a static analysis tool which will analyse any given chrome extension and provide results based on which we can decide whether the extension is vulnerable, malicious or safe. The system can be divided into two modules which are detailed below.

3.1 Analysis of Least Privilege

Fig. 1 illustrates the Least Privilege Analysis module. As discussed in the previous section, every chrome extension has a manifest file in which we define the permissions required by the extension [5]. Each permission is associated with a set of JavaScript methods which can be used only if the corresponding permission is assigned. This module reads the manifest file of an extension and retrieves the set of permissions assigned and JavaScript files used in the extension. Then it parses all the JavaScript files used and looks for functions or attributes associated with the permission. The methods and attributes corresponding to a permission-type are mostly used along with a prefix which is usually **chrome.<permission>.<name>**[5]. We can scan the JavaScript files for attributes and functions with the corresponding prefix. Alternately, we can maintain also a list of functions corresponding to every permission and compare the list of functions retrieved from the JavaScript files with this. If none of the functions corresponding to a permission are used in the extension, the permission should be deemed as unnecessary and we report that the extension violates the principle of least privilege.

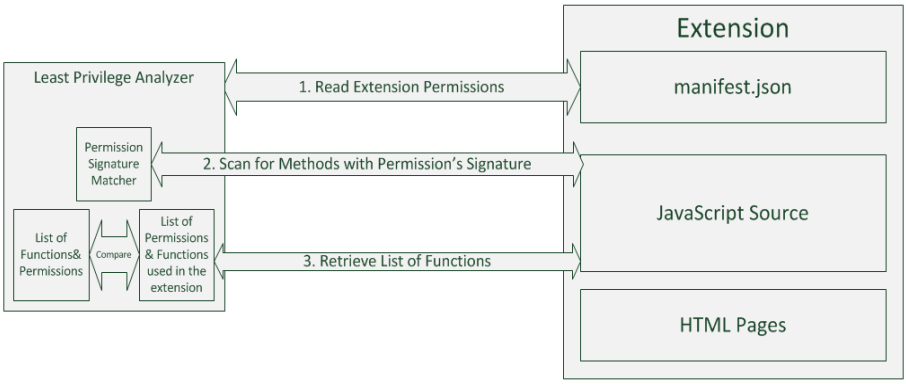

Fig. 1. Block diagram of Least Privilege Analysis Module

3.2 Analysis of Malicious Information Flow

Fig.2 illustrates the Malicious Information Flow Analysis. A browser extension which has permissions to invoke other URLs can send user data from a web page or the browser history/bookmarks to a remote server. A benign end user will never realize this, as the extension would have another utility which requires the same level of permissions. In order to detect such malicious behaviour in a given extension, this module performs an analysis of the information flow. We retrieve the objects and variables which read data from a web page or from the browser internals. It has to be verified whether any of these object values are being forwarded to a different server. This can again be done by analysing the flow of the information. If the data is being passed to some method or URL which is sending a request to a remote server, the extension may be leaking some sensitive user data. Another activity that can be

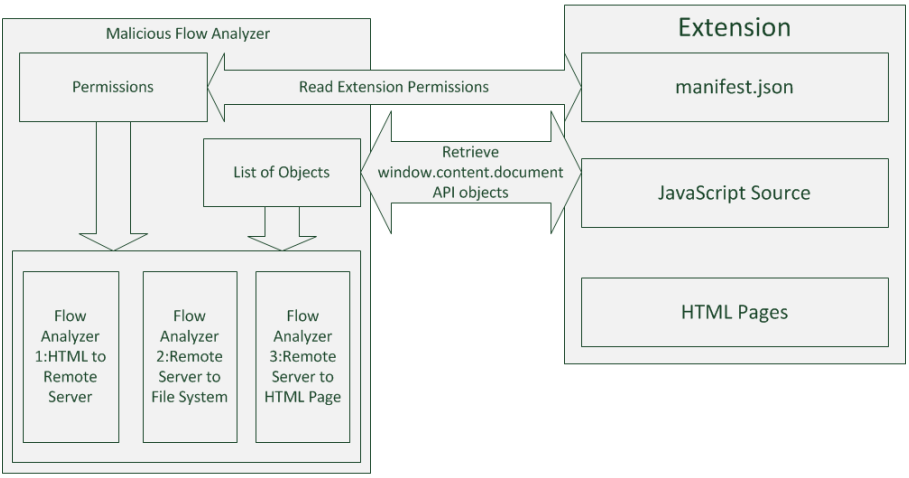

Fig. 2. Block Diagram of Malicious Flow Analysis Module

performed by a malicious extension is downloading some malware from a remote server to the user's system. To detect such a malicious behaviour, first of all we need to check if the extension has privileges to download files to the end user system. If it has the permission, then retrieve JavaScript objects which get their values as a response from a remote server. If any of these objects are being saved to the file system, the extension can be considered malicious. A malicious extension sometimes may act as a bot for attacking a website which the user frequents. The extension may retrieve a payload from a remote server and upload the same to the web page being accessed. To detect such behaviour, we need to check for JavaScript objects/variables which are being uploaded to the current web page. This can be done as simple as running a content-script. If such a JavaScript object gets its value from a remote server, it can be considered as malicious.

3.3 Discussion

The proposed framework helps to detect some of the key issues in Chrome extensions. The privileges assigned to an extension are by default available to a malicious website which is trying to exploit the extension's content script. The proposal makes sure that all extensions follow the principle of least privilege so that the attacks which require any further privileges are stopped at the initial stage itself. The proposal also takes care of detecting malicious extensions so that end user does not become a victim of attacks by malicious extensions. The framework will be able to detect malicious information flows through the extension which may be an attack vector against the user's data or any website. Further, the framework is platform independent. A developer can take this framework and implement a static analysis tool in any language and platform of his choice.

4 Conclusion

The popularity of browser extensions is on rising side and users are always in search of attractive add-ons for a better browsing experience. This trend also attracts malicious developers to explore a new attack surface. Hence a model for analysing the security of these extensions will definitely help developers and users to make sure they are on the safer side. The framework presented here helps in alarming the end users about vulnerabilities that could exist in an extension they are planning to use, so that they can take an informed decision. It can also be used by developers to make sure their product is free from vulnerabilities. Moreover, the framework also detects maliciousnature of extensions. This will protect users from becoming victims of extension based malware attacks as well. However, at the end of the day, the impact of the idea proposed here depends upon the developer or end user who uses the result obtained by this analysis.

References

1. Barth, A., Felt, A.P., Saxena, P., Boodman, A.: Protecting Browsers from Extension Vulnerabilities. In: Proceedings of the 17th Network and Distributed System Security Symposium (2010)
2. Carlini, N., Felt, A.P., Wagner, D.: An evaluation of the google chrome extension security architecture. In: Proceedings of the 21st USENIX Conference on Security (2012)
3. Liu, L., Zhang, X., Yan, G., Chen, S.: Chrome Extensions: Threat Analysis and Countermeasures. In: Proceedings of the 19th Annual Network & Distributed System Security Symposium (2012)
4. Bandhakavi, S., King, S.T., Madhusudan, P., Winslett, M.: VEX: Vetting Browser Extensions For Security Vulnerabilities. In: Proceedings of the 19th USENIX Security Symposium (2010)
5. Chrome Extension Developer Guide, `http://developer.chrome.com/extensions`