IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

An Extensive Study for Development of Browser Extensions in Secure Browser Environment

[1]Yash Agrawal,[2]B.K. Srinivas

[1]Student,[2]Assistant Professor
[1]Department of Information Science and Engineering,
[1]RV College of Engineering, Bengaluru,India

Abstract: Browser extensions are small piece of software that are used to extend the capabilities of browsers. Today all major browsers support extensions to extend their functionalities. However, extensions are also prone to several attacks due to their close relationship with the browser environment. Extensions have been abused to gather private information, password theft etc. Thus, to protect the system and user data from malicious attacks, several practices have been introduced which must be followed by extensions developer to develop a safe extension. This paper presents several of such practices adopted by major browsers to counter various attacks which are done either by targeting extensions or by using extensions.

***Index Terms* - Browser Extension, Add-ons, Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, Content Security Policy.**

I. Introduction

Today almost every Browser provides support for extensions to extend their functionalities. These extensions can be easily downloaded and installed on the browser. Major browsers such as Google Chrome and Mozilla Firefox provides a central repository to download these extensions (Chrome Web Store [8] and Firefox Add-ons [11] respectively). In spite of the fact that extensions are very much useful for the browsers, they are also prone to several threats and misuse. Since extensions are very much connected to the browser working environment like in IE and older version of Firefox the extensions runs in the same process space as the browser itself [15], they can be easily abused to gather private information such as browsing history, cookies and even user passwords [4]. Moreover, extensions can also be used to attack the underlying operating system as they enjoy the same privileges as the browser. Due to all these the amount of research and studies based on security aspect of extensions have increased in recent years. [2, 5, 10, 12, 13, 14, 15]. This Paper aims to provide an extensive study on different threats related to extensions that could be harmful for user's privacy and for the system and the various countermeasures that have been adopted by browsers. The Paper presents the study for some major browsers namely Google Chrome, Mozilla Firefox, Apple Safari and Microsoft Edge in subsequent sections.

II. Google Chrome Extensions

According to [23] Chrome Browser is the most widely used browser across the World with total share of about 62.48% as of April 2020. It also hosts most number of extensions in its Chrome Web Store with more than 188k extensions [9]. Due to its large popularity, providing security is one aspect that can't be neglected. Thus to provide a secure experience to its users, chrome browser has built-in protection mechanism to deal with any type of extensions. A typical extension in Chrome consists of multiple components namely a content script which interacts with the DOM of webpage, an extension core which consists of background scripts and an optional native binary. Every extension in Chrome runs as a separate process from the browser process and access to any system or browser resources is declined unless it's mentioned in the permission profile [3]. The content script of the extension is injected into the tab when webpage is loaded and runs in the same process space as of the webpage and thus can interact with its DOM. The content script then can interact with the extension core via Chrome Inter Process Communication Channel (IPC).

Chrome extension follows three security principles: least privileges, privilege separation and strong isolation [15]. Every extension must include a manifest.json file which includes details about the extension name, version etc. along with different permissions needed. These permissions includes access to different websites, access to browser resources such as tabs, cookies, history etc. The permission to access different websites is provided based on origins as Chrome also follows the basic security mechanism of Same Origin Policy (SOP) [20].

Chrome prevents attacks by following and requesting users to follow best practices. This includes providing all extensions at one single space in the Chrome Web Store so that users are not confused and can download verified safe extensions. Thus Chrome prevents any third party extension from installing in the browser [22]. The user can only download and install extensions which are hosted on Chrome Web Store or as mentioned in [6]. This prevents attackers to create and host malicious extensions on the Store as every extension in the Web Store undergoes a review process which is done by Google before making it available for the intended users.

Extensions make external calls to different websites and APIs to fetch or send data. [7] specifically mentions never to use HTTP for any of these purpose and always to use HTTPS to prevent man-in-the-middle attack. Google also advices all the developers to enable twofactor authentication to keep their account secure as any compromise to the these accounts can proved to be very harmful for millions of users. Also Google charges a one-time fee of $5 to authenticate the account as a developer account. Only after paying and registering the account as developer, one is able to publish any extension.

As previously mentioned, only after registering the request in permission section of manifest.json file, extension can able to contact and use the resources. [7] mentions that this permission should be kept to minimum as more privileges means more area an attacker can exploit. Also it advices to keep the externally connectable and web accessible resources to minimum to avoid any potential attacks.

If the extension is using any external scripts it needs to be included in content security policy. This prevents any cross-site scripting attack. Even if the extension does not use any external scripts, it is advised to declare content security policy by declaring "default-src 'self'".

Most types of attacks are through content scripts as it is the only part of extension which interacts with webpages directly. Any malicious sites can hamper DOM which content scripts depends upon and even use it to gain private information via side channel attacks (e.g. Spectre [21]). Therefore, it is always advised to perform security related operations on the background scripts and never pass any secure information to the content scripts. It is also preferred to validate and sanitize data which comes from content scripts to background scripts as it might be crafted by attackers.

III. Mozilla Firefox Extensions

Firefox extensions or as they say Add-ons can be used to modify or enhance the browser capability. The technology used to develop extensions for Firefox is largely related to WebExtensions API which is supported by chromium-based browsers such as Chrome, Microsoft Edge, Opera. It is therefore very easy to export or import chromium-based extensions to or from Firefox by just making few changes. It also supports use of web-ext [19] which is a command line tool used to speedup extension development process.

As mentioned in [18] Firefox has completely removed the support for previous used tool XPCOM/XUL for developing extensions. As from Firefox 57, only extension developed with WebExtension API would be supported. Similar to Chrome Web Store, Firefox also provides a central repository to download and install extensions. This not only provides convenience to the users but also plays a key role in security by keeping out corrupt and malicious extensions by carrying out a security audit for every release of any extension.

Data inside the extensions could come from different sources but to make extension more secure, Firefox allows to restrict its control over extension by allowing developer to apply some properties to the document holder which is known as the docshell. Also to run the code in the extension with some restricted privileges one can use evalInSandbox().

It is also preferred to sandbox any HTTP connection which the extension is using to interact with any web services so that any normal browsing of that site by the user in the browser is not affected. Also, it is advised to always use HTTPS connection for making API calls and to use JSON as a data format to share any information among each other.

It is also advised to use Login Manager and Firefox Password Manager to handle logins and password in extensions. These are the same which are used to manage logins in the webpages. Extensions should use a chrome URL for site identifier, to avoid any clash with sites login.

It is also mentioned in [18] that any external scripts associated with remote sources are not acceptable as they are very much prone to man-in-the-middle attacks. The only secure way is to run the scripts in a sandbox. It is always advised to obtain these scripts from a trusted source and to always namespace it so that other add-ons could not use it.

IV. Apple Safari Extensions

A Safari app extension can enhance the capability of Safari by interacting with the webpage and any native app. It is build using JavaScript, CSS and a native code written in Objective-C or Swift using Xcode as an integrated development environment. The earlier supported legacy Safari extensions (.safraiextz files) are now converted to Safari App extension format. All these app extensions are available at one place on the Mac App Store or developer should notarize them to make them available outside of Mac App Store.

In iOS and macOS there are several types of extensions. These extensions are enabled by a system area known as extension point. Each of these extension points defines policies and APIs that would be used to develop extension for that particular area.

Before publishing any extension to the App Store, one needs to get a development certificate for the app extension. This is done by signing in for Apple Development Program or configuring in as a developer in Xcode. Without the development certificate, app extension won't show up in App Store. This is done to provide only secured and verified extensions on the App Store.

For every webpages and domains for which the app extension needs access, developer should specify them by adding the SFSafariWebsiteAccess dictionary to the NSExtension element. It is also advised to use the level subkey to restrict the extension's website access [1]. All these is done to prevent extension accessing any malicious websites or webpages.

To modify or override any behavior or style of any webpage, extension can inject a style sheet or script to it. This injected script can access the DOM of page and have same privileges as of any other webpage scripts. These injected scripts are just like content scripts of Google Chrome Extension as mentioned before. One can thus interact with the webpage by using these injected scripts. Even though both the app extension and injected script run on separate sandboxed environment, a common message passing format has been developed which provides an interface for sending and receiving messages. Upon receiving any message from the webpage, the extension must check the sender info to avoid any intruders attack.

V. Microsoft Edge Extensions

Extensions in Microsoft Edge are used to add or modify features of the browser. Since Edge is based on open source Chromium engine just like Chrome, it uses a similar extension API model like chrome. All the Edge extensions are hosted on Microsoft Store to allow users to download and install these extensions from a single space. Though there is an option for Windows users who could also use Windows registry to download extensions if provided by developer.

Similar to chrome extension, Edge extensions also consists of a manifest.json file which defines the various properties and permissions needed for extension. Most of the security issues here are similar to Chrome extensions. Like for safeguarding the extension against cross-site scripting issues, concept of Content Security Policy (CSP) is incorporated [16]. If manifest-version in manifest.json file is declared as 2 then a default content security policy of script-src 'self'; object-stc 'self' is declared. This policy provides security to extension and application by limiting its access in three ways: eval and related functions are disabled, inline javascript is disabled and only local scripts and object resources can be loaded.

Also use of HTTP to access remote script is prohibited and one should always make use of HTTPS. This is because man-in-the-middle attacks are always undetectable over HTTP. Also [16] clearly specifies that the restriction on HTTP are only on resources which can directly run. Developers are still free to make XMLHTTPRequest connection to any origin they like.

Content scripts are not affected by CSP and therefore it is recommended to use content scripts for most of the extension behavior and not the DOM injected scripts as changes on CSP can cause unwanted extension behavior if DOM injected scripts are used.

There is also a lot of security preference given regarding the user data that any extension might collect. It is clearly mentioned in the docs [16] that the extension should first take consent from the user and must clearly specify how the data is been taken, to whom it is disclosed etc. It also mentions that extension should provide an option for user to opt out and withdraw the consent whenever they want. Also, every user data should be stored and transmitted securely by using modern cryptography technology.

VI. Conclusion

The following table gives the overall comparison between different browsers and their extensions.

Table -1: Comparison between different properties of browsers and their extensions

Properties	Chrome	Firefox	Safari	Edge
Chromium Based Browser	✓	✓	✗	✓
Extensions runs as a separate process from Browser	✓	✓	✓	✓
Single space to host extensions	✓	✓	✓	✓
Mandatory manifest.json file in extensions	✓	✓	✗	✓
HTTPS over HTTP	✓	✓	✓	✓
Uses Content Security Policy	✓	✓	✓	✓
Sandbox Environment	✓	✓	✓	✓
Using Xcode	✗	✗	✓	✗
Most Secure Web Browser[17]	2nd	1st	4th	3rd

Extensions are useful for the browsers but are also prone to several attacks if not handled properly. To safeguard these extensions and protect the user privacy and system, different browsers presents the guidelines to the developers of extensions so that from the early stage of development, they could keep security as their top priority. We see in this paper how most browsers support extensions and run them on a separate environment or within a Sandbox to keep both extension and browser secure. Most of the interaction between the browser and extension happen over a communication channel and special attention should be given while making interaction since most of the attacks could occur via this mean. We also see that some attacks like man-in-the-middle attacks, cross-site scripting attacks etc. could occur in any extension and thus different browsers presents different protocols that must be followed by developers to prevent such attacks. Every browser clearly specifies that extensions must try to avoid the use of any external scripts and if required they must specify it and take permissions beforehand. Also, every extension in major browsers first need to pass a review process which is done by the browser company to check for any security issues and thus making the browser working environment safe and secure.

Acknowledgment

A lot of analysis and reading was done prior to the curation of the content of the given paper. This would definitely not be possible without the guidance of my mentor, Prof. B.K. Srinivas, who constantly guided me and gave me a direction in the due course of the given paper. I would also like to thank our Head of Department, Dr. B.M. Sagar, who gave me this opportunity to work on this paper.

REFERENCES

[1]Apple Safari Developer Docs: https://developer.apple.com/documentation/safariservices/safari_app_extensions/safari_app_extension_info_property_list_keys/adjusting_website_access_permissions

[2] Bandhakavi S., Tiku N., Pittman W., King S. T., Madhusudan P., and Winslett M., "Vetting browser ex- tensions for security vulnerabilities with vex. Communications of the ACM 54", 9 (2011), 91–99.

[3] Barth A., A. P. Felt, P. Saxena, and A. Boodman. "Protecing browsers from extension vulnerabilities", Proc. of NDSS, 2010.

[4] Buyukkayhan A. S., Onarlioglu K., Robertson W., and Kirda, E., "CrossFire: An Analysis of Firefox Extension- Reuse Vulnerabilities", Proceedings of the Network and Dis- tributed System Security (NDSS), 2016.

[5] Carlini N., Felt A. P., and Wagner D., "An evaluation of the google chrome extension security architecture", Proceedings of the USENIX Security Symposium (SEC), 2012.

[6] Chrome Developer Docs: https://developer.chrome.com/apps/external_extensions

[7] Chrome Developer Docs: https://developer.chrome.com/extensions/security

[8] Chrome Webstore: https://chrome.google.com/webstore/category/extensions

[9] Chrome Web Store Number of Extensions: https://www.ghacks.net/2019/08/04/chrome-web-store-has-188k-extensions-with-at-least-1-2-billion-installs/

[10] Dhawan M., and Ganapathy V., "Analyzing information flow in JavaScript-based browser extensions", Proceedings of the Annual Computer Security Applications Conference (ACSAC), 2009.

[11] Firefox Add-ons: https://addons.mozilla.org/en-US/firefox/

[12] Guha A., Fredrikson M., Livshits B., and Swamy N., "Verified security for browser extensions", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, 2011.

[13] Iskander Sanchez-Rola, Igor Santos, Davide Balzarotti, "Extension Breakdown: Security Analysis of Browsers Extension Resources Control Policies", 26th USENIX Security Symposium August 16–18, 2017.

[14] Kapravelos A., Grier C., Chachra N., Kruegel C., Vigna G., and Paxson V., "Hulk Eliciting malicious behav- ior in browser extensions", Proceedings of the USENIX Security Symposium (SEC), 2014.

[15] Liu L., Zhang X., Yan G., and Chen S., "Chrome Ex- tensions: Threat Analysis and Countermeasures", Proceedings of the Network and Distributed Systems Security Symposium (NDSS), 2012.

[16] Microsoft Edge Developer Docs: https://docs.microsoft.com/en-gb/microsoft-edge/extensions-chromium/store-policies/csp

[17] Most Secure Web Browsers 2020: https://bestvpn.org/best-secure- web-browsers/

[18] Mozilla Firefox Developer Docs: https://developer.mozilla.org/en-US/docs/Archive/Add-ons/Security_best_practices_in_extensions

[19] Mozilla Webext Docs: https://extensionworkshop.com/documentation/develop/getting-started-with-web-ext/

[20] Same origin policy. https://en.wikipedia.org/wiki/Same-origin_policy

[21] Spectre: https://spectreattack.com/

[22] TechCrunch: https://techcrunch.com/2018/06/12/google-puts-an-end-to-chrome-extension-installs-from-third-party-sites/

[23] Usage Share of Web Resources https://en.wikipedia.org/wiki/Usage_share_of_web_browsers