STRIDE Based Analysis of the Chrome Browser Extensions API

P.K. Akshay Dev and K.P. Jevitha

Abstract Chrome browser extensions have become very popular among the users of Google Chrome and hence they are used by attackers to perform malicious activities which lead to loss of user's sensitive data or damage to the user's system. In this study, we have done an analysis on the security of the Chrome extension development APIs. We have used the STRIDE approach to identify the possible threats of the Chrome specific APIs which are used for extension development. The analysis results show that 23 out of the 63 Chrome specific APIs are having various threats as per the STRIDE approach. Information disclosure is the threat faced by many APIs followed by tampering. This threat analysis result can be used as reference for a tool which can detect whether the extension is malicious or not by deeply analysing the ways in which the APIs having threats are used in the extension code.

Keywords Chrome extensions · STRIDE analysis · Chrome specific APIs · Malicious extensions

1 Introduction

Google Chrome is the most popular web browser available today and many extensions are being developed to enhance the features and functionality of the browser. Extensions are basically small programs developed by browser vendors or third party developers to improve the functionality of browsers. Due to the wide

P.K. Akshay Dev (✉)
TIFAC CORE in Cyber Security, Amrita School of Engineering,
Amrita Vishwa Vidyapeetham, Amrita University, Coimbatore, India
e-mail: akshaydev313@gmail.com

K.P. Jevitha
Department of Computer Science and Engineering,
Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham,
Amrita University, Coimbatore, India
e-mail: kp_jevitha@cb.amrita.edu

S.C. Satapathy et al. (eds.), *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*, Advances in Intelligent Systems and Computing 516, DOI 10.1007/978-981-10-3156-4_17

acceptance of the browser extensions by the users, it has also become the major target of the attackers [1]. Extensions created by third-party developers may not have enough security considerations and become vulnerable to attacks. Extensions once compromised can result in different malicious activities which can affect the user's sensitive data and system. Also, malicious extensions are being developed to attack the user's system and data. Chrome extensions can have the privilege to perform many sensitive actions in the browser using the different permissions and chrome specific APIs available to them. This makes them highly dangerous when used for malicious purposes.

STRIDE is a threat analysis approach developed by Microsoft for analysing the possible threats of web applications or components [2]. STRIDE approach can be used to map the possible threats into six categories namely Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege. Spoofing is impersonating something or someone else. Tampering is malicious modification of code, data or settings. Repudiation is denial of truth or validity of something. Information disclosure is exposing information to someone not authorized to see it. Denial of service is denying or degrading service to users. Elevation of privilege is gaining capabilities without proper authorization [3].

The rest of the paper is organized as follows. Section 2 briefs the related work done in this area of research. Section 3 describes the security model of the chrome extensions and Sect. 4 describes the security analysis of the Chrome specific APIs based on STRIDE. Section 5 explains the statistical results of the analysis and Sect. 6 concludes the work.

2 Related Work

Different research work has been done in the area of browser extension security over the past few years. Due to the wide acceptance of extensions by the users of popular web browsers like Google Chrome and Mozilla Firefox, the security of those extensions has become a major concern and also gained attention of the security researchers. An in-depth analysis of the Chrome's extension security model was done [1] and they conclude that its vulnerabilities are rooted from the violation of the principles of least privilege and privilege separation. Using a series of bot-based attacks, the authors demonstrate that malicious chrome extensions pose serious threat to Chrome browser. A thorough evaluation of how well the Chrome extension's security mechanism defends against the vulnerabilities has been done in [4] and they found out that developers are not always following the security mechanisms for developing the extensions which leads to vulnerabilities. A framework to analyse the security of Chrome extensions has been proposed in [5]. The framework proposed uses the permission feature of Chrome extensions and flow of data through the extension's JavaScript code to detect vulnerabilities and malicious behaviour of extensions. A systematic review of the different research work done in the area of browser extension security has been done and the different

security measures that can be taken for browser extension vulnerabilities have been tabulated in [6].

Various solutions have been proposed to protect the users from different vulnerabilities in extensions and from attacks of malicious extensions. A framework called LvDetector was proposed, which combines static and dynamic program analysis techniques for automatic detection of information leakage vulnerabilities in legitimate browser extensions [7]. A dynamic analysis system called Hulk detects malicious extensions by monitoring the actions and by creating a dynamic environment that adapts to the extension's needs to trigger the intended malicious behaviour [8]. Researchers from Google analysed the use of Chrome web store by criminals to spread malicious extensions and came up with a system called WebEval that broadly identifies malicious extensions [9]. Apolicy enforcer for Firefox browser called Sentinel, gives the control to the users over the actions of existing JavaScript extensions and prevents common attacks on the extensions [10]. A system called Expector automatically inspects and identifies extensions that inject ads and classifies those ads as malicious or benign based on their landing pages [11].

3 Security Model of Chrome Extensions

The security model of Chrome extensions is based mainly on three principles. They are least privilege, privilege separation and strong isolation [1]. Extensions will get only those privileges requested in their manifest. Least privilege does not protect users from malicious extensions. Instead, least privileges helps if a benign extension has a vulnerability. If somehow an extension is compromised, the attacker's privileges will be limited. For privilege separation, instead of dumping all the extension code into a single file, developers divide it into two parts, a background page and a content script [12]. The background page is given the major share of the extension's privileges, but it is isolated from direct contact with webpages. On the other hand, content scripts has direct contact with webpages, but has fewer privileges. For strong isolation, extension's content script is run in a separate environment from that of the websites so that websites can't access content script's variables or functions [4].

Google Chrome is concerned with the security of users of Chrome extensions. As part of Chrome's continuing security efforts, from 2013, it is no more possible to silently install extensions into Chrome on windows using the windows registry mechanism for extension deployment [13]. Also, from 2014, Chrome supports only installation of extensions that are hosted in the Chrome web store, in the windows stable and beta channels [13]. But they continue to support local extension installs in developer mode as well as installs via enterprise policy. In Chrome web store, all extensions go through an automated review process, but in most cases, will be published without any further manual review. Chrome web store will block those

extensions that violate the Chrome's malware policy. Despite all these security checks, malicious extensions are being hosted in the Chrome web store.

4 Security Analysis of Chrome Specific APIs

This section explains the security analysis of the Chrome specific APIs based on STRIDE approach. Google Chrome provides specific APIs for the extensions to use, in addition to giving access to all the APIs that web pages and apps can use. They are often called chrome.* APIs [12]. There are a total of 63 chrome.* APIs. These APIs allow tight integration with the browser. These chrome.* APIs can be used to perform specific functions with extensions. These APIs have different method, events and properties which can be used to do specific tasks. For example, bookmarks API can be used to create, organize and otherwise manipulate bookmarks [14]. These chrome specific APIs are so powerful in terms of the ways they can be used to access or modify user data or perform different actions in the browser. There are different ways in which these APIs can be used in extensions for performing malicious actions. So, we have used the STRIDE threat analysis approach to analyse the possible threats of using chrome specific APIs.

4.1 STRIDE Threat Analysis

STRIDE approach, developed by Microsoft is used to identify the possible threats of each Chrome specific API. In the case of extensions, spoofing can occur in the form of phishing, clickjacking etc. For example, update method of bookmarks API can be used to update an existing bookmark with a new URL which can be used for phishing, a form of spoofing. Tampering can occur in the form of XSS, clickjacking etc. For example, executeScript method of tabs API can be used to perform cross site scripting attack which is a form of tampering. Repudiation can occur when the extension hides some activity it is doing. For example, erase method of downloads API can be used to erase the download information from the browser which can hide the fact that the extension has downloaded something. Any activity of extensions that accesses user's private data can come under information disclosure. For example, getVisits method of history API can be used to retrieve the history information from the browser. Any activity of extensions that deny users normal browsing experience comes under Denial of service. For example, images property of contentSettings API can be used to block images in webpages. Elevation of privilege involves doing things without the proper consent of users or doing things the user thinks the extension can never do. For example, create method of bookmarks API can be used to create new bookmarks and remove method of bookmarks API can be used to remove existing bookmarks from the web browser.

4.2 Threat Analysis of Chrome Specific APIs

We have used the STRIDE approach to analyse the possible threats of using chrome specific APIs. Chrome specific APIs have different methods, events and properties which can be used for different purposes. We have analysed the specific use of different methods, events and properties of each API and identified the possible threats based on the STRIDE approach. We have analysed the functionality of particular methods, events and properties of each API to identify the possible threats as per STRIDE approach and did not consider the combination of methods and events or two or more methods to identify the threats. Table 1 lists the STRIDE analysis results for the Chrome specific APIs that were identified as having threats.

Out of the 40 APIs that are not in the table, 27 were found to have no threats as per STRIDE approach, 11 APIs works only on the Chrome OS which is not so popular yet. We have concentrated on the users of Chrome browser. The APIs that work only on Chrome OS include accessibilityFeatures, documentScan etc. [14]. There are 3 APIs which are called the devTools APIs. They are devtools.inspectedWindow, devtools.network and devtools.panels. They work only in the devTools page which is called only when the devTools window is opened. The rest

Table 1 STRIDE analysis of APIs

Chrome.* APIs	Methods/events/properties	S	T	R	I	D	E
bookmarks	methods—get, getChildren, getRecent, getTree, getSubTree & search	✗	✗	✗	✓	✗	✗
	methods—move, create, remove & removeTree	✗	✓	✗	✗	✗	✓
	method—update	✓	✓	✗	✗	✗	✓
	events—onCreated, onRemoved, onChanged, onMoved, onChildrenReordered, onImportBegan & onImportEnded	✗	✗	✗	✗	✗	✗
browsingData	method—settings	✗	✗	✗	✓	✗	✗
	methods—remove, removeAppcache, removeCache, removeCookies, removeDownloads, removeFileSystems, removeFormData, removeHistory, removeIndexedDB, removeLocalStorage, removePluginData, removePasswords & removeWebSQL	✗	✓	✗	✗	✗	✓
contentSettings	properties—cookies, images, javascript, location, plugins, popups, notifications, fullscreen, mouselock, unsandboxedPlugins&automaticDownloads	✗	✓	✗	✗	✓	✓
cookies	methods—get, getAll & getAllCookieStores	✗	✗	✗	✓	✗	✗
	methods—set, remove	✗	✓	✗	✗	✗	✓
	event—onChanged	✗	✗	✗	✗	✗	✗

(continued)

Table 1 (continued)

Chrome.* APIs	Methods/events/properties	S	T	R	I	D	E
debugger	method—attach	✗	✓	✗	✗	✓	✓
	method—sendCommand	✗	✓	✗	✗	✗	✓
	methods—detach & getTargets	✗	✗	✗	✗	✗	✗
	events—onEvent & onDetach	✗	✗	✗	✗	✗	✗
desktopCapture	method—chooseDesktopMedia	✗	✗	✗	✓	✗	✗
	method—cancelChooseDesktopMedia	✗	✗	✓	✗	✗	✗
downloads	methods—download, pause, resume, cancel, open & removeFile	✗	✓	✗	✗	✗	✓
	method—search	✗	✗	✗	✓	✗	✗
	methods—getFileIcon, show & showDefaultFolder	✗	✗	✗	✗	✗	✗
	method—erase	✗	✗	✓	✗	✗	✓
	methods—acceptDanger, drag & setShelfEnabled	✗	✗	✗	✗	✗	✗
	events—onCreated, onErased, onChanged & onDeterminingFilename	✗	✗	✗	✗	✗	✗
gcm	methods—register & unregister	✗	✗	✗	✗	✗	✗
	method—send	✗	✗	✗	✓	✗	✗
	events—onMessage, onMessagesDeleted & onSendError	✗	✗	✗	✗	✗	✗
history	methods—search & getVisits	✗	✗	✗	✓	✗	✓
	methods—addUrl, deleteUrl, deleteRange & deleteAll	✗	✓	✗	✗	✗	✓
	events—onVisited & onVisitRemoved	✗	✗	✗	✗	✗	✗
management	methods—getAll, get, getPermissionWarningsById & getPermissionWarningsByManifest	✗	✗	✗	✓	✗	✗
	method—setEnabled	✗	✓	✗	✗	✓	✓
	methods—uninstall & launchApp	✗	✓	✗	✗	✗	✓
	methods—getSelf, uninstallSelf, createAppShortcut, setLaunchType & generateAppForLink	✗	✗	✗	✗	✗	✗
	events—onInstalled, onUninstalled, onEnabled & onDisabled	✗	✗	✗	✗	✗	✗
power	method—requestKeepAwake	✗	✓	✗	✗	✗	✗
	method—releaseKeepAwake	✗	✗	✗	✗	✗	✗
privacy	property—network	✗	✗	✗	✗	✗	✗
	properties—services & websites	✗	✓	✗	✗	✗	✓
proxy	property—settings	✗	✓	✗	✗	✓	✓
	event—onProxyError	✗	✗	✗	✗	✗	✗
sessions	methods—getRecentlyClosed & getDevices	✗	✗	✗	✓	✗	✗
	method—restore	✗	✗	✗	✗	✗	✗
	event—onChanged	✗	✗	✗	✗	✗	✗

(continued)

Table 1 (continued)

Chrome.* APIs	Methods/events/properties	S	T	R	I	D	E
storage	property—sync	✗	✗	✗	✓	✗	✗
	properties—local & managed	✗	✗	✗	✗	✗	✗
	event—onChanged	✗	✗	✗	✗	✗	✗
system.cpu	method—getInfo	✗	✗	✗	✓	✗	✗
system.memory	method—getInfo	✗	✗	✗	✓	✗	✗
system.storage	method—getInfo	✗	✗	✗	✓	✗	✗
	methods—ejectDevice & getAvailableCapacity	✗	✗	✗	✗	✗	✗
	events—onAttached & onDetached	✗	✗	✗	✗	✗	✗
tabCapture	method—capture	✗	✗	✗	✓	✗	✗
	method—getCapturedTabs	✗	✗	✗	✗	✗	✗
	event—onStatusChanged	✗	✗	✗	✗	✗	✗
tabs	methods—connect, sendRequest, sendMessage, getSelected, getAllInWindow, create, duplicate, query, highlight, move, reload & detectLanguage	✗	✗	✗	✗	✗	✗
	methods—update & remove	✗	✗	✗	✗	✓	✓
	methods—get, getCurrent & captureVisibleTab	✗	✗	✗	✓	✗	✗
	method—executeScript	✗	✓	✗	✗	✗	✗
	method—insertCSS	✓	✓	✗	✗	✗	✗
	methods—setZoom, getZoom, setZoomSettings & getZoomSettings	✗	✗	✗	✗	✗	✗
	events—onCreated & onActivated	✗	✗	✗	✗	✓	✗
	events—onUpdated, onMoved, onSelectionChanged, onActiveChanged, onHighlightChanged, onHighlighted, onDetached, onAttached, onRemoved, onReplaced & onZoomChange	✗	✗	✗	✗	✗	✗
topSites	method—get	✗	✗	✗	✓	✗	✗
webRequest	method—handlerBehaviorChanged	✗	✗	✗	✗	✗	✗
	event—onBeforeRequest	✓	✓	✗	✗	✓	✓
	events—onBeforeSendHeaders & onHeadersReceived	✗	✓	✗	✗	✓	✓
	events—onSendHeaders, onAuthRequired, onResponseStarted, onBeforeRedirect & onCompleted	✗	✓	✗	✗	✗	✓
	event—onErrorOccurred	✗	✗	✗	✗	✗	✗
windows	method—get, getLastFocused & getAll	✗	✗	✗	✓	✗	✗
	method—getCurrent	✗	✗	✗	✓	✓	✗
	method—create	✗	✗	✗	✗	✗	✗
	methods—update & remove	✗	✓	✗	✗	✗	✓
	event—onCreated	✗	✗	✗	✗	✓	✗
	events—onRemoved & onFocusChanged	✗	✗	✗	✗	✗	✗

3 APIs are events, extensionTypes and types. events API is basically a namespace containing common types used by APIs dispatching events to notify something. extensionTypes API contain type declarations for Chrome extensions and types API contains type declarations for Chrome.

5 Results

This section describes the statistical results obtained from our threat analysis of the Chrome specific APIs based on STRIDE. Chrome provides JavaScript APIs specifically for developing extensions. These APIs are very powerful in terms of the activities they can perform. There are a total of 63 Chrome specific APIs that we have analysed using the STRIDE approach. We have identified that 23 APIs are having various threats based on the STRIDE approach. Table 1 lists the STRIDE analysis results for the Chrome specific APIs that were identified as having threats. In summary, APIs that have threats as per STRIDE approach are: bookmarks, browsingData, contentSettings, cookies, debugger, desktopCapture, downloads, gcm, history, management, power, privacy, proxy, sessions, storage, system.cpu, system.memory, system.storage, tabCapture, tabs, topSites, webRequest and windows. Table 2 lists the count of the APIs, methods, events and properties having various threats as per STRIDE approach. It is the total count of APIs, methods, events and properties affected by each threat as per STRIDE approach.

Table 2 Summary of API threat analysis

Threats (STRIDE)	APIs	Methods	Events	Properties
Spoofing	3	2	1	0
Tampering	14	40	8	14
Repudiation	2	2	0	0
Information disclosure	17	33	0	1
Denial of service	7	5	6	12
Elevation of privilege	13	42	8	14

6 Conclusion

Chrome extensions are getting wide acceptance among the users of Google Chrome and hence are being widely used for malicious purposes by attackers. The research work has revealed the potential dangers the users of Chrome extensions are facing when they use the extensions which can use all the Chrome specific APIs for its own purposes. Table 1 shows the various threats different Chrome APIs are having and the attackers can use any of these APIs in the extensions to perform malicious activities. The APIs having threats are so powerful that when used in a malicious way can lead to loss of user's sensitive data or damage to user's system. This threat analysis results can help in the identification of the malicious behaviour of chrome extensions. This threat analysis has only considered the functionality of each Chrome specific API for identifying the threats. This work can be extended by analysing the possible threats considering the different combinations of Chrome specific APIs, as it can serve different purposes and can be used for more malicious activities in the extension code.

References

1. L. Liu, X. Zhang, G. Yan, S. Chen, Chrome extensions: threat analysis and countermeasures, in *NDSS* (2012)
2. Microsoft STRIDE threat model, https://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx
3. S.F. Burns, Threat modeling: a process to ensure application security, in *GIAC Security Essentials Certification (GSEC) Practical Assignment* (2005)
4. N. Carlini, A. Porter Felt, D. Wagner, An evaluation of the google chrome extension security architecture, in *Presented as Part of the 21st USENIX Security Symposium (USENIX Security 12)*, pp. 97–111 (2012)
5. V. Aravind, M. Sethumadhavan, A framework for analysing the security of chrome extensions. Adv. Comput. Netw. Inf. **2**, 267–272 (2014)
6. J. Arunagiri, S. Rakhi, K.P. Jevitha, A systematic review of security measures for web browser extension vulnerabilities, in *Proceedings of the International Conference on Soft Computing Systems* (Springer India, 2016)
7. R. Zhao, C. Yue, Q. Yi, Automatic detection of information leakage vulnerabilities in browser extensions, in *Proceedings of the 24th International Conference on World Wide Web* (International World Wide Web Conferences Steering Committee, 2015)
8. A. Kapravelos, et al., Hulk: eliciting malicious behavior in browser extensions, in *23rd USENIX Security Symposium (USENIX Security 14)* (2014)
9. N. Jagpal, et al., Trends and lessons from three years fighting malicious extensions, in *24th USENIX Security Symposium (USENIX Security 15)* (2015)
10. K. Onarlioglu, et al., Sentinel: securing legacy firefox extensions. Comput. Secur. **49**, 147–161 (2015)

11. X. Xing, et al., Understanding malvertising through ad-injecting browser extensions, in *Proceedings of the 24th International Conference on World Wide Web* (International World Wide Web Conferences Steering Committee, 2015)
12. Chrome extension developer guide, https://developer.chrome.com/extensions/overview
13. Chromium blog, http://blog.chromium.org/
14. Chrome extension specific API index, https://developer.chrome.com/extensions/api_index