RESTRAINING ADD-ON'S BEHAVIOR IN PRIVATE BROWSING

Bapat Ameya D.
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Karnataka, India
ameyabap@gmail.com

Alwyn Pais
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Karnataka, India
alwyn.pais@gmail.com

ABSTRACT
In this paper we address the privacy issues of add-on mechanism supported by browser in private mode. The add-ons enjoy unrestrained access to user sensitive information at all times. This freedom can be misused to create add-ons with malicious intent of violating privacy of the browser. We have designed and implemented an add-on which performs this task in private mode of the browser. This is a clear violation of the goals of private browsing. Mozilla lacks privacy ensuring mechanism against add-ons at browser level. So we have modified the source code of Mozilla Firefox to prevent such behavior of an add-on. It involves runtime monitoring of add-on's behavior in private mode and notify/block suspicious ones. We have been able to prevent such add-on's activity using our mechanism.

General Terms
Security.

Keywords
Privacy, Private Browsing, Add-ons, Mozilla Firefox.

1. INTRODUCTION
Web browser has been integral part of internet. In modern world it also plays important role in human's professional life. People use browsers for seeking information, social networking, instant messaging, blogging etc. Browsers are designed in such a way that they can record and retrieve information back about user's activities. This information is stored on local machine that can be accessed by anyone who has access to local machine. This information includes visited URLs, cookies, user preferences, searched items etc. Since past 4-5 years most of the browser companies have been concerned regarding user's privacy while surfing on internet.

Most of the browsers included private mode as an extra functionality such as InPrivate by Internet Explorer, Incognito by Google Chrome, Private Browsing by Mozilla Firefox. The primary goal of private browsing is to keep no trace of user's activities on user machine. The amount of privacy actually guaranteed by private mode has always been topic of concerned. Navigation privacy of user majorly depends on browser parameters such as cookies, history, temporary files etc.

Browser add-ons play prominent role in Modern Browsers. Current behavior of browsers clean history and cookies at the end of private session to preserve privacy but at the same time information can stay behind via add-ons which are not addressed by browsers. Scope of add-ons can make private session susceptible to threats. Browsers do not leak cookie or history but its add-ons can have tracking system which can leak it if they want.

The Mozilla Firefox grants equal access to add-ons in private mode as they get in non-private mode. As per the Firefox policy "the add-on must properly respect private browsing mode by not recording sensitive data while private browsing mode is active" [4]. Mozilla is against blocking add-ons as they are open source and user defined. As per Firefox report, "it has not reviewed all of the material contained in such add-ons; it cannot be held accountable for their content or any harm they might cause" [3]. It means when we are in private mode these small applications can hold our data and send it to anyone. It is also possible that some add-ons may reveal this information unintentionally, in any case this is potential privacy risk. It's an open challenge to track suspicious add-ons and restrain them from acting capturer.

Organization of Paper
In section 2 we present the related work in the area of privacy of add-ons in private mode. Section 3 explains how add-ons communicate with browser. In Section 4 we discuss the design and implementation of add-on which exploits the Firefox vulnerability by tracking user's data even in private mode. Section 5 proposes method to detect suspicious add-ons and prevent them. Section 6 highlights sensitive data in Firefox and results of testing phase. We conclude our paper in section 7.

2. RELATED WORK
Several securities related sites have been highlighting the mentioned problem. Technical papers and surveys have registered potential add-ons those might expose private data. In Firefox, binary extensions such as "Cooliris" execute with the same permissions as those of the user, these extensions can read/write to any file on disk hence binary extensions are said to be unsafe for private browsing. Most of the extensions are based on javascript[JS]. They need to be checked for write operation

SecurIT'12, August 17-19, 2012, Kollam, Kerala, India

hence, manual scanning of files that are written then tells whether extension violates privacy or not [1]. The following are results found by Gaurav Aggrawal, Dan Boneh in their paper[1]:

There is an extension such as "1-Click YouTube Video Download" writes list of videos to be downloaded on file, whereas "FastestFox" writes bookmarks on a file. Write operations are most dangerous as they nullifies privacy measures taken by private browsers.

Categories of the most common violations are as follows:

1. URL queues: Extensions such as "DownThemAll" maintain a queue of URLs to download. This queue is maintained to disk even in private mode till download completes. Download might not complete in same private mode session which makes information available outside private mode.

2. URL mapping: Extensions such as "Stylish" allow different CSS styles for viewing pages from different sites. Styles to site pairs are persisted to disk even in private browsing.

3. Timestamp: The extensions store timestamp values on disk. Extension such as "personos" store when was the last time theme changed even in private browsing. This can be used to track the time when private browsing was used based on time in history and time stored by above extension.

Current browsers give an option to allow all or block all add-ons in private mode. Most of the add-ons don't follow private browsing so they might get blocked. Hence, it has been recommended that browser vendor should provide APIs which would help developers to decide which data should be stored during private browsing and which should not.

There have been attempts made to block potential add-ons by building another add-on such as ExtensionBlocker [1]. It disables all unsafe add-ons and enables them when mode is changed. An extension is considered safe for private mode if its manifest file contains a new XML tag <privateModeCompatible/>.

The add-ons operate with the user's full privileges. Adam barth analyzed and found that 88% of 25 popular Firefox extensions require less than the full set of privileges [17]. They also found that 76% of these extensions use unnecessarily powerful APIs. It is also needed to check extensions that unnecessarily use APIs dealing with sensitive information in private mode.

Previous research talk about malicious add-ons or spying add-ons and only few looked at it from private browsing aspect. There is no browser level prevention method against such violations.

3. ADD-ON AND BROWSER INTERACTION

Firefox enables users to expand browser's functionality by allowing them to create own add-ons. Firefox has modular and layered structure which creates basic foundation for users to design add-ons. There are some famous extensions written in Javascript such as NoScript, which disables JavaScript to improve security, Firebug gives various tools for web developers.

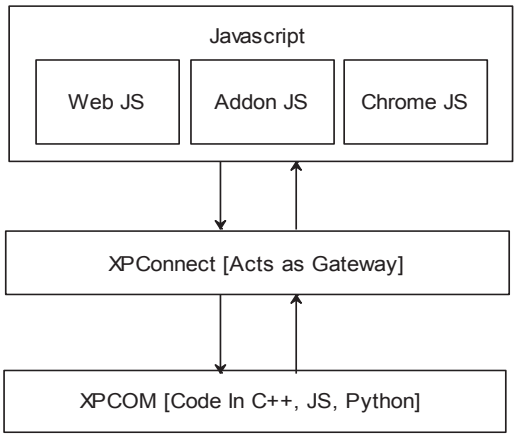

Figure 1. Interaction between JS and XPCOM

The Cross Platform Component Object Model (XPCOM) is a simple, cross platform component model. It contains Interface Description Languages[IDL] which helps programmers to plug their functionality into the framework and connect it with other components [10].

The XPCOM allows JS add-ons to access independent Firefox components written in different languages such as JS, C++, python via common interface [5]. XPCOM helps to build a module in which large task can be broken into smaller pieces which are known as components [13]. Each component is uniquely identified by 128-bit number called interface ID.

The Cross Platform Connect [XPConnect] is intermediate layer allows Javascript in Firefox to access XPCOM components. "With XPConnect, we can use XPCOM components from JavaScript code, and interact with JavaScript objects from within XPCOM components" [16]. It can be considered as bridge between Javascript and XPCOM components. Its service manager calls service manager of XPCOM. XPConnect does not have filtering mechanism as a result it grants both browser [chrome] JSs and add-on JSs full access to browser components.

4. EXPLOITING VULNERABILITY

The Firefox add-on called "Capture_Data" is designed and implemented in such a way to track browsing history, cookies and store them on secondary storage in private browsing mode.

During non-private mode Firefox maintains database of browsing history and cookies but in private mode it does not. The browsing is monitored at real time and corresponding URLs and cookies are tracked and stored.

The following figure shows operation of Firefox Add-on Capture_Data:

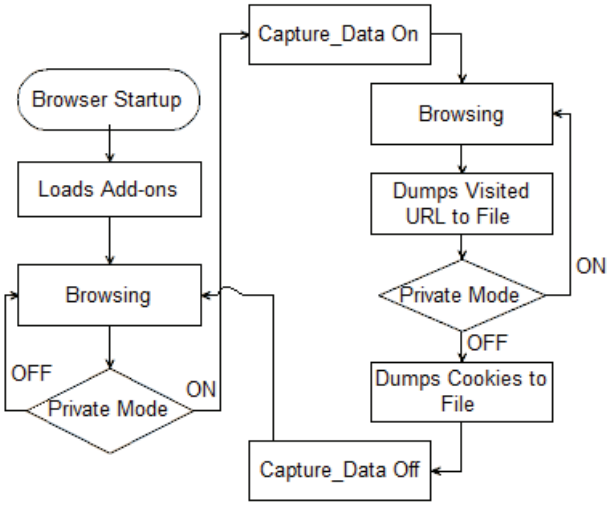

Figure 1. WorkFlow of Capture_Data.

Even though browser clears sensitive data once it is closed, we can easily track behaviour or habits of respective user during active session. It completely violates the goals of private mode as well as launches successful attack on browser.

5. DEFENSE MECHANISM

The differences between claimed and implemented goals make this domain open to research. This attack can be prevented by two ways, one of them is by blocking all add-ons in private mode which is not acceptable as it completely blocks benefits of add-ons. Second approach is to a design mechanism which will monitor behavior of add-ons based on some rules, guidelines and take actions such as block or allow which are necessary to maintain privacy.

The second approach can be designed in two ways:

1. Build an add-on to monitor working of other add-ons and gives an alarm or blocks respective one, if malicious.

2. To tweak browser code itself so that it takes extra care in private mode. The browser can be improved to scrutinize add-on behavior before granting access to sensitive data.

In general add-ons are not trusted completely and it is never preferred that one add-on is used to control other as either can't be trusted. The browser code modification is better approach because browser itself would take care of own privacy.

In order to enforce monitoring on extensions, it is necessary to identify the extension requesting sensitive XPCOM services. "Once an extension is installed, the browser does not differentiate between extension script and browser script because to characteristic of overlays" [2].

If a typical security manager detects a script that is accessing a XPCOM component then also it could not tell whether the script is owned by the browser or by an extension. Even if a script belongs to an add-on, there is no method or function which would tell the owner add-on of a script. It is required to filter script URLs that belong to add-on's code.

The following figure shows workflow of solution:

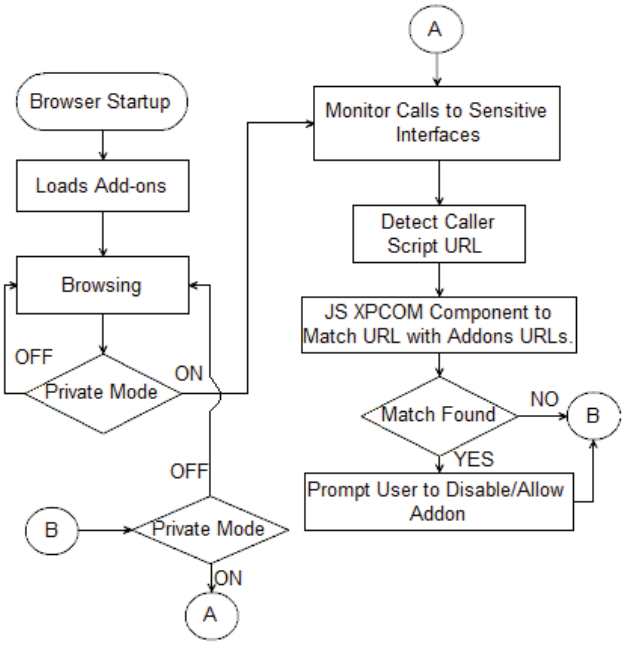

Figure 2. WorkFlow of Filtering Mechanism

The algorithm depicts major modules of filtering mechanism:

1. Start Browser
2. Loads Add-ons
3. Browsing Operation
4. IF Private Mode ON goto 5 Else goto 3
5. Monitor Calls to Sensitive Interfaces listed in Table 1.
6. Identify Caller Script URLs
7. Match Caller Script URL with Add-on's URL.
8. If Match Found YES goto 9, NO goto 4
9. Prompt User to Disable/Allow Add-on
10. Goto 4.

Filtering needs following major steps:

5.1 Identify Caller Script

The objective is to find caller script which accesses specific interfaces. The conventional method to access any service such as cookie manager via interface is as follows:

Ex: CookieManager

Var cookie=
Components.classes["@mozilla.org/cookiemanager;1"].

getService(Components.interfaces.nsICookieManager);

The add-on script which has above code can access cookies irrespective of mode of browsing. The getService method and eventually XPConnect is responsible to call nsICookieManager. The section 4 explains that JS always pass through XPConnect layer. The getService method is modified to get URLs of caller scripts [15]. JS engine maintains stack of calls via nsIStackFrame interface which is accessed in source code of getService. It helps to track script URLs that call getservice method. Those URLs

can be dumped and logged in temporary file. It is possible to identify caller script of nsICookieManager. If URL belongs to an add-on then privacy policies are applied on respective it. Above approach is generalized to remaining user sensitive interfaces discussed in Table 1.

The XPCOM JS component such as AddonMapper is created which is called from getService method. The script urls are classified in schemes such as 'chrome'[8] or 'resource'[9]. In order to match them with add-on's resources, script URLs need to be converted into scheme 'file', which represents their real path.

5.2 Identify Add-on

Next task is to find whether the script that calls sensitive interfaces belongs to Add-on or not. We need URLs of running add-ons to compare with script URL. The JS XPCOM AddonMapper component checks whether passed script URL belongs to add-ons. Mozilla's AddonManager is used to retrieve information regarding add-ons [14].

If suspicious add-on is found then user would be alerted with details and asked for permission to block or allow respective add-on.

5.3 Pre-Private Mode Disabling

The extensibility of Mozilla Firefox is based on overlay features. It means the moment overlays are merged, there is no distinction between extension scripts and browser code. The add-ons could be used to modify graphical user interface of browser. Hence, browser restart is necessary for enabling or disabling of add-on comes in effect.

The flow of disabling mechanism:

1. The user is alerted when a suspicious add-on is found. Add-on's information is prompted and also stored in sqlite ("Sqlite Storage") database such as "blacklist_add-ons.sqlite".
2. The moment user tries to enter into private mode, database is retrieved and stored add-ons are disabled. Browser is forced to restart directly into private mode [PM] such that disabling takes place.
3. Blacklisted add-ons are suspicious only in the scope of PM hence, all disabled add-ons are re-enabled as soon as user comes out of PM.

The restart operation could be said as overhead but it is mandatory for disable/enable operations.

6. IMPLEMENTATION DETAILS

Attack and defense mechanism is implemented on the latest version of browser which is Mozilla Firefox Nightly 12.0a1. The Firefox asks add-on developers to respect private browsing by not accessing or holding the user parameter.

Mozilla Firefox defines data as sensitive based on following parameters [4]:

1. URLs of visited pages.
2. Domains of visited sites.
3. Content of visited pages.
4. All data related to visited pages, including cookies and form data.
5. Data used to customize the Firefox user interface based on activities in private browsing mode.

Following Services can be considered as sensitive [6]:

Table 1. Sensitive Services with their interfaces

Service	Interfaces
Files/Streams	nsIOutputStream, nsILocalFile, nsIFile, nsIPropertises nsIFileOutputStream
History	nsISHistoryListener, nsIBrowserHistory
Cookies	nsICookieManager, nsICookieService, nsICookie2
Bookmarks	nsIRDFDataSource.
Cache Data	nsICacheService
Network	nsIHttpChannelInternal, nsIXMLHttpRequest, nsIHttpChannel
Login	nsILoginInfo, nsILoginManager
Download	nsIDownload, nsIDownloadManager
To Hold a set of name/value pairs	nsIFormHistory2
Accessing i/o streams to resource	nsITransport
To Save preferences for specific websites	nsIContentPrefService
Preferences	nsIPrefService, nsIPrefBranch
Listener on open top-level windows.	nsIWindowWatcher
keeps track of open windows	nsIWindowMediator
Executable process	nsIProcess
Authorised tokens	nsISecretDecoderRing.

Permissions for (cookies, images etc.) on site basis.	nsIPermissionManager
Image Cache	imgICache
plugin data	phInterface

Add-ons are not supposed to access interfaces listed in Table 1. Proposed restraining method is tested on 50 popular add-ons from categories such as Alert and Updates, Privacy and Security, Social Communication, Download management, Bookmarks, Photo and video [11]. Add-ons are certified by Mozilla under Mozilla Public License which says "The entire risk as to the quality and performance of the covered code is with you. Should any covered code prove defective in any respect, you (not the initial developer or any other contributor) assume the cost of any necessary servicing, repair or correction" [7]. So are add-ons really respecting private mode? Our approach makes browser capable of finding an answer.

The following list contains add-ons that found suspicious in private browsing because they access sensitive interfaces:

Table 2. Denotes Suspicious Firefox Add-ons.

Javascript Add-ons	Sensitive Interface	Threat Level
Export Cookies	nsICookieManager	High
PremiumPlay Codec-V	nsICookieService	High
List It	nsILoginManager, nsIWindowMediator	High
Ad blocker	nsICacheservice	High
Download helper	nsIWindowMediator, nsIRDFDataSource,nsILoginManager,nsIProperties	High
pdf download	nsIBrowserHistory	High
Chatzilla	nsIProperties, nsIWindowWatcher	High
All in one sidebar	nsIWindowWatcher, nsIWindowMediator	Medium
MemChaser	nsIWindowWatcher,nsIWindowMediator, nsIProperties	High
Missing e	nsIWindowWatcher, nsIProperties, nsIWindowMediator	High
Xmarks Sync	nsIPrefService	Medium
IE tab 2	nsIFile	High
Password Exporter	nsILoginManager	High
StumbleUpon	nsIPermisiionManager	Medium
Download StatusBar	nsIWindowWatcher	Medium
Session Manager	nsIWindowWatcher	Medium
Ghostery	nsIProperties	High
Tabletools2	nsIProperties	High
Modify Headers	nsIPropertises, nsIWindowMediator	High
Fast video download	nsIProperties, nsIWindowMediator	High
Fastest search	nsIProperties, nsIWindowMediator	High
Zotero	nsLocalFile, nsIWindowMediator	High
FB chat history Manager	nsIProperties	High
Coolpreviews	nsIProperties	High
Stylish	nsIWindowMediator, nsIProperties	Medium
Last pass	nsIWindowMediator	Low
Forecastfox	nsIWindowMediator	Low
Fox tab	nsIWindowMediator	Low
Fastest fox	nsIWindowMediator	Low
FB Chat Sidebar	nsIWindowMediator	Low
Facebook Toolbar	nsIWindowMediator	Low
web mail ad blocker	nsIWindowMediator	Low
Image Like Opera	nsIWindowMediator	Low

6.1 Comparison with Related Research

The results obtained by Adam Barth are tested as per our approach [17]. We tested same 25 extensions on our implementation to evaluate whether they respect private mode or not. Our approach also detects extensions that unnecessarily use sensitive interfaces in private mode. Out of 25 extensions 10 could be classified as suspicious for private browsing.

The list of extensions is as follows:

The Coolpreview and DownloadHelper are already included in Table 2. Fission, WeatherBug are not compatible with Firefox 12.01a which is modified during our approach.

Table 3. Comparison with Related Research

Javascript Add-ons	Sensitive Interface	Threat Level
Twitterfox	nsILoginManager, nsIWindowMediator	High
Delicious Bookmarks	nsIPermissionManager, nsICookieManager, nsIWindowMediator	High
Glue	nsIProperties, nsIWindowMediator	High
AutoPager	nsIProperties, nsIWindowMediator	High
Download	nsIDownloadManager	Medium

Status Bar		
Zemanta	nsIProperties,	High
Multiple Tab Handler	nsIWindowMediator	Medium
Lazarus: Form Recovery	nsIWindowMediator,	Low

The bug/enhancement and patch to the source code of Firefox is filed at bugzilla.mozilla.org for review [18]. The concept is well appreciated by Mozilla community. It is under discussion where modifications are being suggested in order to make it possible for actual inclusion.

All of them do not steal information but they can use interfaces detected in Table 2. to record information. Mozilla scans Firefox add-ons at remote level [12] but it does not step up towards adding filtering or sandboxing at browser level. Our modification in source code adds missing filtering at browser level.

7. CONCLUSION AND FUTURE WORK

We have presented an approach which successfully attempts to exploit vulnerability of Mozilla Firefox's private browsing by creating add-on such as "capture_data". We have also modified the source code of Mozilla Firefox, so that it detects suspicious add-ons that try to access user sensitive data items. Our approach adds privacy protection at browser level which is missing in current implementation. It could also be used to detect poorly coded add-ons which do not respect private browsing.

First task as a future work would be to get it approved by Mozilla Community so that mentioned concept would be incorporated into Firefox. Current implementation of patch would be modified as per requirement of Firefox community.

It is possible to monitor behavior of JS methods owned by Mozilla. Add-on may contain web JS such as window.addEventListener which is not owned by Mozilla. It can track browser's address bar and store that on file even in private mode. It is difficult to monitor such JSs. We would also address this issue in future work. It is required to modify the JS engine of Mozilla in private browsing such that it would categorize the suspicious web JS methods and make extra efforts to maintain its stack. As a result suspicious Add-on would be discovered once sensitive web JS would be accessed, even before their file operation takes place.

8. REFERENCES

[1] Aggrawal. G., E. B., C. J., and Boneh. 2010. An analysis of private browsing modes in modern browsers. *In Proceedings of 19th Usenix Security Symposium*.

[2] Lim. J. S., and Venkatakrishnan. 2007. *Policy-based Runtime Monitor for Browser Extensions*. Technical Report, Dept. of Computer Science, University of Illinois at Chicago.

[3] Is private browsing really private? Identifying Web browser risk, June 2011 http://searchsecurity.techtarget.com/tip/Is-private-browsing-really-private-Identifying-Web-browser-risk

[4] Mozilla Firefox - Supporting private browsing mode https://developer.mozilla.org/En/Supporting_private_browsing_mode

[5] Mozilla XPCOM https://developer.mozilla.org/en/XPCOM

[6] Sanitized Interfaces https://mxr.mozilla.org/mozilla-central/source/browser/base/content/sanitize.js

[7] Mozilla Public License Version 1.1 http://www.mozilla.org/MPL/1.1/

[8] The Chrome URL https://developer.mozilla.org/en/XUL_Tutorial/The_Chrome_URL

[9] Using JavaScript code modules https://developer.mozilla.org/en/JavaScript_code_modules/Using#Extending_resource:_URLs

[10] Mozilla XPCOM https://developer.mozilla.org/cs/XPCOM

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

[12] Add-on Submission Process: Criteria for Submission https://addons.mozilla.org/en-US/developers/docs/policies/submission

[13] Creating XPCOM Component https://developer.mozilla.org/en/Creating_XPCOM_Components

[14] Addon Manager Object. https://developer.mozilla.org/en/Addons/Add-on_Manager/AddonManager

[15] XPCJSID : GetService Method Implementation. http://dxr.lanedo.com/mozilla-central/js/xpconnect/src/XPCJSID.cpp#l783

[16] Mozilla XPConnect https://developer.mozilla.org/en/XPConnect

[17] Barth. A., F. A., S. P and Boodman. 2009. *Protecting Browsers from Extension Vulnerabilities*. Technical Report No. UCB/EECS-2009-185, EECS Department, University of California, Berkeley.

[18] Bug 761950 - Restraining Add-on's behavior in Private Browsing Mode https://bugzilla.mozilla.org/show_bug.cgi?id=761950