I J C T A, 9(7), 2016, pp. 3085-3093 ISSN: 0974-5572

Static analysis of Firefox OS privileged applications to detect permission policy violations

Shahanas P.* and Jevitha K.P.**

ABSTRACT

There is an emerging trend to use web browsers as mobile operating systems initiated by big market players such as Mozilla Firefox and Google Chrome. The applications for Firefox OS are basically web applications developed using HTML, CSS, JavaScript and other technologies. Firefox OS uses a Linux kernel and boots into Gecko runtime engine. It provides security features like sandboxed execution for applications, Content Security Policy and permission management system. In this paper, we present a study on the permission management system in Firefox OS through static analysis of its applications. The results of the study on 16 privileged applications downloaded from Firefox OS marketplace shows that about 7% of the permissions accessed by these applications are unauthorised. 13% of the permissions were requested but not used, 14% of the permissions were never requested but the equivalent WebAPI calls were being made in the application source code. Finally 66% of permissions were requested and used. The results reveal that many code reviewed privileged applications hosted on the Firefox marketplace do not conform to the Firefox OS permission policies and could cause potential threats to the system.

***Keywords**: Firefox OS;Application security; manifest analysis; Permission system*

1. INTRODUCTION

Mozilla launched Firefox OS, an entirely web-based Linux mobile operating system in July 2013. The Firefox OS architectural stack has four layers namely Gaia, Gecko, Gonk and the Mobile device [1]. The Gaia layer is the user interface; Gecko is the application runtime that implements the WebAPIs [2] and provides framework for application execution; Gonk has underlying Linux kernel, system libraries, firmware, and device drivers; and finally the mobile device running the Firefox OS. Applications can access the underlying mobile hardware functionalities only if Gecko accepts the permission request.Gecko is the intermediary layer that enforces the security policies and it prevents unauthorised requests to access the Web APIs.

Firefox OS applications are mainly of three types [3]-certified, privileged and web applications. Certified applications are those that comes along with the mobile phone itself. These can request security sensitive permissions in the application manifest file as they are considered trusted. Privileged applications are developed by third party developers and submitted to the Firefox OS marketplace for code review. After performing the code review, the marketplace digitally signs the applications in contrast to Android OS, where the applications are signed using the developer's certificate. Firefox OS internal/certified applications are signed by device manufacturers or operators. In addition, privileged applications will be having a Content Security Policy [4] that specifies the trusted domains from which the content can be loaded to the application. The third category is the normal web applications whose content is hosted elsewhere. A web

* Dept of Computer Science and Engineering Amrita School of Engineering, Coimbatore Amrita Vishwa Vidyapeetham University India, *Email: shahanasp00@gmail.com*

** Dept of Computer Science and Engineering Amrita School of Engineering, Coimbatore Amrita Vishwa Vidyapeetham University India, *Email: kp_jevitha@cb.amrita.edu*

application can be installed from any website and it does not require any further verification. When one installs this kind of application, only the application manifest file will be downloaded and stored on the mobile device. The application will appear as a shortcut which will load its content from remote host once the user launch the application. The application delivery can be in two ways-either packaged or hosted [2]. Certified and privileged applications are usually packaged while the web applications are hosted.

2. LITERATURE SURVEY

Marta Piekarska [5] et al., studied on the security architecture of Firefox OS and pointed out its shortcomings. Further they presented a threat model with possible attacks such as web attacks, usability based attacks and mobile network based attacks for the web based OS. The paper explained in detail about the permission system, application vetting and distribution process, sandboxing, vendor customization, fragmentation, and advertisements.

Daniel DeFreez [6] et al., have analysed Firefox OS from the perspective of application security considering two important factors which included code review in the Firefox marketplace and Content Security Policy. The paper used lightweight static analysis to detect applications that failed to validate message origin after registering the event handler. Attacks like Cross Site Scripting were also tested along with SSL certificate caching problems.

Olga Gadyatskaya [7] et al., compared the security architecture in Firefox and Tizen OS and concluded that both Firefox and Tizen security architectures were drawn from Android. They compared the operating systems based on memory management features like ASLR and DEP. According to their studies, Android provides address space layout randomization (ASLR) and data execution prevention (DEP) techniques as memory management security enhancements while Firefox OS had not implemented these techniques.

Borting Chen [8] et al., proposed an anomaly detection module for the Firefox OS. The paper addressed the two main drawbacks of Firefox OS which include consumption of abnormal amount of resources by applications and WebAPI being called with unusual frequency. These two internal anomalies were detected by the Anomaly Detection Module (ADM) by running a semi-supervised machine learning algorithm. They initially trained the input for normal behaviour and used this input in analysis phase to detect anomalies. A predefined threshold was used to differentiate normal and abnormal behaviour.

Marta Piekarsha [9] et al., addressed the privacy issues in Firefox OS through a tool called Privacy Dashboard. The tool provided features like remote privacy protection, guest mode for secondary users, permission control, backup and adjustable location accuracy. The tool followed a user centric approach in which initially data was collected based on the privacy awareness of an average user. A prototype solution was defined and then another study was conducted to analyse the impact of the prototype solution on users. The tool was refined with the changes thus solving the privacy concerns.

Mohd Najwadi Yusoff [10] et al., proposed an approach for forensic analysis in Firefox OS so as to investigate on any criminal intentions. The approach comprised three procedures namely Preparation and preservation procedure, Acquisition procedure and Examination/Analysis procedure.In the first phase after acquiring sufficient knowledge about the Firefox OS, forensic hardware and software was set up. After preparing the relevant and irrelevant data list, integrity of data was verified. In the second phase, they acquired the relevant forensic and imaging data and documented it. In the third phase, analysis using forensic tools was performed to determine the origin of data, how and when it was created, modified, accessed, etc.

AnthonyV´erez [11] et al., studied on the security model of the Firefox OS. The paper explained in detail about the security architecture of Firefox OS, applications, manifest file and security implementation

in OS. The paper summarised that information flow leakage problems could be handled in Firefox OS through sandboxing techniques. They also explained about buffer overflow and memory corruption which could be addressed through techniques like ASLR and build-flags hardening respectively.

3. PROPOSED SYSTEM

3.1. Static Analysis of Manifest and Javascript Files

Every Firefox OS application regardless of its type (certified, privileged or web app) will be having a manifest file[12] through which it request permissions to access WebAPIs related to specific hardware or other drivers. The Firefox OS marketplace can allow, deny or grant the permission on prompt [13] (i.e. ask user whether to grant or deny permission each time the app is launched). Almost all permissions can be accessed by the certified applications as their trust level is high. The privileged applications are having medium trust level and have access to fewer permissions. The web applications are having the lowest trust level and so most of permissions are denied for them. They can access least number of permissions. i.e., each kind of application is having a certain trust level anddepending on that there is restriction on the type of permissions that they can access. Every WebAPI has a required level of permission to be accessed. Whenever a WebAPI is called, Gecko performs a check on the permission requirements based on:

- Permissions that are associated with the calling app (as requested in the manifest file and based on the type of application).
- Permissions that are required to execute the requested operation (Web API call.)

For static analysis, a mapping is done between the permissions requested in the manifest file and corresponding Web API calls in the javascript file of the application.The proposed system architecture is depicted in Figure I. Firefox OS applications have HTML, XML, JavaScript, CSS and manifest files. In addition, they may also include json and signature files. The static analysis concentrates on the manifest and JavaScript files only, as they contain information related to permissions.Application manifest file follows a json structure. So, the set of permissions are retrieved initially from manifest using the key called “permissions”. The javascript files are analysed to find out the permissions used, by performing a check on the WebAPI calls made by the applications. “mapper.json” file has the permission and WebAPI call mappings which can be used to correlate the requested and used permissions. Table I shows the mapping of all permissions with its equivalent WebAPI calls.All the permissions are accessed using “navigator” object that is retrieved by “window.navigator” property [14]. It is observed that in some cases, applications should specify individual permissions to access a specific Web API. For e.g. applications requesting permissions like “device-storage” should specify in manifest file what storage it want to access (i.e. apps, crashes, video, pictures, music, sdcard) though all these are accessed using “get Device Storage” or “get Device Storages” Web API calls. For some permissions like “contacts” there is a one to mapping i.e. “contacts” accessed using “mozContacts”. Now considering the WebAPI calls, same permission may make different calls in different contexts in some cases. For e.g. “alarms” uses “mozAlarms” or “mozSet Message Handler”. The table also shows the type of application which can request these permissions [15]. For each application, based on the permissions requested in the manifest file and Web API calls in JavaScript file, the following four permission classifications are done.

1. Unauthorised permissions accessed (depending upon the trust level of application).
2. Permissions requested and used.
3. Permissions requested but never used.
4. Permissions never requested but used.

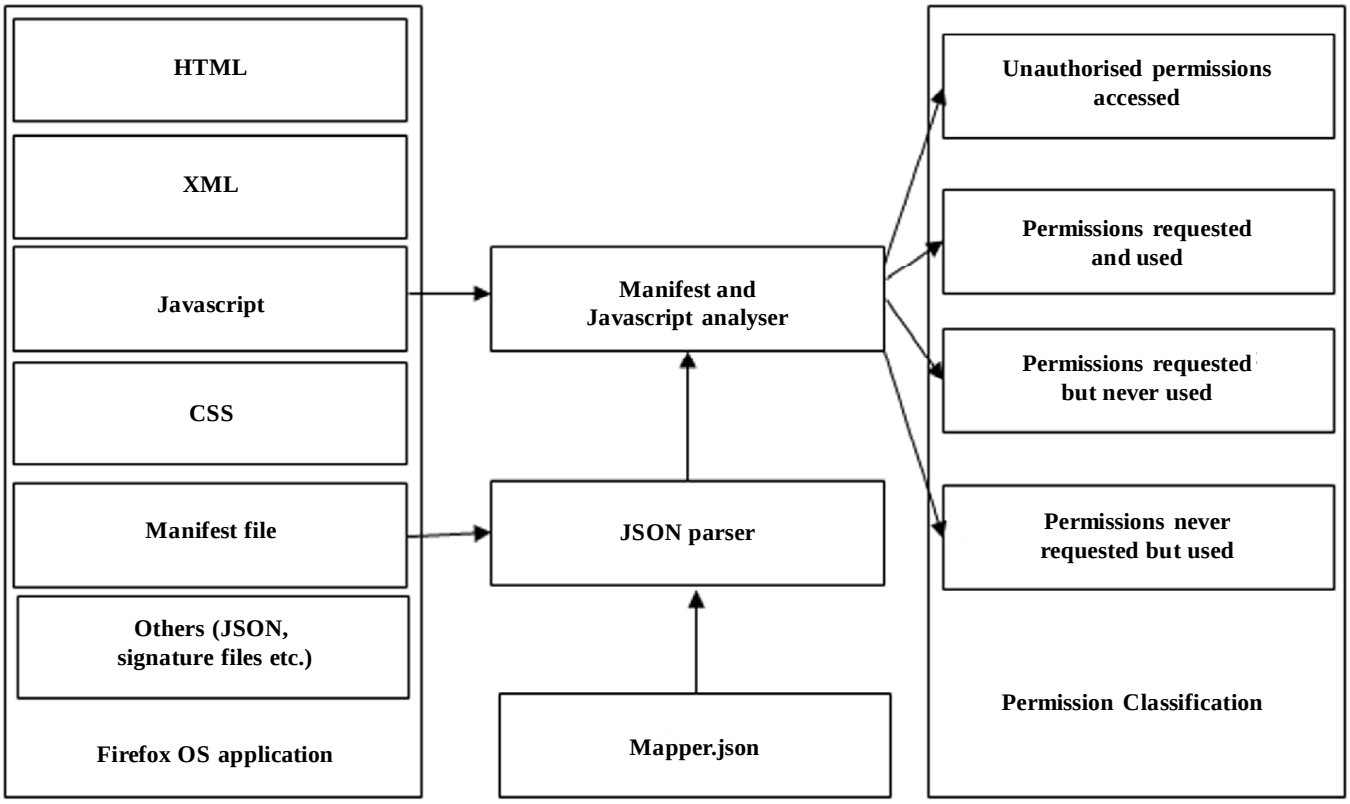

Figure 1: Proposed System Architecture

Table 1
Permission Mapping

Permission	*WebAPI call*	*Allowed access*		
		Web	*Privileged*	*Certified*
geolocation	geolocation, getCurrentPosition, watchPosition, clear Watch	User prompt	User prompt	User prompt
geolocation-noprompt	geolocation, getCurrentPosition, watchPosition, clear Watch	No	No	Yes
alarms	mozAlarms, mozSetMessageHandler	Yes	Yes	Yes
camera	mozCamera, mozCameras	No	User prompt	Yes
tcp-socket	mozTCPSocket	No	Yes	Yes
udp-socket	mozUDPSocket	No	Yes	Yes
contacts	mozContacts	No	User prompt	Yes
device-storage:apps	getDeviceStorage, getDeviceStorages	No	No	Yes
device-storage:crashes	getDeviceStorage, getDeviceStorages	No	No	Yes
device-storage:pictures	getDeviceStorage, getDeviceStorages	No	User prompt	Yes
device-storage:videos	getDeviceStorage, getDeviceStorages	No	User prompt	Yes
device-storage:music	getDeviceStorage, getDeviceStorages	No	User prompt	Yes
device-storage:sdcard	getDeviceStorage, getDeviceStorages	No	User prompt	Yes
Sms	mozMobileMessage, mozSMS	No	No	Yes
telephony	mozTelephony	No	No	Yes
bluetooth	mozBluetooth	No	No	Yes
mobileconnection	mozMobileConnections	No	No	Yes

(*contd*...)

(*Table 1 contd...*)

Permission	*WebAPI call*	*Allowed access*		
		Web	*Privileged*	*Certified*
mobilenetwork	mozMobileConnections	No	Yes	Yes
power	mozPower	No	No	Yes
push	push	Yes	Yes	Yes
settings	mozSettings	No	No	Yes
settings-clear	mozSettings	No	No	No
permissions	mozPermissionSettings	No	No	Yes
fmradio	mozFMRadio	No	Yes	Yes
browser	mozbrowser	No	Yes	Yes
desktop-notification	mozNotification, Notification	Yes	Yes	Yes
networkstats-manage	mozNetworkStats	No	No	Yes
systemXHR	XMLHttpRequest	No	Yes	Yes
idle	addIdleObserver, removeIdleObserver	No	No	Yes
time	mozTime, onmoztimechange	No	No	Yes
embed-apps	mozapp, mozApps	No	No	Yes
webapps-manage	mozApps.mgmt	No	No	Yes
wifi-manage	mozWifiManager	No	No	Yes
voicemail	mozVoicemail	No	No	Yes
cellbroadcast	mozCellBroadcast	No	No	Yes
audio-channel-normal	mozAudioChannelManager	Yes	Yes	Yes
audio-channel-content	mozAudioChannelManager	Yes	Yes	Yes
audio-channel-notification	mozAudioChannelManager	No	Yes	Yes
audio-channel-alarm	mozAudioChannelManager	No	Yes	Yes
audio-channel-telephony	mozAudioChannelManager	No	No	Yes
moz-audio-channel-telephony	mozAudioChannelManager	No	Yes	Yes
audio-channel-ringer	mozAudioChannelManager	No	No	Yes
moz-audio-channel-ringer	mozAudioChannelManager	No	Yes	Yes
audio-channel-publicnotification	mozAudioChannelManager	No	No	Yes
open-remote-window	window.open	No	No	Yes
input	mozKeyboard	No	Yes	Yes
audio-capture	getUserMedia	User prompt	User prompt	Yes
audio-capture:3gpp	getUserMedia	No	Yes	Yes
video-capture	getUserMedia	User prompt	User prompt	Yes
nfc	mozNfc	No	Yes	Yes
nfc-share	mozNfc	No	No	Yes
nfc-manager	mozNfc	No	No	Yes
nfc-hci-events	mozNfc	No	No	Yes
network-events	moznetworkupload, moznetworkdownload	No	No	Yes
firefox-accounts	mozId	No	No	Yes
moz-firefox-accounts	mozId	No	User prompt	Yes

4. RESULTS AND ANALYSIS

Static analysis is performed on set of privileged applications downloaded from Firefox OS marketplace. The analysis is explained using an example privileged application named “WhatsApp”. The permissions requested by “Whatsapp” application is presented in Figure II.

The results after analysis of “Whatsapp” is shown in Figure III.

The permission “settings” can be accessed through mozSettings API call by certified applications only. The privileged “Whatsapp” app access this permission without even specifying it in manifest file. The code snippet from Whatsapp application trying to use “settings” permission is shown in Figure IV.

The Firefox OS source code contains Permission Table [15] corresponding to Firefox OS API permissions and the code snippet corresponding to “settings” from the table is represented in Figure V.


```
"permissions": {"tcp-socket" : {"description" :
"Create TCP sockets and communicate over
them."},"device-
storage:sdcard":{"access":"readwrite",
"description": "storage"}, "device-
storage:pictures" : {"access":"readwrite",
"description": "Pictures"}, "device-
storage:videos" : {"access":"readwrite",
"description": "Video"}, "desktop-notification"
: {"description": "Desktop notification"}
,"feature-detection" : {"description": "Feature
detection"},"external-app" : {"description" :
"Access external app"},
 "moz-external-app" : { "description" :
"Additional permissions" },
 "contacts" : {"description" : "because i say
so","access":"readwrite"},
 "alarms" : {"description" : "Required to
schedule alarms"}}
```


Figure 2: Whatsapp manifest file

```
No. of unauthorised permissions accessed 1
No. of permissions requested and accessed 7
No. of permissions requested and never
accessed 1
No. of permissions never requested but
accessed 1
```

Figure 3: Whatsapp analysis results

```
function initSMSListener() {
 var settings = navigator.mozSettings;
  if(settings !== null) {
    if(DEBUG) console.log('Adding SMS listener');
    settings.addObserver('acl.sms.received',
smsReceived); }
  else {
    if(DEBUG) console.log('initSMSListener: cannot
access mozSettings');}}
function removeSMSListener() {
  var settings = navigator.mozSettings;
  if(settings !== null) {
    if(DEBUG) console.log('Removing SMS
listener');
    settings.removeObserver('acl.sms.received',
smsReceived);
  }
 else {
    if(DEBUG) console.log('initSMSListener: cannot
access mozSettings');
  }}
```

Figure 4: Code snippet from Whatsapp Javascript file

```
settings: {app: DENY_ACTION,
           privileged:DENY_ACTION,
           certified:ALLOW_ACTION,
           access: [“read”,”write”],
           additional: [“indexedDB-
chrome-settings”]}
```

Figure 5: Permission table code snippet for “settings”

For privileged application the action specified in the Permission table of Firefox OS is DENY_ACTION but the application is trying to use the permission by making a WEBAPI call. This indicates that application bypassed the code review process without getting detected. For certified applications the action is ALLOW_ACTION.The manifest analyser is run on 16 different applications downloaded from Firefox OS marketplace and the results are summarised in Table II.

Table 2
Application Statistics based on Manifest Analysis

Application name	*Description*	*Unauthorised permissions accessed by privileged applications*	*Authorised permissions accessed by privileged applications*		
			Permissions requested and accessed	*Permissions requested and never accessed*	*Permission never requested but accessed*
Battery Notifier	Notify battery level		alarms, desktop-notification		
BirthDay App	Birthday reminder		alarms	desktop-notification	
Contacts++	Smarter way to add face to contacts		contacts, system XHR		
Fire Wallet	Manage income and expenses			systemXHR	
Frases y citas célebres – Nex Cono	Knowledge sharing		system XHR		
Hawk	File manager		geolocation		device-storage
Net Tools	Penetration and security testing		tcp-socket, system XHR	browser	
Sección Amarilla	Hotel list for gps location		geolocation, system XHR		
Whats App	Messaging application	Settings	alarms, tcp-socket, device-storage, contacts, desktop-notification, moz-external-app, external-app	feature-detection	Settings
Telegram	Social networking app	open-remote-window	contacts, device-storage, push, desktop-notification		geolocation, system XHR, embed-apps/moz-external-app/ external-app
Jongla IM	Free messaging app	settings	contacts, device-storage, storage, desktop-notification, systemXHR, push	browser, audio-channel, mobile-connection/mobile-network	geolocation, embed-apps/moz-external-apps/ external-app
Skater Kid	Game app in which players should avoid obstacles		Storage		
Wish Buddy	Reminder app about special events of loved ones		storage, alarms, desktop-notification		

Macaw	Open source micro-blogging client app	open-remote-window	geolocation, system XHR, desktop-notification	embed-apps/moz-external-app/ external-app
zMaps	Allows users to view maps		geolocation, storage	
Viewfinder for Instagram	Instagram client		system XHR, browser	

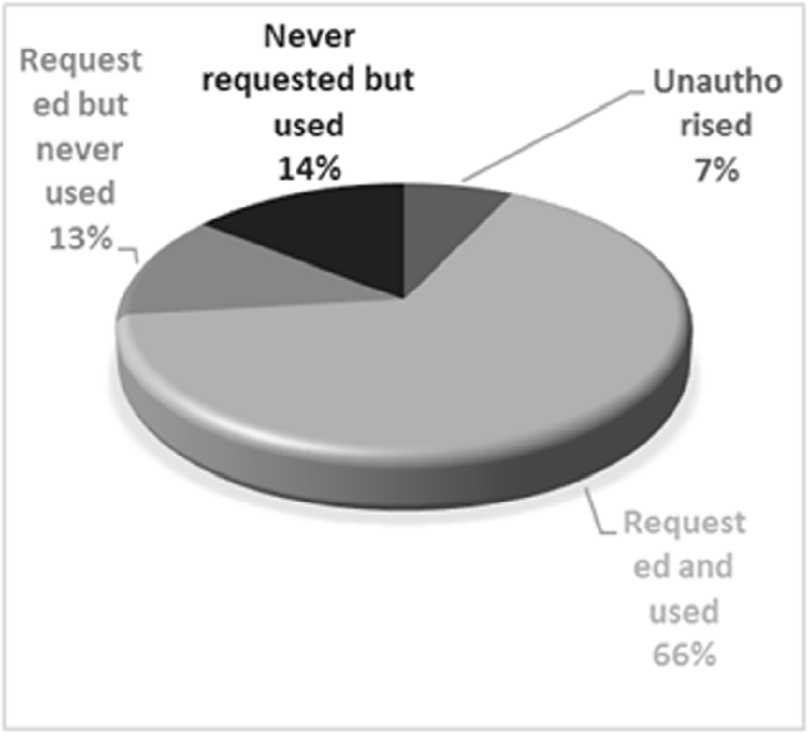

Figure 6: Permission Distribution

The percentage distribution for the four categories is depicted in Figure VI. Total number of permissions coming under the category “Permissions requested and used” is high which is about 66%. Permissions never requested but used accounts to 14% and about 7% of permissions accessed by the Firefox OS test applications are unauthorised. Permissions requested and never used comes with 13%. The unauthorised permissions requested and requested but never used permissions are the attacker targets for launching code injection attacks like cross site scripting. The permissions never requested but used shows the inefficiency of the code review process to detect the security violations. Figure VI. Permission Distribution

5. CONCLUSION

Firefox OS is an attempt made by Mozilla foundation to merge the web and mobile experiences. The web applications are portable across heterogeneous platforms and therefore both application developers and attackers concentrate on developing web applications rather than native applications. The results of the analysis shows that privileged applications are violating the permission manager policies and requesting unauthorised permissions. Also, the applications requesting more permissions than required could potentially be exploited by the attackers, to cause damage to the users of these applications. Firefox OS marketplace code review process is not detecting such security violations before the applications are digitally signed and released into the marketplace. Hence, there is more scope for the development of tools that uses static and dynamic analysis techniques to test the applications which violate the policies and this could help the marketplace reviewers and users in assessing the applications.

REFERENCES

[1] "Firefox OS architecture", https://developer.mozilla.org/en/docs/Mozilla/Firefox_OS/Platform/Architecture

[2] "Web API", https://en.wikipedia.org/wiki/Web_API

[3] "Firefox OS security overview", https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Security/Security_model

[4] "CSP (Content Security Policy)", https://developer.mozilla.org/en-US/docs/Web/Security/CSP

[5] M. Piekarska, B. Shastry and R. Borgaonka, "What Does the Fox Say? On the Security Architecture of Firefox OS," IEEE 9th International Conference on Availability, Reliability and Security, pp. 172177, 8-12 Sept. 2014

[6] D. Defreez, Shastry B., Chen H. and Seifert J.P., "A First Look at Firefox OS Security," Mobile Security Technologies (MoST), San Jose, CA, USA, May 17. 2014

[7] O. Gadyatskaya., Massacci F., Zhauniarovich Y., "Security in the Firefox OS and Tizen Mobile Platforms," IEEE Journals and Magazines, vol. 57, pp. 5763, June 18. 2014.

[8] B. Chen, M. Shih, Y. Huang, "An Anomaly Detection Module for Firefox OS," Eighth International Conference on Software Security and Reliability-Companion, pp. 176184, July 2014.

[9] M. Piekarsha, Y. Zhou, D. Strohmeier, A. Raake, "Because We Care: Privacy Dashboard on FirefoxOS", arXiv preprint arXiv:1506.04105, 2015.

[10] M. N. Yusoff, R. Mahmod, A. Dehghantanha, M. T. Abdullah, "An approach for forensic investigation in Firefox OS", "Third International Conference on Cyber Security, Cyber War Fare and Digital Forensic(Cyber Sec), May 2014.

[11] A. Vérez, G. Hugues, "Security Model of Firefox OS", 2013.

[12] "App manifest", https://developer.mozilla.org/en-US/Apps/Build/Manifest

[13] "App permissions", https://developer.mozilla.org/en-US/Apps/Build/App_permissions

[14] "Navigator", https://developer.mozilla.org/en-US/docs/Web/API/Navigator

[15] "Permission Table", https://mxr.mozilla.org/mozilla-b2g32_v2_0/source/dom/apps/src/PermissionsTable.jsm