

メタ API: プライバシポリシーに準じた機能制約を備えた API

鈴木富明† 小林真也†† 可児潤也† 川端秀明††† 磯原隆将††† 竹森敬祐††† 西垣正勝†

†静岡大学大学院情報学研究科 432-8011 静岡県浜松市中区城北 3-5-1

††静岡大学情報学部 432-8011 静岡県浜松市中区城北 3-5-1

†††株式会社 KDDI 研究所 356-8502 埼玉県ふじみ野市大原 2-1-15

あらまし 本稿では、端末情報アクセスに関するAPI(プライバシ関連API)に対し、プライバシポリシーに準じた機能制約を与える命令群を付加することにより、APIのプライバシ強化を実現する。命令群を付加することでラッピングされたAPIを「メタAPI」と呼ぶ。アプリ開発者がアプリのプライバシポリシーをチェックシート形式で入力するだけで、それに準じたメタAPIが自動的に用意される。アプリ開発者はこのメタAPIを使用してアプリを作成するだけでプライバシポリシーに準じたアプリが実装できる。メタAPIの使用(かつ、それ以外のプライバシ関連APIの不使用)をチェックするだけでアプリがプライバシポリシーに準じていることが確認できるため、マーケット側/端末側でのアプリとプライバシポリシーの整合性(アプリがプライバシポリシー通りの動作をするか)検査も容易となる。

Meta-APIs: Functionally limited APIs in accordance with privacy policy

Tomiaki Suzuki† Shinya Kobayashi†† Junya Kani† Hideaki Kawabata†††
Takamasa Isohara††† Keisuke Takemori††† Masakatsu Nishigaki†

†Graduate school of Informatics, Shizuoka University
3-5-1 Johoku, Naka, Hamamatsu, Shizuoka, 432-8011 Japan

††Faculty of Informatics, Shizuoka University

†††KDDI R&D Laboratories, Inc. 2-1-15 Ohara, Fujimino, Saitama, 356-8502 Japan

Abstract This paper proposes privacy-enhanced APIs, “Meta-APIs”, by adding functionally limited instructions in accordance with privacy policy to privacy related APIs that access privacy information in user’s mobile device. Meta-API is automatically generated from the application privacy policy that application developers make by filling in the check-sheet. Application developers have only to embed the Meta-API into the application, and the implemented application would be in accordance with privacy policy. A consistency check between application and privacy policy in an application market or a device would also become easier, because the consistency can be verified just by confirming the use of the Meta-API.

1 はじめに

近年のスマートフォンの普及によって、全ユーザが各自のモバイル端末を所持する世界が到来している。このようなモバイル環境では、端末とユーザが強く結びついているため、端末上で動作するアプリケーションにおける端末情報の取り扱いに対してプライバシーへの配慮が強く求められる。近年の不正アプリによる端末情報（プライバシー情報）の漏えいが数多く報告されていることを受け、最近ではモバイル端末アプリに対してもプライバシーポリシーを表示することが総務省により推奨されている[1]。そして、これとともにアプリマーケットがアプリのプライバシーポリシー作成を支援したり、第三者機関としてアプリのプライバシーポリシーを検査するような動きも始まっている[2]。

しかし、アプリ開発者がアプリのプログラムとプライバシーポリシーを別々に作成する形になっている以上、開発者の誤解やミスによって不正確なプライバシーポリシーが記述されてしまう可能性が残る。プログラムの静的／動的解析だけではアプリ内における端末情報（プライバシー情報）の利用状況を完全に把握することは容易でないため、端末側またはアプリマーケット側で、アプリとプライバシーポリシーの整合性を検査するには限界がある。

そこで本稿では、アプリ内で使用されるプライバシー関連 API（端末情報アクセスに関する API）に対して、プライバシーポリシーに準じた機能制約を与える命令群を付加することによって、API のプライバシー強化を果たす。プライバシー制約の命令群によってラッピングされた API を「メタ API」と呼ぶ。アプリ開発者がアプリのプライバシーポリシーをチェックシート形式で入力するだけで、それに準じたメタ API が自動的に用意される。アプリ開発者がこのメタ API を組み込んでアプリを作成することによって、プライバシーポリシーに準じたアプリが実装できる。メタ API が使用されていれば（かつ、メタ API 以外のプライバシー関連 API が使用されていないければ）、アプリがプライバシーポリシーに準じていることが確認できるため、

端末側／アプリマーケット側でアプリとプライバシーポリシーの整合性（アプリがプライバシーポリシー通りの動作をするか）検査を行うことも容易となる。

本稿では、次世代のスマートフォン向け OS として発展が期待されている Web OS の一つである Firefox OS [3] を提案方式の適用対象として考え、正規のアプリマーケットを通じて公開されるアプリの保護を実現する。

2 プライバシポリシー

2.1 課題

モバイル端末アプリにおける端末情報（プライバシー情報）の取り扱いの健全化に対する社会的な要求を受け、総務省によりスマートフォンプライバシーイニシアティブが取りまとめられた[1]。その中で、スマートフォン向けアプリに対してもプライバシーポリシー（アプリがどの情報を何の目的で使用するか）を表示することが推奨されている。しかし、プライバシーポリシーの運用にあたっては、下記の課題がある。

(1) アプリ開発者側の課題：

アプリ開発者は、アプリを作成する際に、そのプライバシーポリシーについても記述しなければならない。すなわち、アプリ開発にあたっての負担がその分増大することになる（課題 1a）。また、アプリ開発の作業とプライバシーポリシー作成の作業が分離されているため、アプリ開発の際にプログラムにバグが混入してしまった場合や、プライバシーポリシー作成の際に誤りが混入してしまった場合などに、アプリとプライバシーポリシーの間の整合性（一貫性）が崩れることなる（課題 1b）。更には、不正なアプリ開発者が、実際のアプリの動作とは異なるプライバシーポリシーを故意に作成する可能性もある（課題 1c）。

(2) ユーザ側の課題：

ユーザがアプリを自身の端末にインストー

ルする際には必ずプライバシーポリシーが表示されるようになっていたとしても、ユーザがその内容を理解できなかったり、十分に確認せずにインストールを許可してしまったりすることがある[1](課題 2)。

(3) マーケット側の課題:

プログラムの静的／動的解析によってアプリ内における端末情報(プライバシー情報)の利用状況を完全に把握することは容易でない。このため、アプリマーケット側で、アプリとプライバシーポリシーとの整合性(アプリが本当にプライバシーポリシー通りの動作をするのかどうか)を確認することが難しい(課題 3)。同様の理由で、アプリとプライバシーポリシーの整合性をユーザの端末内で検査することも容易ではない。

2.2 既存対策

課題 1a に対しては、アプリ開発者のプライバシーポリシー作成を支援する方法が提案されている[4]。文献[4]では、チェックシート形式の入力方法を採用することによって、開発者がチェックシートにチェックを入れるだけでスマートフォンプライバシーイニシアティブに沿ったプライバシーポリシーが作成される仕組みを実装している。また、アプリ開発者にプライバシーポリシー作成に対するインセンティブを与えるというアプローチも、課題 1a の対策として有効だと考えられる。文献[5]では、実際に、プライバシーポリシーを正しく記述しているアプリの開発者に対してアプリマーケットが奨励金を支払うという試みを運用した結果、プライバシーポリシーの充足率の向上が確認されたことが報告されている。課題 1b、課題 1c に対する既存対策については、著者らが調べた限りでは見当たらなかった。

課題 2 に対しては、文献[4]の中で、ユーザにプライバシーポリシーを分かりやすく表示する工夫についても示されている。また、関連研究として、アプリに関するワーニング情報やパーミッション情報をユーザに分かりやすく説明するための方法が種々提案されている[6][7]。これら関連研

究の知見は、プライバシーポリシーの表示に対しても活用できるのではないかと考えられる。

課題 3 に対しては、文献[8]が、アプリコードの静的検査によって広告モジュールやプライバシー関連 API の使用を確認し、プライバシーポリシーに記述すべき内容を提示する機構を提案している。また、アプリ内の端末情報(プライバシー情報)に「色付け」をすることによって、端末情報のトレースを可能にする方法が提案されている[9]。この技術を活用して、アプリにおける端末情報の利用がプライバシーポリシーに準じているかどうかの検査を端末側で動的に実施することができる可能性がある。ただし、一般的に、プログラムの静的／動的解析には高い技術が必要となるため、アプリの挙動を完全に把握することは難しい問題といえる。

3 提案方式

3.1 コンセプト

課題 1b、課題 1c の解決には、アプリ開発とプライバシーポリシー作成を一体化させるというアプローチが有効であると考えられる。すなわち、アプリからプライバシーポリシーを自動生成することができれば、プライバシーポリシー作成に関するアプリ開発者の負担を完全にゼロにすることができる。しかし、前述の通り、プログラムの静的／動的解析は容易ではない。このため、アプリのコードから完全な形のプライバシーポリシーを自動生成させることは難しいと言えよう。

そこで本稿では、プライバシーポリシーに準じたアプリを自動生成するという方法を検討する。すなわち、プライバシーポリシーに記載されている「アプリ内の端末情報(プライバシー情報)の取り扱い方法」に関する情報を利用することによって、アプリ内で使用されるプライバシー関連 API(端末情報アクセスに関する API)の自動生成を試みる。具体的には、プライバシー関連 API に対して、プライバシーポリシーに準じた機能制約を与える命令群を付加することによってプライバシー関連 API のプライバシー強化を果たす。プライバシー制

約の命令群によってラッピングされた API を「メタ API」と呼ぶ。

本稿では、チェックシートを利用したプライバシーポリシー作成支援が課題 1a の対策として効果を発揮していることに鑑み、アプリ開発者がプライバシーポリシー用チェックシートの記入を行うだけで、それに準じたメタ API が自動生成される仕組みを実装する。開発者は生成されたメタ API を自身のアプリに組み込むだけで、プライバシーポリシーに準じたアプリを作成できる。

アプリ開発者は、完成したアプリをプライバシーポリシーとともにアプリマーケットに公開する。マーケットでは、アプリの中でメタ API が使用されている（かつ、メタ API 以外のプライバシー関連 API が使用されていない）ことを検査するだけで、当該アプリがプライバシーポリシーに準じていることが確認できる。すなわち、アプリマーケット（または端末）側でアプリの静的検査を行うだけで、容易にアプリとプライバシーポリシーの整合性（アプリがプライバシーポリシー通りの動作をするか）検査が可能である。このように、提案方式は課題 3 の対策としての効果も有した方式となっている。

提案方式を 2.2 節に示した既存対策と併用することによって、2.1 節に示したすべての課題に対処することが可能となる。

3.2 プロシージャ

提案方式によって、正規のアプリマーケットを通じて公開されるアプリの保護を実現する。提案方式を運用した場合のアプリ開発の流れを、図 1 を用いて説明する。なお、ここでは、マーケットがプライバシーポリシーのチェックシートを用意するとともにメタ API の生成を担うという前提で話を進めるが、アプリ開発環境の中にこれらの機能を組み込むような運用も可能である。

- ① アプリ開発者は、アプリマーケットから提供されるチェックシートを利用して、プライバシーポリシーの記入を行う。
- ② マーケットは、開発者から提出されたプライバシーポリシーに準じたメタ API を開発者に提供

する。

- ③ 開発者は、マーケットから提供されたメタ API を利用してアプリを開発する。
- ④ 開発者は、完成したアプリをプライバシーポリシーとともにマーケットに提出する。
- ⑤ マーケットは、提出されたアプリとプライバシーポリシーの整合性検査を行う。具体的には、アプリのソースコードに対する静的検査によって、プライバシーポリシーで宣言されたメタ API がアプリの中で実際に使用されているか、かつ、それ以外のプライバシー関連 API が使用されていないかの確認が行われる。本稿では Web OS 上で動作する Web アプリを提案方式の適用対象としているため、マーケットがアプリのソースコードに対して静的検査を行うことが可能であることに留意されたい。
- ⑥ 整合性検査を通過したアプリは、プライバシーポリシーとともにマーケットに公開される。
- ⑦ ユーザは、マーケットに公開されたアプリを自身の端末にダウンロードする。アプリのインストールにあたっては、そのプライバシーポリシーがユーザに分かりやすい形で表示される。ユーザがプライバシーポリシーに対して了承した場合のみ、実際にアプリがユーザの端末にインストールされる。

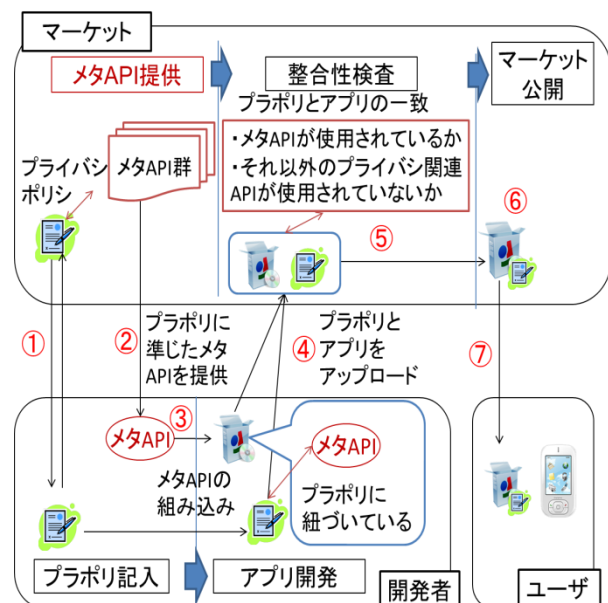


図 1 メタ API を利用したアプリ開発の流れ

4 実現例

提案方式の Firefox OS [3]への適応を例に採り, Firefox アプリにおけるメタ API の実現例を示す.

4.1 Firefox OS アプリの API

Firefox OS アプリは, HTML5, JavaScript, CSS などによって構成される Web アプリケーションプログラムであり[10], Web サイト上で稼働するホスト型アプリと Web ブラウザ内で稼働するパッケージ型アプリが存在する[11]. パッケージ型アプリは, API を介して端末側の情報を利用することができるようになっており, API の呼出し権限によって 3 つの種類に分かれる. 通常のアプリは最低限の API 呼出し権限しか有しておらず(Plain), マーケットの審査に通過したアプリは追加の API 呼出し権限を得ることができる(Privileged). プレインストールアプリなどに対しては, 通信事業者や端末メーカーが特別に高い API 呼出し権限を与えることができる(Certified)[11]. それぞれの権限ごとに, アプリが使用可能な API(API 呼出しによって取得できる情報)の一覧を表 1 に示す[12].

本研究の第一段階である現時点においては, 「一般のアプリ開発者によって作成され, 正規のアプリマーケットを通じて公開されるアプリ」を保護することによって, まずはマーケットを健全化することを目的としている. すなわち, Plain 型および Privileged 型のアプリが今回の保護対象である.

表 1 アプリの種類と呼出し可能 API

権限	使用可能なAPI
Plain	位置情報, 画面方向, 電池情報,
	加速度・近接・環境光センサー, 振動, データベースなど
Privileged	Plain型アプリが呼び出し可能なAPI,
	アドレス帳, SDカードの読み書き, XHR, TCP Socket
Certified	Privileged型アプリが呼び出し可能なAPI,
	電話, SMS, Bluetooth,
	料金発生サービス, Background Service など

表 2 プライバシ関連 API

GeolocationAPI	
オブジェクト	メソッド
navigator. geolocation	getCurrentPosition(A1,A2,A3)
	watchPosition(B1,B2,B3)
	clearWatch(C1)
Contacts API	
navigator. mozContacts	clear()
	find(D1)
	getAll(E1)
	getCount()
	getRevision()
	remove(F1)
	save(G1)

4.2 Firefox OS のプライバシー関連 API

Plain 型アプリと Privileged 型アプリが呼び出すことのできる API(表 1)の内, ユーザのプライバシーとの結び付きが強いと考えられる情報に関係する API(プライバシー関連 API)として, 今回は, 位置情報を取得する API(Geolocation API)とアドレス帳情報を取得する API(Contacts API)の 2 つを取り上げる.

Geolocation API と Contacts API の仕様を表 2 に示す. ここでは, 紙面の都合で, Geolocation API の getCurrentPosition メソッドのみを説明する. Geolocation API は, navigator.geolocation オブジェクトを通じて提供され, getCurrentPosition(A1,A2,A3)メソッ

ドを呼ぶことによって端末の現在位置を取得することができる。メソッドの引数 A1, A2, A3 は、それぞれ、位置情報を取得できたときのコールバック関数、位置情報を取得できなかったときのコールバック関数、位置情報を指定取得するオプションである。A1 のコールバック関数は position オブジェクトを引数にとり、取得された端末の位置情報が position オブジェクト内の各変数（緯度：position.coords.latitude, 経度：position.coords.longitude, 等）にそれぞれ格納されるようになっている。

4.3 API とメタ API

アプリの中で呼び出される API を検査することによって、アプリが取得する情報の種類を知ることができる。しかし、通常、アプリが情報を取得する目的は一意ではない。例えば Geolocation API を呼び出して位置情報を取得する目的は、地域連動型広告配信アプリとナビゲーションアプリで異なる。

これに対し、情報の利用目的（アプリがその情報を何のために取得するのか）をユーザに明示するものが、プライバシーポリシーである。すなわち、プライバシーポリシーは、API（取得される情報種別）ごとに記されるものではなく、情報の利用目的ごとに記されることに留意されたい。したがって、呼び出される API が同じであっても、プライバシーポリシー（情報の利用目的）が違えば、その API をベースにして生成されるメタ API も異なるものになる。

端末情報（プライバシー情報）の利用目的はアプリによって多岐に渡るが、ある程度の類型化は可能であると考えられる。特に今回は、検討対象となるプライバシー関連 API が（Geolocation API と Contacts API の）2 つだけであるので、それぞれの情報に対して、考えられ得る利用目的を洗い出して列挙しておけば、それに対応するすべてのメタ API を事前に用意しておくことができるだろう。

表 3 に、（Geolocation API によって取得される）位置情報と（Contacts API によって取得さ

れる）電話帳情報の利用目的を列挙した。ただし、表 3 は現時点では暫定的なものであり、今後更なる精査が必要である。

3.1 節にて、提案方式においては、チェックシートを利用したプライバシーポリシー作成支援を併用することを述べた。情報の利用目的を記すものがプライバシーポリシーであるので、利用目的の数だけプライバシーポリシーが存在することになる。すなわち、利用目的に応じて、チェックシートに記入する内容が異なり得る。このため、端末情報（プライバシー情報）の利用目的の洗い出しは、プライバシーポリシー入力用チェックシートのフォーマットを検討する上でも必要となる作業である。

表3 プライバシ関連APIの使用目的

プライバシー情報	利用目的
位置情報	サーバに送信
	情報を付加してサーバに送信
	外部メモリに出力
	画面に表示
電話帳情報	サーバに送信
	ハッシュ化してサーバに送信
	外部メモリに出力
	画面に表示
	メールの宛先に使用

4.4 メタ API の実装例

本節では、

- ・ アプリ開発者 D が広告アプリ A を作成する。
- ・ アプリ A は、起動時に、ユーザ端末の位置情報（緯度と経度）を取得し、広告サーバ S（url:www.s.jp, port:80）に送信する。
- ・ アプリ A はアプリマーケット M にて公開される。

というケースを例に取り、メタ API の実装例を示す。本稿では紙面の都合上、他のメタ API について説明する余裕はないが、4.3 節で述べたように、端末情報（プライバシー情報）の利用目的の種類の数だけメタ API が用意されることになる。

マーケット M には、すべてのメタ API が用意

されている。この内、「端末の緯度・経度を外部に送信」というプライバシーポリシーに準じたメタAPIが、図2のGPS_send_to_#url_#port()である。GPS_send_to_#url_#port()は、プライバシー関連APIである Geolocation API が、TCP 通信用APIである TCP Socket APIによってラッピングされた形となっており、navigator.geolocation オブジェクトのgetCurrentPosition メソッドによって取得された端末の位置情報(緯度, 経度)を、navigator.mozTCPSocket オブジェクトのopen メソッドとTCPSocket オブジェクトのsend メソッドによって特定の通信先に送信する。位置情報の送信の直後、TCPSocket オブジェクトのclose メソッドによって、soc オブジェクトに格納されている位置情報はクリアされる。

3.2 節の手順①にて、マーケット M から開発者 D に提示されるプライバシー作成用チェックシートを図3に示す。チェックシートは、「取得情報」と「利用目的」のみを記す簡易なフォーマットとなっており、今回であれば、開発者 D は「緯度」および「経度」のチェックボックスと「外部に送信」のチェックボックスにチェックをし、「送信先アドレス」および「ポート」にサーバ S の url (www.s.jp)と port (80)を記入するだけである。

手順②において、マーケット M は、開発者 D から提出されたプライバシーポリシーを確認する。チェックシートから、アプリ A が「端末の位置情報(緯度, 経度)を外部(url:www.s.jp, port:80)に送信」することが分かるので、マーケット M は、メタAPIの集合の中から今回のプライバシーポリシーに該当するGPS_send_to_#url_#port()を取り出し、これを開発者 D に提供する。ここで、図3のコードの中に含まれる「#url」および「#port」の文字列に対しては、それぞれ「www.s.jp」、「80」に置換された上で開発者 D に届けられることに注意されたい。

開発者 D は、GPS_send_to_www.s.jp_80()を利用して、アプリ A を開発し(手順③)、完成したアプリ A をマーケット M に提出する(手順④)。

手順⑤では、マーケット M が、アプリ A とプライバシーポリシーの整合性検査を行う。具体的には、アプリ A のソースコードを静的検査し、(i) アプリ A の中でGPS_send_to_www.s.jp_80()のメタAPIが使用されていること、(ii) このメタAPIが改変されていないこと、(iii) このメタAPI以外でプライバシー関連APIが使用されていないことを確認すればよい。

```
GPS_send_to_#url_#port(){
  navigator.geolocation.getCurrentPosition(
    function(position){
      // サーバ:server,ポート番号:portとの通信ソケット
      // をオープン
      var soc =
        navigator.mozTCPSocket.open("#url",#port);
      soc.onopen=function(){
        // ソケットオープン時,"緯度,経度"を送信
        soc.send(position.coords.latitude+","+
          position.coords.longitude);
      };
      soc.close();
    }
  );
}
```

図2 メタAPIの例

取得する情報	アドレス帳: <input type="checkbox"/> 電話番号 <input type="checkbox"/> 名前 <input type="checkbox"/> メールアドレス 位置情報: <input checked="" type="checkbox"/> 緯度 <input checked="" type="checkbox"/> 経度 <input type="checkbox"/> 高度 <input type="checkbox"/> 方角 <input type="checkbox"/> 速度
利用形態	<input checked="" type="checkbox"/> 外部に送信 送信先アドレス <input <input=""],="" type="text" value="80" ポート=""/> <input type="checkbox"/> 情報を付加して外部に送信 <input type="checkbox"/> 外部メモリに出力 <input type="checkbox"/> ディスプレイに表示

図3 プライバシーポリシー記入用チェックシートの例

5 まとめ

本稿では、プライバシー関連APIに対してプライバシーポリシーに準じたラッピングを施すことによってAPIのプライバシー強化を果たしたメタAPI

の導入を提案し, Firefox OS における実装例を示した. 提案手法は, プライバシポリシーをすべてのアプリに付帯させるにあたり, アプリ開発時の負担の増加や記載ミスの混入を抑えるとともに, マーケット(または端末)でのアプリとプライバシーポリシーの整合性検査を容易にする.

今後は, 端末情報(プライバシー情報)の利用目的の洗い出し作業を進め, メタ API を拡充させるとともに, 提案方式の統合開発環境化を推進いく予定である. 提案方式の実運用を通じて, 提案方式の可用性を評価していきたい. また, 提案方式の Firefox OS 以外のプラットフォームへの適用も検討したい.

参考文献

- [1] 総務省, “「スマートフォン プライバシーイニシアティブ –利用者情報の適正な取扱いとテラシー向上による新時代イノベーション–」の公表”:http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000087.html
- [2] 総務省, “利用者視点を踏まえた ICT サービスに係る諸問題に関する研究会提言「スマートフォン安心安全強化戦略」(案)に対する意見募集”:http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000111.html
- [3] Firefox OS:
<http://www.mozilla.jp/firefoxos/>
- [4] 磯原隆将 川端秀明 竹森敬祐 窪田歩: “XML を用いたモバイルアプリのプライバシーポリシーの運用”, 2013 年暗号とセキュリティシンポジウム(SCIS2013), 2013.01
- [5] 竹森敬祐 磯原隆将 川端秀明 窪田歩 高野智秋 可児潤也 西垣正勝: “アプリ/コンテンツ向けプライバシーポリシーの第三者検証フレームワーク”, 情報処理学会研究報告, 2013-CSEC-62, 2013.07
- [6] Adrienne Porter Felt, Serge Egelman, Matthew Finifter, Devdatta Akhawe, David Wagner : “How to Ask For Permission”, In Proc. USENIX Workshop on Hot Topics in Security, 2012
- [7] Sanae Rosen, Zhiyun Qian, Z. Morley Mao : “AppProfiler: A Flexible Method of Exposing Privacy-Related Behavior in Android Applications to End Users”, Proceedings of the third ACM conference on data and application security and privacy, ACM(2013), pp. 221-232
- [8] 磯原隆将 川端秀明 竹森敬祐 窪田歩: “Android アプリの静的解析を用いた利用 API と外部モジュール検知によるプライバシーポリシー作成支援機構”, 情報処理学会研究報告, 2013-CSEC-62, 2013.07
- [9] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth : “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones”, Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation(OSDI10), Canada, 2010
- [10] Firefox OS developers apps:
<https://developer.mozilla.org/ja/docs/Web/apps?redirectlocale=ja&redirectslug=Apps>
- [11] Firefox OS developers packaged apps:
https://developer.mozilla.org/en-US/docs/Web/Apps/Packaged_apps
- [12] Firefox OS Web API: <https://developer.mozilla.org/ja/docs/WebAPI>