# スマートフォン等のスマート・デバイスにおけるセキュリティ：プラットフォーム化によるリスクの現状と展望

磯部光平（いそべこうへい）／宇根正志（うねまさし）

## 要　旨

*近年、スマートフォンや IoT（Internet-of-Things）機器に代表されるスマート・デバイスが急速に普及している。特に、スマートフォンは、金融・決済サービス用のツールとして重要な位置を占めている。スマートフォン上で動作するアプリケーション・ソフトウェアには、サービスに関するデータの暗号化や認証等、セキュリティ機能を実現することが求められる。こうしたセキュリティ機能は、通常、スマートフォンに標準装備されている OS（Operating System）やハードウェア等のプラットフォームによって API（Application Programming Interface）として提供されている。そうした API の利用には、プラットフォームのセキュリティ機能が期待通りに動作するという状態が前提となるが、近年、プラットフォームのセキュリティに関する問題を示唆する研究成果が報告されている。アプリケーション・ソフトウェアのセキュリティを確保して安全なサービスを提供しつづけるためには、こうした研究動向をフォローしつつ、リスクの所在や影響について検討することが重要である。本稿では、スマート・デバイスのプラットフォームのセキュリティに関する最近の主な研究成果を紹介するとともに、アプリケーション・ソフトウェアの提供者の立場から、問題への対応のあり方について考察する。*

キーワード：　スマート・デバイス、スマートフォン、脆弱性、セキュリティ、プラットフォーム、API、IoT 機器

本稿の作成に当たっては、森達哉教授（早稲田大学）から有益なコメントを頂戴した。ただし、本稿に示されている意見は、筆者たち個人に属し、セコム株式会社や日本銀行の公式見解を示すものではない。また、ありうべき誤りはすべて筆者たち個人に属する。

磯部光平　セコム株式会社 IS 研究所（E-mail: ko-isobe@secom.co.jp）
宇根正志　日本銀行金融研究所企画役（E-mail: masashi.une@boj.or.jp）

## 1. はじめに

近年、スマートフォンやIoT（Internet-of-Things）機器に代表されるスマート・デバイスが急速に普及している[1]。特に、金融・決済分野では、個人が各種サービスを利用する際の主要なデバイスとして、スマートフォンが重要な役割を担っている。ユーザーは、自分のスマートフォンに専用のアプリケーション・ソフトウェア（以下、アプリ）をダウンロードして、さまざまなサービスの利用を手軽に開始することができる。例えば、スマートフォンが普及する前から利用されてきたモバイル・バンキングに加えて、複数の銀行口座の取引記録等をまとめて参照する家計簿サービスや、SNS（Social Networking Service）と連携して銀行口座間で送金を行うサービスが挙げられる。店頭での決済に関しては、スマートフォンの近距離無線通信を利用した電子マネーによる決済サービスが提供されてきたほか、スマートフォンのカメラを利用した店頭のバーコードによる決済サービスも提供されている。

スマート・デバイス上のアプリを開発しそれによってサービスを提供する主体（以下、アプリ・サービス提供者）は、データの盗聴や改変等、想定されるセキュリティ上の脅威への対策を予め講じておく必要がある。暗号化や認証等のセキュリティ機能はスマート・デバイスのプラットフォームによって準備されており、アプリに対してAPI（Application Programming Interface）として提供されている。プラットフォームは、OS（Operating System）、SoC（System on a Chip）[2]、周辺機器、ファームウェア等、スマート・デバイスに標準装備されたさまざまなソフトウェアやハードウェアから構成されている（Spensky *et al.* [2016]）[3]。アプリ・サービス提供者は、APIを利用することにより、自らセキュリティ機能を実装することなく、また、プラットフォームの個々の構成要素の仕様に立ち入ることもなく、一定のセキュリティ機能を実現することができる（Dessouky *et al.* [2020]、図1を参照）。

こうしたプラットフォームが標準的に提供するセキュリティ機能を利用するうえでは、それが期待通りに動作することが前提となる[4]。前述のとおり、プラット

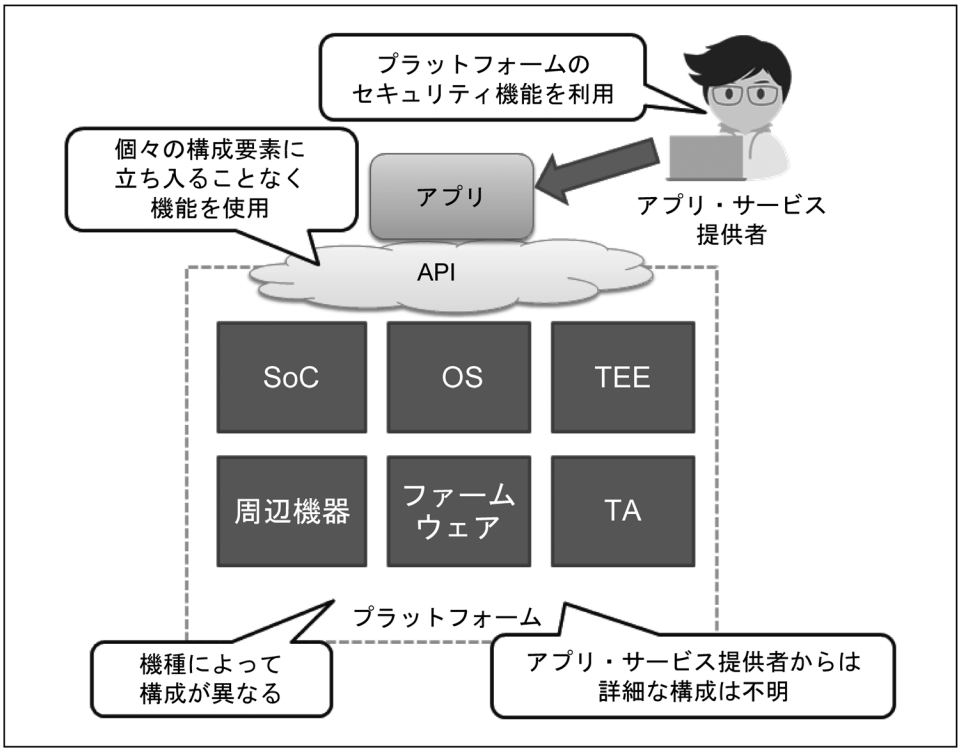
1 本稿執筆時点（2020年9月18日）では、スマート・デバイスには明確な定義が存在しないようであるが、スマートフォンに加えて、それと基本的に類似の構造を有するタブレット、ウェアラブル・デバイス（スマート・ウォッチ等）、IoT機器等のデバイスの集合体を指すことが多い。こうしたことから、一般にスマートフォンのセキュリティにかかわる事項は、スマート・デバイスに関連があると考えられる。

2 SoCは、1つの物理的なICチップの内部に複数のICチップの機能を回路として集約したものを指す。

3 これらに加えて、TEE（Trusted Execution Environment）を実現するハードウェアやソフトウェアもプラットフォームの構成要素に含まれる。TEEは、暗号処理等、比較的高いセキュリティが要求される処理のための実行環境（TEE空間）を提供する機能であり、TEE空間上で動作するアプリはTA（Trusted Application）と呼ばれる。

4 実際には、プラットフォームがアプリのセキュリティに影響を及ぼす脆弱性を含まないことも前提

図１　プラットフォームとアプリ間の関係




フォームの構成要素が多岐にわたり、アプリ・サービス提供者がそれらを正確に理解して動作を確認することは容易でない。また、アプリを動作させたい機種（プラットフォームの構成が異なるもの）が数多く存在する場合、それらすべてを確認することは手間やコスト面で事実上困難であると考えられる。さらに、構成要素の一部の仕様が公開されていないケースでは、アプリ・サービス提供者がプラットフォームの動作を確認することができないという問題もある。

こうしたなか、最近、プラットフォームに期待されているセキュリティ機能が十分には果たされないケースがあるとの研究成果が報告されている。例えば、一部のスマートフォンを対象に、OS によって提供される暗号鍵管理の機能について調査したところ、OS がサポートしている一部の暗号アルゴリズムにおいて、当該機能が適切に動作せず、アプリ・サービス提供者が期待する機能を使用できないケースがあるとの研究成果が示されている（磯部・坂本・葛野［2019］）。また、近年、TEE（Trusted Execution Environment）を実装するスマート・デバイスが増加するなかで、実装形態によっては TEE が提供する機能に脆弱性が存在することを指摘す

として求められる。

る研究報告や、それを悪用した攻撃や対策に関する研究報告も注目を集めている（Cerdeira *et al.* [2020]、Murdock *et al.* [2020]、Schwarz and Gruss [2020]）[5]。これらを踏まえると、アプリ・サービス提供者は、脆弱性のみならず、プラットフォームのセキュリティ機能等が有効でない場合や予想外の動作が生じる場合のリスクも想定しておくことが必要であろう[6]。

スマート・デバイスのプラットフォームは、デバイス・メーカーの開発体制やビジネスモデルに依存しており、アプリ・サービス提供者がプラットフォームの仕様の策定に関与する余地は小さい。そうしたもとで、アプリ・サービス提供者は、期待通りに動作しない可能性があるプラットフォームを使用せざるをえない場合があることに留意し、アプリによって提供するサービスのビジネスリスクを適切に評価し管理することが必要不可欠である[7]。

本稿では、アプリ・サービス提供者の視点から、スマートフォンを中心に、スマート・デバイスのプラットフォームにおけるセキュリティに関する問題を概観する。2節では、スマート・デバイスのプラットフォームの構成について説明し、3節では、プラットフォーム側の要因によって、アプリのセキュリティが低下するリスクに関する最近の主な研究成果を紹介する。4節では、3節に基づき、プラットフォームがアプリのリスクに影響を及ぼす状況を整理するとともに、アプリ・サービス開発者がそのセキュリティ・リスクを評価・管理するうえで生じうる課題や、ステークホルダー間での連携や情報共有といった対応について既存の事例とともに考察する。

5 SoCにおいても、ハードウェア上の脆弱性が存在しうるとともに、それが悪用されてセキュリティ上の問題につながりうることを指摘する研究成果が発表されている（例えば、Dessouky *et al.* [2019]）。

6 こうした問題に関連して、ユーザーが端末を購入する前にインストール済みのアプリ（プリインストール・アプリ）によるセキュリティ上の問題も注目されている。プリインストール・アプリは、基本的にデバイス・メーカーによって構成が管理されており、ソフトウェアのみで構成されているが、アプリのセキュリティ・リスクに影響を及ぼす可能性があるという点で、プラットフォームに関する問題と共通する部分がある。プリインストール・アプリのなかには、他のアプリが取り扱うデータを収集して外部に送信するケースがある旨が報告されている（Gamba *et al.* [2020]）。仮に、それらに脆弱性が存在し攻撃に悪用されたならば、データの流出や改変等のリスクが生じうる（Wu *et al.* [2019]）。これらに関連する最近の研究については補論を参照されたい。

7 こうした状況は、IoT機器においても生じている。IoT機器は、メモリ、センサー、OS、マイクロコントローラ、周辺機器やファームウェア等から構成されており、これらがそれぞれ異なるベンダーによって提供されるケースが少なくない（IoT推進コンソーシアム・総務省・経済産業省［2016］）。こうした点を踏まえると、スマート・デバイスにおけるプラットフォームに関する問題は、IoT分野における問題ともいえる。

## 2. スマート・デバイスのプラットフォーム

本節では、まず、スマート・デバイスのプラットフォームの構成における主な特徴を紹介する。次に、ソフトウェアとハードウェアにおける主なセキュリティ対策をそれぞれ説明し、最後に、スマート・デバイスやそのセキュリティ機能の開発プロセスをステークホルダーの側面から説明する。

### (1) 構成の主な特徴

スマート・デバイスのプラットフォームの構成の主な特徴として、① SoC の採用、②ソフトウェアの配布や開発環境の一元化、③スマートフォンをベースとした各スマート・デバイスの開発についてそれぞれ取り上げて説明する。

#### イ. SoC の採用

第 1 の特徴は、ハードウェアにおける高性能な SoC の採用である。Spensky *et al.* [2016] の分析によると、スマート・デバイス向け SoC の場合は、CPU（Central Processing Unit）や RAM（Random Access Memory）、GPU（Graphics Processing Unit）等を統合し、1 つのチップに収めている。これにより、基板の小型化や、消費電力の削減等のメリットを実現している。近年では、ハイエンド製品からローエンド製品まで、基本的な設計を共通とした SoC のラインナップ整備が進んだ。その結果、高機能な SoC が量産によって低価格化し、各デバイス・メーカーが開発する複数のモデルのスマート・デバイスに共通した SoC が搭載されるようになった。また、スマートフォン向け SoC から派生する形で、タブレットやウェアラブル・デバイス等のスマート・デバイス向け SoC が開発されており、同様に幅広いデバイスへの SoC の搭載が進んでいる。

#### ロ. ソフトウェアの配布や開発環境の一元化

第 2 の特徴は、OS ベンダーによるソフトウェアの配布や開発環境の一元化である。スマート・デバイスでは、その OS ベンダーが運営するアプリ・ストアから一元的にアプリを入手する形態を取っている[8,9]。これにより、スマート・デバイス向けのアプリ・サービス提供者は基本的にアプリ・ストアを通じてアプリを配布す

8 このような形態は総務省［2012］において示されている。また、OS ベンダー以外の主体（サード・パーティ）がアプリ・ストアを運営するケースもある。

9 パーソナル・コンピュータの場合、パッケージ・ソフトウェアの購入やインターネットからのダウンロード等、複数の手段でアプリを入手することができる。

る形になっている。アプリ開発プロセスにおいても、OS がプラットフォームとして各デバイスのハードウェアの差異を吸収し、アクセス方法や制御方法を統一した API をアプリ・サービス提供者に提供している[10]。これによって、アプリの開発者は、各デバイスの内部構成を意識することなく、多くの種類のデバイスで実行可能なアプリを短期間で開発することが可能となっている[11]。

ハ．スマートフォンをベースとした各スマート・デバイスの開発

第 3 の特徴は、スマートフォン向けのハードウェアやソフトウェアから他のスマート・デバイスが派生している点である。SoC ベンダーは、スマートフォン向けに開発された SoC をベースとして、さまざまなスマート・デバイス向けの新しい SoC を開発している。その際、スマートフォン向けの SoC の基本的な設計を流用しつつ、性能や消費電力を各スマート・デバイスの要件に合致するように設計する。こうして開発された新しい SoC は、スマートフォンと同様に、各社のスマート・デバイス向けに提供されている。

ソフトウェアに関しても、OS ベンダーは、スマートフォン向け OS について、各スマート・デバイスの画面や操作の特性に合わせた機能の拡充を行ったうえで、それぞれのスマート・デバイス向け OS として提供している[12]。アプリの配信は、スマートフォン向けのアプリ・ストアと共通化されている。アプリの開発においても、同一の環境で開発可能となるようにしている。

## (2) セキュリティ・リスクと対策

スマート・デバイスのプラットフォームにおける主なセキュリティ・リスクと代表的な対策を紹介する。対策については、OS での対策とハードウェア面での対策についてそれぞれ説明する。

10 実際のアプリの開発の際には、OS の API に加えて、サード・パーティが提供したライブラリ（例えば、広告サービスを提供する機能を有するもの）を利用する場合も多い。これらライブラリにも OS の API を利用するものがある。

11 例えば Google LLC が提供するスマートフォン向け OS の Android では、OS のなかにアプリケーション・フレームワークを設けており、アプリからハードウェアに対する呼出しはフレームワークを介して OS が受け付け、該当するハードウェアを呼び出す構造になっていることが開発者向けガイド（https://developer.android.com/guide/platform、2020 年 10 月 16 日）に記載されている。なお、Android は Google LLC の商標である。

12 例えば、Android から、スマート・ウォッチ向け OS である Wear OS by Google（https://developer.android.com/training/wearables、2020 年 10 月 16 日）やスマート・テレビ向け OS である Android TV（https://developer.android.com/training/tv、2020 年 10 月 16 日）が派生している。いずれも開発者向けガイドに Android をベースとしている旨の記述がある。なお、Wear OS by Google と Android TV はいずれも Google LLC の商標である。

### イ．OS での対策

スマート・デバイスのアプリは外部から入手して実行可能なため、マルウェアがアプリとしてスマート・デバイスにインストールされるリスクがある。さらに、個人情報や位置情報といったセンシティブなデータを取り扱うため、登場時からセキュリティ機能が OS に盛り込まれてきた。Han *et al.* [2013] の調査によると、初期のスマート・デバイス向け OS では、アプリの権限を制御する機能やアプリ間の通信を分離する機構が盛り込まれた。

具体的にみると、アプリの権限はそのプログラム内部で定義され、インストール時や実行時にそのデバイスの所有者（ユーザー）の同意が得られた場合に、当該権限が OS によってアプリに開放される。また、動作するアプリ間での直接の通信を禁止し、OS を経由させたり、実行可能な権限を限定したサンドボックスを利用したりすることで、マルウェアによる被害の低減を図るケースもある。さらに、ユーザーに対してもファイル・システムへのアクセスや設定変更が可能な範囲が限定されており、セキュリティに関する知識がないユーザーが不注意等でデバイスを危険な状態にすることがないようにしている。このように、各デバイスにおいて、ユーザーにセキュリティ対策を任せるのではなく、OS ベンダー等がセキュリティ対策を行うという仕組みが取り入れられてきた。

もっとも、OS が備えるセキュリティ機能の実装上の不備や脆弱性を悪用するマルウェアが登場している[13]。また、スマート・デバイスのユーザーが、前述した制限を回避するためにデバイスの改ざんを試みる点にも留意する必要がある[14]。OS によって実現されるセキュリティ機能は、その改ざんによって無効化されるため、その効果には一定の限界があるといえる。

### ロ．ハードウェア面での対策

ハードウェア面での主な対策として、ハードウェア RoT（Root of Trust）[15] と TEE について説明する。これらは SoC、すなわちハードウェア・プラットフォームによって実現されており、OS 等のソフトウェア・プラットフォームがこうしたセ

13 こうしたマルウェアを排除する方法の 1 つとして、OS ベンダーは、アプリ・ストアを通じたアプリの配布に当たって、マルウェアに特有のコードが含まれていないかなどをチェックするケースが多い。もっとも、未知のマルウェア等、事前のチェックで検知・排除できないケースもありうる。

14 ここでの改ざんはルート（root）化や脱獄（jailbreak）と呼ばれ、ユーザーが通常変更不可とされている設定の変更や、ゲーム等のアプリへの介入を目的としており、OS やファームウェアの改ざんによってセキュリティ機能を無効化する。

15 RoT とは、デバイスの内部全体の信頼性を確認・検証するための基点となる箇所を指す。RoT は、デバイス内部の別のコンポーネントを検証し、問題がなければ信頼関係を結ぶ。RoT によって信頼されたコンポーネントは、さらに別のコンポーネントを検証し、信頼関係を結ぶ。このように信頼関係を連鎖させることで、最終的にデバイス全体の信頼性を検証することにつながる。RoT はこうした信頼関係の構築の出発点であり、その安全性がデバイス全体の信頼性に影響を与える。

キュリティ機能を活用する対応が進んでいる[16]。言い換えると、スマート・デバイスに関するセキュリティ技術はソフトウェア・ハードウェア両面のプラットフォームで機能拡充が進んでいると捉えることができる。

### （イ） ハードウェア RoT

昨今のスマート・デバイスでは、ハードウェア RoT を設けることで、スマート・デバイス製造後に生じうる上述の攻撃によるソフトウェアの一貫性（integrity）の喪失に備えている。例えば、ソフトウェアの改ざんを検出する機能としてセキュア・ブート（Secure Boot）がある。セキュア・ブートでは、デバイス起動時に読み込まれるソフトウェアの改ざんの有無を電子署名の検証によって確認する。

ハードウェア RoT を備えるスマート・デバイスにおいては、この電子署名の検証用の鍵をスマート・デバイスのハードウェアに格納する。これによって、改ざんしたソフトウェアによって鍵を盗取するなどの攻撃から鍵を保護することが可能となり、セキュア・ブートの機能がより堅牢になる。ハードウェア RoT の実現には、SIM（Subscriber Identity Module）カードを端末に装着して利用するものや専用チップを搭載するもの等いくつかの手法があるものの、昨今では、SoC にハードウェア RoT の機能を統合させたものが多く登場している[17]。

### （ロ） TEE

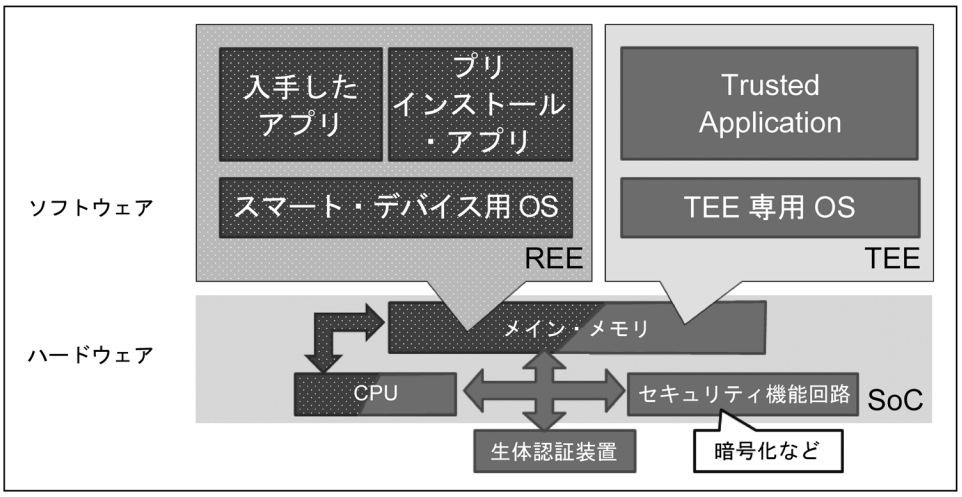
スマート・デバイス向け SoC に統合される特徴的なセキュリティ機能の 1 つとして、TEE が挙げられる。TEE は、メモリに展開されるプログラムの実行空間を分割し、通常の OS やアプリを動作させる REE（Rich Execution Environment）空間と、より高い安全性が求められる認証や暗号化の機能を動作させる TEE 空間を生成する。この分割は、ハードウェアによるアクセス制御機構によって実現されている。SoC ベンダーは、CPU やメモリ、それらを結ぶバスに対して、REE ／ TEE 空間の割当てや切替えをハードウェアとして実装することによって、ソフトウェアの機構に依存しない、すなわち、マルウェア等によってソフトウェアの機構に対する介入を受けても空間の分離を維持する仕組みとしている（図 2 を参照）。

TEE 空間で動作するアプリである TA（Trusted Application）は、DRM（Digital

16 ソフトウェア・プラットフォームは、ハードウェア・プラットフォームのセキュリティ機能を呼び出すインタフェースを備えている。そのため、アプリ・サービス提供者は、ハードウェア・プラットフォームの内部仕様を特別に意識せず、ソフトウェア・プラットフォームが提供する API を通じて、ハードウェアが備えるセキュリティ機能を利用することができる。

17 ハードウェア RoT は、SoC の回路の一部として搭載されるため、他の IC チップと同様にコモン・クライテリア（Common Criteria）等のセキュリティ機能の認証・評価制度の対象とすることができる。実際に、認証を取得しているものがある。また、回路として構成する際に、サイドチャネル攻撃（脚注 22 を参照）等の物理的な介入に対して耐性をもたせることもできるため、従来 IC チップで確保されてきたものと同等水準のセキュリティ機能が SoC に統合されて、スマート・デバイスに搭載されているといえる。

図２　TEE の概念図




Rights Management）の機能によって保護されているデータの復号、REE 空間で動作するアプリの暗号鍵管理、認証や決済データの生成機能の実装等に用いられる。また、TEE による空間の分離はスマート・デバイスに搭載されているセンサー等にも利用できる。例えば、指紋データの読取機器を TEE 空間にのみ接続する構成を採用して、生体認証データや認証ロジックを保護することが可能である。

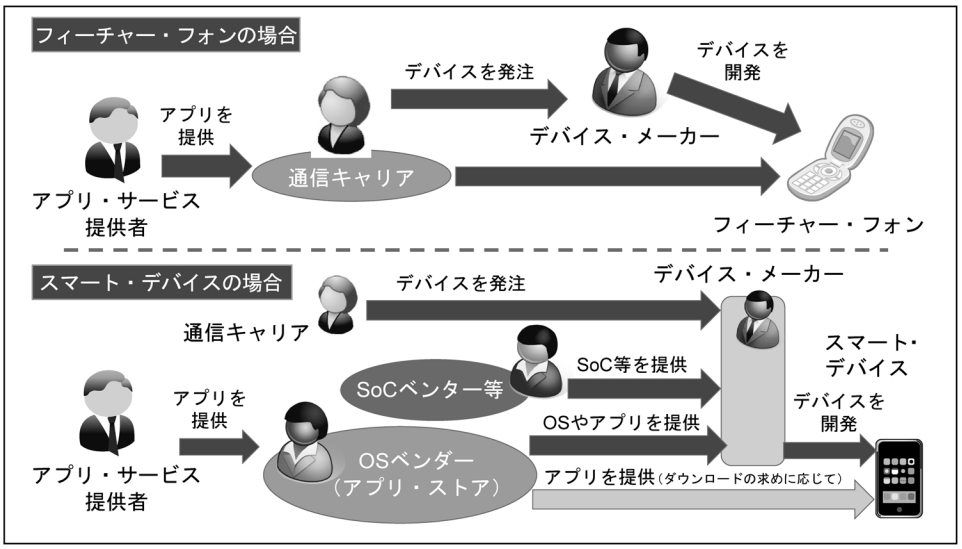
## (3)　ステークホルダー

スマート・デバイスの設計・開発にかかわるステークホルダーについて、従来型の携帯電話であるフィーチャー・フォンの場合とスマート・デバイスの場合をそれぞれ紹介した後、セキュリティ機能についてアプリ・サービス提供者からみた場合の差異を説明する。

### イ．フィーチャー・フォンの場合

わが国で 1990 年代から普及した携帯電話は、フィーチャー・フォン（いわゆるガラケー）と呼ばれるタイプであり、2010 年頃のスマートフォンの登場とともに、そのシェアは大きく低下した。フィーチャー・フォンからスマートフォンに代表されるスマート・デバイスへの移行は、デバイスそのものの変化に限らず、ステークホルダーの変化ももたらした。フィーチャー・フォンの調達者は通信キャリアであり、納入元であるデバイス・メーカーによって開発される形態が一般的であった。これらのデバイス・メーカーは、フィーチャー・フォンのハードウェアやソフト

図３　フィーチャー・フォンやスマート・デバイスのステークホルダー




ウェアの大部分を自社で開発していた。具体的には、ハードウェアをチップも含め自社で開発するところから始まり、ソフトウェアも同様にデバイス・メーカー主導、あるいは通信キャリアとの共同開発で進められてきた（図 3 上を参照）。

フィーチャー・フォンで実現されたセキュリティ機能には、通信キャリアが提供する SIM カードを利用した TLS（Transport Layer Security）のクライアント認証や、非接触 IC チップを利用した決済機能等が存在する。これらは、いずれもセキュリティ機能を専用のチップとして付け足す、または既存のセキュリティ機能を持った IC チップをデバイス内部に移植するアプローチで開発されてきた。

### ロ．スマート・デバイスの場合

一方、スマート・デバイスにおいては、ソフトウェアおよびハードウェアのプラットフォーム化が進み、開発にかかるステークホルダーおよびそれらの関係が、フィーチャー・フォンと大きく異なる状況となった（図 3 下を参照）。すなわち、スマート・デバイスのハードウェアにおいて、SoC ベンダーがデバイス・メーカーに提供した SoC が組み込まれ、ソフトウェアについては、OS ベンダーが提供した OS が組み込まれるという形態が生まれた[18]。デバイス・メーカーは、SoC と OS をそれぞれのベンダーから調達した後、自身のデバイスの特徴となる工夫を加えた形

18 こうした形態だけでなく、ハードウェア・プラットフォームからソフトウェア・プラットフォーム、さらにアプリ・ストアまでをすべて自社で開発する形態のデバイス・メーカーも存在する。一方で、すべてを外部から調達し、それらを組み合わせる形で開発を行うデバイス・メーカーも多数存在している。このように、デバイス・メーカーがどこまで自社で開発しているかについては、さまざまなバリエーションが想定される。本稿では、図 3 に示すように、OS および SoC を他のベンダーからそれ

でスマート・デバイスを開発して通信キャリアに納入する、あるいは通信キャリアを経由せずユーザーに直接販売する形態に変化した。

セキュリティ機能についても、ハードウェア・プラットフォームに搭載されたセキュリティ機能を活用するという方法が主流になった。これは、OS ベンダーが、アプリ・ストアへの対応要件の 1 つとして、プラットフォームによるセキュリティ機能への対応をデバイス・メーカーに求めるようになったことが主因といえる。もっとも、プラットフォームによるセキュリティ機能への対応度合いはデバイス・メーカーにより幅があるとみられる。

### ハ. アプリ・サービス提供者からみた差異

アプリ・サービス提供者の視点からフィーチャー・フォンとスマート・デバイスの状況を比較する。

まず、フィーチャー・フォンにおいては、アプリやサービスの提供は、基本的に通信キャリアが整備したプラットフォーム上で行う形となっており、安全性の管理も通信キャリアと共同で行われてきた。特に、通信キャリアがデバイスの仕様を決定したうえで、デバイス・メーカーから調達して販売するため、セキュリティ機能の種類や各デバイスにおける利用可能性についても通信キャリアによって管理される[19]。したがって、アプリ・サービス提供者は、通信キャリアからの情報を参照することによって、必要とするセキュリティ強度に対応するデバイスを選定することができた。

一方、スマート・デバイスにおいては、アプリ・サービス提供者は通信キャリアの介在なしに、アプリ・ストアを運営する OS ベンダーの審査を経てアプリをリリースする形となった[20]。スマート・デバイスは世界中に多くのメーカーが存在するが、各デバイスでの動作は OS ベンダーが提供する API を利用することで一定のレベルは保証される。もっとも、ハードウェア RoT の利用等を含めた各デバイスのセキュリティ機能については、デバイス・メーカーがすべてを把握して適切に動作することを保証しているとは必ずしもいえない。そのため、本稿執筆時点（2020 年 9 月 18 日）においては、アプリ・サービス提供者自身が、アプリに必要な安全性とセキュリティ機能を見極め、それに見合うセキュリティ機能等を利用できるスマート・デバイスを選定することが求められている。

このように考えると、スマート・デバイス向けのアプリやサービス提供において

---

ぞれ調達しつつ、他の構成要素を独自に開発しながらそれらを組み合わせてスマート・デバイスを開発する形態を前提に議論を進める。

19 通信キャリアによる管理の対象はそのキャリアが調達したデバイスに限定される。

20 SIM ロック・フリーのスマートフォンやタブレット、セルラー通信機能を必ずしも持たない IoT 機器においては、通信キャリアによる管理の対象がフィーチャー・フォンと比較すると小さくなっていると考えられる。

は、セキュリティ要求やその対応方法に関してアプリ・サービス提供者が管理すべき領域がより広範になっているといえる。

## 3. セキュリティ・リスクに関する主な研究事例

本節では、アプリ・サービス提供者がスマート・デバイスのプラットフォームのセキュリティ機能を利用する際のリスクに関して、鍵管理と TEE を対象とする最近の主な研究成果をそれぞれ紹介する。

### (1) プラットフォームにおける鍵管理に関するリスク

磯部・坂本・葛野［2019］は、スマートフォンにおいて、OS が提供している暗号鍵管理用の API をアプリが適切に使用できるか否かを調査した結果を報告している。鍵管理のセキュリティは、近年、OS の脆弱性を悪用した攻撃の高度化等を背景に、ソフトウェア・レベルでの対策に加えて、セキュア・エレメント等のハードウェアを活用した管理が標準的である。こうしたハードウェアによって実現される機能は通常 API として提供され、アプリは、ハードウェアの機能の詳細に立ち入ることなく比較的強固な鍵管理を実現することができる。

磯部らが調査対象としたスマートフォンの端末（Android）においては、OS の API（Keystore）が鍵の生成等を行う機能を有し、その鍵をハードウェアが管理する。そこで、Keystore を用いて、実際に暗号用の鍵を端末内で生成し、その後、鍵がハードウェアによって管理されているか否かを調査した。具体的には、Android を搭載した国内の 71 機種を対象に、仕様上 Keystore によってサポートされているとされる各暗号アルゴリズム（RSA、ECDSA、AES、HMAC〈ハッシュ関数は 5 種類〉）の鍵を生成して確認を行った。

調査の結果、RSA についてはすべての機種においてハードウェアによる鍵管理が行われていた。また、ECDSA と AES に関しても、1 機種を除き、ハードウェアによる鍵管理が行われていることが明らかになった。一方、HMAC については、ハッシュ関数の種類によってはハードウェアによる鍵管理が行われていないケースが散見された。そのようなケースにおいては、ソフトウェア・ベースで鍵管理が行われていた。具体的には、ハッシュ関数として、SHA-224、SHA-384、SHA-512 を使用する場合に、11 の機種において上記の事象が観察された。これらの機種においては、一部の HMAC 用の鍵管理をハードウェアによって実現できないため、鍵管理のセキュリティに関して、アプリ・サービス提供者が想定していたレベルを満たす

ことができないリスクが想定される。

磯部・坂本・葛野［2019］の調査結果により、スマート・デバイスのハードウェア・プラットフォームが提供している鍵管理用の API がアプリ・サービス提供者によって期待通りに活用できないケースがあることが示された。このようなケースがセキュリティを直接損なうわけではないものの、鍵管理が適切に実行されないリスクを示唆する。こうしたリスクを解消するために、API による鍵管理が適切に実施されていることを、事前にアプリ・サービス提供者が確認できるようにすることが望まれる。

## (2) TEE の脆弱性

2 節（2）ロ.（ロ）において説明したように、TEE は、スマート・デバイス向けの SoC に統合されるセキュリティ機能の 1 つとして注目されている。TEE は、ハードウェアの機能を利用してプログラムの実行空間を分離し、より高いセキュリティが要求される処理を保護することを目的としている。しかしながら、近年では、TEE を実装した製品の脆弱性や問題点を指摘する研究成果も報告されており、アプリ・サービス提供者においても TEE における脆弱性に留意することが肝要である（Murdock *et al.* [2020]、Schwarz and Gruss [2020]）。

例えば、Cerdeira *et al.* [2020] は、TEE を搭載した Android のスマートフォンにおける TEE 由来の脆弱性に関する研究を調査した。また、TEE を採用したいくつかのスマートフォンを解析し、TEE のセキュリティにかかる課題を指摘した。具体的には、TEE における課題を、設計、実装、ハードウェアの 3 つの観点から分類している。

まず、設計上の課題について、Cerdeira *et al.* [2020] は、TEE の機能が必要以上に多くなってきており、脆弱性が発生しうる余地も拡大している点を指摘している。具体的には、① TEE 空間と REE 空間を結ぶインタフェースが広く、REE 空間から TEE 空間にさまざまな形態のアクセスが可能となっている場合がある、② TEE 空間で動作する TA の権限が必要以上に設定され、悪用されるリスクが高まっているケースがみられる、③既にパーソナル・コンピュータにおいて採用されている不正なアプリの介入を防止する機構等が採用されていないものがあるというものである。

次に、実装上の課題については、脆弱性情報を識別する際に用いられる CVE（Common Vulnerabilities and Exposures）の番号が既に付与され公開されている脆弱性も対象に加えて調査・分析している。その結果、①入力値の検証が不十分である

ことに起因する脆弱性（例えば、バッファ・オーバーフロー[21]につながるもの）が存在するケースや、② SoC ベンダーが提供するセキュリティ機能の利用方法に誤りがあり、仕様通りの保護が実現されていないケース（例えば、疑似乱数生成器が適切に設定されていない）があることが示されている。また、③複数の TA を 1 つの TEE に含めて動作させた際に競合が発生し、TA のデータが誤って削除されたり、当初実行された処理が無効化されたりするケースも指摘されている。

ハードウェア上の課題については、TEE と連動するハードウェアを介した問題に焦点を当てている。具体的には、①プログラミング可能なハードウェアである FPGA（Field-Programmable Gate Array）等を追加した場合、それが新たな脆弱性をもたらす事例が紹介されているほか、② CPU やメモリに対するサイドチャネル攻撃[22]によって、REE 空間から TEE 空間を攻撃することが可能となる事例が取り上げられている[23]。

Cerdeira *et al.* [2020] によって指摘されている上記の問題は、いずれも、TEE の基本的なコンセプトをベースに実際のデバイスを設計・実装する過程で生じていると考えることができる。こうした設計・実装に関連する問題の背景には、2 節（2）ロ.（ロ）で述べたように、TEE が、OS ベンダーではなく、SoC ベンダーやデバイス・メーカーによって開発されているという事情もあるとみられる。SoC ベンダーやデバイス・メーカーは、第三者の TA が動作することを前提とすることなく、TEE を独自に設計・実装している場合があるとみられる。そのため、第三者のアプリの動作を前提とした対策が求められるスマート・デバイスの OS やパーソナル・コンピュータと比べると、TEE のセキュリティ機能や評価手法は各 SoC ベンダー等によってばらつきが生じると考えられる。

Cerdeira *et al.* [2020] は、こうした問題の解決策も紹介しているほか、SoC ベンダー等が対策を講じていくうえで、TEE を利用した製品開発や脆弱性の改善策を提案する研究者らと一層密に連携して検討していくことが重要であると指摘している。

21 バッファ・オーバーフローは、プログラムが入力許容容量を超えたデータの処理を行う際に、処理しきれないデータが本来想定していないメモリ領域に格納され、プログラムを上書きしてしまうという脆弱性である。

22 サイドチャネル攻撃とは、ハードウェアの動作に伴う周辺情報（消費電力や電磁波、熱等）からハードウェア内部の処理を推定する攻撃を指す。CPU やメモリが持つキャッシュ機構から本来アクセスが禁止されている処理の内容を推定する手法もサイドチャネル攻撃に含まれる。

23 サイドチャネル攻撃については、Dessouky *et al.* [2020] においても指摘されている。そのほか、デバイス上の不正なアプリを用いてメモリ等を意図的に誤動作させ（例えば、メモリ上のデータの書換え等を実施）、TEE によって管理されているデータを推定するなどの攻撃が可能になるケースがあるとの研究成果も報告されている（Murdock *et al.* [2020]）。

## 4. プラットフォームによるセキュリティ・リスクへの対応の方向性

3節でみたように、スマート・デバイスのプラットフォームは、アプリ・サービス提供者が期待した通りに動作しないケースがあるほか、脆弱性を有しているケースがある。これらは、プラットフォームのセキュリティ機能を利用するアプリにセキュリティ上の悪影響を及ぼす可能性がある。本節では、アプリ・サービス提供者がこうした事象によるアプリのセキュリティ・リスクを評価する際に問題となりうる状況を整理し、適切なリスク評価に向けた対応とそれに資する最近の事例を説明する。

### (1) アプリのセキュリティ・リスクに関して問題となりうる状況

以下では、プラットフォームがアプリ・サービス提供者の期待通りに動作しないケースと脆弱性を有するケースに分けて整理する。

#### イ. プラットフォームが期待通りに動作しないケース

このケースを想定したアプリのセキュリティ・リスクの評価は、通常、次のフローに基づいて行われると考えられる。

① アプリが利用するプラットフォームの機能を特定する。
② ①の機能が使用するプラットフォームのコンポーネントを特定する。
③ プラットフォームにおいて期待通りに動作しない機能が②のコンポーネントおよびアプリに及ぼす影響を特定する。
④ 期待通りに動作しない機能が悪用される蓋然性（確率）と、アプリによって提供されるサービスがその悪用によって被る金銭的な損失額を見積り、損失額の期待値をリスクの評価結果として算出する。

①に関しては、アプリ・サービス提供者は、アプリの開発時に、利用するプラットフォームの機能を特定する。

②については、プラットフォームの構成や仕様を把握していない限り、コンポーネントの特定は困難である。③に関しても、プラットフォームの構成や仕様に関する情報を十分に把握していることが前提となる。

④については、アプリ・サービス提供者がサービスにおける損失額をある程度推

定することは可能であろう。一方、期待通りに動作しない機能が悪用される蓋然性を検討するためには、プラットフォームに関する知識（例えば、プラットフォームにおけるセキュリティ対策の情報）が必要となる。

以上を整理すると、アプリ・サービス提供者がアプリのリスクを評価するうえで重要なことは、プラットフォームの構成や仕様等に関する情報を得る手段の確保である。こうした手段を確保できれば、リスク評価の結果に応じて対策を講じ、セキュリティ・リスクを軽減させることができる可能性が高い。一方、確保できなければ、アプリ・サービス提供者は、プラットフォームの構成や仕様等を十分に理解できず、リスク評価を行ったとしても不完全なものとなり、適切な対策を講じることは困難である。

### ロ．プラットフォームが脆弱性を有するケース

アプリのセキュリティ・リスクは、アプリおよびそれが動作するプラットフォームに脆弱性が存在するか否か、そして、これらのベンダーが脆弱性を認識して対策を講じているか否かに大きく依存する。以下では、プラットフォームの脆弱性に焦点を当て、アプリ自体の脆弱性は検討対象外とする。

プラットフォームを構成するコンポーネントのうち、OS の構成要素等のソフトウェアにおける脆弱性への対応については、新たな脆弱性に関する情報の報告に基づき、対象となっている OS や各種ソフトウェアのベンダーが、まず、必要な改修やパッチの作成を行うことになる。そのうえで、脆弱性とそれに対応するパッチ等の情報を公開しつつ、新しい OS 等をネットワーク経由で配布し、各スマート・デバイスのユーザーにダウンロードおよびアップデートを促して実行させる流れが一般的である[24]。

一方、ハードウェアにおける脆弱性に関しては、ソフトウェアのアップデートのような迅速な対応が困難な場合が想定される[25]。例えば、スマート・デバイスに内蔵されている TEE において脆弱性が存在し、それを解消するためにはセキュリティ機能のための回路の一部を改修する必要があるケースが想定される。このようなケースでは、デバイス・メーカーが脆弱な回路を含むスマート・デバイスを回収し、それぞれに新しい回路を組み込んだ後、各ユーザーに返却するといった対応が考えられる。もっとも、回収する対象のスマート・デバイスが大量である場合は、そのような対応の実現性は低い。したがって、当該機種の後継の機種を開発する際

24 デバイス・メーカーのなかには、自社の機種に関連する脆弱性の情報やパッチの対応状況等を定期的に公開する先もある（例えば、Android Security Bulletin）。脆弱性に関する情報が公表されてからパッチが準備されるまでの期間は、メーカーによって差異が存在しているほか、脆弱性の対象となるコンポーネントの種類にも依存しているとの調査結果が発表されている（Farhang *et al.* [2020]）。

25 こうした課題への対応として、新たな脆弱性が発見された場合等にハードウェアのコンポーネントを効率的に再構成する手法（例えば、Just-in-Time Reconfigurable Hardware Element）の研究が進められている（Dessouky *et al.* [2020]）。

に、脆弱性を解消した新しい回路等を組み込んで提供するアプローチの採用が現実的であると考えられる。その場合、脆弱性を解消した後継の機種が普及し（脆弱性を有する）旧機種が使われなくなるまで、旧機種にインストールされているアプリのセキュリティ・リスクが残存することになる[26]。

プラットフォームのベンダーが脆弱性への対策（パッチの生成と配布等）を迅速に実施する場合、プラットフォームの脆弱性によるアプリのセキュリティ・リスクは小さい。一方、上記のハードウェアにおける脆弱性のケースのように、迅速な対応が困難な場合には、アプリのセキュリティ上のリスクが増大する可能性がある[27]。アプリ・サービス提供者がプラットフォームの構成や仕様、脆弱性の内容を理解しているならば、本節（1）イ．と同様のフローに基づいて、脆弱性が悪用された際に想定される当該アプリのリスクをある程度評価し、そのリスクの多寡に応じて何らかの対策を講じることができる可能性がある。一方、プラットフォームの構成等を理解していない部分があるならば、リスクを適切に評価できず、結果として有効な対策を講じることができない。

## （2） 適切なリスク評価の実現に資するプラットフォーム・アプリ間連携

アプリ・サービス提供者は、プラットフォームにおいて期待通りに動作しない機能や脆弱性が発見された際に、それがアプリに及ぼす影響を迅速に理解することがビジネスリスク管理上重要となる。こうした影響についてデバイス・メーカーに問い合わせるという対応が考えられるものの、2節（3）ロ．でみたように、OSやSoCがそれぞれOSベンダーやSoCベンダーによって提供されているため、各コンポーネントの詳細な仕様をデバイス・メーカー自身でもすべて把握していない可能性がある。また、期待通りに動作しない機能や脆弱性が複数のコンポーネントに関係しているケースでは、複数のベンダーが連携して影響度合いを評価し対応することが必要となる。このように、アプリ・サービス提供者は、デバイス・メーカーや

26 近年、スマート・デバイスの性能向上に伴って、スマート・デバイスのライフ・サイクルが長期化する傾向にあるとの見方もある（磯部・坂本・葛野［2019］）。また、スマート・デバイスのなかでも、IoT機器等、他の機器に組み込まれるタイプのものに関しては、そのライフ・サイクルが組込機器のライフ・サイクルに依存することとなり、長期間交換することが困難なケースも想定される。

27 プラットフォームの脆弱性をそのベンダーが認識していない場合も含まれる。スマート・デバイスを解析したりリバース・エンジニアリングを実行したりして特定の第三者のみが脆弱性を発見する（ベンダー等に連絡しない）という状況は否定できず、密かに脆弱性が悪用されるおそれがある。もっとも、実際には、こうした未知の脆弱性への対応を予め検討・実施することは容易でなく、実務上の対応としては、ベンダーが脆弱性の存在を認識しているケースへの対応についてまず検討することが有用であると考えられる。

各コンポーネントのベンダーとどのように連携してスマート・デバイスの構成等に関する情報を共有していくかが、今後の重要な課題となる。

こうした状況に対し、関係者の円滑な連携や情報共有を可能とする試みがある。以下では、①アプリ・サービス提供者によるセキュリティ機能等の基準の整備、②スマート・デバイスの構成情報の公開、③プラットフォームのセキュリティ機能をオープン・アーキテクチャによって構成・管理する技術の開発というアプローチをそれぞれ紹介する。

### イ．アプリ・サービス提供者が基準を整備するもの

アプリ・サービス提供者が、必要とするセキュリティ機能を整備し、その基準や（基準を達成した）実装物をもとに、デバイス・メーカーに対応を要請するというアプローチが存在する。アプリ・サービス提供者は、個々のデバイス・メーカーに対し、基準への準拠や実装物のスマート・デバイスへの取入れを要請し、対応状況についても継続して管理する。こうした体制が実現すれば、アプリ・サービス提供者は、要請に対応したデバイス・メーカーが製造するスマート・デバイスについては必要なセキュリティ機能を確保することができる。また、アプリ・サービス提供者間で、共通した基準や実装物を利用するアプローチも考えられる。

上記の方法を実施した例として、Tencent の SOTER[28] が挙げられる。この取組みでは、スマート・デバイスの生体認証機能の互換性や一定の安全性の確保を目的とし、アプリ・サービス提供者としての Tencent が整備した基準の適用や TA のインストールをデバイス・メーカーに求めている。また、製造されるスマート・デバイスには上記機能の一貫性を検証する鍵も備えており、製造と同時に Tencent に検証用の鍵を登録する。Tencent は自社サービスが生体認証機能を利用する際に、この鍵でデバイスのセキュリティ機能の一貫性を検証している。この仕組みは、他のアプリ・サービス提供者にも開放されている。

### ロ．プラットフォームの構成情報を公開するもの

スマート・デバイスはさまざまなハードウェア、およびソフトウェアから構成される。この構成情報がデータとしてまとめられ、登録機関（レジストラ）に保存されていれば、アプリ・サービス提供者は、レジストラのデータベースを参照することによって、自身が必要とするセキュリティ機能を満たすデバイスを検索することが可能となる。さらに、サービス提供の対象とするスマート・デバイスの絞込みやリスク管理が必要となるデバイスのリストアップが容易になる利点がある。

こうした試みとして、GlobalPlatform, Inc. の DLOA（Digital Letter Of Approval）が

28 SOTER についての情報は GitHub（https://github.com/Tencent/soter、2020 年 9 月 14 日）に掲載されている。

ある[29]。DLOA では、TEE やセキュア・エレメント、それらを使うソフトウェア、TA の単位で分類されたコンポーネントに対し、セキュリティ機能やそれに対する認定等の情報が電子文書として記述される。この電子文書は登録機関に保存され、アプリ・サービス提供者は電子文書を一括取得し、自身の管理システムに取り込む。サービス提供時には、提供対象のスマート・デバイスと電子文書の内容を照合し、提供の可否を決定するとともに、当該デバイスのセキュリティ機能に応じたサービス提供を実施することができる。

ハ. プラットフォームそのものをオープン化するもの

これは、プラットフォーム内に含まれるセキュリティ機能を予め開示し、オープン・ソース・ソフトウェアと同様に、アプリ・サービス提供者を含む第三者による検証を容易とするものである。

最近の実践例として、セキュアオープンアーキテクチャ・エッジ基盤技術研究組合（TRASIO）の取組みがある[30]。この取組みでは、IoT 機器を対象に、SoC、TEE、これらのセキュリティ機能を管理する機能を、仕様が公開されているハードウェアやソフトウェアで構成し、プラットフォームに必要とされる一連の技術群の確立と公開を目的としている。今後、セキュリティ機能やその管理手法のオープン化により、プラットフォームの脆弱性の発見を容易にするとともに、その影響範囲を明確化できるようになること等が期待される。

## 5. おわりに

本稿では、2 節において、スマート・デバイスの基本的な構造やプラットフォーム、スマート・デバイスを取り巻くステークホルダーについて説明した後、3 節では、アプリのセキュリティ・リスクを巡る最近の主な研究成果を紹介した。4 節では、スマート・デバイスのプラットフォームがアプリのセキュリティ・リスクに影響を及ぼしうる状況を整理したうえで、リスクを適切に評価・管理する方法について考察した。特に、プラットフォームのセキュリティ機能が期待通りに動作しないケース、および、脆弱性が存在するケースにおいて、アプリ・サービス提供者がプラットフォームの構成等を理解できていない場合、アプリのリスク評価と管理を適

29 DLOA に関する文書が、GlobalPlatform, Inc. のウェブサイト（https://globalplatform.org/wp-content/uploads/2015/12/GPC_DigitalLetterOfApproval_v1.0.pdf、2020 年 9 月 14 日）に掲載されている。

30 セキュアオープンアーキテクチャ・エッジ基盤技術研究組合は、半導体チップのセキュリティを検証可能とするために、オープン・アーキテクチャを活用したエッジ向けのセキュリティ技術の試験・研究を目的とする研究組合であり、2019 年 8 月に設立された。詳細な情報は、同組合のウェブサイト（http://trasio.org/home/、2020 年 9 月 14 日）に掲載されている。

切に行うことは困難であるという問題があることを提起した。

リスク評価・管理に関する問題への対応としては、ステークホルダー間の連携による情報共有が重要である。そのような連携に向けた有望な取組みとして、①アプリ・サービス提供者によるセキュリティ機能等にかかる基準の整備とスマート・デバイスへの実装、②スマート・デバイスの構成情報の公開とそれに基づく（サービス提供対象の）スマート・デバイスの選定、③プラットフォームのセキュリティ機能をオープン・アーキテクチャによって構成・管理する技術の開発について概略と実践例を説明した。これらの取組みはいずれも比較的最近開始されたものであり、本稿執筆時点において金融機関や決済事業者等が直ちに活用できるわけではない点に留意する必要がある。

プラットフォームによるアプリのセキュリティ・リスクに関する課題は残されている。今後、スマート・デバイスのアプリを用いた金融・決済サービスが普及の一途を辿るとすれば、アプリのセキュリティ・リスクも大きくなる可能性がある。セキュリティ強度の向上には、スマート・デバイスのプラットフォームに具備されるセキュリティ機能の利用が不可欠である。金融機関や決済事業者は、こうした点に留意しつつ、スマート・デバイスのアプリによるサービスのリスク管理を行っていくことが求められる。その際、ステークホルダー間での連携や情報共有が重要であり、まずは、4 節で紹介した取組みを調査・研究しリスクの評価・管理への活用方法を検討することが有用であろう。

## 参考文献

磯部光平・坂本一仁・葛野弘樹、「ハードウェアベース暗号鍵管理に関する日本向け Android プラットフォームの調査」、コンピュータセキュリティシンポジウム 2019 論文集、情報処理学会、2019 年、1140〜1147 頁

総務省、『平成 24 年版 情報通信白書』、総務省、2012 年

IoT 推進コンソーシアム・総務省・経済産業省、「IoT セキュリティガイドライン ver 1.0」、総務省、2016 年

Cerdeira, David, Nuno Santos, Pedro Fonseca, and Sandro Pinto, "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-Assisted TEE Systems," Proceedings of IEEE Symposium on Security and Privacy (SP) 2020, IEEE, 2020, pp. 1416–1432 (available at https://www.cs.purdue.edu/homes/pfonseca/papers/sp2020-tees.pdf、2021 年 2 月 18 日).

Dessouky, Ghada, Tommaso Frassetto, Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf, "With Great Complexity Comes Great Vulnerability: From Stand-Alone Fixes to Reconfigurable Security," *IEEE Security & Privacy*, 18(5), IEEE, 2020, pp. 57–66.

———, David Gens, Patrick Haney, Garrett Persyn, Arun Kanuparthi, Hareesh Khattri, Jason M. Fung, Ahmad-Reza Sadeghi, and Jeyavijayan Rajendran, "HardFails: Insights into Software-Exploitable Hardware Bugs," Proceedings of the 28th USENIX Security Symposium, USENIX Association, 2019, pp. 213–230 (available at https://www.usenix.org/system/files/sec19-dessouky.pdf、2021 年 2 月 18 日).

Farhang, Sadegh, Mehmet Bahadir Kirdan, Aron Laszka, and Jens Grossklags, "An Empirical Study of Android Security Bulletins in Different Vendors," Proceedings of The Web Conference (WWW) 2020, Association for Computing Machinery, 2020, pp. 3063–3069 (available at https://arxiv.org/pdf/2002.09629.pdf、2021 年 2 月 18 日).

Gamba, Julien, Mohammed Rashed, Abbas Razaghpanah, Juan Tapiador, and Narseo Vallina-Rodriguez, "An Analysis of Pre-Installed Android Software," Proceedings of IEEE Symposium on Security and Privacy (SP) 2020, IEEE, 2020, pp. 1039–1055 (available at https://arxiv.org/pdf/1905.02713.pdf、2021 年 2 月 18 日).

Han, Jin, Qiang Yan, Debin Gao, Jianying Zhou, and Robert H. Deng, "Comparing Mobile Privacy Protection through Cross-Platform Applications," Proceedings of Network and Distributed System Security Symposium (NDSS) 2013, Internet Society, 2013 (available at https://www.ndss-symposium.org/wp-content/uploads/2017/09/06_2_0.pdf、2021 年 2 月 18 日).

Murdock, Kit, David Oswald, Flavio D. Garcia, Jo Van Bulck, Frank Piessens, and Daniel Gruss, "Plundervolt: How a Little Bit of Undervolting Can Create a Lot of Trouble,"

*IEEE Security & Privacy*, 18(5), IEEE, 2020, pp. 28–37.

Schwarz, Michael, and Daniel Gruss, "How Trusted Execution Environments Fuel Research on Microarchitectural Attacks," *IEEE Security & Privacy*, 18(5), IEEE, 2020, pp. 18–27.

Spensky, Chad, Jeffrey Stewart, Arkady Yerukhimovich, Richard Shay, Ari Trachtenberg, Rick Housley, and Robert K. Cunningham, "SoK: Privacy on Mobile Devices—It's Complicated," Proceedings on Privacy Enhancing Technologies, 2016(3), De Gruyter Open, 2016, pp. 96–116 (available at https://www.researchgate.net/publication/302065591_SoK_Privacy_on_Mobile_Devices_-_It's_Complicated/fulltext/572fa0fe08ae3736095c1d3f/SoK-Privacy-on-Mobile-Devices-Its-Complicated.pdf、2021 年 2 月 18 日).

Wu, Daoyuan, Debin Gao, Rocky K. C. Chang, En He, Eric K. T. Cheng, and Robert H. Deng, "Understanding Open Ports in Android Applications: Discovery, Diagnosis, and Security Assessment," Proceedings of Network and Distributed System Security Symposium (NDSS) 2019, Internet Society, 2019 (available at https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_06B-5_Wu_paper.pdf、2021 年 2 月 18 日).

## 補論.　プリインストール・アプリに関するセキュリティ上の問題

スマート・デバイスのプリインストール・アプリによるセキュリティに関して、最近の主な研究報告2件を取り上げ、概要を紹介する。プリインストール・アプリは、基本的にデバイス・メーカーによって構成が管理されており、ソフトウェアのみで構成されているため、本論で取り上げたプラットフォームと比較した場合、複数のステークホルダー間の連携や脆弱性修正は比較的容易と想定される。もっとも、アプリのセキュリティ・リスクに影響を及ぼす可能性があるという点では同一であり、アプリ・サービス提供者のリスク評価や管理において考慮することが求められる可能性がある[31]。

### (1)　プリインストール・アプリ全般に関するリスク

#### イ.　調査の対象と結果

スマートフォンには、ユーザーが使用を開始する前に、さまざまなアプリが既にインストールされていることが多い。ただし、これらのなかには、ユーザーが「プリインストールされている」ことを認識していないものも存在しうる。Gamba *et al.* [2020] は、約2,700のユーザーの協力を得て、約1,700種類の端末（Android。214のデバイス・メーカーが提供しているもの）から合計で約82,000のプリインストール・アプリの情報を収集して分析している[32]。

まず、プリインストール・アプリに付与された権限（カスタム権限）を分析した[33]。その結果、①端末の通話記録、ユーザーの識別情報、位置情報等にアクセスできるようにする、②システム・ログの読出しを可能にする、③他のソフトウェアを端末にダウンロードさせることができる、④自分（プリインストール・アプリ）の機能の一部を他のアプリに使用させることができるといった権限がアプリに付与

31 仮に、他のアプリにおいて処理されているデータにアクセスしたり、外部から動的にコードをダウンロードしたりする機能を有するアプリが端末にプリインストールされており、それらに脆弱性が存在したとすれば、それが悪用され、他のアプリのデータが外部に流出するなどの事象が発生する可能性が想定される。

32 Gamba *et al.* [2020] によれば、分析対象のアプリ群には、ユーザーが独自にインストールしたものではないアプリ（広告・宣伝、端末のユーザーの動作追跡等）、ライブラリ（ソフトウェア開発用キット〈Software Development Kit〉、スクリーン・キャプチャ等）、ミドルウェア（ルート化されていないもの）等が含まれている。これらのアプリを開発したベンダーをコード署名に紐づく証明書の属性情報に基づいて確認したところ、デバイス・メーカー、通信キャリア、ハードウェア・ベンダー、SNSや広告・宣伝等のサービス提供業者、セキュリティ・ベンダー等、多岐にわたっていたと報告されている。

33 収集したプリインストール・アプリのうち、約2,900件のアプリがカスタム権限を有していた。

されていることが判明した。

次に、プリインストール・アプリの動作を既存の分析ツール（静的解析）によって分析したところ、ユーザーや端末の識別情報、位置情報、デバイスの設定情報等を収集して別のアプリに送信する、通話や SMS（Short Message Service）の内容を収集し発信するなどの機能を有するアプリが存在していた[34]。また、インターネットにアクセスする権限を有する約 1,000 件のプリインストール・アプリを対象に、通信フローに関するデータを取得して接続先の組織（ドメイン）を分析したところ、全体で約 54,000 件のドメインへの接続が確認され、広告・宣伝、端末の動作の追跡・分析、SNS 等のサービスを提供する事業者等、さまざまな組織に接続していたことも判明した[35]。

### ロ．想定されるリスクと含意

このように、スマートフォンには、多種多様な機能・権限を有するアプリがプリインストールされており、それらがさまざまな外部の組織と通信しつつ端末内部のデータを外部に送信したり変更したりするケースが存在しうることが示された。こうしたアプリのなかには、遠隔からスマート・デバイスの動作を監視し疑わしい動作を検知するとその動作を停止させる MDM（Mobile Device Management）等、端末のセキュリティを確保するうえで必要な機能を実現するものも存在している。

ただし、セキュリティ機能のためのアプリであっても、未知の脆弱性が存在する可能性は否定できない。仮に、脆弱性が存在し悪用されてしまうことがあるとすれば、端末内部のデータの盗取、不正なアプリのインストール、他のアプリの機能の悪用といったリスクが生じることになる[36]。

プリインストールではないアプリを提供する立場からみると、当該アプリの動作や取り扱うデータがプリインストール・アプリによってセキュリティ上どのような影響を受けるかという観点から上記のリスクに留意しておく必要があると考えられる。

34 具体的には、各アプリの動作に関する情報を約 40,000 件の APK（Android Application Package）ファイル（Android のアプリの実行形式ファイル）から抽出して分析を実施した。その結果、分析対象のアプリ群のうち 25％以上が上記の動作を実行しうるとの結果が示されている。

35 接続先のドメインについては、FQDN（Fully Qualified Domain Name）によってアプリが最初に接続したサーバを特定した。

36 Gamba *et al.* [2020] では、分析対象のアプリ群（ただし、Google Play〈Google LLC の商標〉等で公開されていないアプリを除く）のうち、約 74％のアプリが端末に導入されてからアップデートされていなかったほか、約 11％のアプリが 5 年以上アップデートされていなかったとの調査結果が示されている。こうしたアプリにおいて、仮に、脆弱性が存在しパッチが作成されていたとすれば、パッチが適用されず脆弱性も長期間放置されていたことになる。

## (2) 外部からアクセス可能なプリインストール・アプリに関するリスク

### イ. 調査の対象と結果

スマートフォンのアプリの多くは、サービスを提供する際に、モバイル通信網やWi-Fi（Wi-Fi Allianceの商標）等のネットワークを介して外部のサーバ等とデータをやり取りする。そうした設定がなされたアプリの場合、端末外部からの不正なアクセスの対象になりうる。こうしたリスクに関して、Wu *et al.* [2019]は、約3,200の端末（Android）に搭載されていたアプリを対象に、通信の形態や脆弱性の有無について調査・分析した結果を発表している。

Wu *et al.* [2019]は、調査対象の端末において、外部のサーバ（IPアドレスで特定）からデータ（パケット）を受信できるように設定されていたアプリ（全体で約2,700件）を識別しつつ、それらの動作に関するデータを収集して分析した[37]。これらのアプリには約700のプリインストール・アプリ（built-in apps）が含まれており、それらは約20のデバイス・メーカーによって提供されていた。

Wu *et al.* [2019]は、これらのアプリにおける通信の設定を確認した。その結果、約65%のアプリが、接続先のIPアドレスを動的に設定するためのプロトコルDHCP（Dynamic Host Configuration Protocol）等を利用するために、外部からアクセス可能な設定となっていたことが判明した。また、約23%のアプリについては、音声や動画等をやり取りするために、外部からアクセス可能な設定であった[38]。さらに、後者のアプリ群のなかには、端末の動作を制御するためのアプリであって、音声等を通信する必要性がないものが含まれていた旨を指摘している。

### ロ. 想定されるリスクと含意

端末のプリインストール・アプリに対して外部からアクセス可能な設定になっていたとしても、アプリに脆弱性が存在しておらず、かつ、認証等によってアクセス元を適切に制御することができていれば、実際には不正なアクセスを防止することができると考えられる。

これに関して、Wu *et al.* [2019]は、プリインストール・アプリの一部をリバー

37 具体的には、Wu *et al.* [2019]は、インターネットにおけるデータの転送制御プロトコルTCP（Transmission Control Protocol）またはユーザー・データグラム・プロトコルUDP（User Datagram Protocol）を用いて通信するアプリに焦点を当てて、アプリによる通信の種類（ポート番号）、通信の時間（ポート継続時間）、通信相手（IPアドレス）等の情報を、各端末に導入した通信監視アプリによって収集した。

38 これらのアプリでは、音声通話のためのプロトコルであるVoIP（Voice over Internet Protocol）や、音声や動画を通信する際のセッションを確立するためのプロトコルSIP（Session Initiation Protocol）を利用していた。

ス・エンジニアリングによって分析したところ、データを転送する際の転送先の確認が不十分とみられるアプリが存在していた旨を指摘している。そのアプリは、周囲に存在するデバイスのなかから当該端末に接続可能なものを自動で探索する機能（Automatic Service Discovery）を有しており、探索の際に接続した任意のデバイスに対して、当該端末の機種やユーザーの識別情報等を含むデータを送信していたとみられている。

また、調査対象アプリ全体の脆弱性に関して、Wu *et al.* [2019] は、脆弱性の発生につながりうる事象を抽出して整理しているほか[39]、モバイル通信網等における端末間通信の可能性についても分析している[40]。

こうした調査結果を踏まえて、Wu *et al.* [2019] は、プリインストール・アプリにおける外部からのアクセスによる影響やリスクを、デバイス・メーカーがより注意深く評価することが有益であるとしている。プリインストール・アプリの開発者に対しては、そのアプリに外部からのアクセスが必要か否かを見極めるとともに、必要最小限のアクセスのみ許容する、アクセス元を認証するなどの対応が有用であるとしている。通信キャリアに対しては、同一ネットワークにおける端末間の直接的な接続を制限するなどの対応が望ましいとしている。

39 Wu *et al.* [2019] は、アプリの脆弱性に関する分析を行った結果、発見した事象を以下のカテゴリーに分類している。すなわち、①データ転送時に求められる確認が不十分なケース（認証の処理が適切に実行されていないなど）、②コマンド実行時に求められる確認が不十分なケース（デバッグのためのポートの開閉が適切に行われていないなど）、③アプリの動作が阻害されるケース（アプリが不正な形式のデータを受信して停止する〈Crash-of-Service〉、アプリが処理するデータの量を急速に増加させるなど）、④データ分析用のインタフェースが安全でないケース（不正なアクセスによって分析用のデータが盗取されるなど）が紹介されている。

40 攻撃対象となりうる端末に別の端末からアクセス可能か否かについて、Wu *et al.* [2019] は、約 200 のモバイル通信網と約 2,000 の Wi-Fi ネットワークを対象に検証（ピング〈Ping〉による疎通確認や TCP/UDP パケット通信の可否の確認）を行った結果を報告している。約半数のモバイル通信網と約 80％の Wi-Fi ネットワークにおいて、端末間通信での疎通確認が成功したほか、TCP パケットの通信も成功したとしている。また、約 20 のモバイル通信網と約 10 の Wi-Fi ネットワークにおいては、IP アドレスを端末に割り当てる設定となっており、当該ネットワーク内での端末間接続だけでなく、インターネット上のホスト・コンピュータから接続されてしまう可能性があったと報告している。