Google I/O 2018
What's new in Android Security

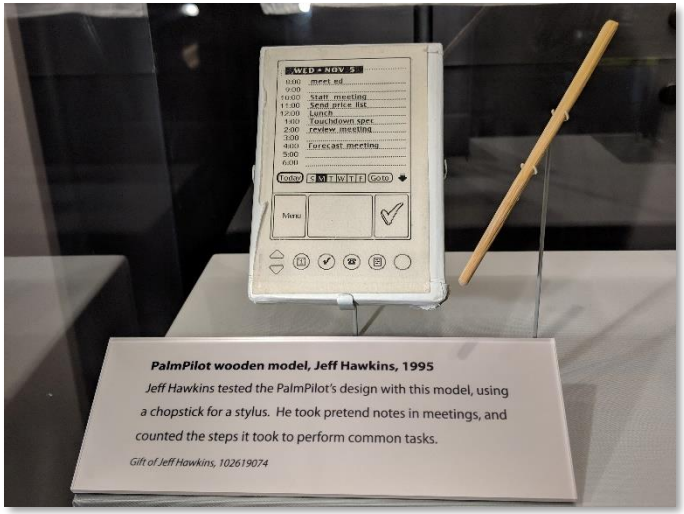
ABC Spring 2018
2018/6/9(土) 16:00〜16:50
246講義室

タオソフトウェア株式会社
代表取締役
谷口岳
@tao_gaku

RiskFinder

本セッションについて

Google I/O 2018で「What new in Android Security」セッションがありました。どのような事が話されたのかを解説を踏まえながら説明致します。



自己紹介

タオソフトウェア株式会社

- 代表取締役　谷口岳
- ２００５年創業、Android関連システム受託開発
- Google Playにアプリを多数公開
- ブログにて開発者向け情報を発信（昔）
- 雑誌執筆、講演
- セキュリティ色々

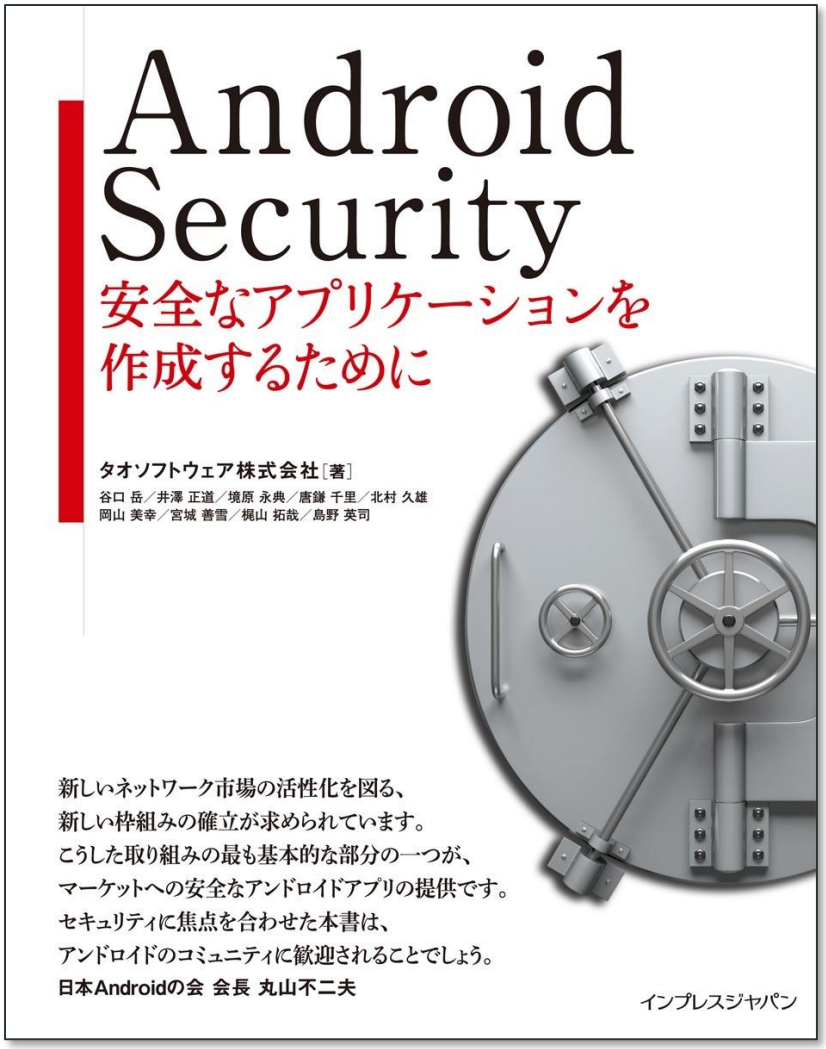
自己紹介

タオバイザー

3DVRゴーグル

- クラウドファンディングで資金調達
- http://taovisor.com

Android Security 安全なアプリケーションを作成するために



２０１２年１月１日発刊

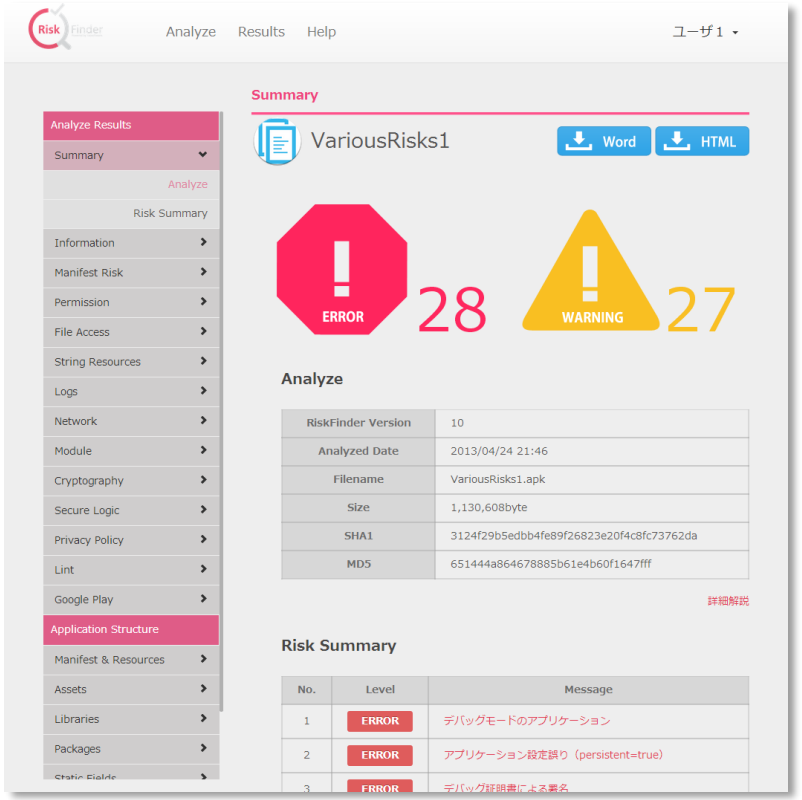
開発者向け

出版社：インプレスジャパン

RiskFinder　アンドロイドアプリの脆弱性発見ツール

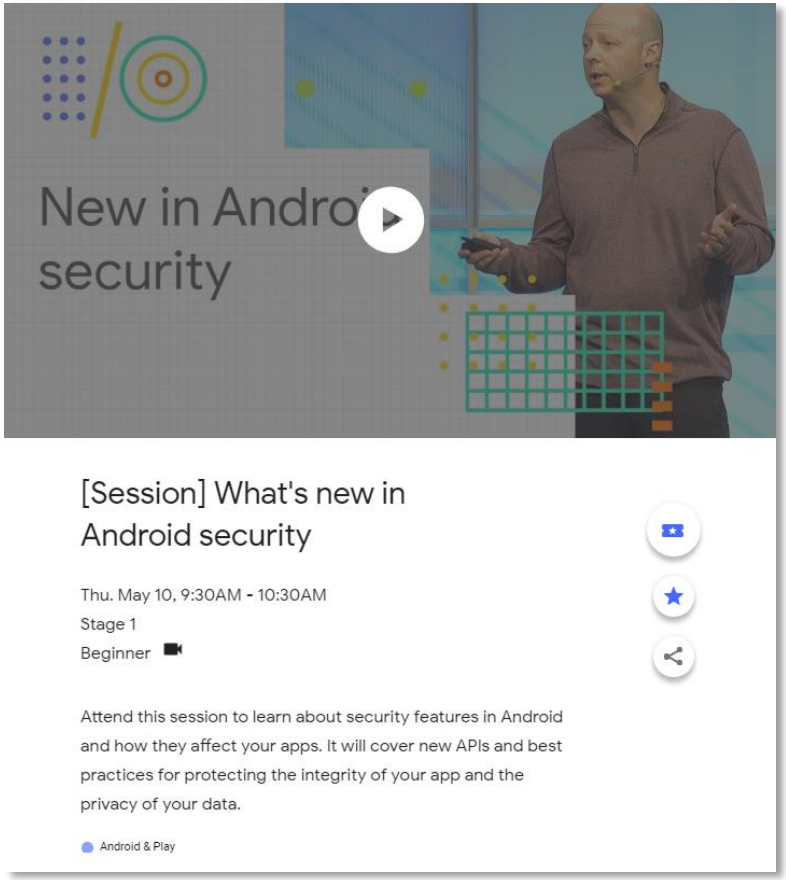
ＡＰＫファイルをアップロードするだけで脆弱性レポートが作成されます。

講演をする中で、
「気を付ける事が沢山あるのは分かった。
でも全てのプログラマが理解するのは
難しい何かいい方法はないか？」
という声があったので作ってみました。



1. **プログラマでなくても使える**
2. **ソースコード不要**
3. **ウエブサービス型**
4. **脆弱性以外も検出**

What’s new in Android Security Session



Google I/O 2018

5月10日(木) 9:30～

Youtube:
https://www.youtube.com/watch?time_continue=12&v=r54roADX2MI

Androidのセキュリティ戦略

Android Security Strategy

Android Security Strategy

Google Play Protect

Defend against Internet-borne threats

User experience that offers security CCC (comprehension, control, confidence)

Platform Engineering

Feature dev

OS hardening, leverage HW

SDLC

Vulnerability management

Full cycle, e.g. fuzzing and security reviews, for AOSP and partners

Androidの３つのセキュリティ戦略

1. Google Play Protect
    1. マルウエアからの保護
    2. モバイルセキュリティサービス
2. Platform Engineering
    1. OSのセキュリティ強化、ハードウエア活用
    2. SELinux,セキュアブート、暗号化…
3. SDLC (Software Development Life Cycle)
    1. セキュリティテスト
    2. セキュリティパッチ（毎月）

セキュリティパッチOEM義務付け

毎月セキュリティパッチを出しているが、OEM端末等は総てが適用されるわけではない

Androidの定例セキュリティパッチをOEMに義務付け

今後は多くの端末でセキュリティパッチが適用される事となる

Project Treble

- デバイスベンダーがOSを効率的にアップデートできるようにする仕組み（Android Oから）
- https://source.android.com/devices/architecture/treble

Androidは安全である

Androidのセキュリティは他のモバイルプラットフォームと同じレベルである。

by Google

3つの証明？

1 Transparency（透明性）

2 Measurability（測定可能性）

- 1．PHAをインストールした発生率を見る
- 2．エクスプロイトコード価格を見る

1. Transparency

AndroidOSが安全である理由1

Transparency（透明性）＝オープン

Androidプラットフォーム

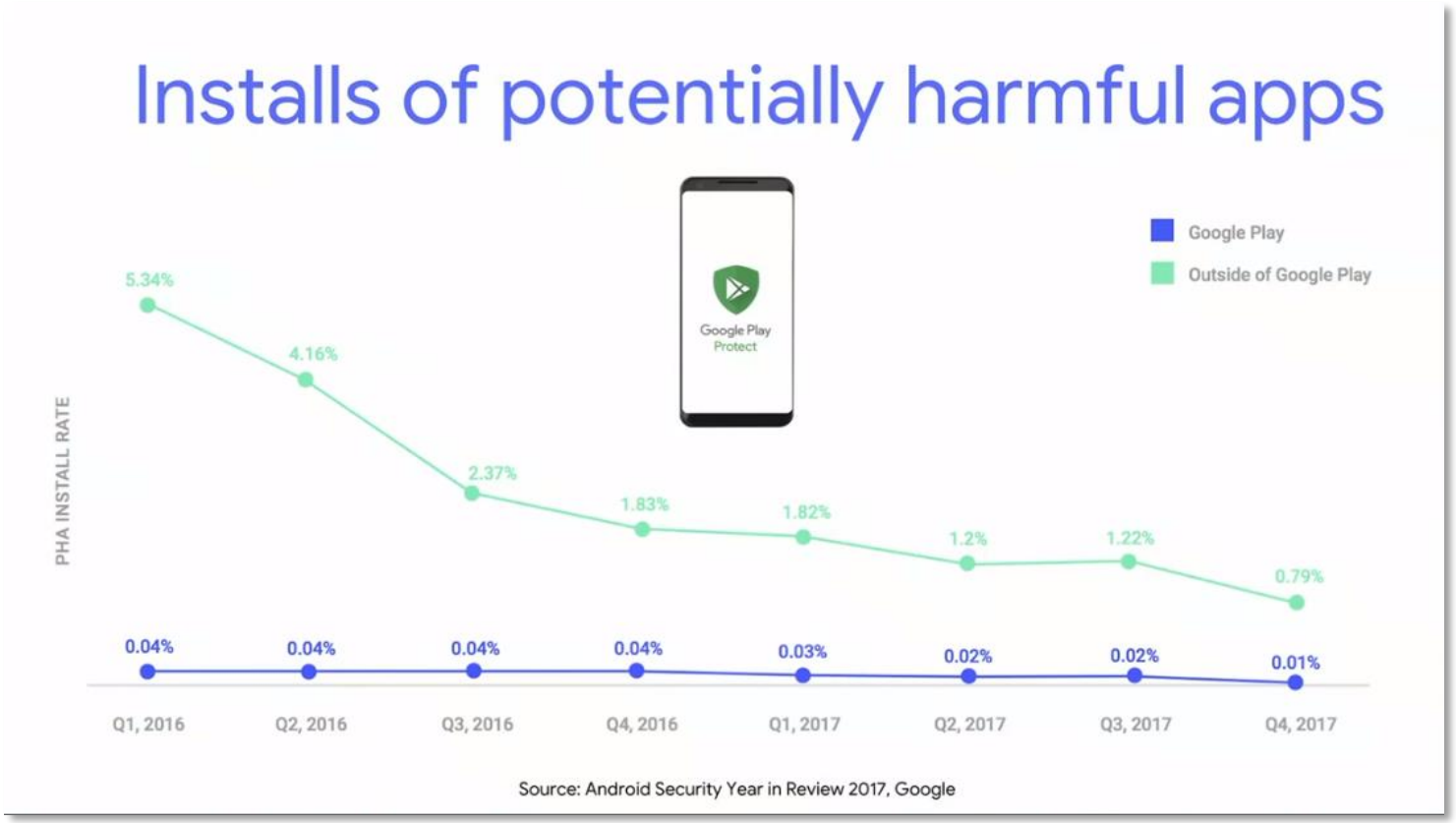
- オープンプラットフォームはセキュリティに貢献する
- 防御側は、数千人のGoole社員＋アーム＋インテル＋クアルコム＋ブロードコム＋Linuxコミュニティ＋学術研究コミュニティ

クローズドプラットフォーム

- 防御側は１社の従業員

2. PHAインストール率

AndroidOSが安全である理由2



PHA（Potentially Harmful Applications）：
マルウエアまたは潜在的に有害なアプリケーション

緑：Google Play以外からのPHAインストール

青：Google PlayからのPHAインストール

Google Play Protect

端末やアプリに不正な動作がないかを定期的に確認するアプリ。

- https://www.android.com/intl/ja_jp/play-protect/
- Google I/O 2017発表（2017/7/22頃提供）
- 設定→Google→セキュリティ→Google Play Protect→ON

PHAアプリのインストール率

GooglePlayからPHAアプリがインストールされた割合

- 0.2から0.1に減っている２倍減っていることになる。
  （グラフをもっと減っているように見えるように加工すればいいのに）

Google Play以外からのPHAアプリインストール率

- Google Protectがクライアントで動作するようになったので明らかに減少している

機械学習を使ってマルウエアの６０%が検知されている

PHAをインストールする確率は雷にあたるのと同じ確率(ﾟДﾟ)

Tip:Android Developer Blog: 2018/5/24

- Keeping 2 billion Android devices safe with machine learning
- 機械学習してるとか言ってるだけで特に有益な情報が出ているわけではない
- https://android-developers.googleblog.com/2018/05/keeping-2-billion-android-devices-safe.html

3．エクスプロイトコード価格

AndroidOSが安全である理由3

Exploit（エクスプロイト）コード

- 脆弱性検証するための実証コード
- 脆弱性を利用した悪意ある行為のために書かれたコード

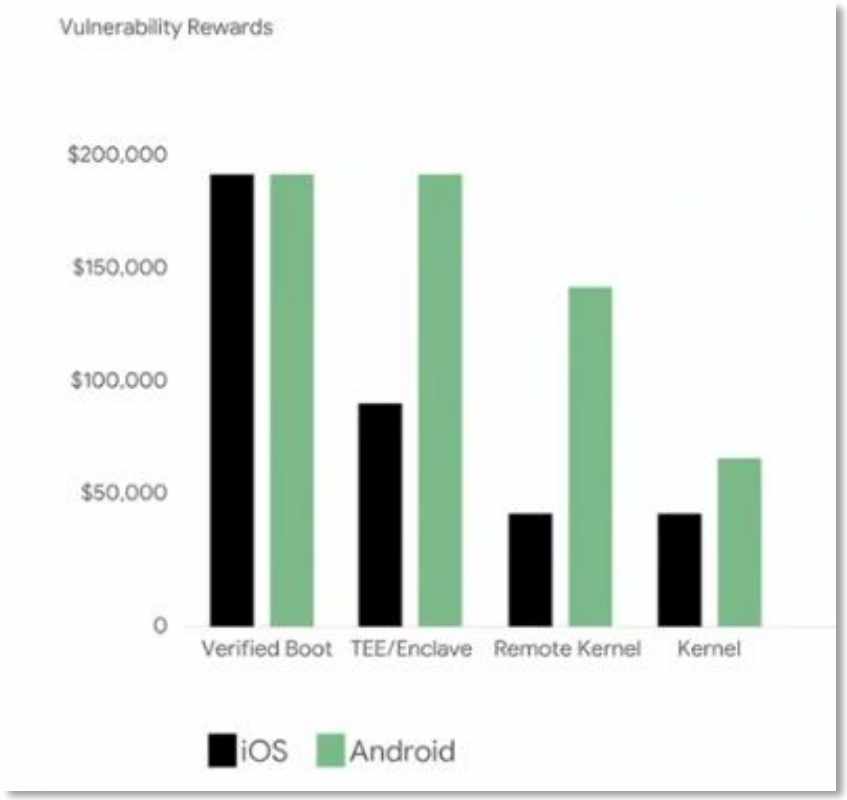
エクスプロイトコード価格

- 売買されている。
- 攻撃性が高いものほど高価
- 攻撃が困難な物ほど高価

※Androidのエクスプロイトコードは高価→安全であるとGoogleは言っている。

- 言いたいことはわかるけど、ちょっと強引かなと言う気もする。

企業のバグ発見報酬プログラム



緑：Android

黒：iOS

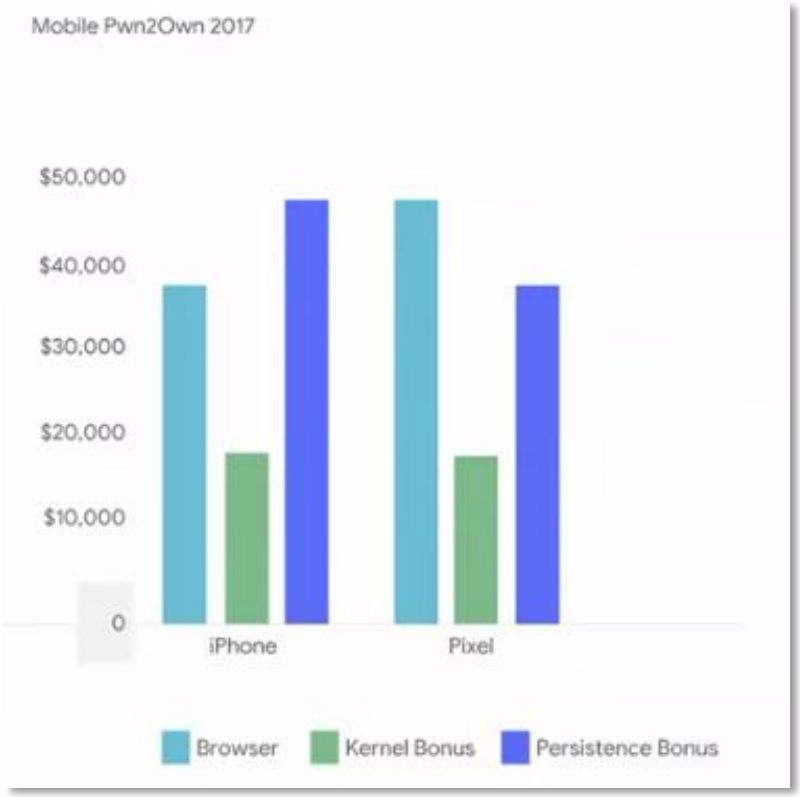
Androidの方が高額を支払っている

- Verified Boot
- TEE/Enclave
- Remote Kernel
- Kernel

Android Security Rewards

- セキュリティバグを報告すると賞金が貰える
- https://www.google.com/about/appsecurity/android-rewards/

ハッキングコンテスト



Mobile Pwn2Own 2017(賞金付き脆弱性発見コンテスト)

AndroidとiPhoneはほぼ同額

アンダーグラウンドマーケット

アンダーグラウンドマーケットでの価格も、他のプラットフォームと同じかそれ以上の価格が観測されている。

よって安全である！

1．Transparency

2．PHAをインストールした発生率

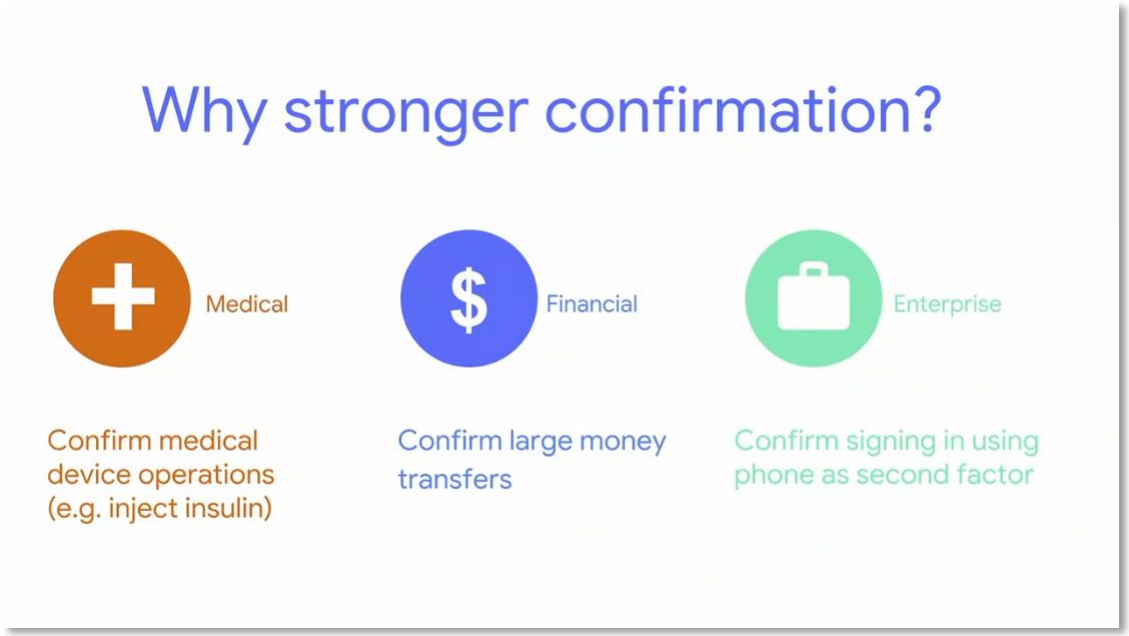
3．エクスプロイトコード価格

Androidのセキュリティは他のモバイルプラットフォームと同じレベルにある。by Google

Android P

Android Protected Confirmation

Protected Confirmation



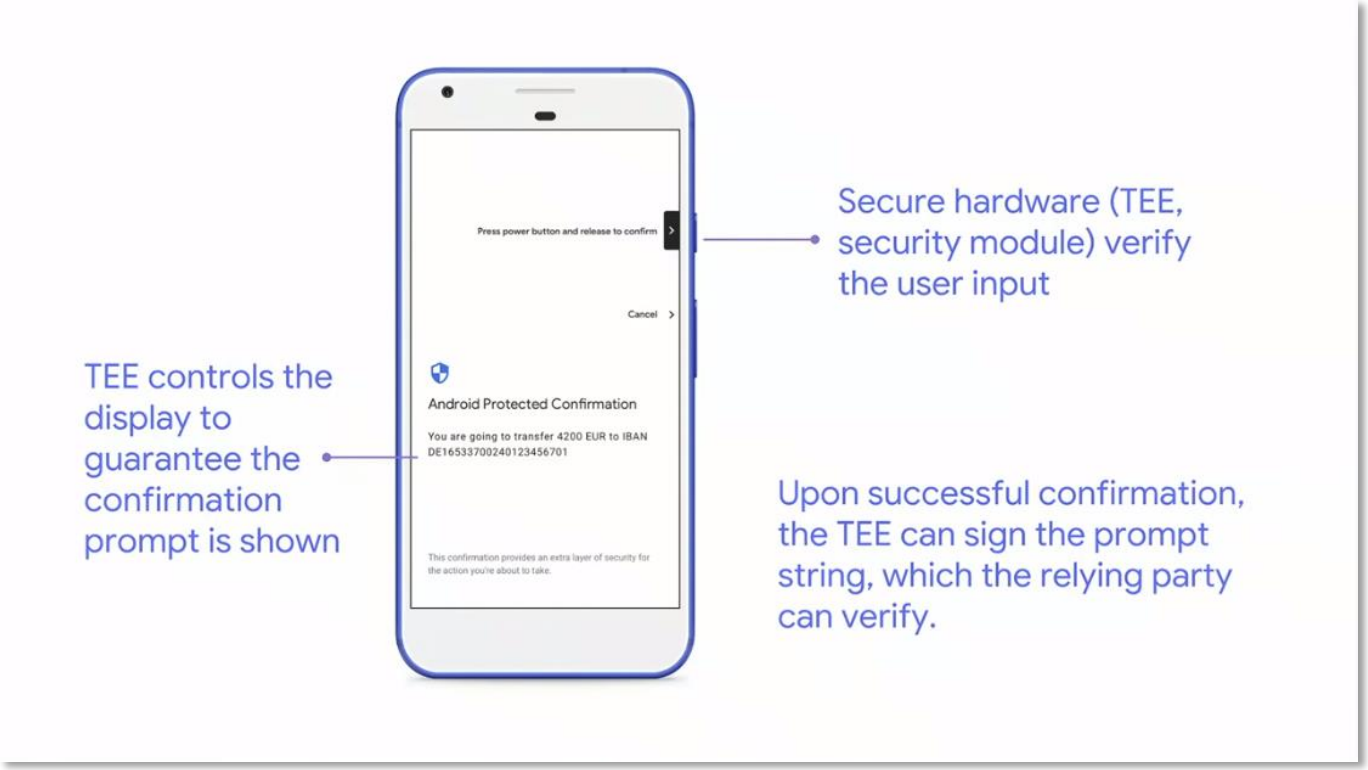
スマートフォンのセキュリティは良くなっているが

- スマートフォンから投票はしない
- インシュリンのような生命に重要な医療機器をプログラムしない
- パスポート機能もない

医療、金融、企業なので安心して使えるようにさらなるセキュリティ機能として

Protected Confirmation機能を作成した

Confirmation Prompt動作



Android P には、Trusted Execution Environment（TEE）を使用して任意のプロンプト文字列がユーザーに確実に表示および確認されるようにする Android Protected Confirmation が導入されています。TEE では、ユーザーが正常に確認した場合にのみ、アプリで検証できるプロンプト文字列が署名されます。

その他

- Android Protection ConfirmationがサポートされているAndroidP端末が必要
- ソフトウェアではなくハードウェアボタンによる確認
- 主要OSでは初めてAPIを実現
- ConfirmationDialogはConfirmationPromptに名前変更（Dp1→Dp2）

  https://developer.android.com/sdk/api_diff/p-dp2-incr/changes/pkg_android.security#ConfirmationDialog

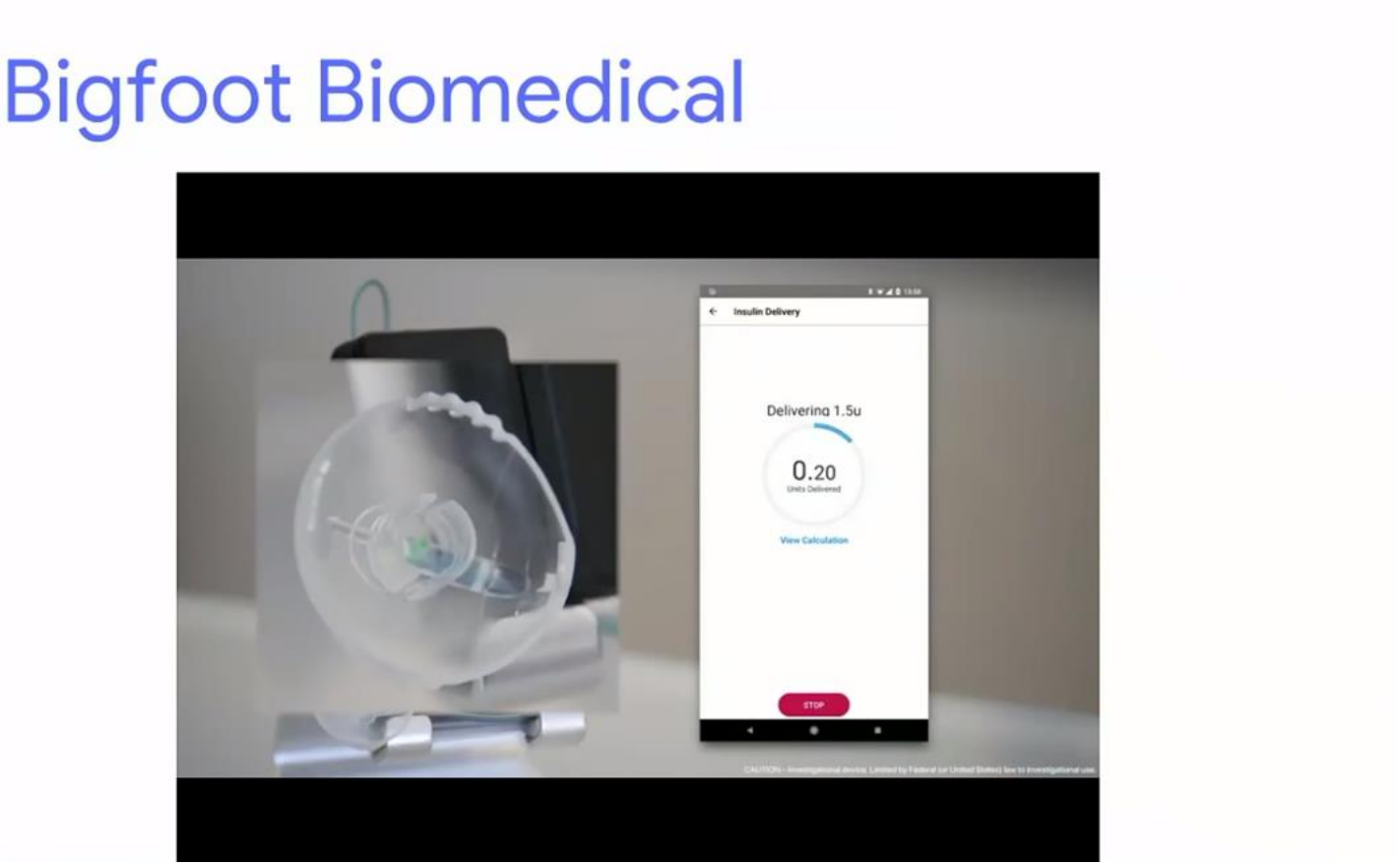
Confirmation Prompt

```
// Create and use a key that requires confirmation

keyPairGenerator.initialize(KeyGenParameterSpec.Builder(
        alias,
        KeyProperties.PURPOSE_SIGN or KeyProperties.PURPOSE_VERIFY)
    .setDigests(KeyProperties.DIGEST_SHA512)
    .setUserConfirmationRequired(true)
    .build())
val keyPair = keyPairGenerator.generateKeyPair()

val dialog = ConfirmationDialog.Builder()
        .setPromptText("Send $10,000 to Greg?")
        .setExtraData(myExtraData)
        .build(context)
dialog.presentPrompt(threadReceivingCallback, callback)
```

Require user confirmation before key can be used

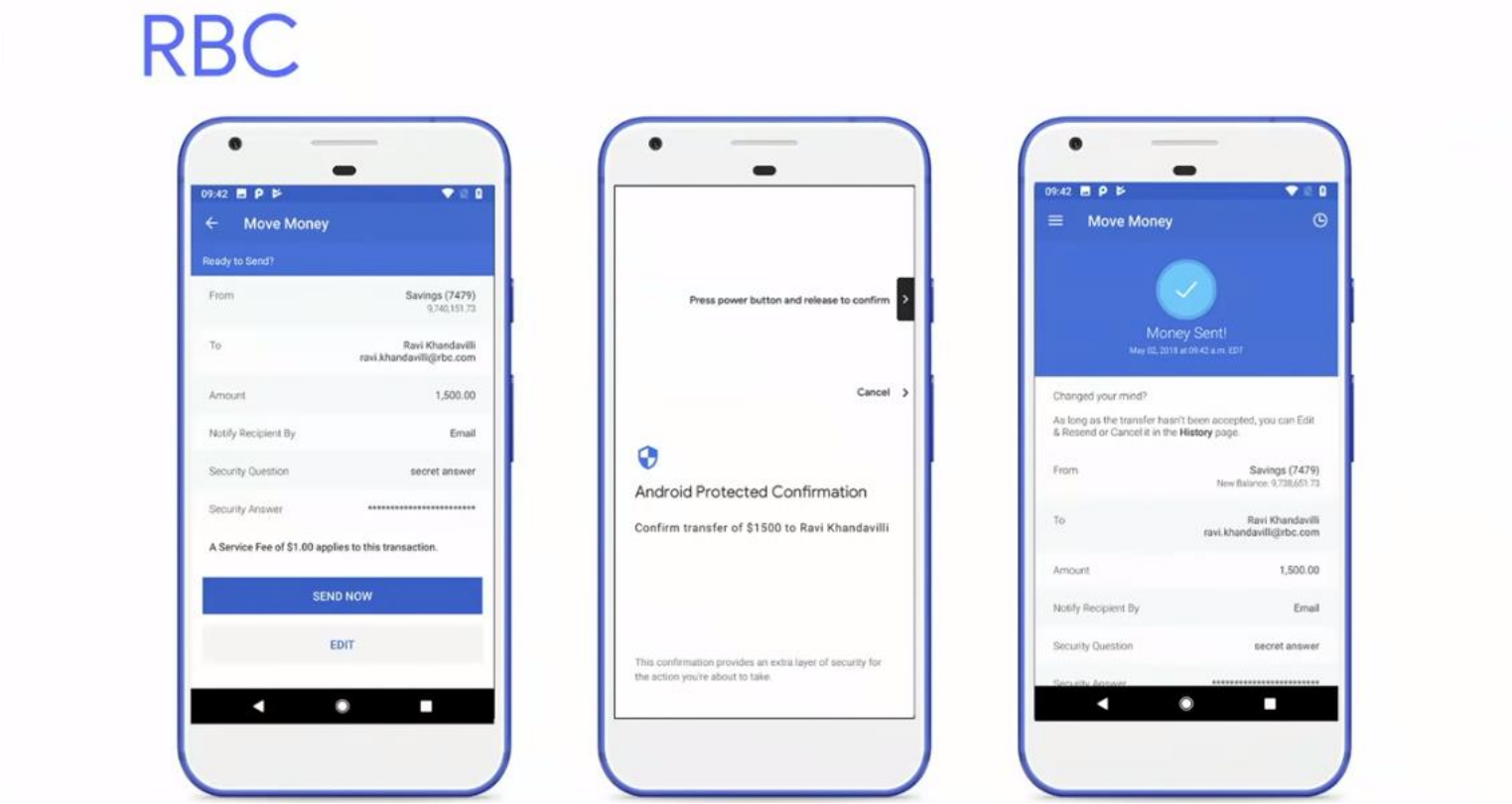
Show confirmation prompt

適用例:Bigfoot biomedical



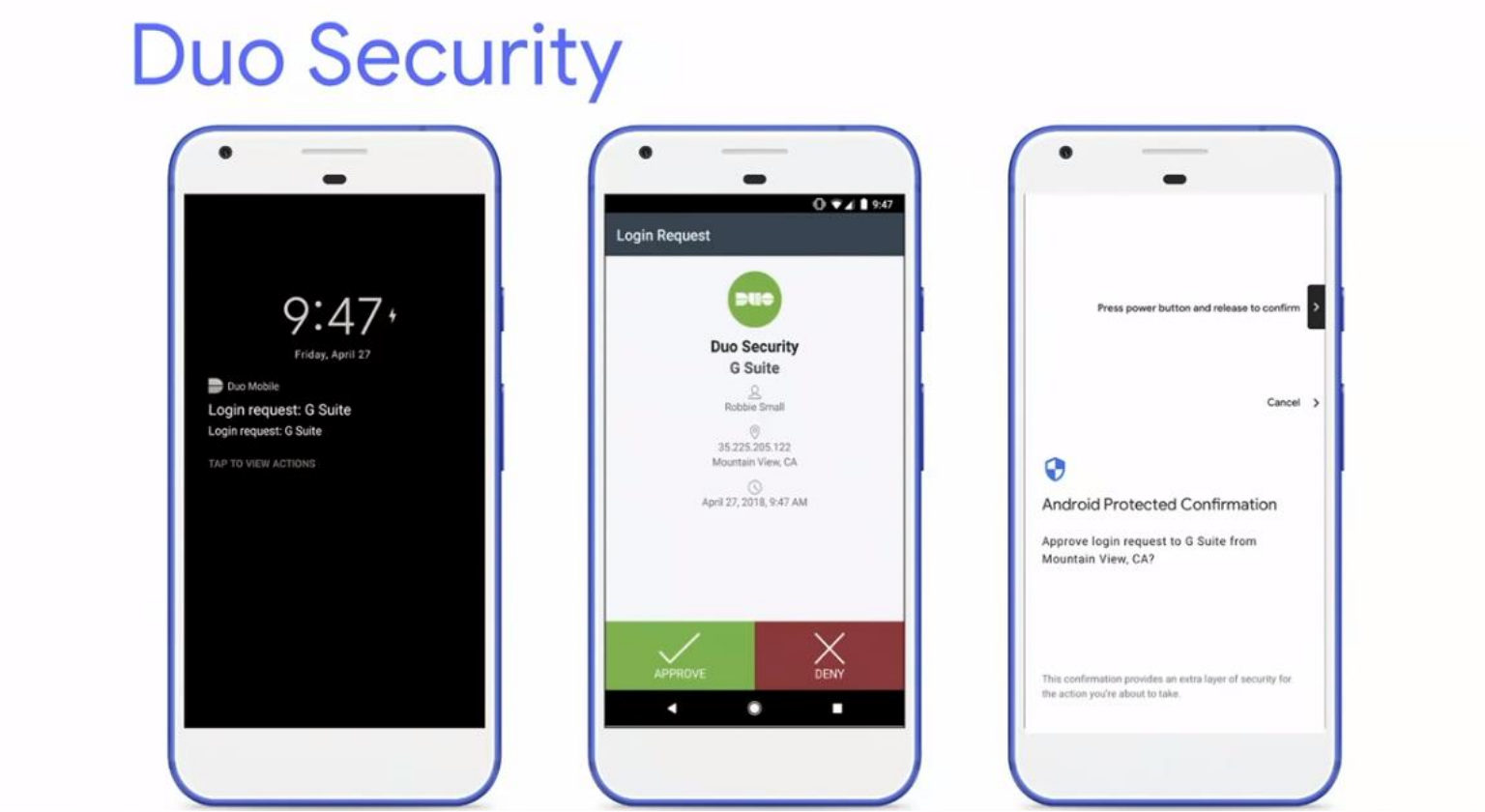
インシュリンの投与（糖尿病）

適用例:RBC



お金を1500ドル転送

適用例: Duo Security



https://duo.com/

ランチパートナー

Launch Partners

Requires Supported Android P Device

StrongBox

About Android KeyStore

Android KeyStoreとは、暗号で使用する鍵を安全に保存する場所及び機能

鍵は暗号化された上でKeyStoreに格納されるため、Root端末であっても鍵を取り出す事はできない。

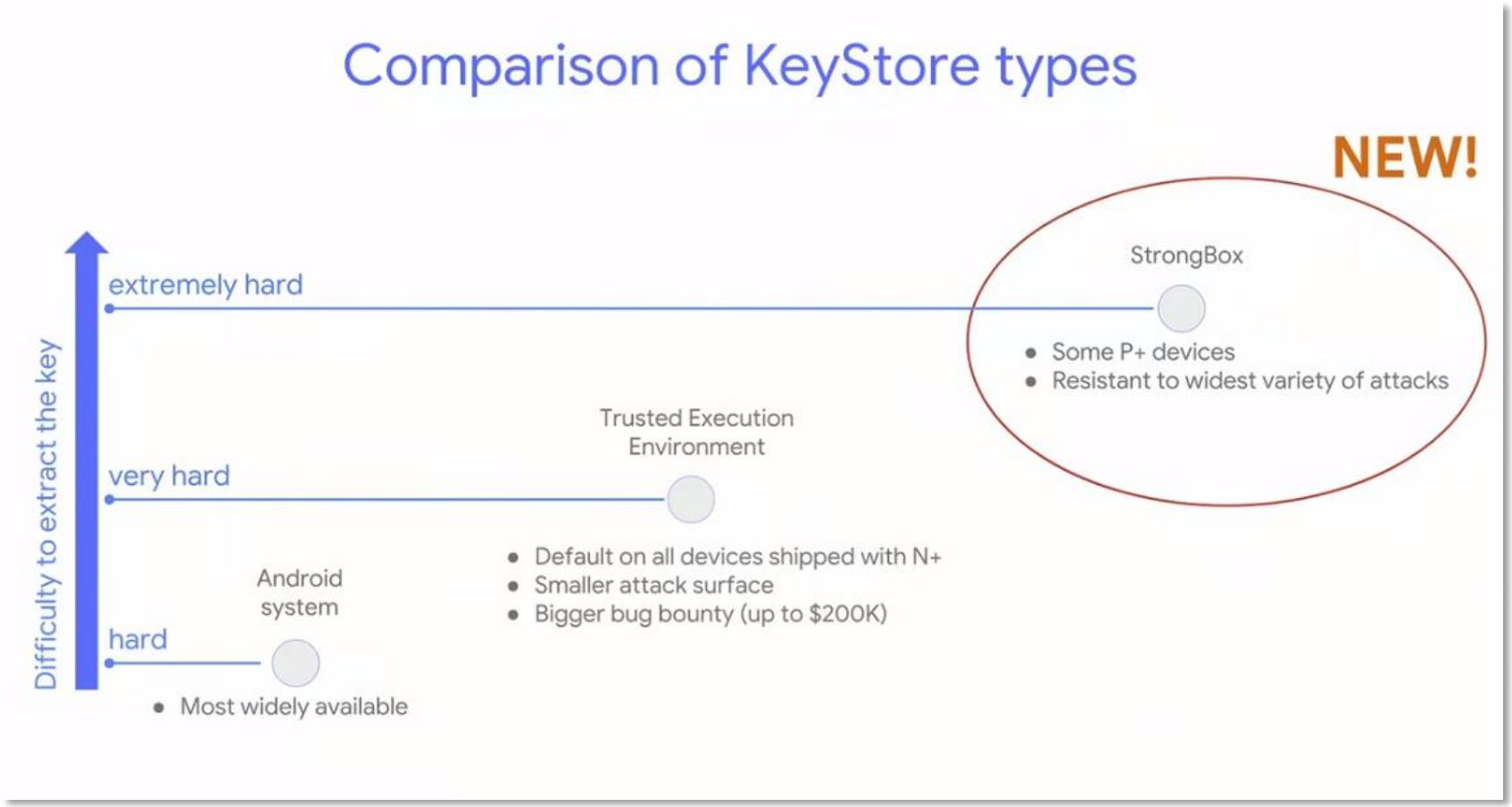
Android4.3

- 導入されたが遅いかつ、鍵の生存期間が不明瞭だったため利用に耐えれなかった

Android6.0　リニューアル

- OSに指紋認証機能追加
- ユーザのロック画面知識ファクタ（Lock Screen Knowledge Factor :LSKF)とAndroid　KeyStoreの関連機能追加
- AES（共通鍵暗号）が利用可能
- 一般的な利用に耐えられるようになった。

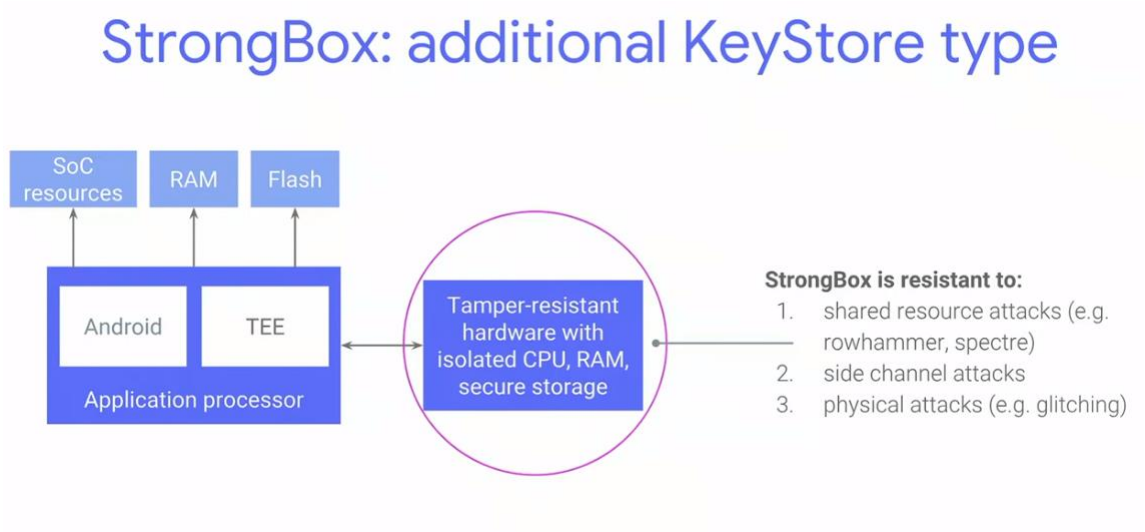
AndroidKeyStoreタイプ



StrongBox:OSはP以上で対応端末のみ利用可能

3つのタイプのキーストア

1. Android System
   - OSがサポート（ソフトウエア）
2. TEE(Trusted Execution Environent)
   - N以上で総ての端末がサポート
3. Strong Box (New!)
   - P以上の端末でサポート（必須ではない）
   - 改ざんに強いハードウエアで実現
   - hasSystemFeature(FEATURE_STRONGBOX_KEYSTORE)



https://developer.android.com/preview/features/security#hardware-security-module

Android P がインストールされたサポート対象の端末では、ハードウェアのセキュリティ モジュール内にある Keymaster HAL の実装である *StrongBox Keymaster* を使用できます。 このモジュールには、独自の CPU、安全なストレージ、真性乱数ジェネレータのほか、パッケージの改ざんやアプリの不正なサイドローディングを防ぐ追加のメカニズムが含まれています。 システムでは、StrongBox Keymaster に格納されている鍵をチェックするときに、信頼できる実行環境（TEE）で鍵の整合性を保証します。

StrongBoxの利用するソースコード

```
// Set StrongBox when generating the key

val kpg = KeyPairGenerator.getInstance(
        KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore")

kpg.initialize(KeyGenParameterSpec.Builder(
        alias,
        KeyProperties.PURPOSE_SIGN or KeyProperties.PURPOSE_VERIFY)
        .setDigests(KeyProperties.DIGEST_SHA512)
        .setIsStrongBoxBacked(true)
        .build())

val kp = kpg.generateKeyPair()
```

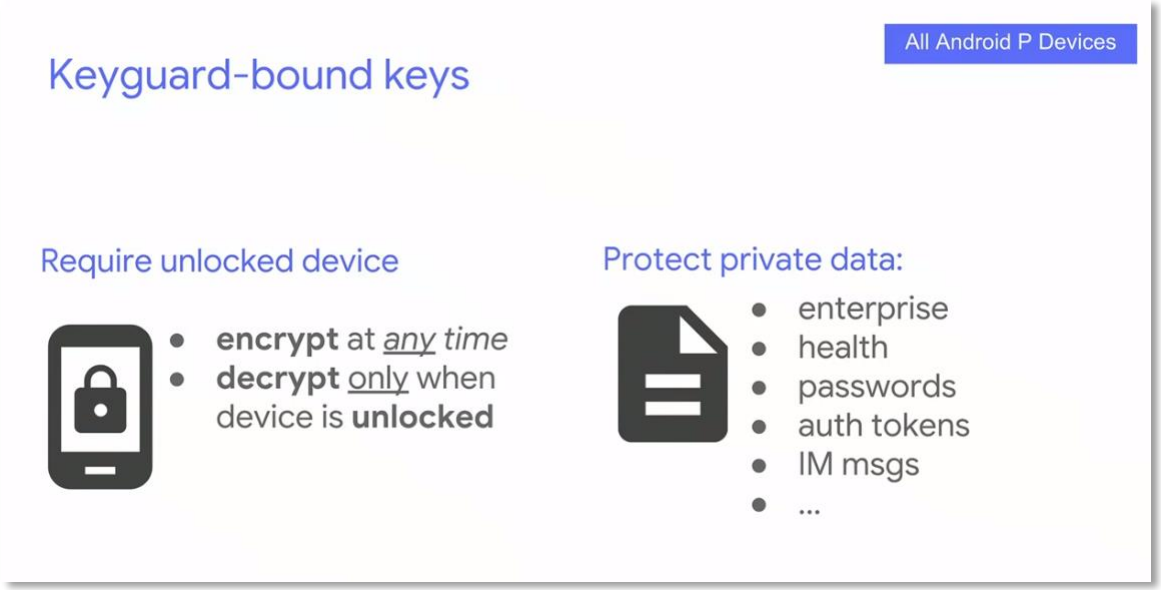
Use StrongBox

setIsStrongBoxBacked(true)を設定

端末がStrongBox非対応の場合はStrongBoxUnavailableExceptionとなる

Keyguard-bound keys

Keyguard-bound keys



端末がロックされていない状態では復号化できなくする。

ロック画面のライフサイクルに関連付けられる

- 暗号化はいつでもできる
- 復号化はアンロックした時のみ可能
- 端末を紛失したり盗まれた時に有効

Keyguard-bound keys　ソース

```
// Using keyguard-bound keys

val kg = KeyGenerator.getInstance(KEY_ALGORITHM_AES, ...)
kg.init(
  KeyGenParameterSpec.Builder("k", PURPOSE_ENCRYPT or PURPOSE_DECRYPT)
   .setBlockModes(BLOCK_MODE_GCM)
   .setEncryptionPaddings(ENCRYPTION_PADDING_NONE)
   .setUnlockedDeviceRequired(true)
   .build())
val key = kg.generateKey()
...
cipher.init(Cipher.ENCRYPT_MODE, key)
...
cipher.init(Cipher.DECRYPT_MODE, key)
```

The key can only be used to decrypt when the device is unlocked

Initialize a cipher to encrypt sensitive data

Initialize a cipher to decrypt sensitive data when unlocked

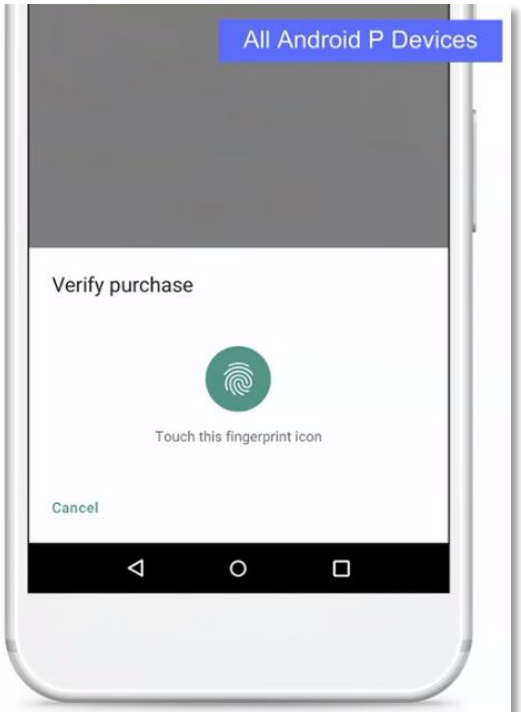
setUnlockedDeviceRequired (true)を設定

Biometric Prompt

Biometric Prompt

BiometricPrompt

- Replacement for FingerprintManager
- Chooses the most appropriate biometric
  - fingerprint, face, iris
- System-wide consistent experience
- Support library version coming



FingerprintManagerに代わる物

- 指紋認証だけでなく、顔認証、虹彩認証、他利用
- USE_FINGERPRINTパミッション非推奨、USE_BIOMETRICパミッション新規追加
- 指紋認証UIがサードパーティアプリによってバラバラで指紋認証処理に一貫性がなくユーザの利便性が悪かったので、標準UIを用意
- PとP以下と一つのコードで記載できるように、サポートライブラリが用意される予定

FingerPrint系クラス

FingerPrintDialogクラス

- Android P DP１で追加

  https://developer.android.com/sdk/api_diff/p-dp1/changes/classes_index_additions

- Android P DP2で削除

  https://developer.android.com/sdk/api_diff/p-dp2-incr/changes/classes_index_removals

FingerPrintManagerクラス

- API Level23（Android 6.0:M)で追加
- API Level28 (Android P)で非推奨

BiometricPrompt

- 生体認証機能が存在するかの確認
- hasSystemFeature(FEATURE_FINGERPRINT)
- hasSystemFeature(FEATURE_IRIS)
- hasSystemFeature(FEATURE_FACE)

BiometricPrompt ソースコード

```
// Using BiometricPrompt

val prompt = BiometricPrompt.Builder(this)
    .setTitle("Verify purchase")
    .setSubtitle("...")
    .setDescription("...")
    .setNegativeButton("Cancel", executor, listener)
    .build()

prompt.authenticate(
    cryptoObject, cancellationSignal, executor, callback);
```

Use a crypto object so user auth properly gates data access

USE_BIOMETRICパミッション追加

USE_BIOMETRIC

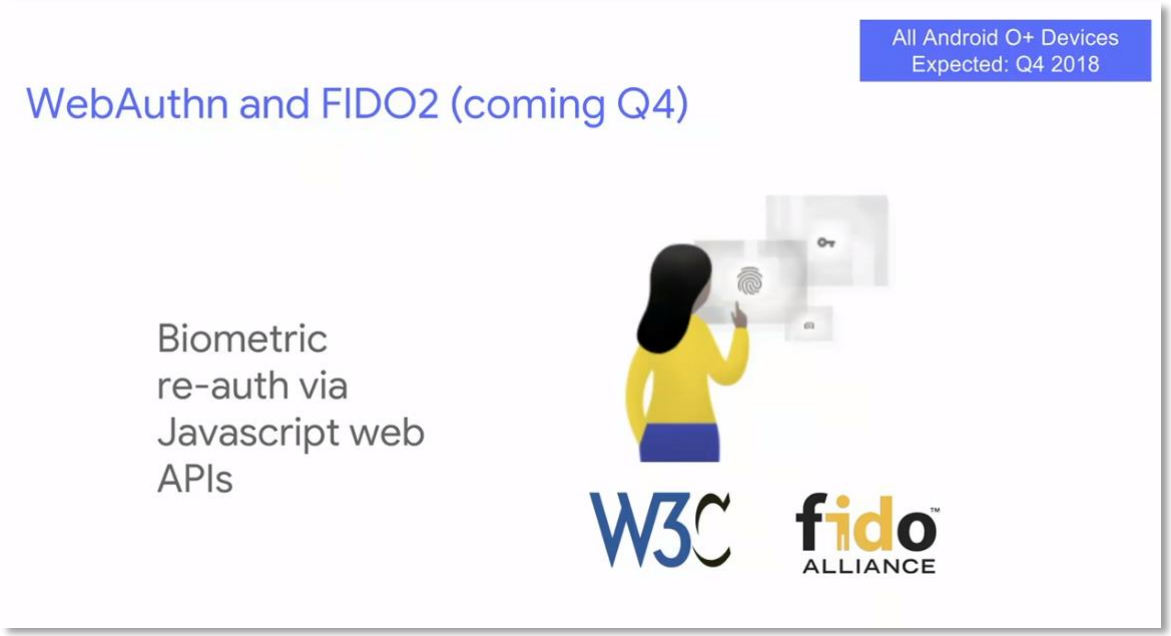
- 端末が持っている総ての生体認証に対応したパミッション
- API Level28(AndroidP)で追加
- protection Level Normal

USE_FINGERPRINT

- 非推奨
- API Level23で追加、28で非推奨
- Protection Level Normal

WebAuthn and FIDO2

About Web Authn(Web Authentication)



- 現在Webサービス上で認証にパスワードを使用しているが、フィッシング詐欺、パスワードの使いまわし問題、脆弱性によるパスワード流出問題がある。
- FIDOアライアンスとW3CがWEB認証の新しいプロトコルとなる、「WebAuthn」を作成（2018/4/10:現在勧告候補）
- ウエブサイトにログインする時、ロックスクリーンや生体認証を利用してウエブサイトへの認証を行う。
- 対応にはWebサイトとブラウザが対応している必要がある。ブラウザは、Gooogle（Q4),Mozilla,Microsoftサポートを表明

パスワードなしプロトコルイメージ

ユーザ登録時

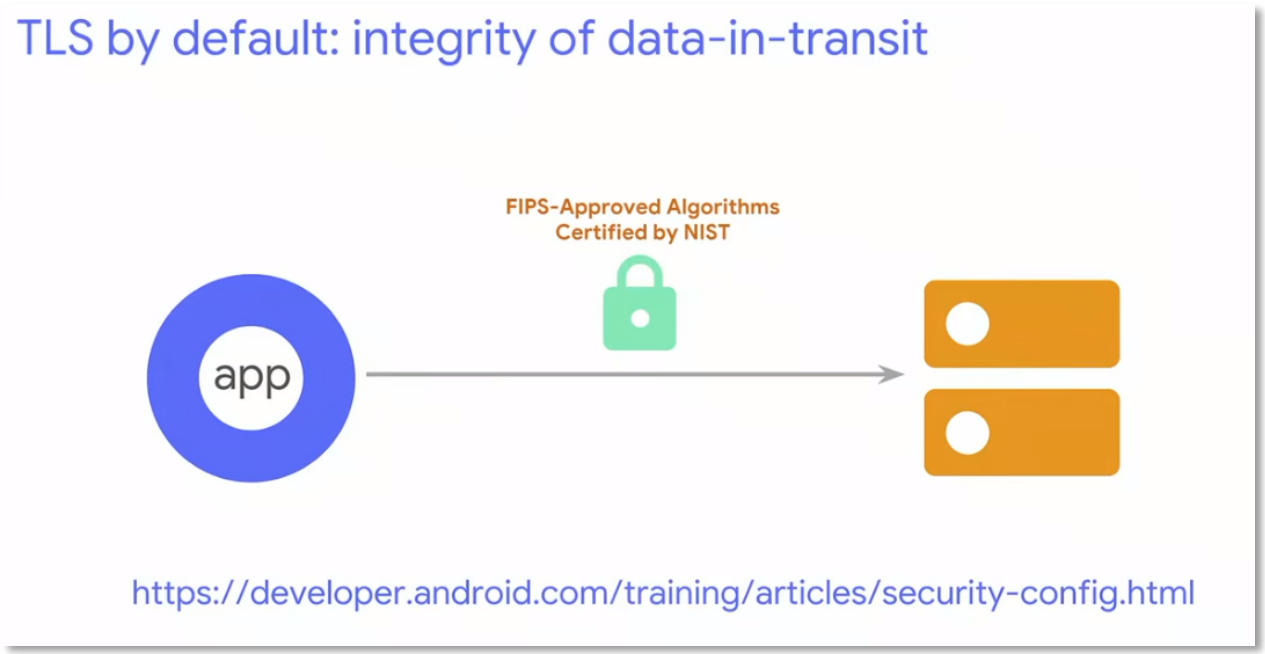
- クライアントはサーバ毎に公開鍵と秘密鍵を作成する
- 公開鍵をサーバーに送り、サーバーは公開鍵を保存する

ログイン時

- サーバはクライアントに、適当なメッセージを送る
- クライアントはメッセージを秘密鍵で暗号化しサーバに送る
- サーバは公開会で復号化して、最初に送ったメッセージと同じならログインとする。

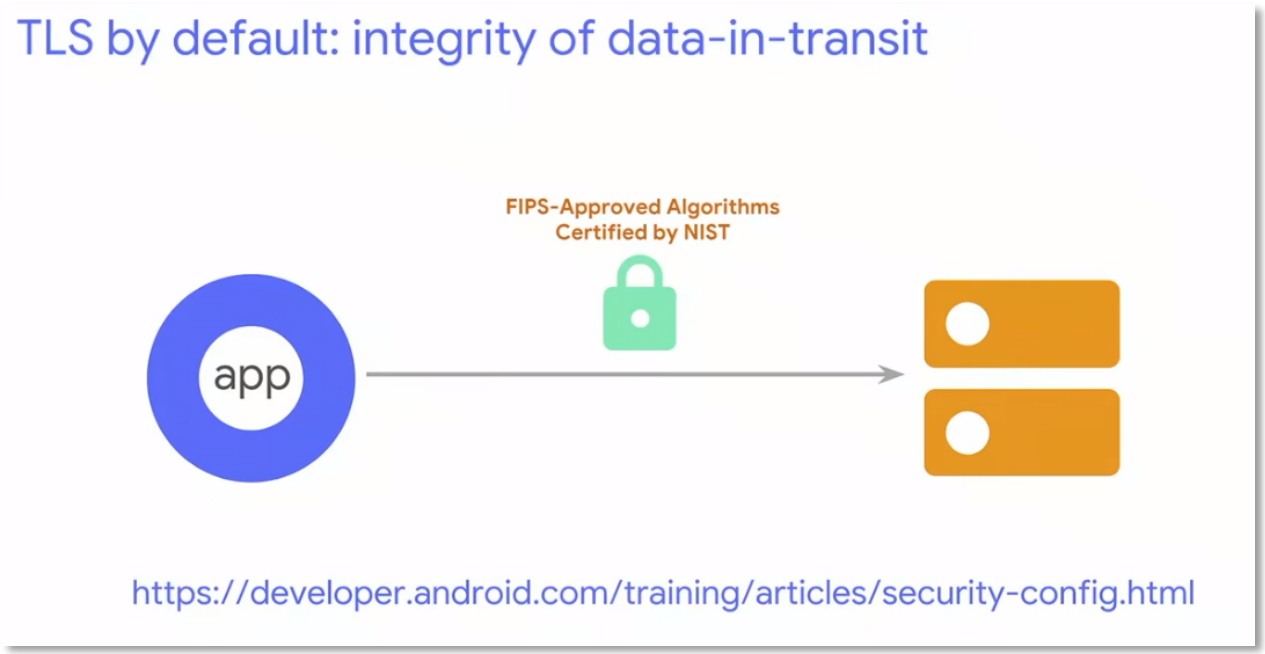
TLS by Default

TLS by Default



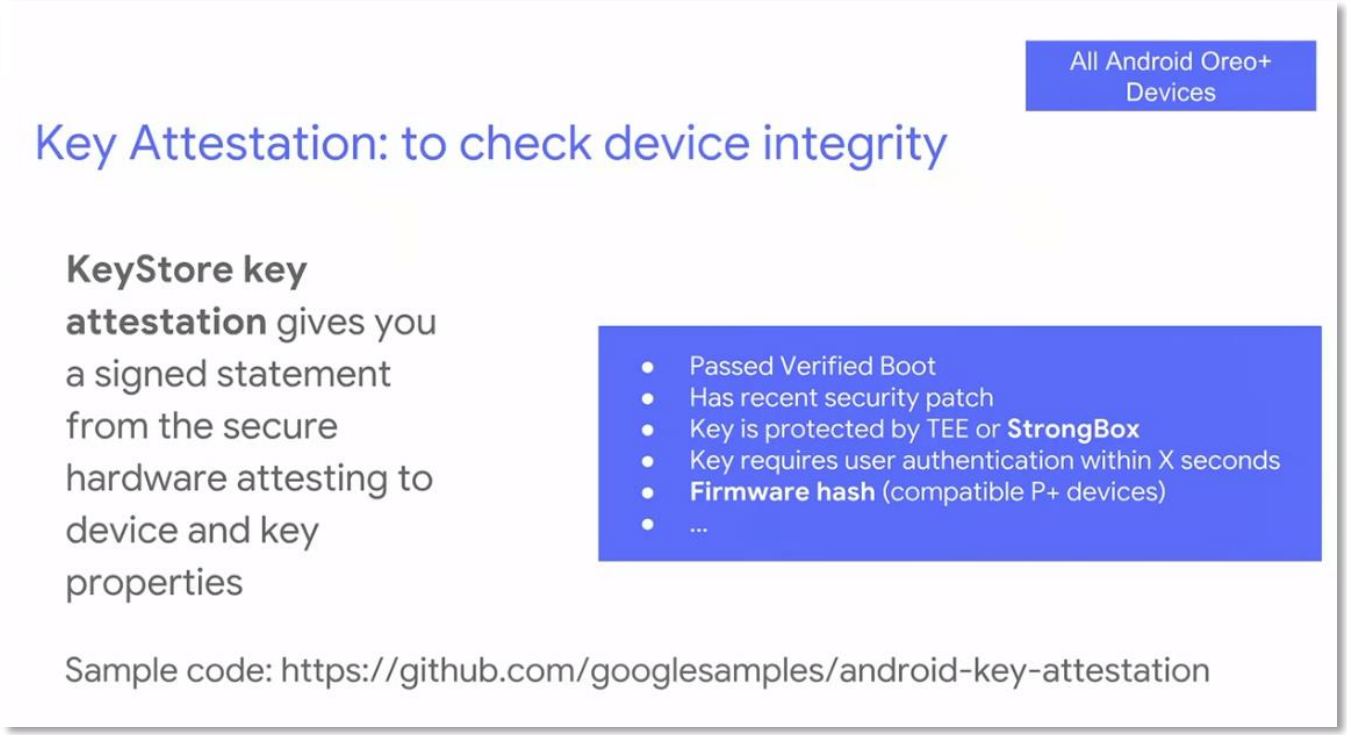
- TargtSDKがP以上の時Network Security Configurationファイルで指定するuseCleartextTrafficのデフォルトがtrueからfalseに変更
- 平文通信を禁止するので、通信をしているライブラリを利用している場合は対応済か確認が必要

FIPS対応



- FIPS（フィップス）：Federal Information Procession Standard)アメリカ政府が取り扱う情報機器や情報技術に求められる基準
- Boring SSLがNISTからCAVP署名を受け取ったのでFIPS準拠となった。

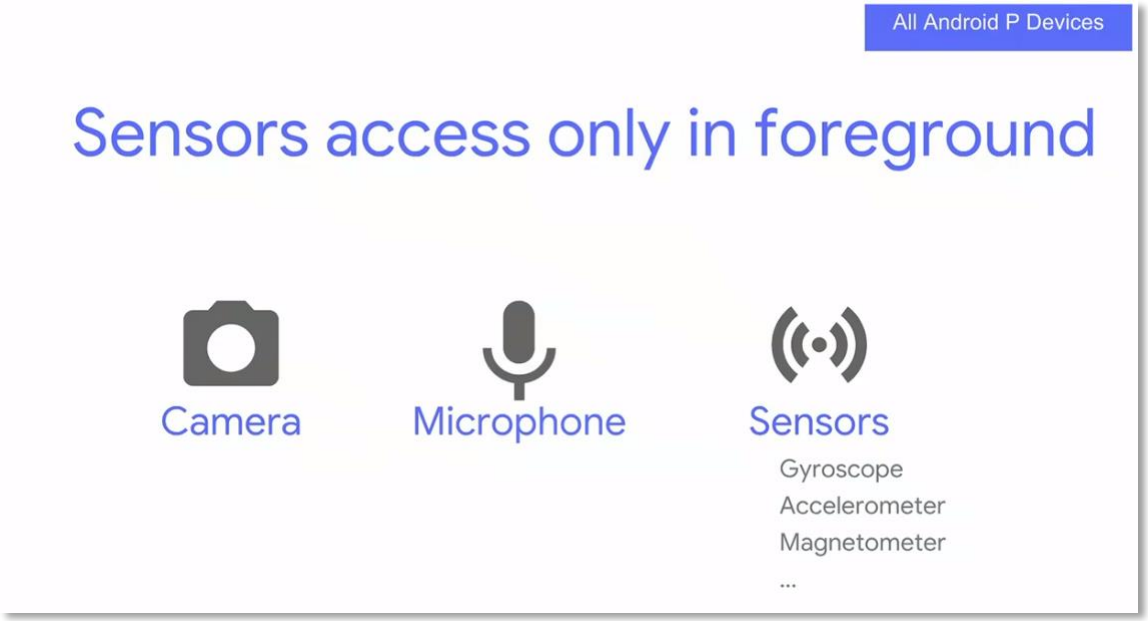
KeyAttestation



- **デバイスが改竄されていないかを判断**
- **Root化の判断に使用可能**

バックグラウンド動作

バックグラウンド盗聴禁止



- **AndroidP端末上では、APIレベルに関係なく、アプリがバックグラウンドでアイドル状態の時に、カメラやマイクロフォン、センサー（加速度センサー、ジャイロセンサー他）からデータを取得できなくなる。**
- **targetSDK等は関係しない。**

フォアグランドサービス　ソース

```
// Foreground service

val notifIntent = Intent(this, ExampleActivity::java.class)
val pendingIntent = PendingIntent.getActivity(this, 0, notifIntent, 0)
val notification =
    Notification.Builder(this, CHANNEL_DEFAULT_IMPORTANCE)
        .setContentTitle("...")
        .setContentText("...")
        .setSmallIcon(icon)
        .setContentIntent(pendingIntent)
        .setTicker("...")
        .build()
startForeground(ONGOING_NOTIFICATION_ID, notification)
```

Start a foreground service with a persistent notification to use sensors

アクセスする場合は、フォアグランドサービスにする。

- **setForgtound()永続的なNotificationが表示される。**
- **AndroidPでFORGROUND_SERVICEパミッションが必要（プロテクションレベル Normal)**

一時的（１分）有効なホワイトリストへのアプリ登録が可能

- **AlarmManager.setAndAllowWhiteIdele()**
- **AlarmManager.setExactAndAllowWhiteIdle()**

DNS over TLS

セキュアなDNS

DNSとは、ドメイン名からIPアドレスを調べる仕組み

DNSは平文通信を行うため（UDP)、中間者攻撃が可能であり盗聴や改ざんが可能。

対策

- **DNS over TLS**
- **DNS over DTLS**
- **DNS over HTTP**
- **DNSSEC(改竄のみ対応)**

DNS over TLS

RFC7858, RFC8310

Google Security Blog

- DNS over TLS support in Android P Developer Preview(April 17, 2018)
- https://developers-jp.googleblog.com/2018/05/dns-over-tls-support-in-android-p.html (日本語)

DNS over TLS動作

「ネットワークとインターネット」 →「詳細設定」→「 プライベート DNS 」（B1→B2で変更）

デフォルトON（Automatic)

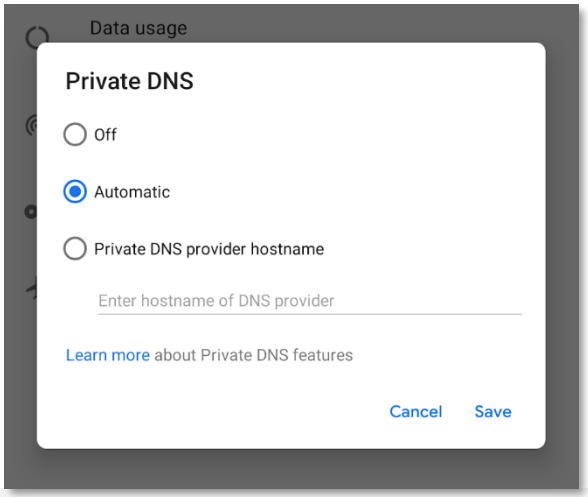
通信するDNSサーバーが、DNS over TLSをサポートしているならDNS over TLSで通信、それ以外なら今でどおり通信する。

Private DNS provider hostname

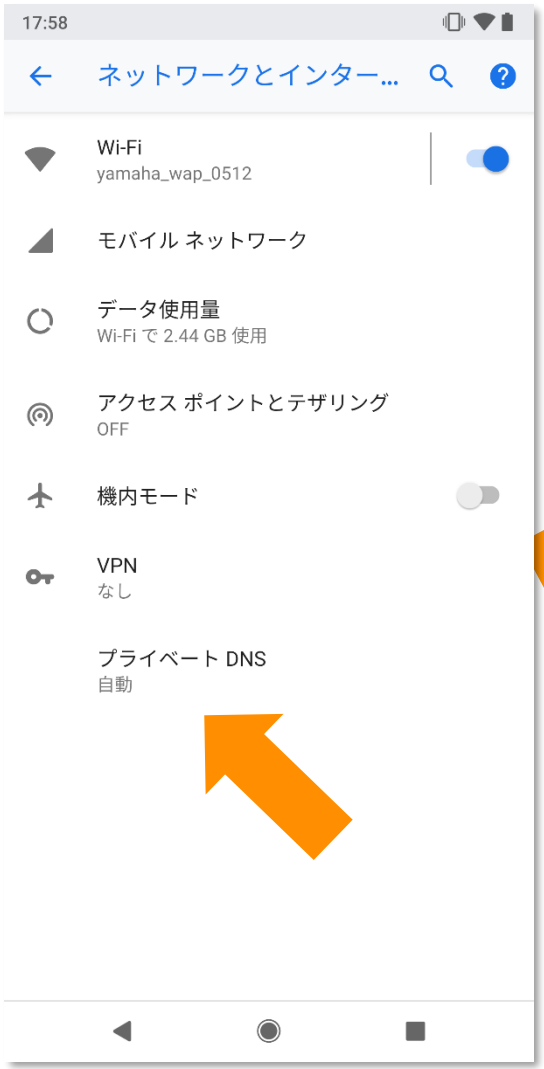
- 常にDNS over TLSを使いたい時に、対応サーバーを指定する使い方か？

追加API

- LinkProperties.isPrivateDnsActive()
- LinkProperties. getPrivateDnsServerName()



Android P Beta 日本語表記

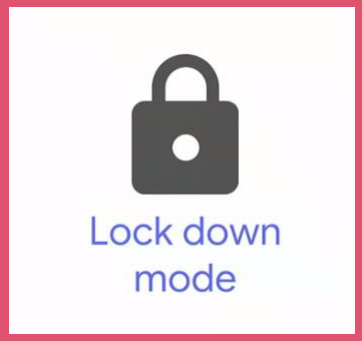


β2で「自動」に治ってました

DNS over HTTP

Google Public DNS(8.8.8.8)

- DNS-over-HTTPS API
- DNS over TLSとは違う
- https://developers.google.com/speed/public-dns/docs/dns-over-https



Lock Down mode

Lock Down Mode

一時的にデバイスにアクセスできなくするモード

セキュリティゲートで端末を人に渡したりする時、警察に端末を押収された、盗難等レアケースではあるが、生体認証、他によって端末にアクセス可能にできるケースがある。（就寝時にも有効かもしれない）

- パスワード,PIN,パターン以外使用できなくなる。
- Smart Lock(身に着けている物、場所、顔認識、音声認識等でロックを外す機能）が無効
- 指紋認証（生体認証）が無効

Lock Down Mode動作

1. 「設定」→「セキュリティと現在地情報」→「ロック画面の設定」→「ロックダウンオプションの表示」
2. 電源長押しでメニューを出して、ロックダウンを入力でロックダウンモード

まとめ

AndroidOS危険危険と言うほどではなくなってきています。

AndroidPのセキュリティ変更点については、まだまだ色々ありますが、大体細かい所です。

タオソフトウェア株式会社　谷口岳

ありがとうございました。