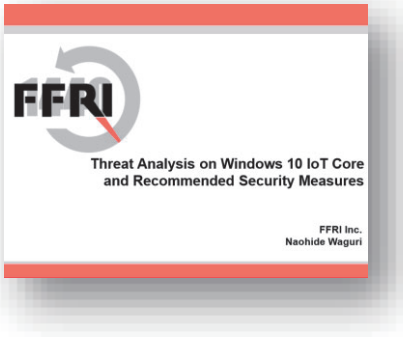
October 21, 2016
CODE BLUE 2016

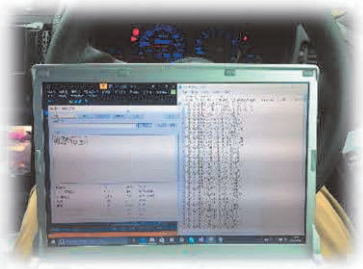
IoTとしての自動車とセキュリティ：リモートサービスのセキュリティ評価とその対策の検討

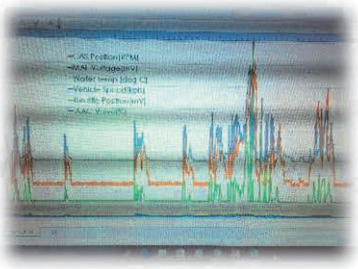
株式会社ＦＦＲＩ

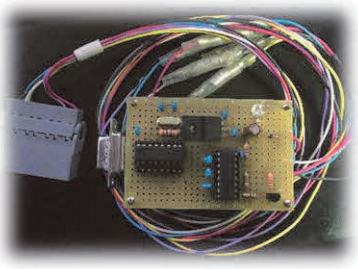
自己紹介

- 元ネットワーク屋。
- 主に自動車関連のセキュリティについて調査・研究。
- CODE BLUE 2015 では Windows 10 IoT Core の話をした。
- 趣味で CAN トランシーバや車の故障診断ツールを作ったり。



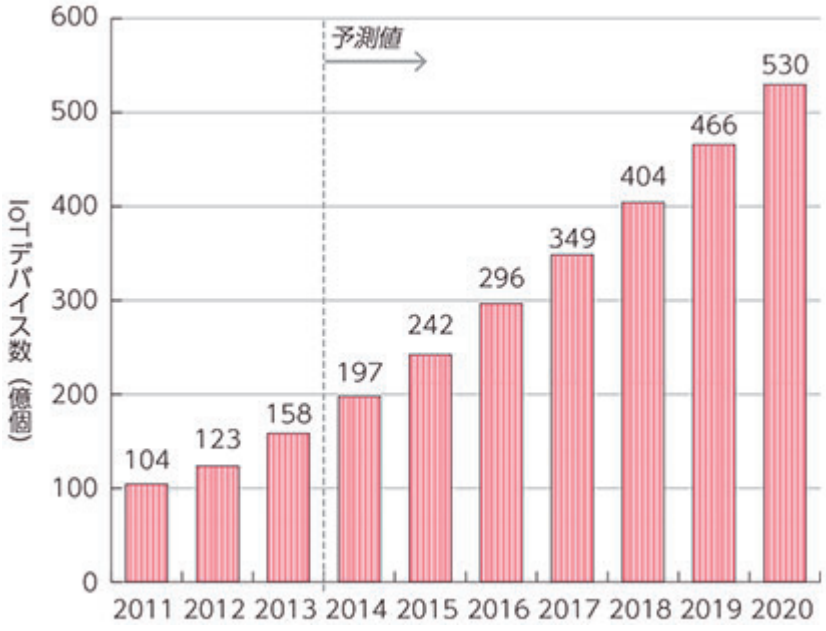


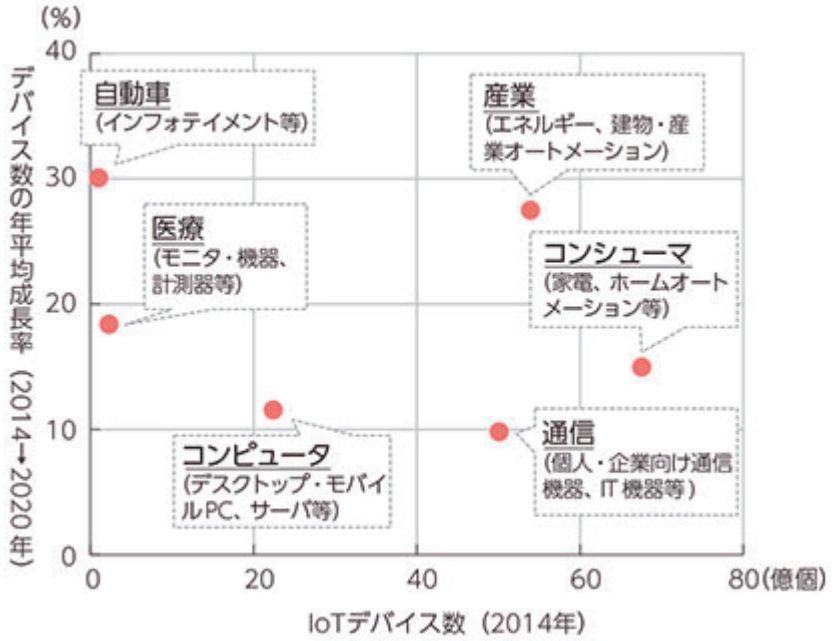




IoT [Internet of Things]

- IoT [*Internet of Things*] という言葉が世の中に出てきて数年、様々な機器がインターネットに繋がる時代。
- 特に**自動車**・産業・医療といった人命に関わる分野・領域における成長率は著しい。





総務省　平成27年度版　情報通信白書より抜粋
（出展）　IHS Technology

自動車セキュリティの現状

- 自動車に対する調査・研究・攻撃のエントリポイントは大きく２つある。

バス上に追加されたデバイスからのメッセージ(インジェクション)

外部との通信を行うシステムの脆弱性悪用

自動車セキュリティの現状

- 自動車に対する調査・研究・攻撃のエントリポイントは大きく２つある。

バス上に追加されたデバイスからのメッセージ(インジェクション)

外部との通信を行うシステムの脆弱性悪用

今回はこの領域に対するお話。

自動車セキュリティの現状

- 自動車に対する調査・研究・攻撃のエントリポイントは大きく２つある。

バス上に追加されたデバイスからのメッセージ(インジェクション)

外部との通信を行うシステムの脆弱性悪用

その前に

今回はこの領域に対するお話。

自動車セキュリティの現状

- 自動車に対する調査・研究・攻撃のエントリポイントは大きく２つある。

バス上に追加されたデバイスからのメッセージ(インジェクション)

こちらについても少し触れます。

今回はこの領域に対するお話。

自動車セキュリティの現状

バス上に追加されたデバイスからのメッセージ(インジェクション)

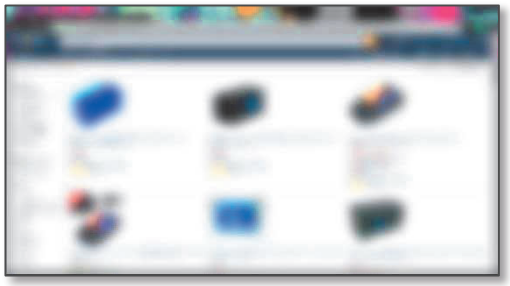
- このエントリポイントは、主に外部（車内）に露出している故障診断用ポートを指す。

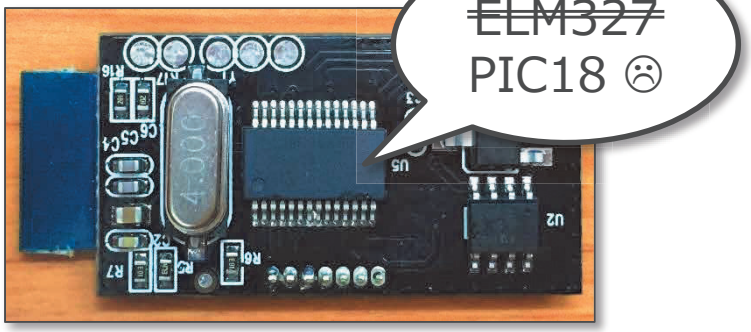
- 昨今、外部故障診断用ポートは単にメンテナンス用途だけではなく、様々な場面での利活用が進んでいる。

OBD-IIドングルなどは車両オーナーが意図的に設置するデバイス。
そのため、各オーナーは接続した**デバイスが脆弱、もしくは悪意のあるものだった場合に車両のセキュリティレベルが下がってしまう可能性**がある事に留意するべき。
（利用する場合は、**信頼できるメーカーや開発元のデバイスの利用**を推奨）

自動車セキュリティの現状

大手ネット通販やオークションで販売されている OBD-II ドングルを分解してみたところ
商品の表記とは異なる部品が利用されている偽物だった。





BluetoothのPINも固定かつ変更が出来ない仕様。

ドングル接続前は物理的に接続しなければメッセージインジェクションが成立しないが
ドングル接続によって脅威のレベルが「物理」から「隣接」に変化する可能性。

OBD-IIドングルなどは車両オーナーが意図的に設置するデバイスである。
そのため、各オーナーは接続したデバイスが悪意のあるものだった場合に
車両のセキュリティレベルが下がってしまう可能性がある事に留意すべきである。
（利用する場合は、**信頼できるメーカーや開発元のデバイスの利用**を推奨）

自動車セキュリティの現状

- 自動車に対する調査・研究・攻撃のエントリポイントは大きく２つある。

バス上に追加されたデバイスからのメッセージ(インジェクション)

外部との通信を行うシステムの脆弱性悪用

ここからが本題。

自動車セキュリティの現状

- インターネットに繋がる自動車の脅威といえば・・・

（出展）https://www.wired.com/wp-content/uploads/2015/07/150701_car_hackers_43-vscocam-photo-1.jpg

自動車セキュリティの現状

- 最近の事例だと・・・

2016-09-19

Car Hacking Research: Remote Attack Tesla Motors

by Keen Security Lab of Tencent

With several months of in-depth research on Tesla Cars, we have discovered multiple security vulnerabilities and successfully implemented remote, aka none physical contact, control on Tesla Model S in both Parking and Driving Mode. It is worth to note that we used an unmodified car with latest firmware to demonstrate the attack.

Following the global industry practice on "responsible disclosure" of product security vulnerabilities, we have reported the technical details of all the vulnerabilities discovered in the research to Tesla. The vulnerabilities have been confirmed by Tesla Product Security Team.

Keen Security Lab appreciates the proactive attitude and efforts of Tesla Security Team, leading by Chris Evans, on responding our vulnerability report and taking actions to fix the issues efficiently. Keen Security Lab is coordinating with Tesla on issue fixing to ensure the driving safety of Tesla users.

As far as we know, this is the first case of remote attack which compromises CAN Bus to achieve remote controls on Tesla cars. We have verified the attack vector on multiple varieties of Tesla Model S. It is reasonable to assume that other Tesla models are affected. Keen Security Lab would like to send out this reminder to all Tesla car owners:

PLEASE DO UPDATE THE FIRMWARE OF YOUR TESLA CAR TO THE LATEST VERSION TO ENSURE THAT THE ISSUES ARE FIXED AND AVOID POTENTIAL DRIVING SAFETY RISKS.

(出展) http://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/

自動車セキュリティの現状

2015

Aug

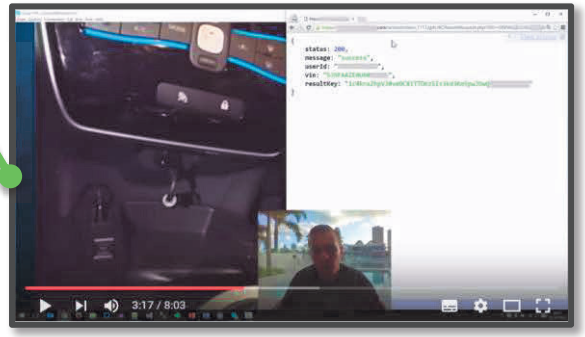
出展:*Samy Kamkar,*
https://www.youtube.com/watch?v=3olXUbS-prU
Drive It Like You Hacked It: New Attacks And Tools to Wirelessly Steal Cars, DEFCON 23

Oct

出展:*Jianhao Liu, Jason Yan,*
https://www.syscan360.org/en/archives/,
Car Hacking: Witness Theory to Scary and Recover From Scare, SyScan360 2015

2016

Feb



出展:*Troy Hunt,*
https://www.youtube.com/watch?v=Nt33m7G_42Q

Jun

出展:*Pen Test Partners LLP,*
https://www.youtube.com/watch?v=NSioTiaX_-Q

モチベーション

- 「自動車もIoTの一部である」と定義付けるにふさわしいシステムに対する脆弱性が2015年以降、相次いで報告されている。

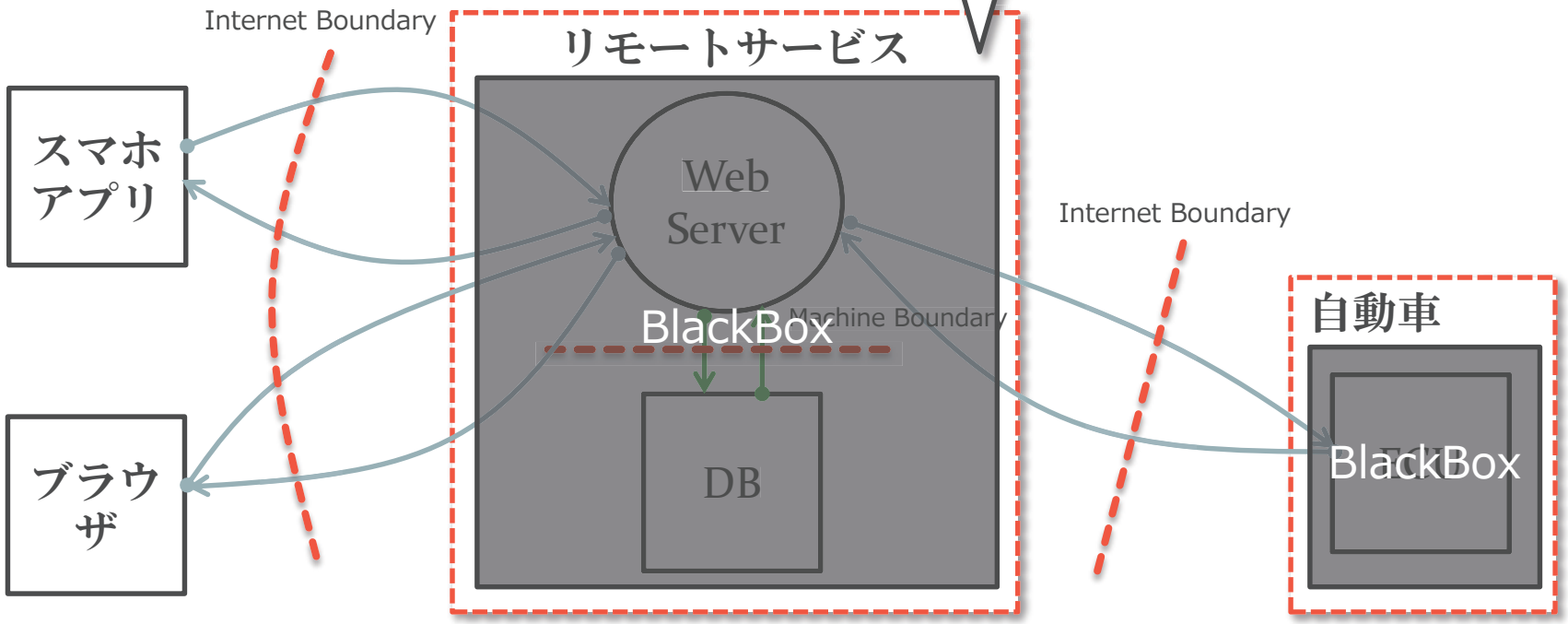
- Jeep Hack の問題に比べ、車両制御がのっとられるレベルの重大な脅威とはいえないが・・・

  ・第三者に個人情報（氏名や住所、電話番号など）を窃取される
  ・第三者に車両の位置情報や走行履歴を見られる
  ・車上荒らし等の目的で勝手にドアを開けられる

  などは、車両オーナーの（情報）資産に対して十分な脅威といえる。

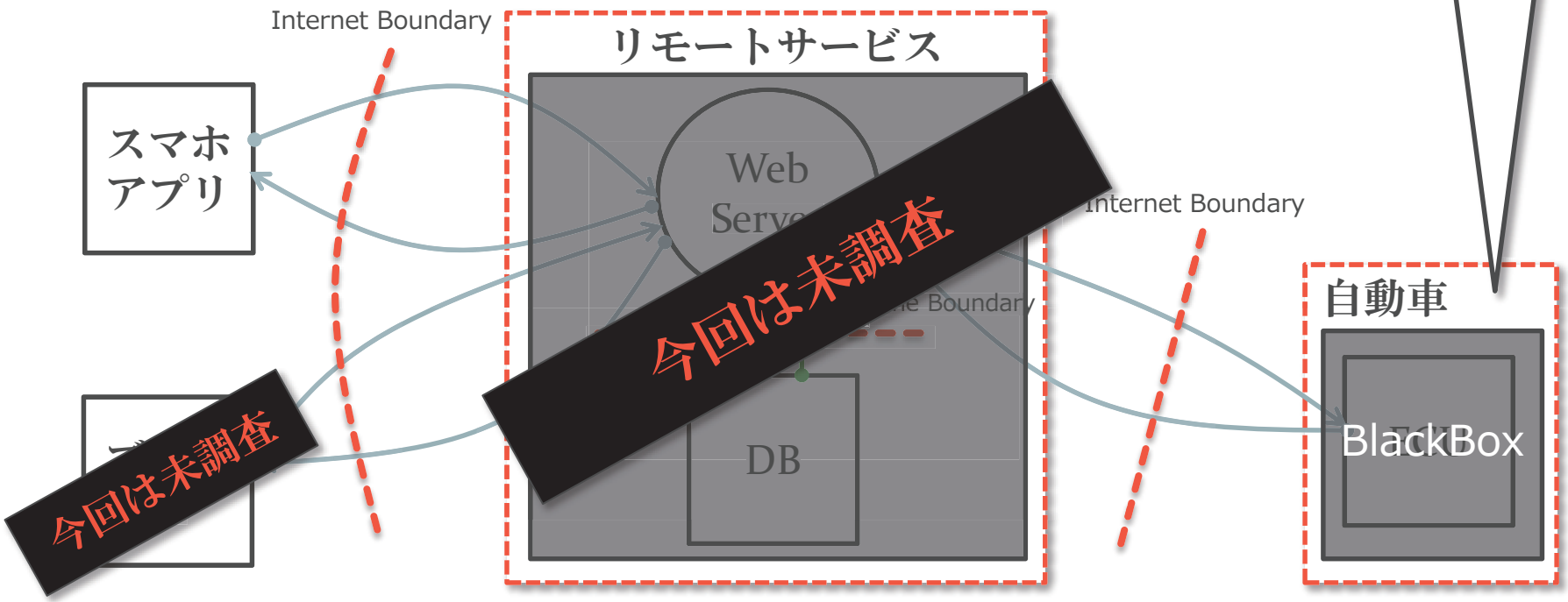
モチベーション

インターネットから先のシステムは各社によって実装は様々であり、基本ブラックボックス。
また、ブラウザからアクセスできる範囲において検証を試みる行為は攻撃とみなされる可能性があるため、許可無く実施することは避けたい。



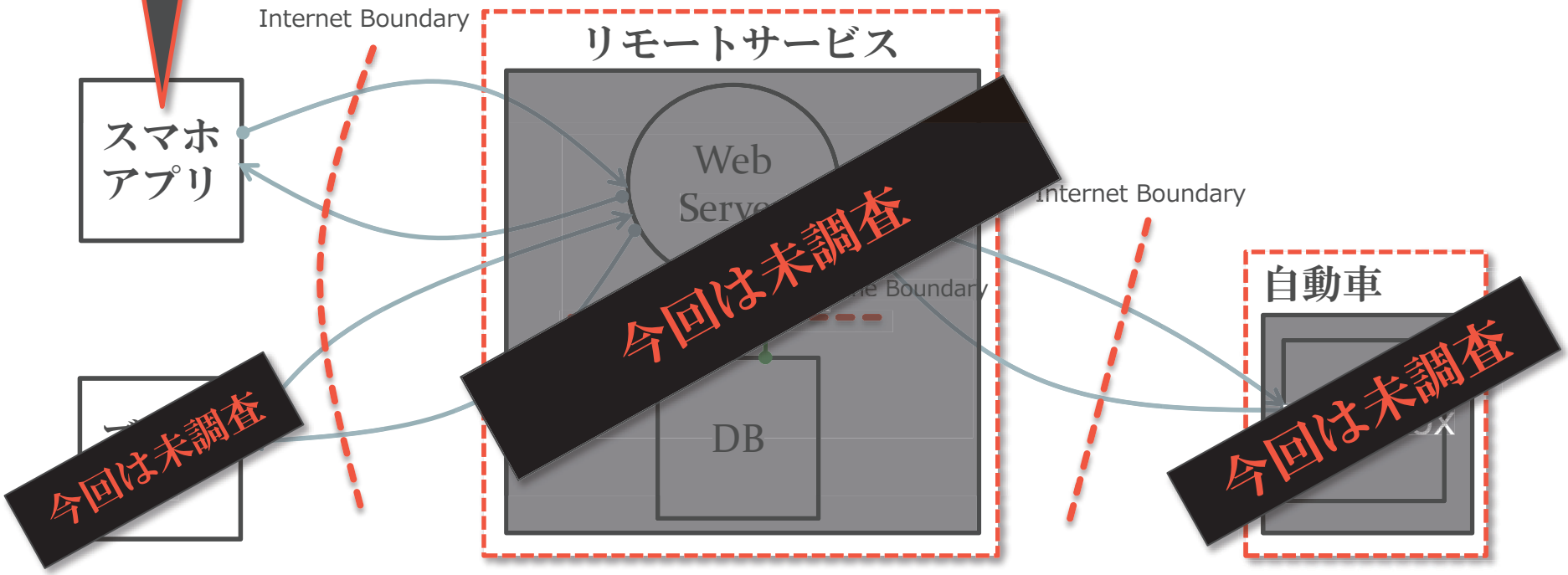
モチベーション

自動車とサービスの通信の検証においては、車両の取得およびサービスへの加入が必要になる。（現状、リモートコントロール系のサービスはオプション加入が必要なケースが多い）
また、検証の技術的難易度も高い。



モチベーション

アプリは容易に入手する事が出来る（攻撃者視点でも同様）。
サービスを構成するエンティティの中でも脆弱性を作りこむ可能性が高い。



調査対象&ゴール

[Phase0] 各OEMが提供しているサービスと連携する Android アプリを対象に収集

[Phase1] AndroBugs を利用して、アプリ毎のレポートを作成

AndroBugs?

- Black Hat EUROPE 2015 で Yu-Cheng Lin 氏が発表した Android アプリの脆弱性スキャナで以下の様な特徴がある：

- **Android アプリの脆弱性を発見する為のツール**
- **Python で記述されていてオープンソース**
- **Android APK に対して静的解析を行う（ソースコードは不要）**
- **効率的にバグを見つけるために、一括解析出来るように設計**
- **新たな機能や脆弱性ベクタの追加が容易**

(出展) https://www.blackhat.com/docs/eu-15/materials/eu-15-Lin-Androbugs-Framework-An-Android-Application-Security-Vulnerability-Scanner.pdf

調査対象&ゴール

[Phase0] 各OEMが提供しているサービスと連携するAndroidアプリを対象に収集

[Phase1] AndroBugsを利用して、アプリ毎のレポートを作成

[Phase2]作成したレポートにもとづいて各アプリを解析

各OEMが提供しているAndroidアプリに対する現状のセキュリティレベルの把握と、今後必要な対策の検討

モバイルアプリの代表的なリスク

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

M4 – Insecure Authentication

OWASP Mobile Top 10 Risks (2016 RC)

M5 – Insufficient Cryptography

M6 – Insecure Authorization

M7 – Client Code Quality

M8 – Code Tampering

M9 – Reverse Engineering

M10 – Extraneous Functionality

https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10

調査対象とした脆弱性リスク

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

M4 – Insecure Authentication

OWASP Mobile Top 10 Risks (2016 RC)

M5 – Insufficient Cryptography

M6 – Insecure Authorization

M7 – Client Code Quality

M8 – Code Tampering

M9 – Reverse Engineering

M10 – Extraneous Functionality

https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10

調査対象とした脆弱性リスク概要

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

アクティビティの公開範囲やフラグメント処理（フラグメントインジェクション対策）などのプラットフォームの機能やセキュリティコントロールの利用方法に問題があるケース。

例：公開アクティビティが PreferenceActivity を継承している場合は、isValidFragment(String fragmentName) をオーバーライドしないとフラグメントインジェクションが行われた際にセキュリティ例外でアプリが強制終了する。

調査対象とした脆弱性リスク概要

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

アクティビティの公開範囲やフラグメント処理（フラグメントインジェクション対策）などのプラットフォームの機能やセキュリ

3つの脆弱性のリスクのうち、AndroBugs で2番目に多く指摘されたが、現状問題にはならない。

場合は、isValidFragment(String fragmentName)をオーバーライドしないとフラグメントインジェクションが行われた際にセキュリティ例外でアプリが強制終了する。

調査対象とした脆弱性リスク概要

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

機微なデータを外部ストレージに保存したり、ログ出力するなどの、データの取り扱いに問題があるケース。

例：送信データに含まれる機微な情報をデバッグ用にログ出力していた。または SharedPreference のインスタンス取得時に MODE_WORLD_READABLE/WRITABNLE を使用して他のアプリからも参照できる状態にしているなど。

調査対象とした脆弱性リスク概要

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

機微なデータを外部ストレージに保存したり、ログ出力するなどの、データの取り扱いに問題があるケース

3つの脆弱性のリスクのうち、AndroBugs による指摘が最も少なかった。
また、調査の結果現状は特に問題ないと判断。

取得時に MODE_WORLD_READABLE/WRITABNLE を使用して他のアプリからも参照できる状態にしているなど。

調査対象とした脆弱性リスク概要

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

SSL通信の実装が不適切であり、結果として中間者攻撃を許してしまうケース。
特にサーバ証明書の検証を省略しているパターンが多く報告されている。

例：ホストネーム（CN）の検証を行わない、独自（空）の TrustManager を実装することで証明書の検証をスキップしているなど。

調査対象とした脆弱性リスク概要

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

SSL通信の実装が不適切であり、結果として中間者攻撃を許してしまうケース。

AndroBugs が最も多く指摘した脆弱性リスクであり、実際に問題のあるアプリも存在した・・・

例：ホストネーム（CN）の検証を行わない、独自（空）の TrustManager を実装することで証明書の検証をスキップしているなど。

実際に問題のあったアプリは？

アプリ名は書けない・・・☹

SSL/TLS 通信の実装に関する問題はモバイルアプリの脆弱性としてはよく届出られるタイプ。

今回見つかった問題例
～Case1. 送信データにユーザ情報の一部を含むHTTP通信～

- 今回確認した内容
  - あるアクティビティ上の WebView に表示するページが HTTP だった。
  - ユーザ情報の一部を POST するページだった為、クリアテキストでその情報がネットワーク上に流れる事になる。
  - 同じホスト上に存在するほかのページは HTTPS となっていたため何らかのポリシーに基いているとも考えられるが・・・。

今回見つかった問題例
～Case1. 送信データにユーザ情報の一部を含むHTTP通信～

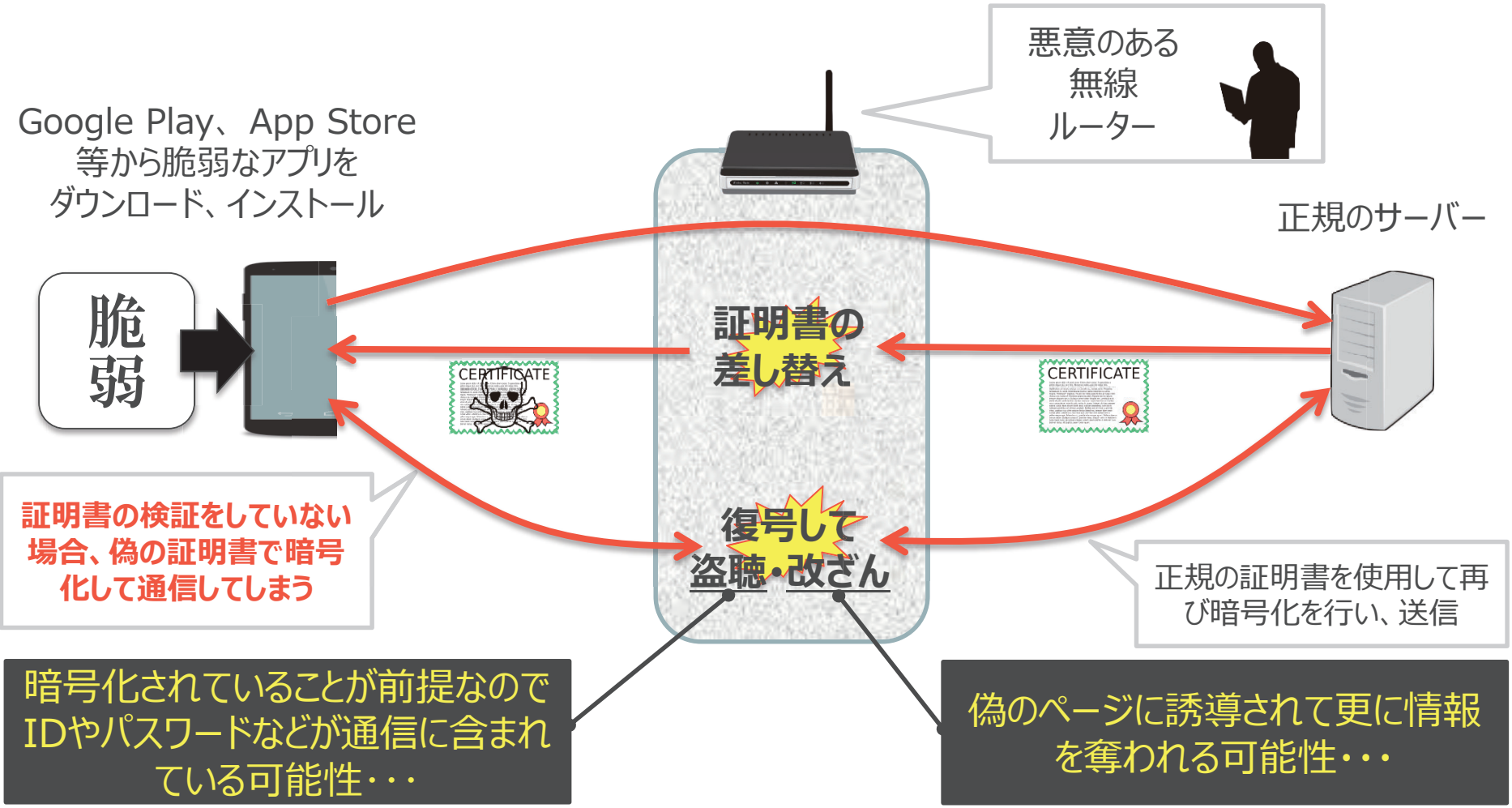
今回見つかった問題例
～Case2. サーバ証明書の検証不備～

https://www.ipa.go.jp/about/press/20140919_1.html
http://www.kb.cert.org/vuls/id/582497

今回見つかった問題例
～Case2. サーバ証明書の検証不備～

そもそも、サーバ証明書の検証不備ってどんな問題？
想定されるリスクは？

今回見つかった問題例
～Case2. サーバ証明書の検証不備～



今回見つかった問題例 ～Case2. サーバ証明書の検証不備～

- 今回確認した内容
  - ALLOW_ALL_HOSTNAME_VERIFIERが利用されているためホスト名の検証が行われない。
  - 空の TrustManager を利用しているため証明書の検証が行われない。
  - WebView のエラーハンドリング処理にて SslErrorHandler.proceed() で問題のあるページでも表示するようにしている。

今回見つかった問題例
～Case2. サーバ証明書の検証不備～

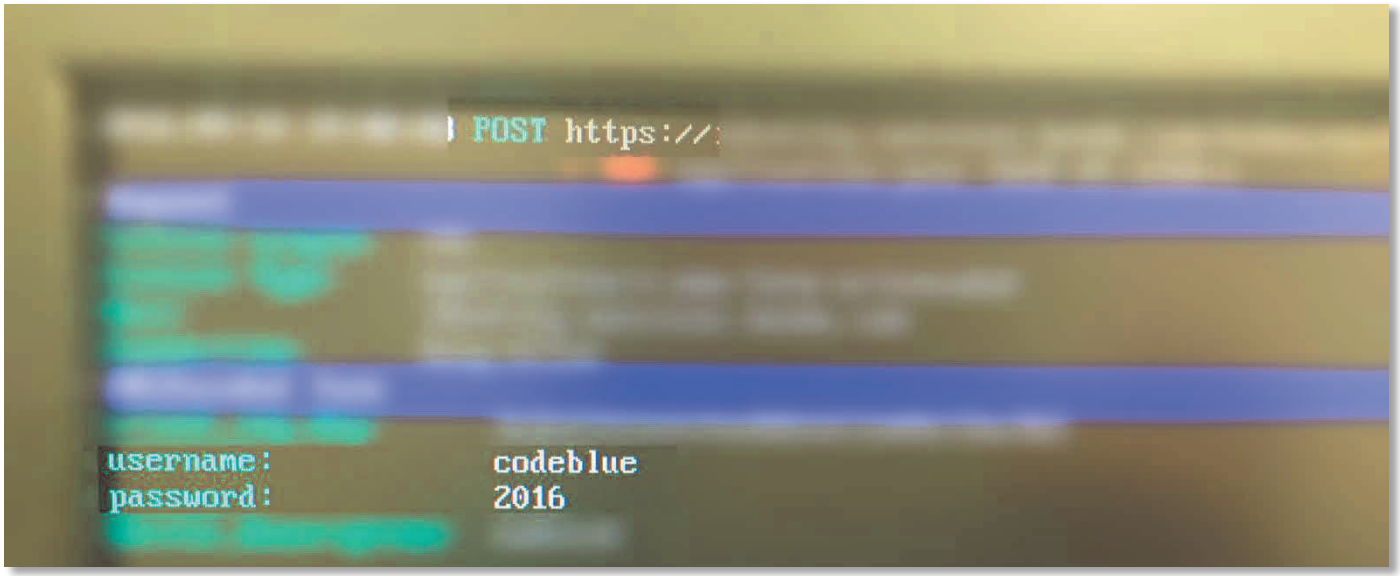
```
setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER)
```

```
X509TrustManager local1 = new X509TrustManager()
{
  public void checkClientTrusted(X509Certificate[] paramAnonymousArrayOfX509Certificate, String paramAnonymousString)
    throws CertificateException
  {
  }

  public void checkServerTrusted(X509Certificate[] paramAnonymousArrayOfX509Certificate, String paramAnonymousString)
    throws CertificateException
  {
  }

  public X509Certificate[] getAcceptedIssuers()
  {
    return null;
  }
```

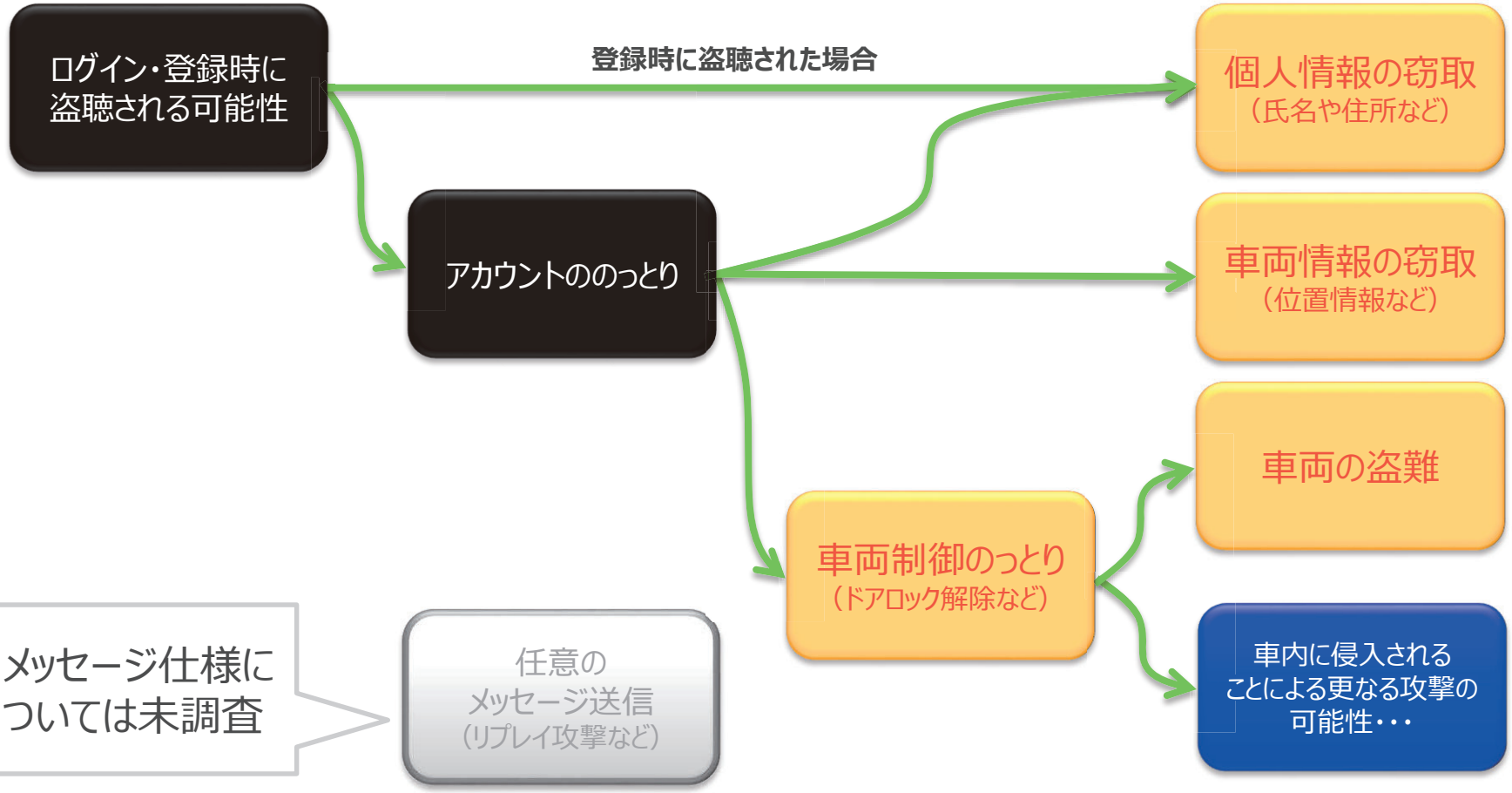
```
public void onReceivedSslError
    SslErrorHandler.proceed()
```

今回見つかった問題例 ～Case2. サーバ証明書の検証不備～

- この問題は結果として下記のようにサービスへのログイン時にユーザIDやパスワードが盗聴されてしまう可能性がある。



脆弱性が見つかったアプリで想定されるリスクまとめ



見つかった脆弱性に対する考察と対策

- 今回見つかった問題はなぜ作りこまれてしまったのか？

開発中におけるデバッグ用として利用していた処理を
そのままリリースしてしまった。

インターネット上などで公開されているサンプルコードを
そのまま流用してしまった。

仕様。
（セキュア設計やコーディングの意識、認識不足）

見つかった問題に対する考察と対策

- Android アプリにおける脆弱性とその要因は様々。今回は、それらの中でも実際に問題が見つかったアプリのHTTP/HTTPS通信における対策（ルール）を紹介する。

センシティブな情報は HTTPS 通信で送受信する

HTTP 通信では受信データの安全性を確認する

SSLException に対しユーザーに通知する等の適切な例外処理をする

独自の TrustManager を作らない

独自の HostnameVerifier は作らない

(出展)Androidアプリのセキュア設計・セキュアコーディングガイド 2016年9月1日版
http://www.jssec.org/dl/android_securecoding.pdf

見つかった問題に対する考察と対策

- Android アプリにおける脆弱性とその要因は様々。今回は、それらの中でも実際に問題が見つかったアプリのHTTP/HTTPS通信における対策（ルール）を紹介する。

センシティブな情報は HTTPS 通信で送受信する

HTTP 通信では受信データの安全性を確認する

SSLE をする

独自の HostnameVerifier は作らない

センシティブ（機微）な情報とは？
あらかじめシステム、ユーザにとって奪われてはいけない
情報を整理しておく必要がある。

(出展)Androidアプリのセキュア設計・セキュアコーディングガイド 2016年9月1日版
http://www.jssec.org/dl/android_securecoding.pdf

見つかった問題に対する考察と対策

- Android アプリにおける脆弱性とその要因は様々。今回は、それらの中でも実際に問題が見つかったアプリのHTTP/HTTPS通信における対策（ルール）を紹介する。

センシティブな情報は HTTPS 通信で送受信する

HTTP 通信では受信データの安全性を確認する

SSLExcep 理をする

独自の HostnameVerifier は作らない

受信データの処理方法が脆弱な場合に攻撃の入り口とされる可能性があるため、あらゆるデータが入力される可能性を考慮する。

(出展)Androidアプリのセキュア設計・セキュアコーディングガイド 2016年9月1日版
http://www.jssec.org/dl/android_securecoding.pdf

見つかった問題に対する考察と対策

- Android アプリにおける脆弱性とその要因は様々。今回は、それらの中でも実際に問題が見つかったアプリのHTTP/HTTPS通信における対策（ルール）を紹介する。

証明書の不備が
原因で発生する

アプリが持つ機能に応じて
検討（考慮）が必要

SSLException に対しユーザーに通知する等の適切な例外処理をする

証明書の不備によって起こる＝MITM攻撃を受けている
可能性がある点に留意

(出展)Androidアプリのセキュア設計・セキュアコーディングガイド 2016年9月1日版
http://www.jssec.org/dl/android_securecoding.pdf

見つかった問題に対する考察と対策

- Android アプリにおける脆弱性とその要因は様々。今回は、それらの中でも実際に問題が見つかったアプリのHTTP/HTTPS通信における対策（ルール）を紹介する。

センシティブな情報は HTTPS 通信で送受信する

SSLException に対しユーザーに通知する等の適切な例外処理をする

証明書の検証を回避する下記の実装はデバッグ段階においても、安易に利用するべきではない。
プライベート証明書を利用する場合は、プライベートCAのルート証明書を使用してサーバー証明書を検証する

独自の TrustManager を作らない

独自の HostnameVerifier は作らない

(出展)Androidアプリのセキュア設計・セキュアコーディングガイド 2016年9月1日版
http://www.jssec.org/dl/android_securecoding.pdf

・他にはどんな対策が必要なのか知りたい人
・まだ見たこと無い人
・これから Android アプリ開発、公開を考えている人、等は・・・

Android アプリのセキュア設計・セキュアコーディングガイド
（http://www.jssec.org/dl/android_securecoding.pdf）

まずは読んでみましょう。

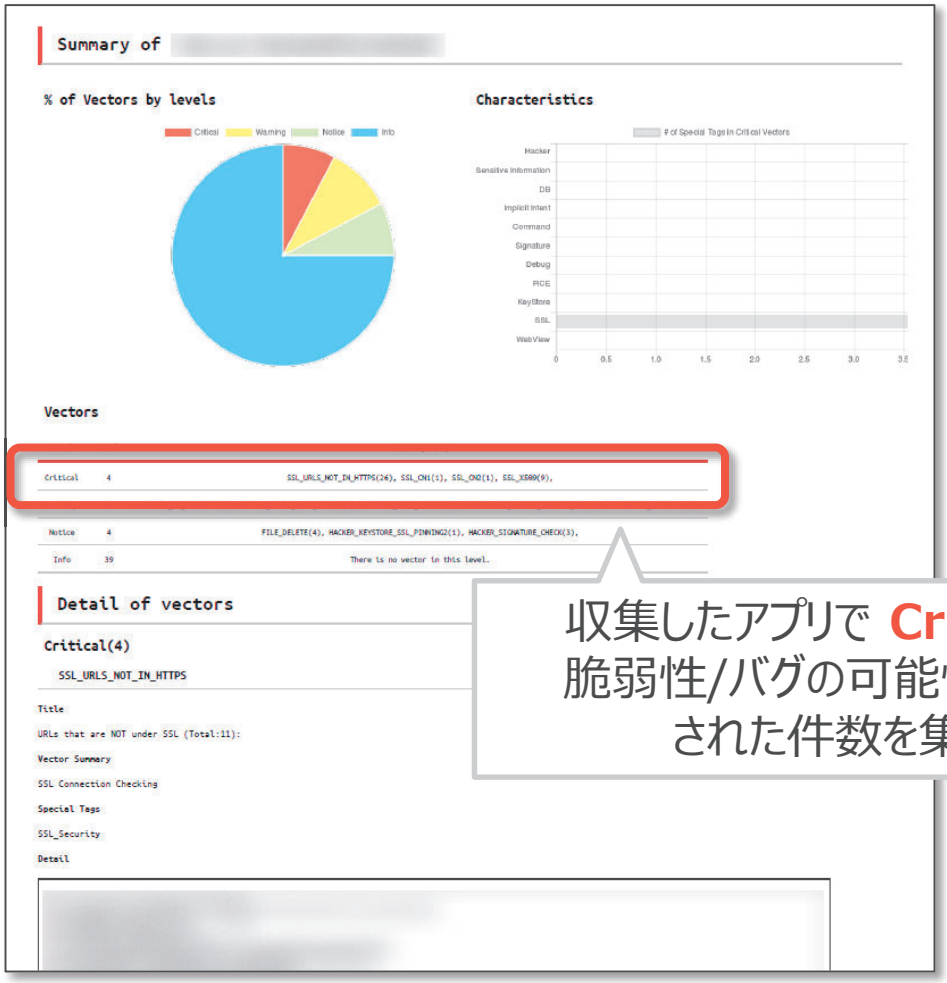
調査結果まとめ [スキャンレポート]

- スキャン結果は機械的にチェック、警告されているものの為この結果が全て脆弱性（バグ）であるとは限らない。

- 傾向としては、SSLに関するセキュリティチェックの警告が多く出ているが、その一部は単純に HTTP 通信であることに起因している。

- スキャンレポートは、冒頭で述べたとおり AndroBugs が出力したレポートをアプリ毎に集計しなおした上で Web ブラウザに出力させて作成した。

調査結果まとめ [スキャンレポート]



収集したアプリで **Critical** な脆弱性/バグの可能性が指摘された件数を集計

作成したスキャンレポートのサンプル

調査結果まとめ [スキャンレポート]

アプリ提供元	リモートコントロール機能の有無	Criticalな脆弱性リスクの件数	不適切なプラットフォーム利用 (M1)	安全でないデータ保存 (M2)	安全でない通信 (M3)
A社	Yes	11	5	1	5
B社	No	5	3	1	1
C社	No	5	2	1	2
D社	No	5	2	1	2
E社	Yes	4	0	0	4
F社	Yes	4	1	1	2
G社	Yes	3	0	0	3
H社	Yes	2	1	0	1
I社	Partial	1	0	0	1
J社	Yes	1	0	0	1
K社	Partial	0	0	0	0

調査結果まとめ [将来問題になる可能性]

アプリ提供元	評価
A社	脆弱性の存在を確認。MITM攻撃を受けるリスクが存在する。 PreferenceActivity を継承したクラスで isValidFragment() を実装していないため、関連アクティビティを公開した場合に Fragment Injection によってアプリがクラッシュさせられる可能性。
B社	現状において、攻撃が可能となる問題は見つからなかった。
C社	現状において、攻撃が可能となる問題は見つからなかった。
D社	現状において、攻撃が可能となる問題は見つからなかった。 難読化されているため、詳細な解析には時間がかかる。
E社	現状において、攻撃が可能となる問題は見つからなかった。 セキュリティ脆弱性では無いが、Android M 以降の新たなパーミッションモデルに対応していないため一部機能にてアプリがクラッシュする。
F社	現状において、攻撃が可能となる問題は見つからなかった。 難読化されているため、詳細な解析には時間がかかる。
G社	現状において、攻撃が可能となる問題は見つからなかった。
H社	現状において、攻撃が可能となる問題は見つからなかった。
I社	現状において、攻撃が可能となる問題は見つからなかった。
J社	現状において、攻撃が可能となる問題は見つからなかった。
K社	現状において、攻撃が可能となる問題は見つからなかった。

まとめ

- 近年多数報告されているリモートコントロール系サービスの問題について、アプリ側にも問題があるケースが実際に存在する事を確認した。
- 今回紹介したありがちな実装ミスはほとんどのアプリで対応出来ている。
- 難読化を採用しているアプリはあまり多くなく、比較的解析しやすい。

少なくとも今回は Android アプリのみの検証結果。
アプリがセキュアだったとしてもサーバや車両側に問題があればそこが狙われる。

リモートコントロール系のサービスは自動運転車両などの
複数の車両を監視するようなシステムにスケールしていく事も予想される。
その場合、アプリや車両だけではなくシステムとしてセキュリティを考える必要がある。

今後やりたいこと

- アプリの調査範囲は限定的。
  - AndroBugs が検出した全ての箇所をチェックしたわけではないため、引き続き調査を継続したい。

- 調査した Android アプリは一部。
  - 他にもリモートコントロールアプリは存在しているため、それらも調査したい。

- 今回はシステムを構成するエンティティのうち Android アプリを対象とした。
  - 機会があればサーバー側や自動車側も検証してみたい。

ご清聴ありがとうございました！