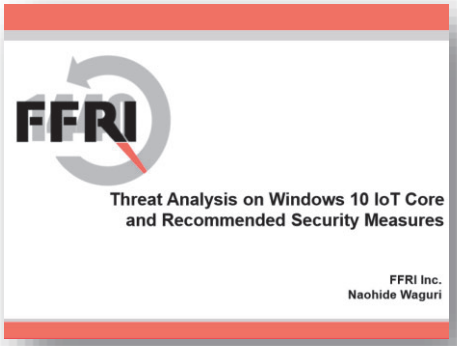
October 21, 2016
CODE BLUE 2016

Security in the IoT World: Analyzing the Security of Mobile Apps for Automobiles

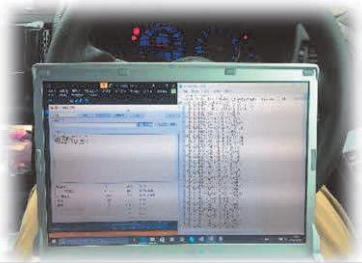
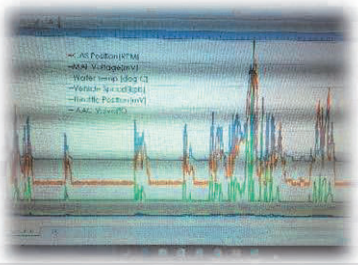
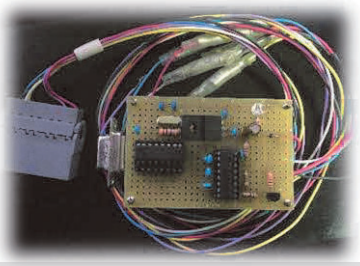
FFRI, Inc.

Who am I?

- I previously worked as a network engineer.
  (Software QA, Software Developer)
- My current job is investigating and researching automotive security.
- I talked about Windows 10 IoT Core
  at CODE BLUE 2015.

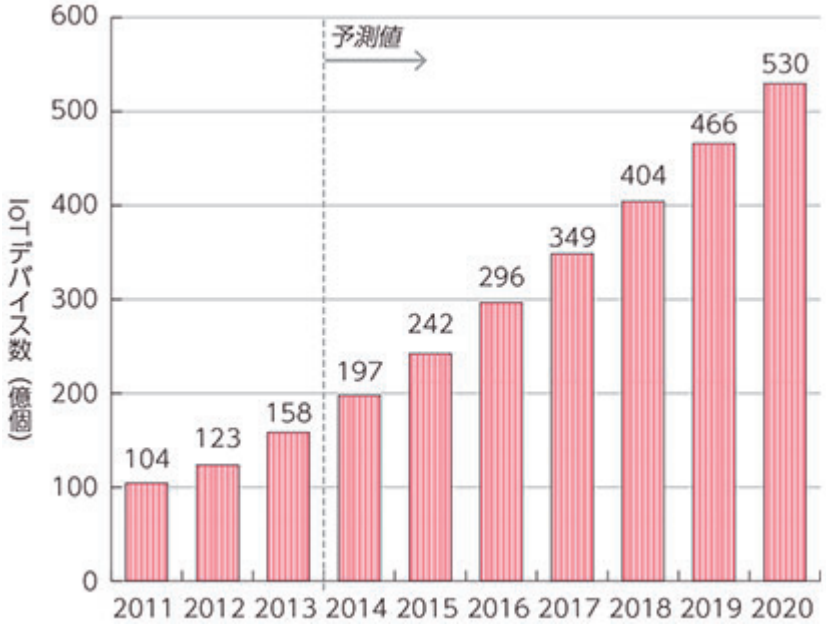


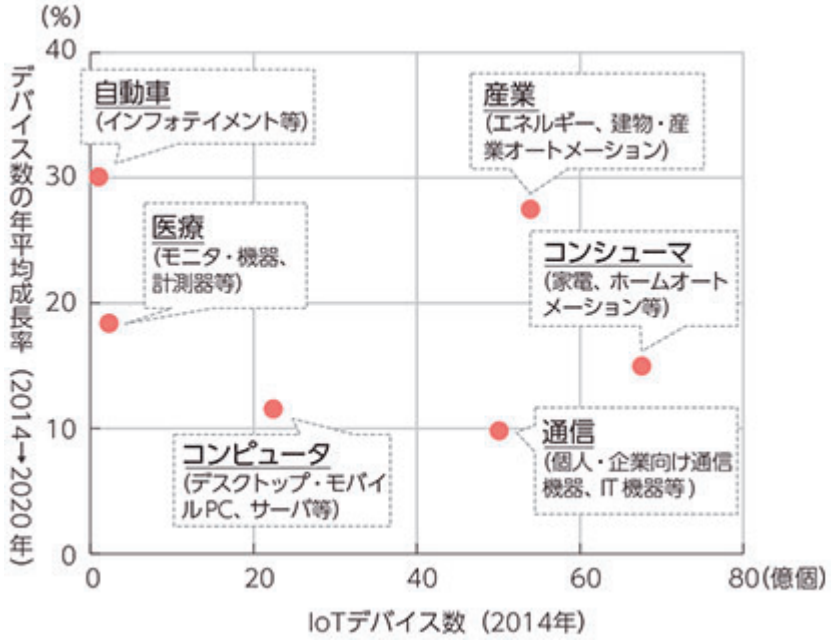
- I build CAN transceivers and diagnostic tools as a hobby.
  (to repair my cars… ☹)

Internet of Things (IoT)

- Several years have passed since use of this term started.
- A wide variety of devices are now connected to the Internet.
- The growth rate is particularly high in sectors related to human life, such as the **automotive**, industrial, and medical sectors.





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

Current State of Automotive Security

- There are two entry points for researching and investigating attacks on automobiles.

Message injection to (CAN) buses

Exploiting vulnerabilities in systems (or devices) connected to the Internet

Current State of Automotive Security

- There are two entry points for researching and investigating attacks on automobiles.

Message injection to (CAN) buses

Exploiting vulnerabilities in systems (or devices) connected to the Internet

We will talk about this entry point. But…

Current State of Automotive Security

- There are two entry points for researching and investigating attacks on automobiles.

Message injection to (CAN) buses

Exploiting vulnerabilities in systems (or devices) connected to the

First of all

We will talk about this entry point. But…

Current State of Automotive Security

- There are two entry points for researching and investigating attacks on automobiles.

Message injection to (CAN) buses

We will talk **A LITTLE** about this entry point.

We will talk about this entry point… But,

Current State of Automotive Security

Message injection to (CAN) buses

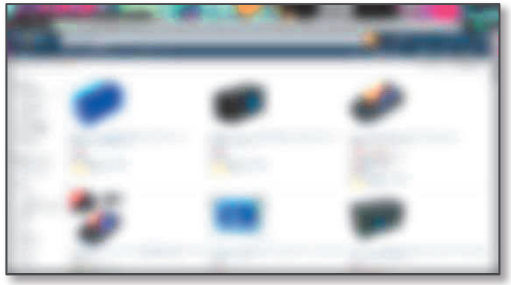
- In most cases, the target of this entry point is a diagnostic port.

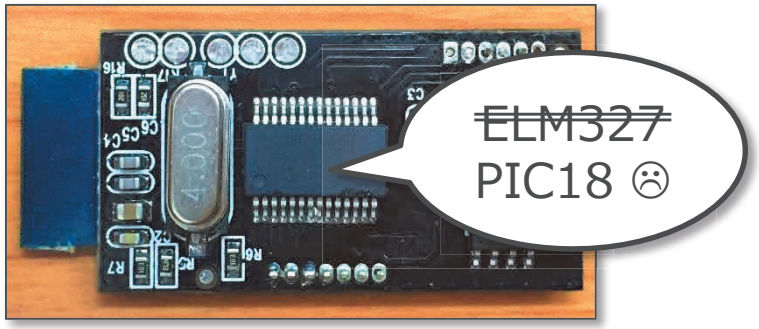
- Now, diagnostic ports are also used for various applications other than maintenance.

For example, owners intentionally connect the OBD-II dongle to vehicles. They need to pay attention before connecting a dongle to their vehicle **because the security level of the vehicle decreases if the device is vulnerable or malicious**.
We recommend the use of **devices from reliable manufactures and developers**.

Current State of Automotive Security

I disassembled an OBD-II dongle sold at popular online shops and auctions.
I found that it was a **FAKE** because it was using a microcontroller
different from the item description…





Also, the Bluetooth PIN cannot be changed…

The threat classification can change from "Physical" to "Adjacent"
if a vulnerable or malicious dongle is connected.

For example, owners intentionally connect the OBD-II dongle to vehicles. They need to pay attention before connecting a dongle to their vehicle **because the security level of the vehicle decreases if the device is vulnerable or malicious**.
We recommend the use of **devices from reliable manufactures and developers**.

Current State of Automotive Security

- There are two entry points for researching and investigating attacks on automobiles.

Message injection to (CAN) buses

Exploiting vulnerabilities in systems (or devices) connected to the Internet

Here is the main subject!

Current State of Automotive Security

- The most famous case of a threat to a connected car…

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

Current State of Automotive Security

- A recent case…

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.

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

Current State of Automotive Security

2015

Aug

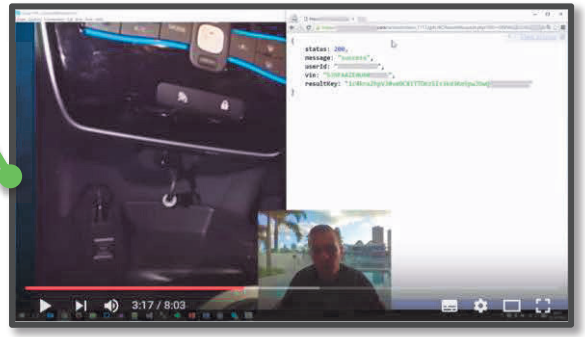
Source: Samy Kamkar,
https://www.youtube.com/watch?v=3olXUbS-prU
Drive It Like You Hacked It: New Attacks And Tools to Wirelessly Steal Cars, DEFCON 23

Oct

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

2016

Feb



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

Jun

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

Motivation

- Vulnerabilities in systems where an "automobile is part of the IoT" have been reported, one after another, from 2015 and beyond.

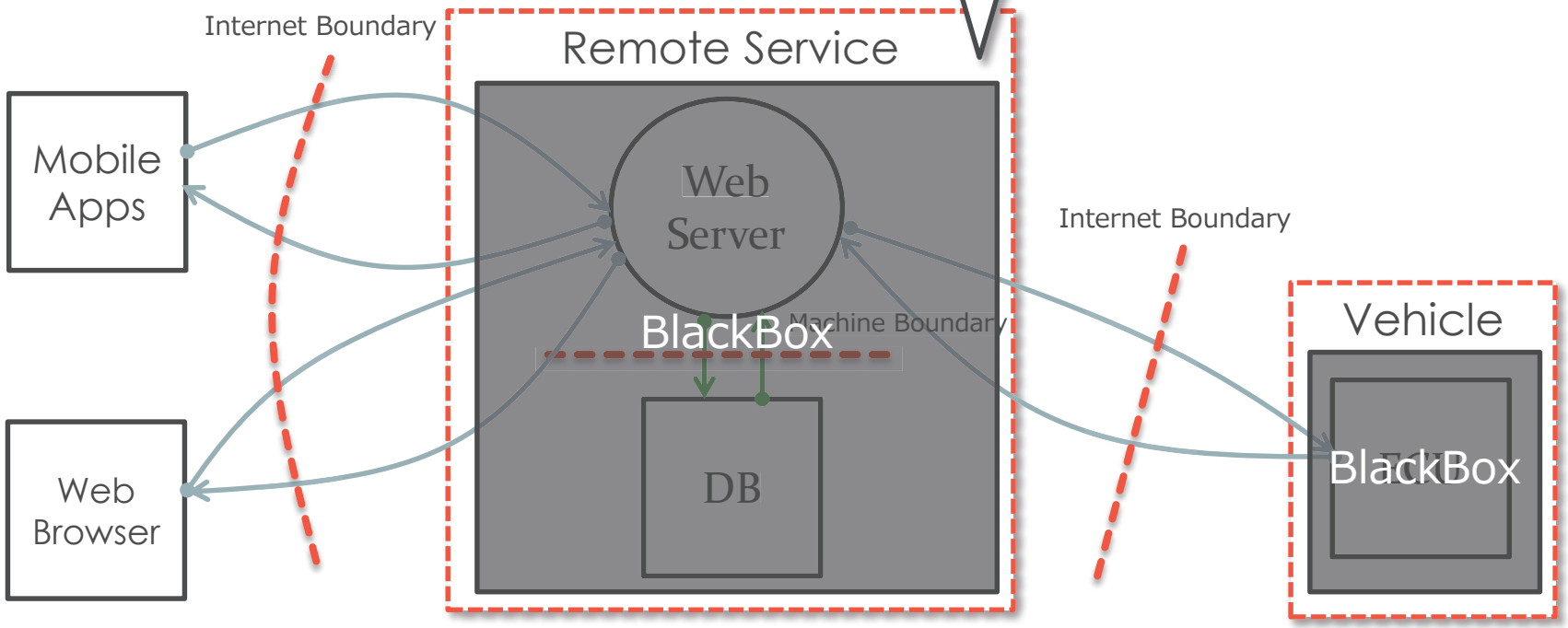
- These threats are not as serious as the “Jeep Hack” vulnerability, but…

  - Personal Information is stolen by attackers.
  - Vehicle position and travel history are stolen by attackers.
  - Doors are unlocked by attackers, allowing vandalism of cars.

  These are threats to the (information) assets of the vehicle owner.

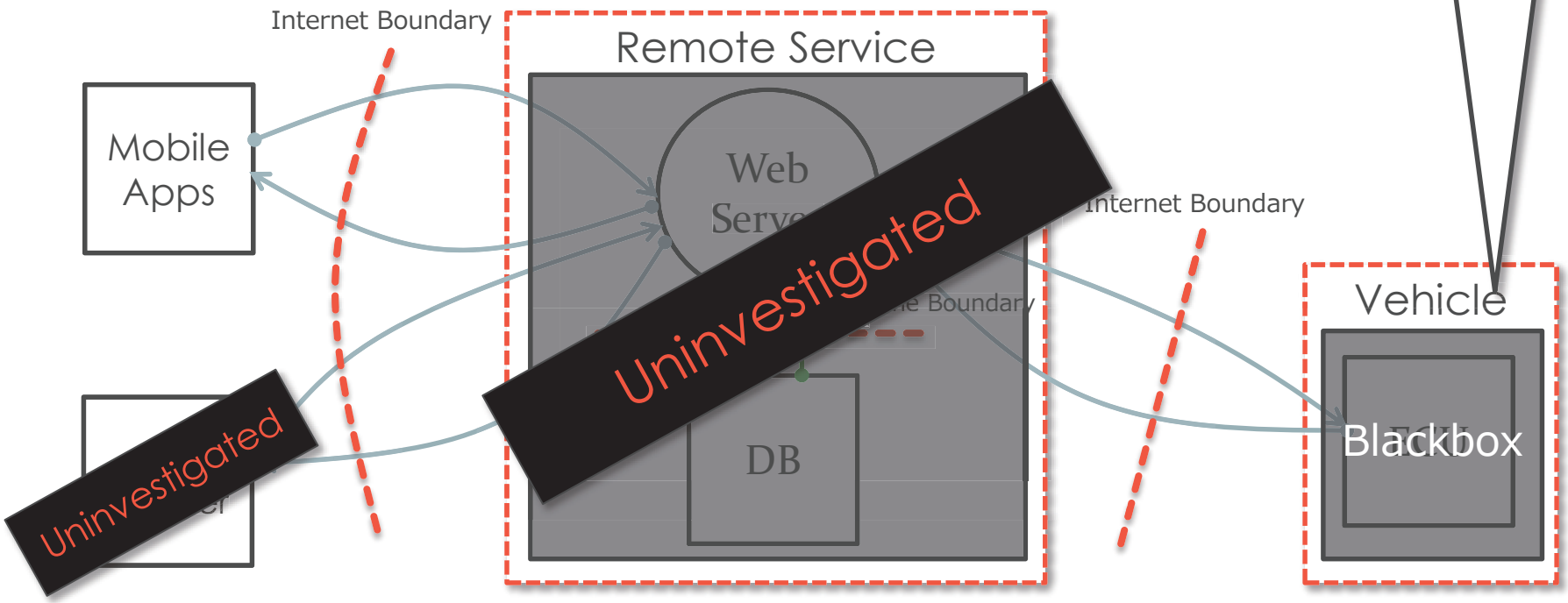
Motivation

The system on the other side of the Internet boundary is basically a black box because each OEM uses its own unique system.
Furthermore, we should not attempt penetration testing via a web browser because it could be deemed a cyber-attack if attempted without permission.



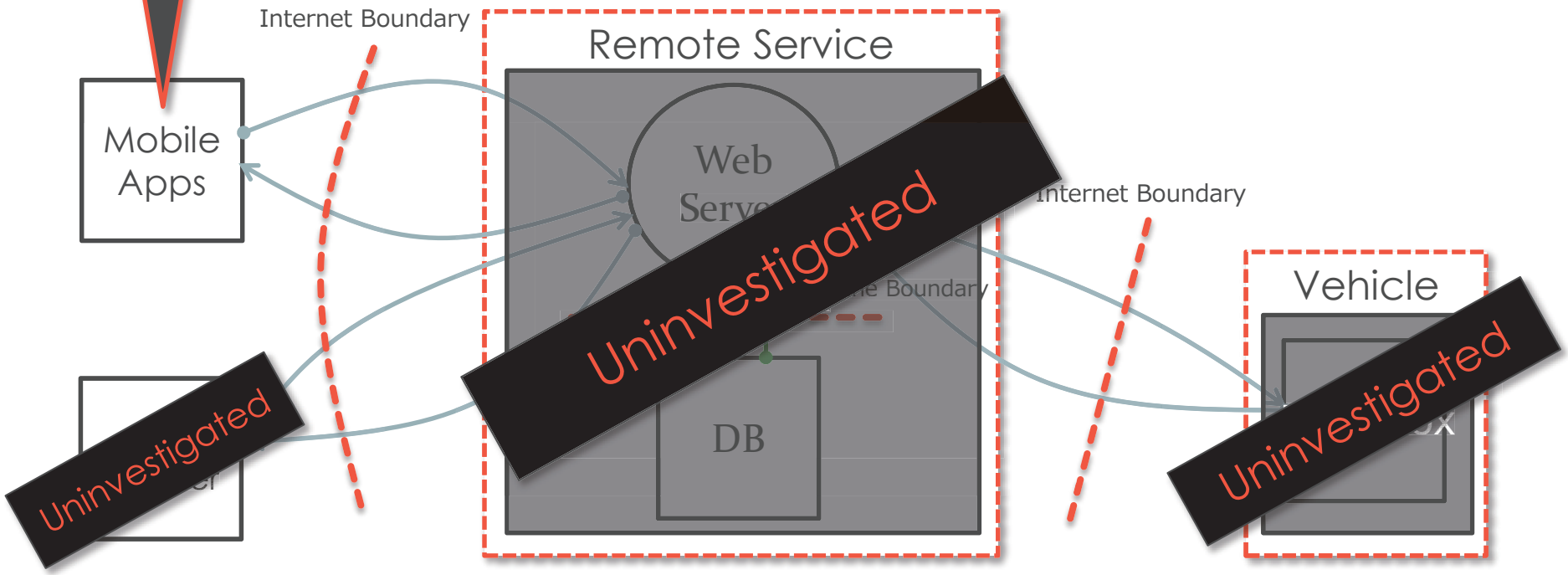
Motivation

This approach is possible but highly challenging.
I would need vehicles and subscriptions to remote control services in order to analyze the communication between the vehicle and system.
(Also, we need to purchase the remote control service in most cases.)



Motivation

Apps can be easily obtained.
(This is also true from the standpoint of the attacker.)
Among the entities that make up the service, this is likely to be the cause of a vulnerability.



Investigation Target and Goal

Phase 0: Collect apps that integrate with the services provided by each OEM.

Phase 1: Create a report on each app using AndroBugs.

What is AndroBugs?

- AndroBugs is a vulnerability scanner for Android apps that was presented at Black Hat EUROPE 2015 by Mr. Yu-Cheng Lin.

- **A system that helps find actual security vulnerabilities in Android Apps.**
- **Open source and written in Python.**
- **A static analysis tool that consumes Android APK (no source code).**
- **Scan for "known common coding vulnerabilities"**
- **Designed for massive analysis and to efficiently finding bugs.**
- **You can easily extend new features or vulnerability vectors.**

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

Investigation Target and Goal

Phase 0: Collect apps that integrate with the services provided by each OEM.

Phase 1: Create a report for each app using AndroBugs.

Phase 2: Analyze the analysis reports for each app.

Understand the security level of apps provided by OEMs and consider necessary corrective measures.

Typical Risks in Mobile Apps

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

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

Vulnerabilities in Investigation Targets

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

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

Overview: M1 – Improper Platform Usage

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

Platform security controls are improperly used, such as the scope of public activities or fragment activity handling.

Example:
The app will crash with a security exception if fragment injection occurs because the activity class that inherits from PreferenceActivity does not override isValidFragment().

Overview: M1 – Improper Platform Usage

M1 – Improper Platform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

Platform security controls are improperly used, such as the scope of public activities or fragment activity handling

Among the three vulnerability risks, this was detected second-most by AndroBugs. However, the results of our investigation did not find any vulnerabilities.

class that inherits from PreferenceActivity does not override isValidFragment().

Overview: M2 – Insecure Data Storage

- M1 – Improper Platform Usage
- M2 – Insecure Data Storage
- M3 – Insecure Communication

Sensitive data is handled insecurely by saving it to external storage, outputting it to logs, etc.

Examples:
Outputting transmission data that contains sensitive information to the debug log.

Using MODE_WORLD_READABLE/WRITABLE to enable access from other apps when getting the instance of SharedPreference.

Overview: M2 – Insecure Data Storage

- M1 – Improper Platform Usage
- M2 – Insecure Data Storage
- M3 – Insecure Communication

Sensitive data is handled insecurely by saving it to external storage, outputting it to logs, etc.

Among the three vulnerability risks, this was the least detected by AndroBugs.
And the results of our investigation did not find any vulnerabilities.

Using MODE_WORLD_READABLE/WRITABLE to enable access from other apps when getting the instance of SharedPreference.

Overview: M3 – Insecure Communication

- M1 – Improper Platform Usage
- M2 – Insecure Data Storage
- M3 – Insecure Communication

Man-in-the-middle (MITM) attacks are allowed because SSL communication is implemented incorrectly.
Many cases have been reported where verification of the server certificate is skipped.

Examples:
Skip hostname verification.

Implement a custom (empty) trustmanager in order to skip certificate validation.

Overview: M3 – Insecure Communication

- M1 – Improper Platform Usage
- M2 – Insecure Data Storage
- M3 – Insecure Communication

Man-in-the-middle (MITM) attacks are allowed because SSL communication is implemented ir
M
v

E
Skip hostname verification.

Implement a custom (empty) TrustManager in order to skip certificate validation.

Among the three of vulnerabilities risk, this was the most detected by AndroBugs. In addition, we confirmed actually vulnerable apps.

Vulnerabilities Found?

YES!

But I cannot disclose the app name… ☹

It is an extremely common vulnerability
in the implementation of
SSL/TLS communication.

Example of Vulnerabilities Found

Case 1: HTTP communication that contains user information

- One activity loads an HTTP URL into WebView.
- The URL posts user information to the server in clear text.
- The other URLs on the same host use HTTPS.
  So, this might be based on some policy, but…

Example of Vulnerabilities Found

Case 1: HTTP communication that contains user information

Example of Vulnerabilities Found

Case 2: Server certificate validation flaw

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

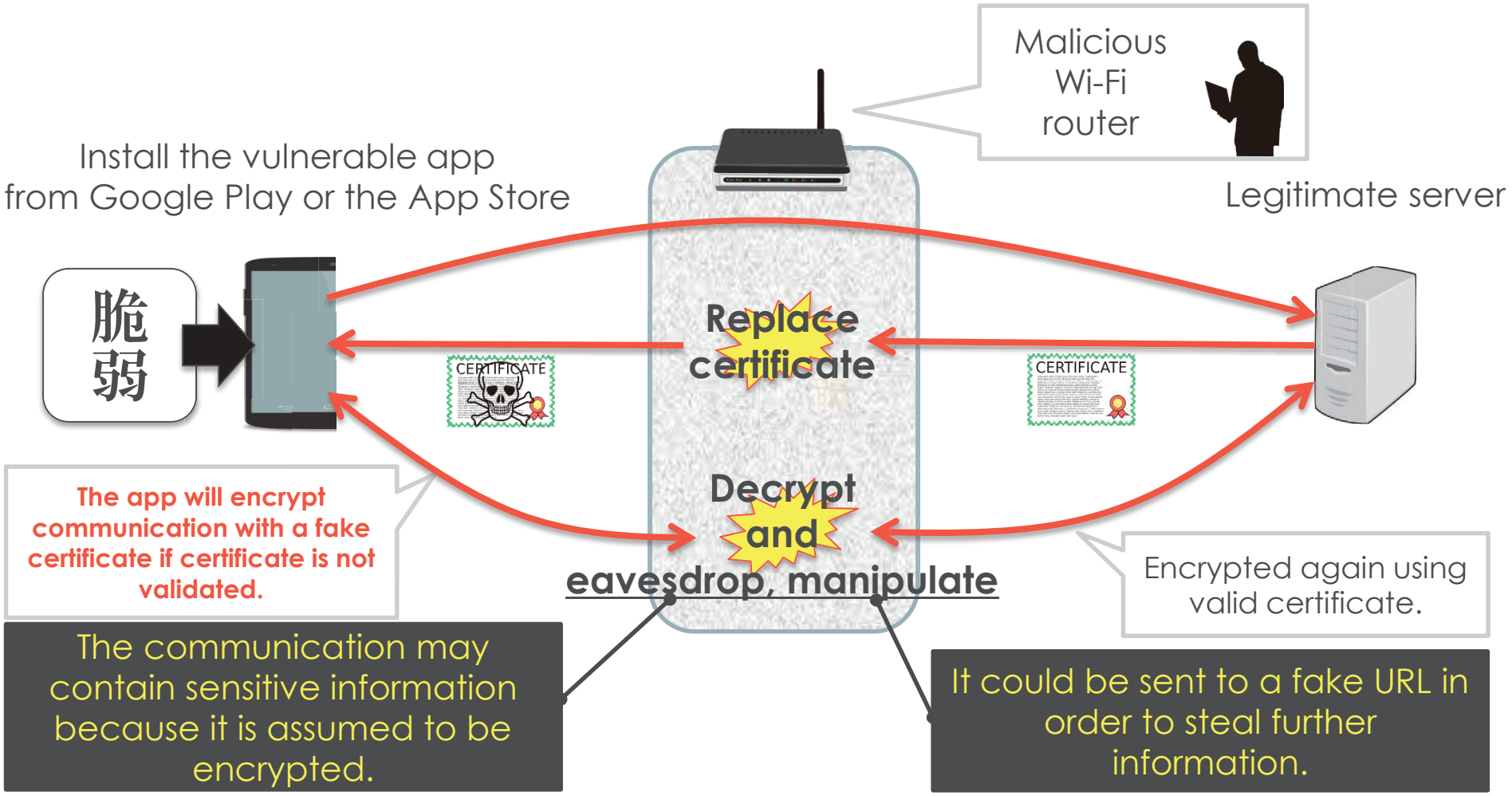
Example of Vulnerabilities Found

Case 2: Server certificate validation flaw

What kind of vulnerability is a server certificate validation flaw?
And what kind of risk does it pose?

Example of Vulnerabilities Found

Case 2: Server certificate validation flaw



Example of Vulnerabilities Found

Case 2: Server certificate validation flaw

- Hostname verification is skipped because ALLOW_ALL_HOSTNAME_VERIFIER is used.
- Certificate verification is skipped because a custom (empty) TrustManager is used.
- WebView displays the page even if it is malicious because of SslErrorHandler.proceed() in certificate verification.

Example of Vulnerabilities Found

Case 2: Server certificate validation flaw

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

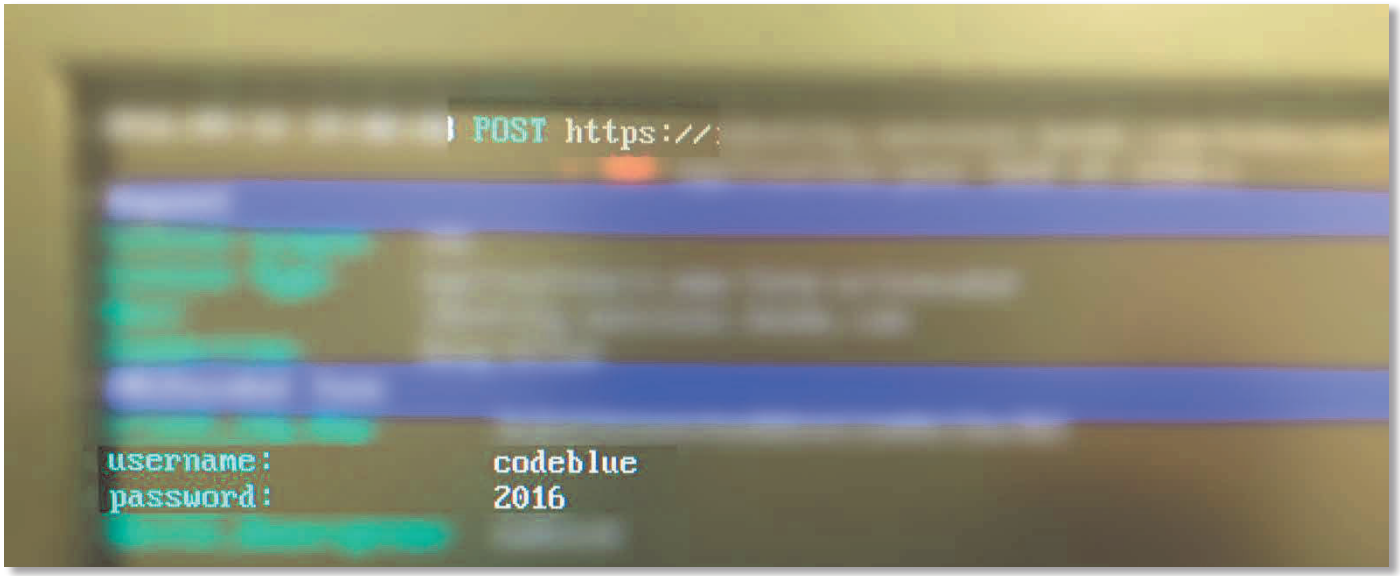
```
X509TrustManager local1 = new X509TrustManager()
{
  public void checkClientTrusted(X509Certificate[] paramAnonymousArrayOfX509Certificate, String paramAnonymousString)
    throws CertificateException
  {
  }

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

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

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

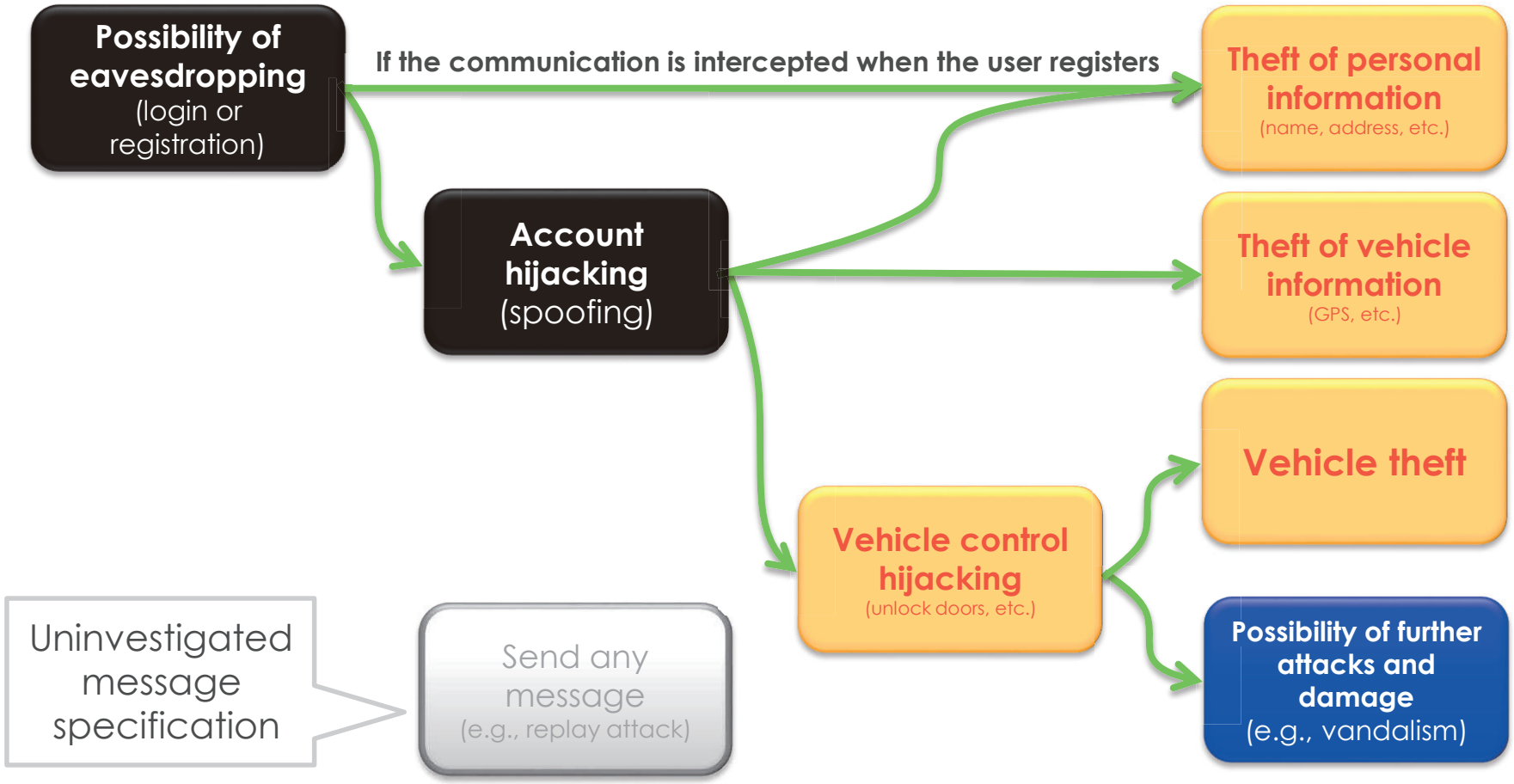
Example of Vulnerabilities Found

Case 2: Server certificate validation flaw

- This vulnerability poses the risk that the user ID and password are intercepted when the user logs into the service using the app.



Summary of risks of the app in which vulnerabilities were found



Corrective Measures and Considerations for Vulnerable Apps

- Why did these vulnerabilities occur?

Debugging code in the release build

Sample code copy & pasted (some might say “appropriated”) from the Internet

Bad specifications
(Lack of understanding of secure design and coding)

Corrective Measures and Considerations for Vulnerable Apps

- There are various vulnerabilities and enabling factors in Android. We will introduce implementation rules for HTTP/HTTPS communication to prevent the vulnerability found in the app we investigated.

Use HTTPS communication when sending sensitive information

Verify the safety of the received data if it uses HTTP communication

Implement appropriate exception handling for SSLException
(e.g., user notification)

Do not implement a custom TrustManager

Do not implement a custom HostnameVerifier

(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

Corrective Measures and Considerations for Vulnerable Apps

- There are various vulnerabilities and enabling factors in Android. We will introduce implementation rules for HTTP/HTTPS communication to prevent the vulnerability found in the app we investigated.

Use HTTPS communication when sending **sensitive information**

Verify the safety of the received data if it use TTP communication

Im opn

Do not implement custom HostnameVerifier

What is sensitive information?
We need to understand the system and user information that must be protected in advance.

(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

Corrective Measures and Considerations for Vulnerable Apps

- There are various vulnerabilities and enabling factors in Android. We will introduce implementation rules for HTTP/HTTPS communication to prevent the vulnerability found in the app we investigated.

Use the HTTPS communication if it contains sensitive information

Verify the **safety of the received data** if it uses HTTP communication

Implement the appropriate exception handling to SSLExceptiopn

Vulnerable processing of incoming data may be targeted by attackers.
We need to perform fuzz testing.

Do not implement custom HostnameVerifier

(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

Corrective Measures and Considerations for Vulnerable Apps

- There are various vulnerabilities and enabling factors in Android. We will introduce implementation rules for HTTP/HTTPS communication to prevent the vulnerability found in the app we investigated.

Necessary to consider the behavior for each feature specification

Caused by certificate error

Implement the **appropriate exception handling** for **SSLException**
(e.g., user notification)

Exception occurs if there is a certificate error
→ There may be an MITM attack in progress

(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

Corrective Measures and Considerations for Vulnerable Apps

- There are various vulnerabilities and enabling factors in Android. We will introduce implementation rules for HTTP/HTTPS communication to prevent the vulnerability found in the app we investigated.

Use HTTPS communication when sending sensitive information

Verif ation

Im on
(e.g., use ification)

Use the private CA root certificate to validate the server certificate if you are using a private certificate.
Do not do this to skip certificate validation even if you are debugging.

Do not implement a custom TrustManager

Do not implement a custom HostnameVerifier

(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

- If you are going to develop an Android app…
- If you want to know other rules for Android apps…
- If you have not read these yet…

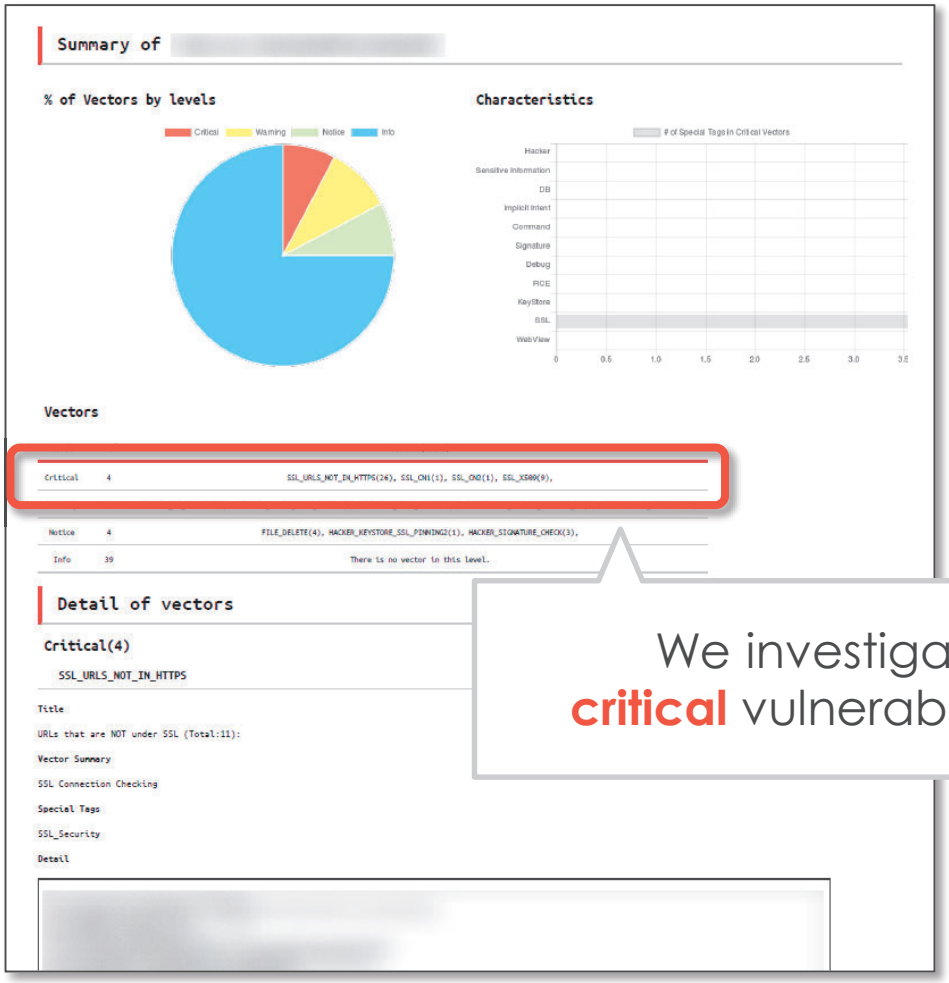
Android アプリのセキュア設計・セキュアコーディングガイド
(http://www.jssec.org/dl/android_securecoding.pdf)

Check it out and give it a try! (Sorry, this is Japanese only.)

Summary: Scan Reports

- Not all positive scan results are true vulnerabilities because AndroBugs reports common vulnerabilities mechanically.
- AndroBugs detects many SSL security alerts because most risk factors arise from apps using HTTP communication.
- We created a web-based custom scan report for each app by using the reports output by AndroBugs.

Summary: Scan Reports



We investigated **critical** vulnerability risks

Scan report sample

Summary: Scan Reports

App Author	Remote Control Feature	No. of Critical Vulnerabiltiy Risks	M1 – Improper Platform Usage	M2 – Insecure Data Storage	M3 – Insecure Communication
A	Yes	11	5	1	5
B	No	5	3	1	1
C	No	5	2	1	2
D	No	5	2	1	2
E	Yes	4	0	0	4
F	Yes	4	1	1	2
G	Yes	3	0	0	3
H	Yes	2	1	0	1
I	Partial	1	0	0	1
J	Yes	1	0	0	1
K	Partial	0	0	0	0

Summary: Possibility of Exploits in the Future

App Author	Note
A	We confirmed there are exploitable vulnerabilities. There is a risk of MITM attacks. Classes that inherit from PreferenceActivity do not implement isValidFragment(). The app might crash due to fragment injection if the class becomes a public activity.
B	We could not find any exploitable vulnerabilities in this app.
C	We could not find any exploitable vulnerabilities in this app.
D	We could not find any exploitable vulnerabilities in this app. Analysis will take a long time because the app was obfuscated.
E	We could not find any exploitable vulnerabilities in this app. This app crashes as part of an activity because it does not support the new permission model starting from Android M.
F	We could not find any exploitable vulnerabilities in this app. Analysis will take a long time because the app was obfuscated.
G	We could not find any exploitable vulnerabilities in this app.
H	We could not find any exploitable vulnerabilities in this app.
I	We could not find any exploitable vulnerabilities in this app.
J	We could not find any exploitable vulnerabilities in this app.
K	We could not find any exploitable vulnerabilities in this app.

Conclusions

- Client-side vulnerabilities were confirmed in remote control services for which other vulnerabilities have been reported recently.
- However, most of the apps did not have the above implementation errors.
- Most apps have been easy to analyze because they are not obfuscated.

The results of this report apply to only Android apps.
Even if the client app is secure, if the server or vehicle is vulnerable, then attackers will target those vulnerable points.

Services like remote control have the potential of being scaled to monitoring of many vehicles at the same time for autonomous cars.
Therefore, we need to consider the security of the system as a whole, not only the individual apps and vehicles.

Future Work

- **Continue the analysis…**
  - We have not yet finished analyzing all of the vulnerability risks detected by AndroBugs.
- **This investigation did not cover all of the available apps.**
  - We also want to investigate other apps that were outside of the scope of this investigation.
- **Our investigation scope at this time was only Android apps.**
  - We also want to investigate server/vehicle-side applications if we have a chance.

Thank you!