Security of Mobile Platforms: Android Security

Techkriti-2019

OUTLINE

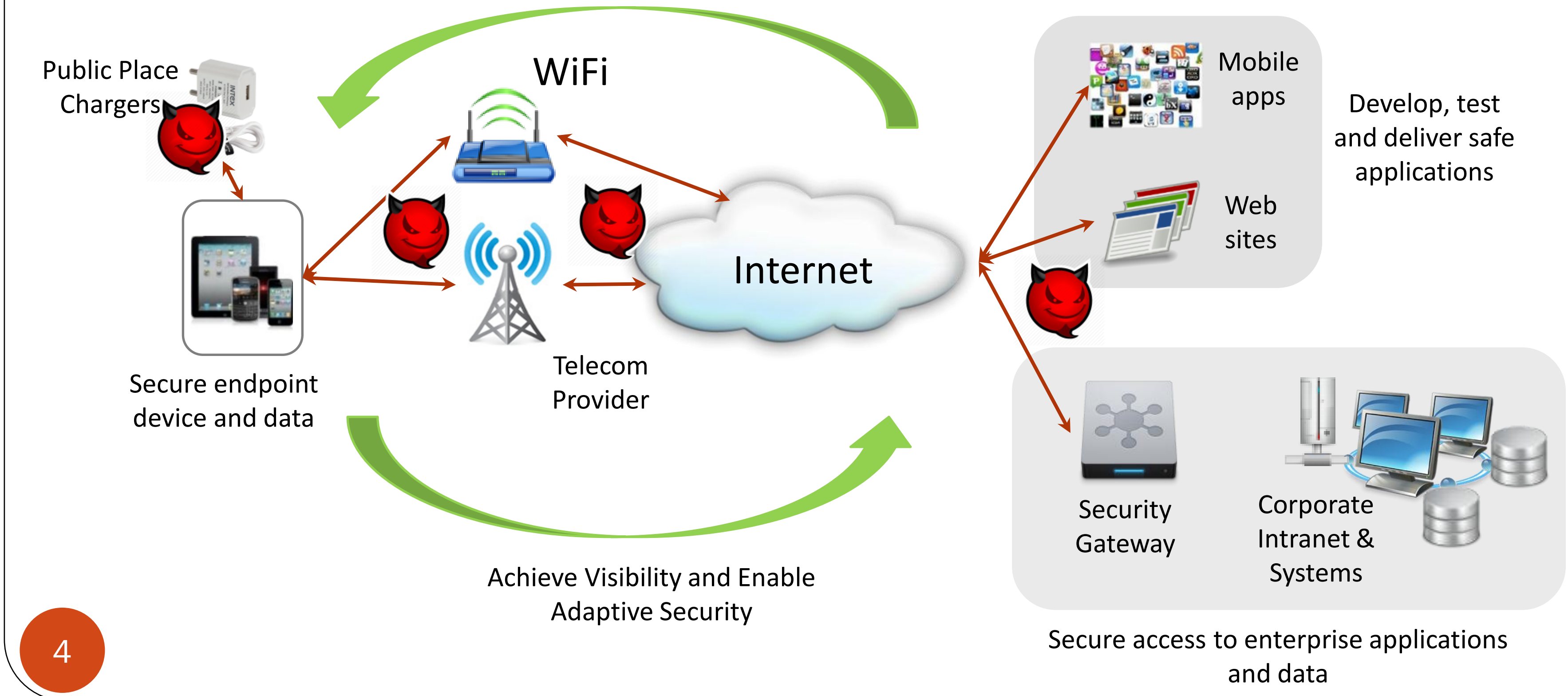
- Motivation
- Android Application
- Android Security Architecture
- Android Vulnerability
- Advanced Threat
- Malware Analysis
- Hands On

MOTIVATION

- Why Mobile Security?
- Why Android?
- Android Ecosystem

Why Mobile Security?

- Technology improvements
- User activity
- Always on
- Valuable data
- Multiple Attack Surfaces



Public Place Chargers
WiFi
Internet
Mobile apps
Develop, test and deliver safe applications
Web sites
Secure endpoint device and data
Telecom Provider
Security Gateway
Corporate Intranet & Systems
Achieve Visibility and Enable Adaptive Security
Secure access to enterprise applications and data

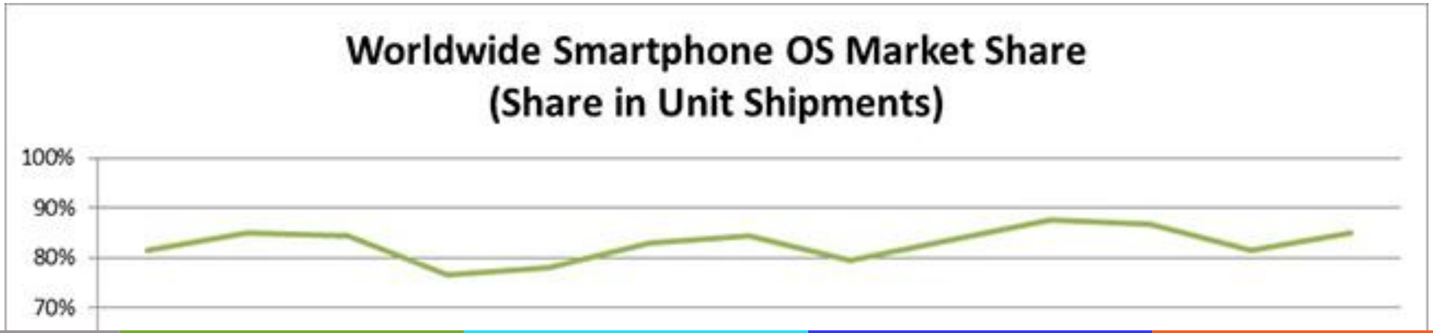
Motivation

- Why Mobile Security?
- Why Android?
- Android Ecosystem

1. Almost completely open source

Source: https://giphy.com/gifs/southparkgifs-3o6ZtqprcPDOkDru5W

2. The Market
Global Smartphone Market Trends



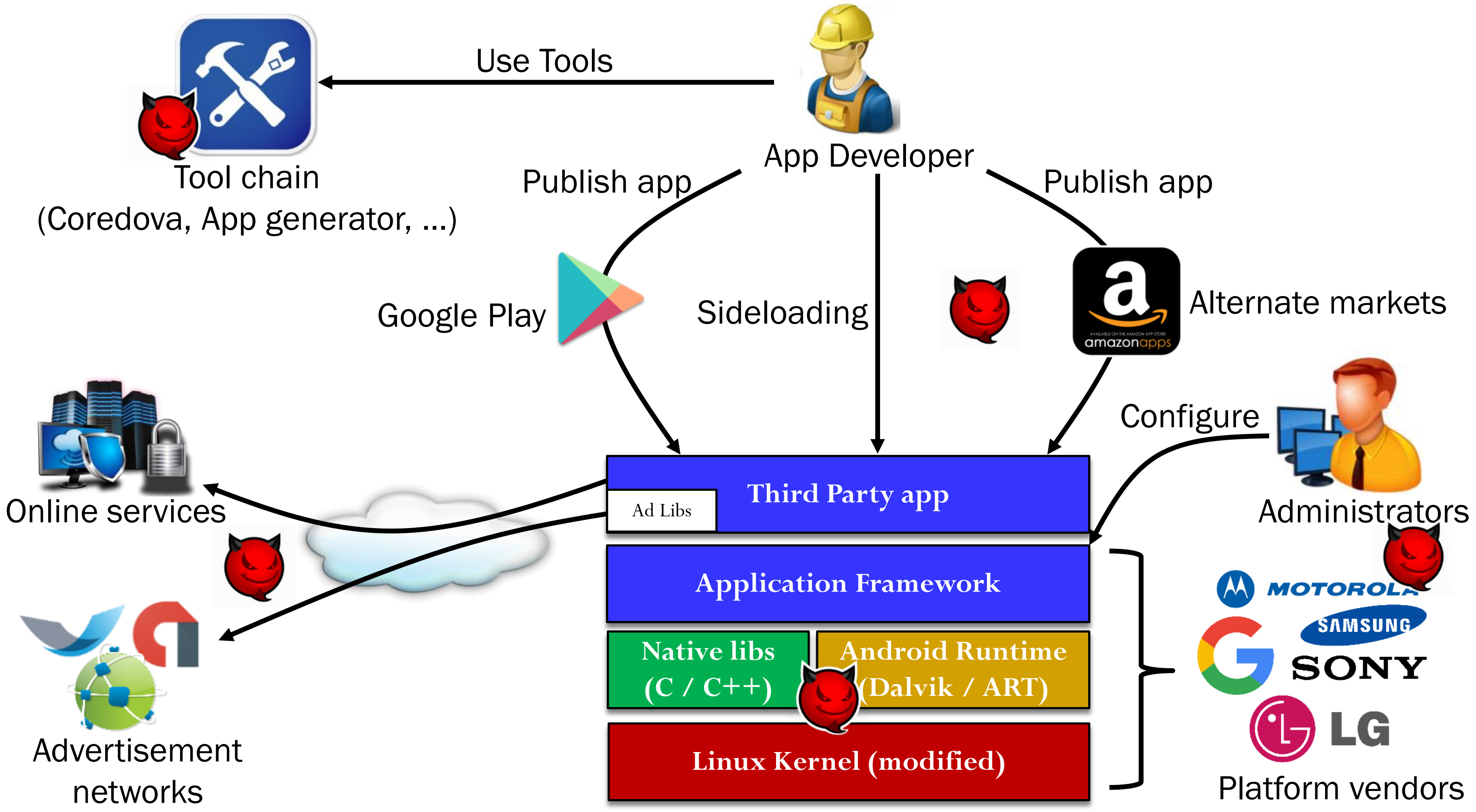
Period	Android	iOS	Windows	Others
Q1 2016	83.4%	15.4%	0.8%	0.4%
Q2 2016	87.6%	11.7%	0.4%	0.3%
Q3 2016	86.6%	12.5%	0.3%	0.4%
Q4 2016	81.4%	18.2%	0.2%	0.2%
Q1 2017	85%	14.7%	0.1%	0.1%

Source: International Data Corporation (IDC), May 2017

MOTIVATION

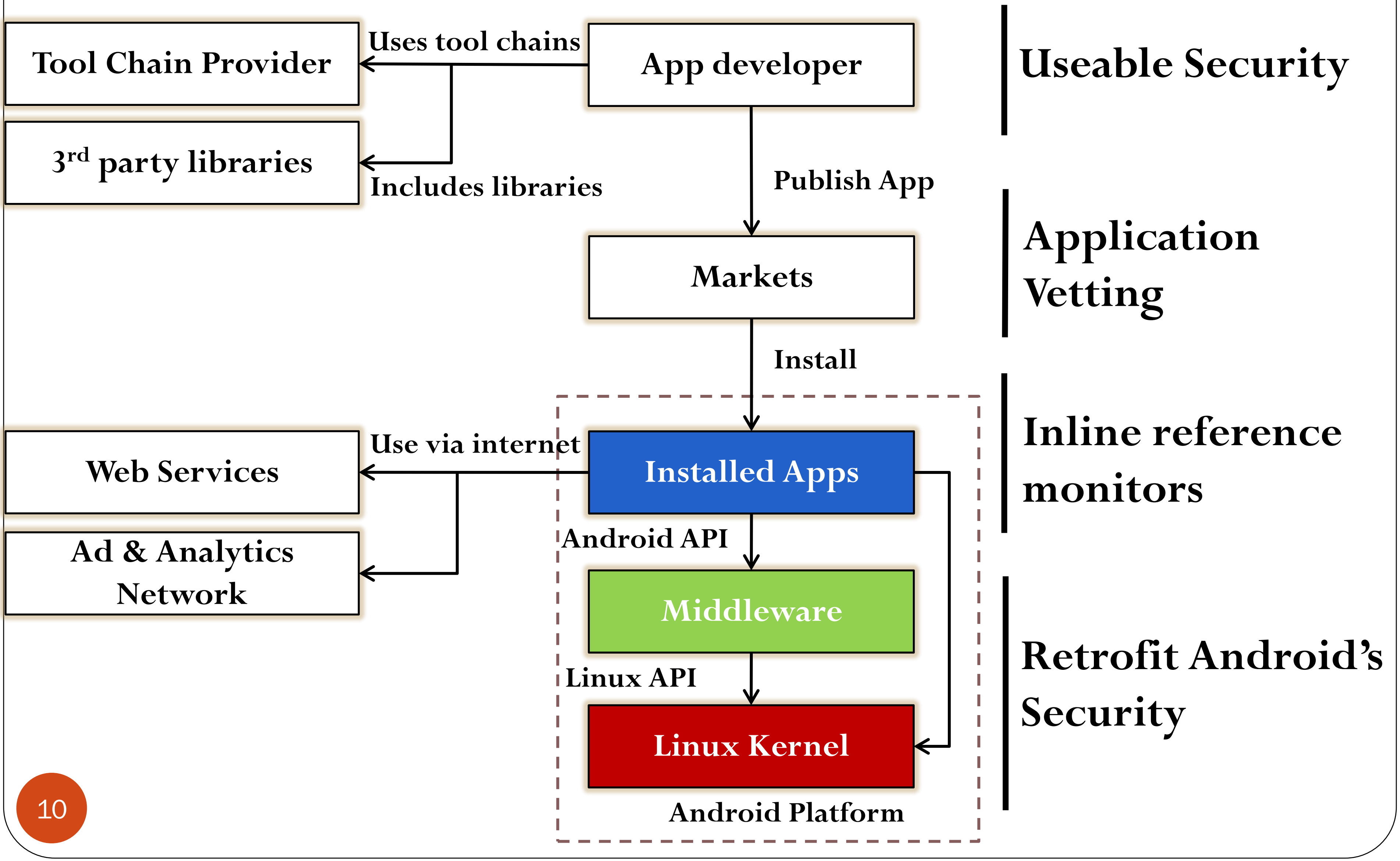
- Why Mobile Security?
- Why Android?
- Android Ecosystem

Actors in the Android Ecosystem



1. Y. Acar et al., "SoK: Lessons Learned From Android Security Research For Appified Software Platforms," SP ’16

Where to improve security & privacy protection?



Security Impact of an Actor over others[1]

Actor	OS Developer	H/W Vendor	Library Provider	S/W Developer	Toolchain Provider	S/W Publisher	S/W Market	End User
OS Developer	--	**Partial**	Full	Full	**Partial**	Full	Full	Full
H/W Vendor	None	--	Full	Full	None	None	None	Full
Library Provider	None	None	--	Full	None	None	None	Full
S/W Developer	None	None	Partial	--	None	None	None	Full
Toolchain Provider	None	None	None	Full	--	None	None	Partial
S/W Publisher	None	None	Partial	Partial	None	--	Partial	Full
S/W Market	None	None	Partial	Partial	None	None	--	Full
End User	None	None	None	None	None	None	None	--

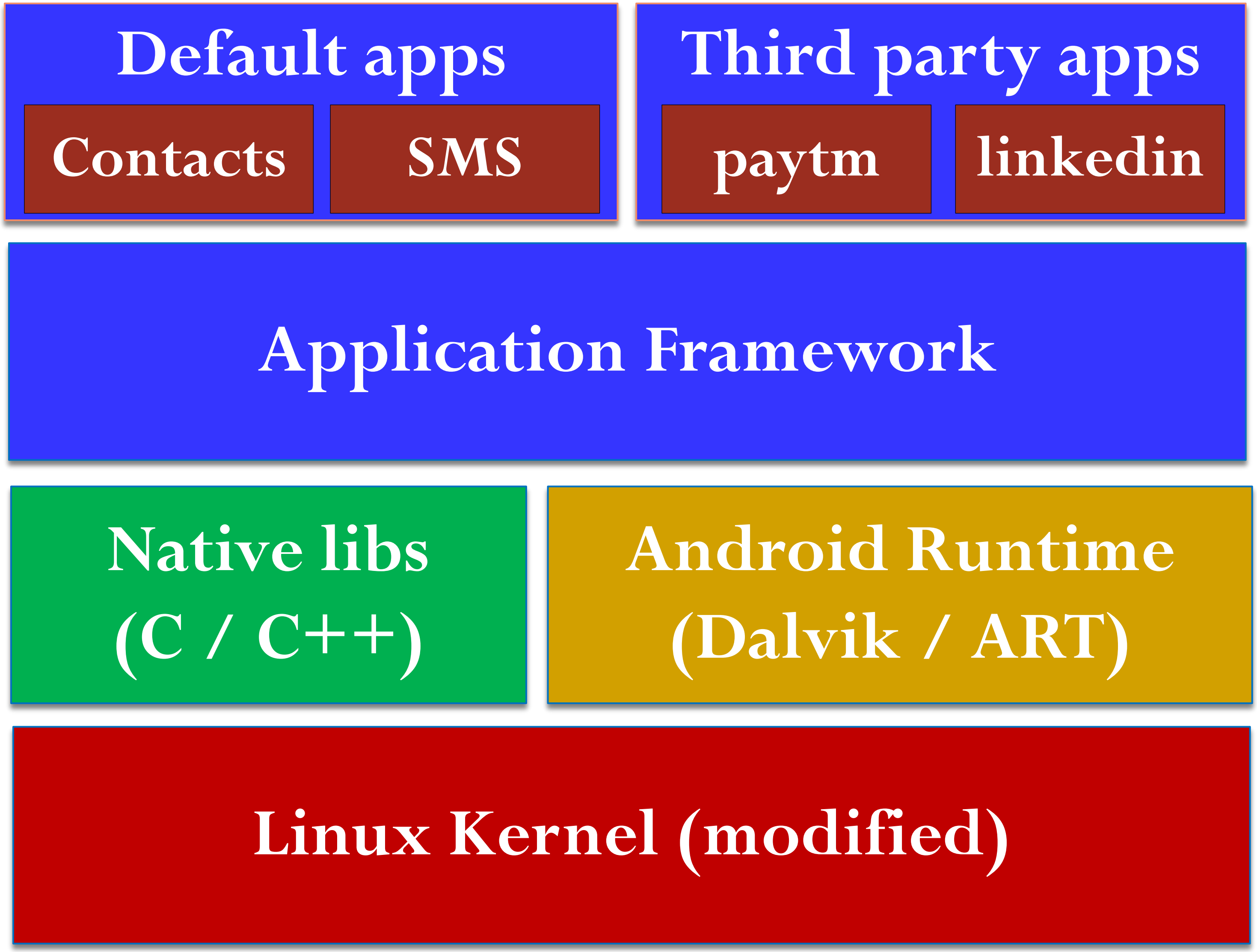
[1] Y. Acar et al., "SoK: Lessons Learned From Android Security Research For Appified Software Platforms," SP '16

Motivation: Summary

- **Feature-rich smartphones** and **appification** have induced security research on various new aspects
- Android's **market share** has made Android the **#1 target** for malware authors and makes improved security & privacy mechanisms imperative
- Various actors in the **ecosystem** with (strong) influence on **security and privacy**

Android Applications

ANDROID SOFTWARE STACK



Default apps

Contacts

SMS

Third party apps

paytm

linkedin

Application Framework

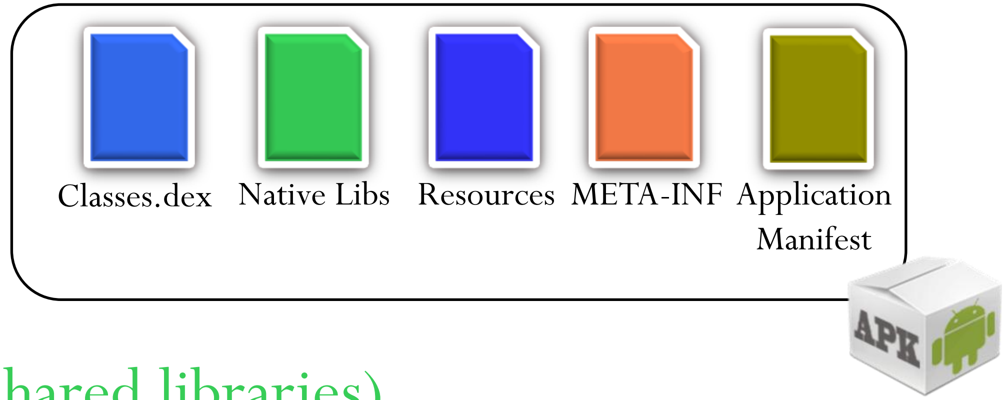
Native libs (C / C++)

Android Runtime (Dalvik / ART)

Linux Kernel (modified)

APPLICATION PACKAGES (APK)

- APK is simply a packaging format like **JAR**, ZIP and TAR
- Component of Application
  - Activity
  - Content Provider
  - Services
  - Broadcast Receiver



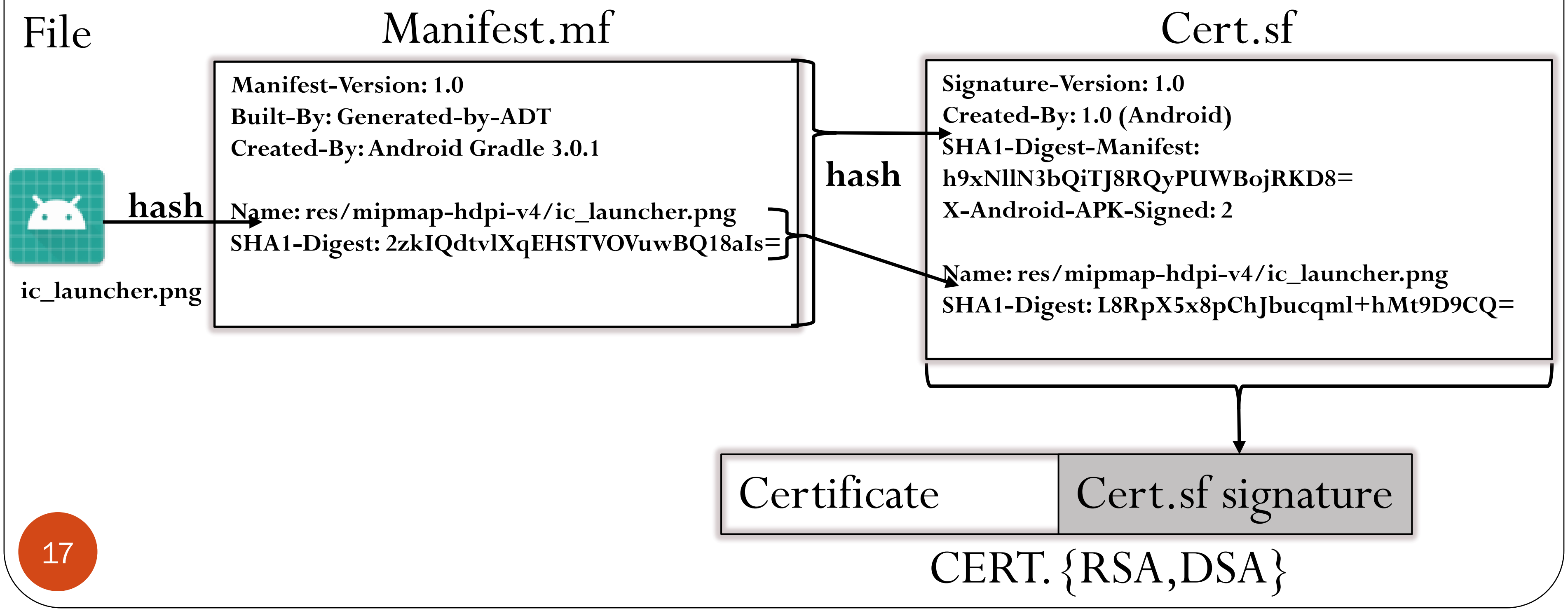
- Native Code (C/C++ shared libraries)
- Resources
- META-INF
- Application Manifest

Android Security Architecture

- Package Integrity
- Sandboxing
- Permission and Least Privilege

Package Integrity: Package Manifest

- Created with **jarsigner**
- META-INF
  - Manifest.mf
  - Cert.sf
  - Cert.{RSA,DSA}

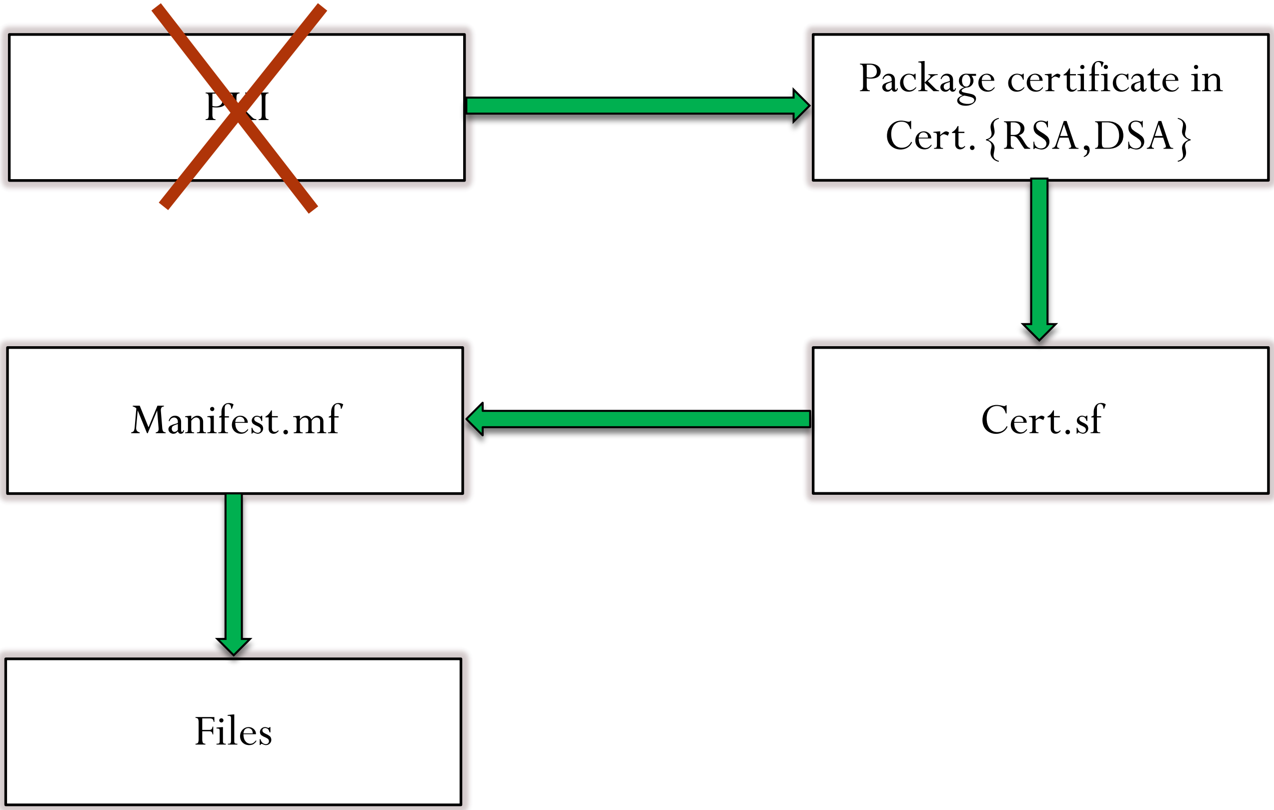


```
Manifest-Version: 1.0
Built-By: Generated-by-ADT
Created-By: Android Gradle 3.0.1

Name: res/mipmap-hdpi-v4/ic_launcher.png
SHA1-Digest: 2zkIQdtvlXqEHSTVOVuwBQ18aIs=
```

```
Signature-Version: 1.0
Created-By: 1.0 (Android)
SHA1-Digest-Manifest:
h9xNllN3bQiTJ8RQyPUWBojRKD8=
X-Android-APK-Signed: 2

Name: res/mipmap-hdpi-v4/ic_launcher.png
SHA1-Digest: L8RpX5x8pChJbucqml+hMt9D9CQ=
```

VERIFYING OF PACKAGE MANIFEST

Chain of trust:

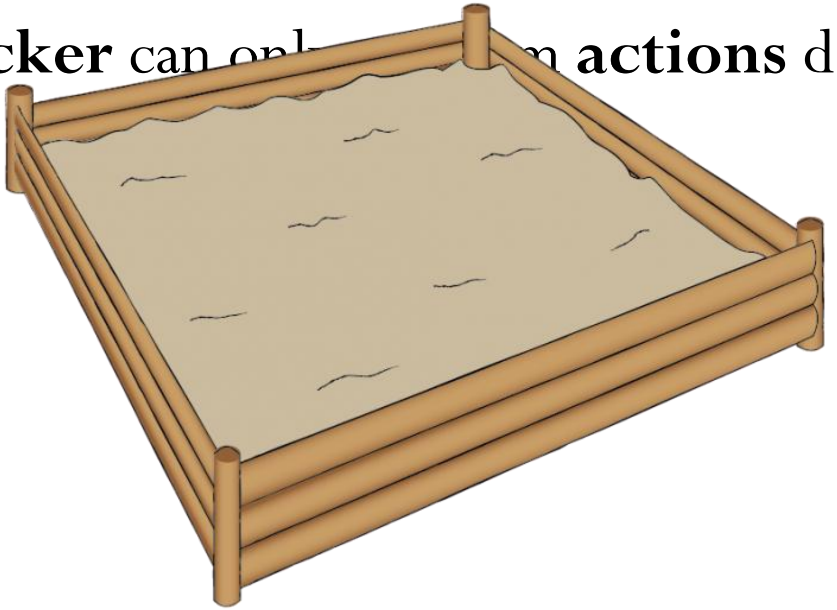


Android Security Architecture

- Package Integrity
- Sandboxing
- Permission and Least Privilege

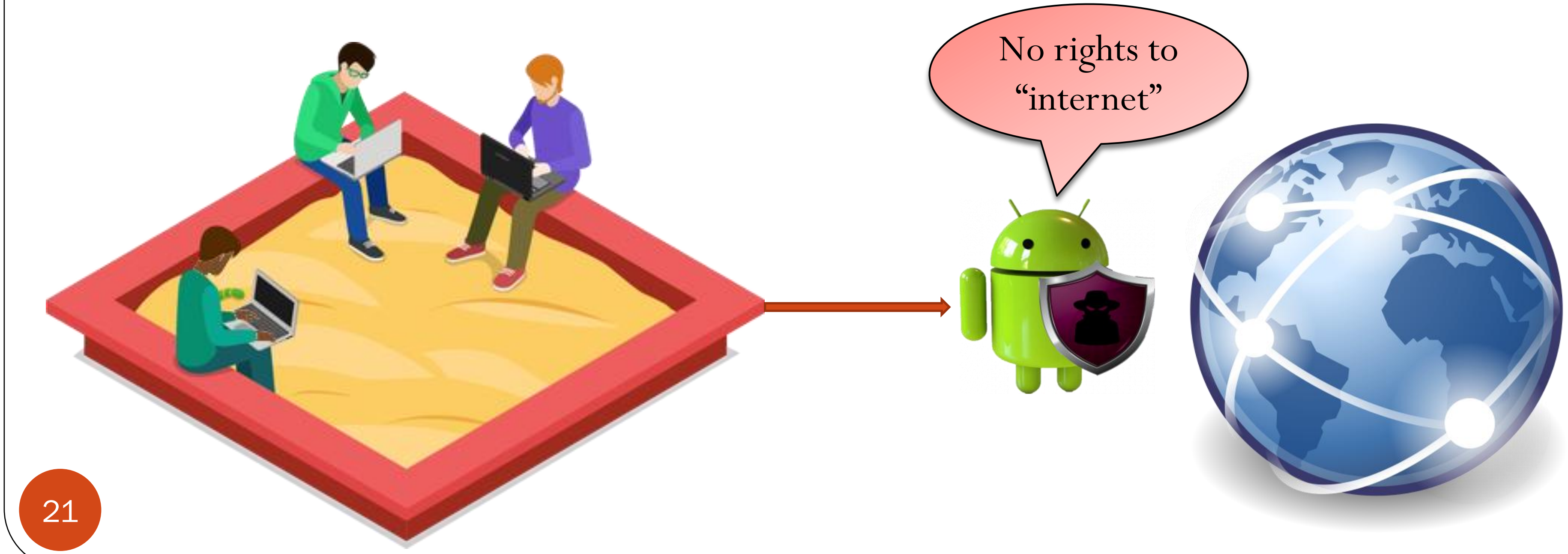
SANDBOXING

- The application sandbox **specifies** which system **resources** the application is allowed to access
- An **attacker** can onl m **actions** defined in the sandbox



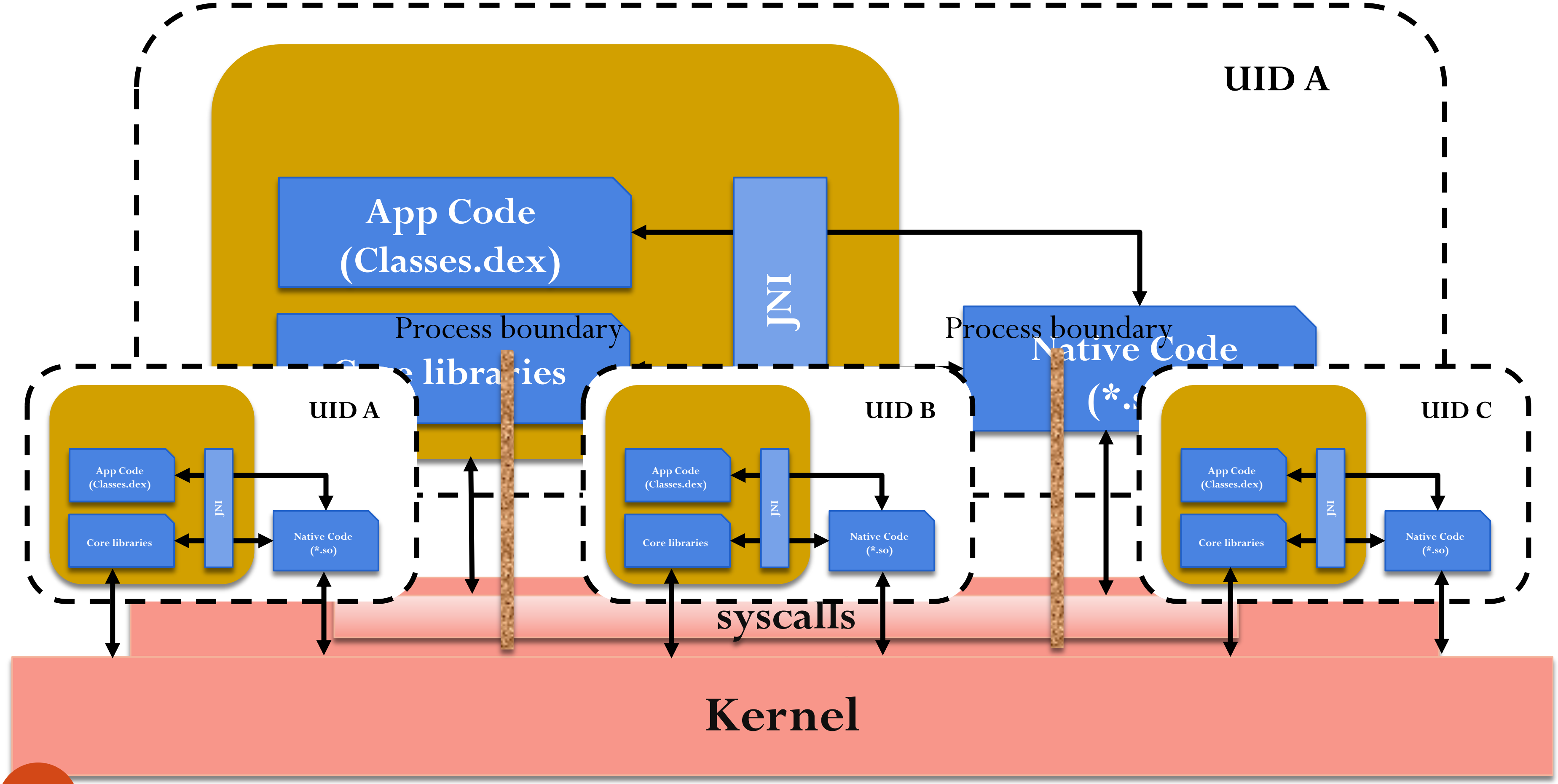
Application Isolation by Sandboxing

- Each Application is **isolated** in its own **environment**
  - **Applications** can access only its **own resources**
  - Access to **sensitive resources** depends on the **application's rights**
- **Sandboxing** is enforced by **Linux**



APPLICATION SANDBOX

- Isolation: Each installed App



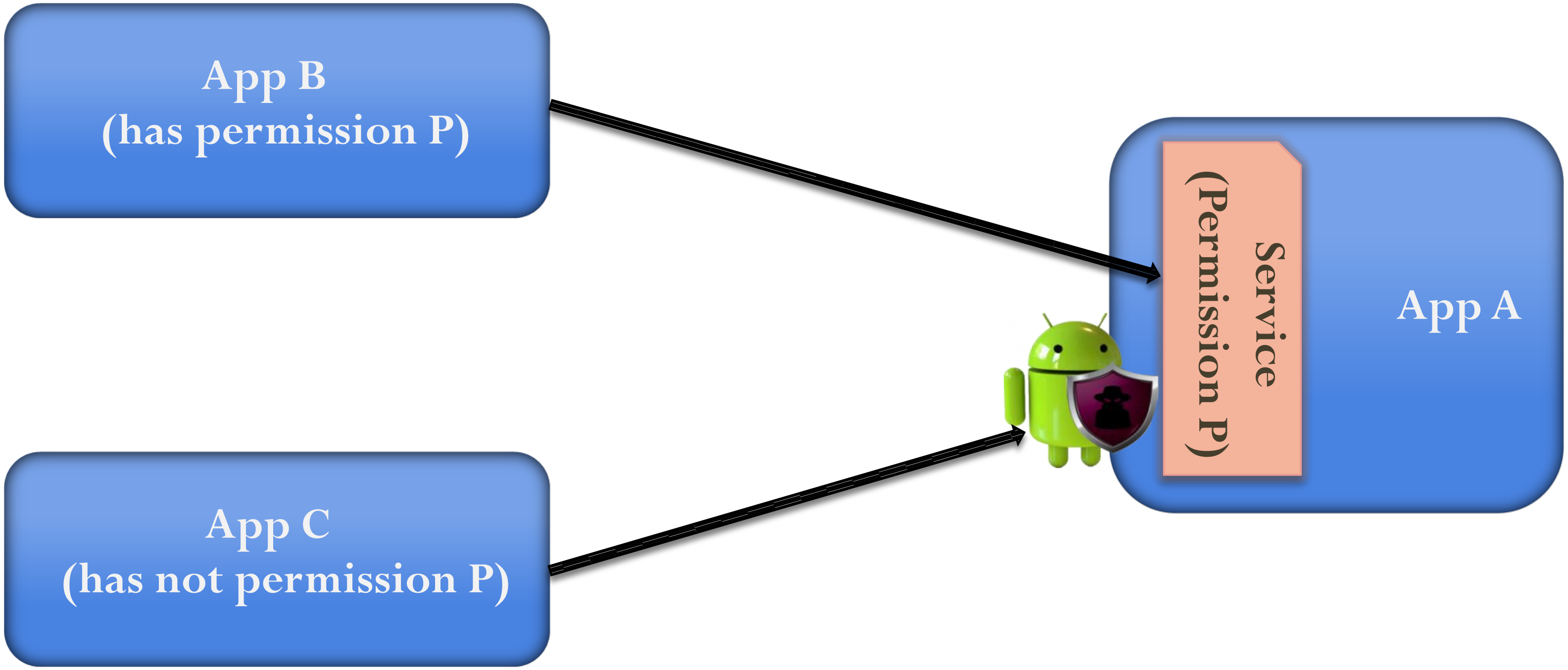
Android Security Architecture

- Package Integrity
- Sandboxing
- Permission and Least Privilege

Android Permission System

- **Access rights** in Android's application framework
  - Permissions are required to **gain** access to
    - System interfaces (Internet, send SMS, etc.)
    - System resources (logs, battery, etc.)
    - Sensitive data (SMS, contacts, etc.)
  - Currently more than 140 default permissions defined in Android
- Permissions are **assigned** to sandbox
- Application developers can also **define** their **own** permissions

Android Permission: Example

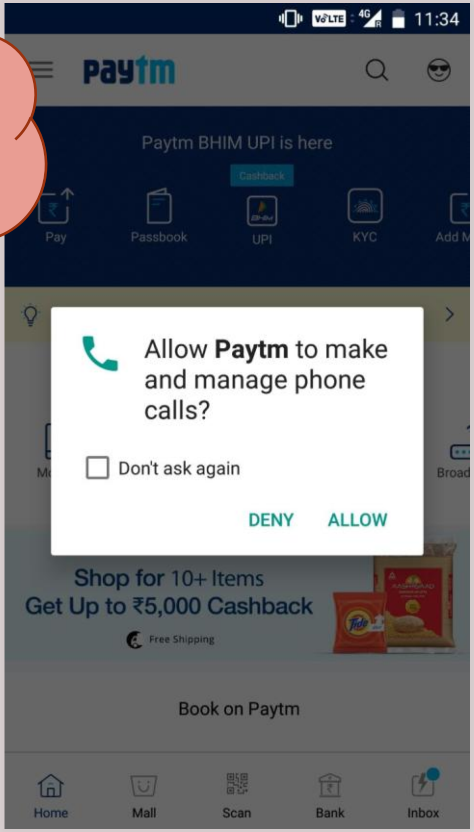


Permissions' Protection Level

- Normal
- Dangerous
- Signature
- SignatureOrSystem

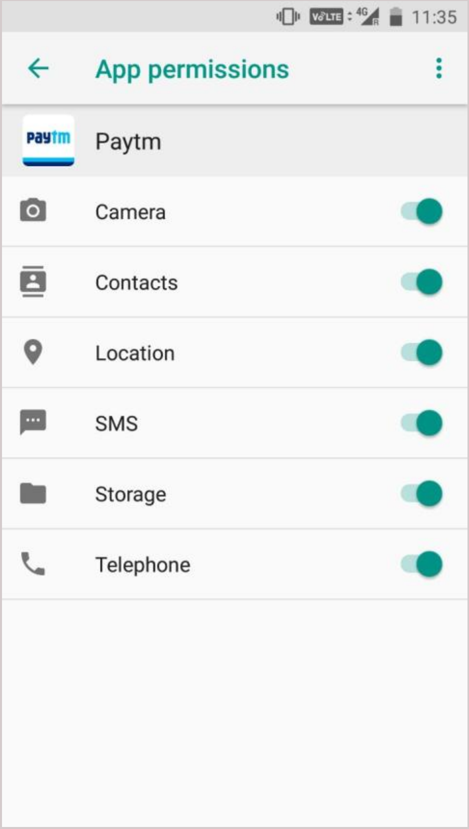
Dynamic Permissions (≥ Android 6.0)

- App developers must **check** if their apps hold required **dangerous** permission, otherwise request them at runtime
- User can **grant** permissions at runtime and also **revoke** once granted permissions again



Allow **Paytm** to make and manage phone calls?

□ Don't ask again



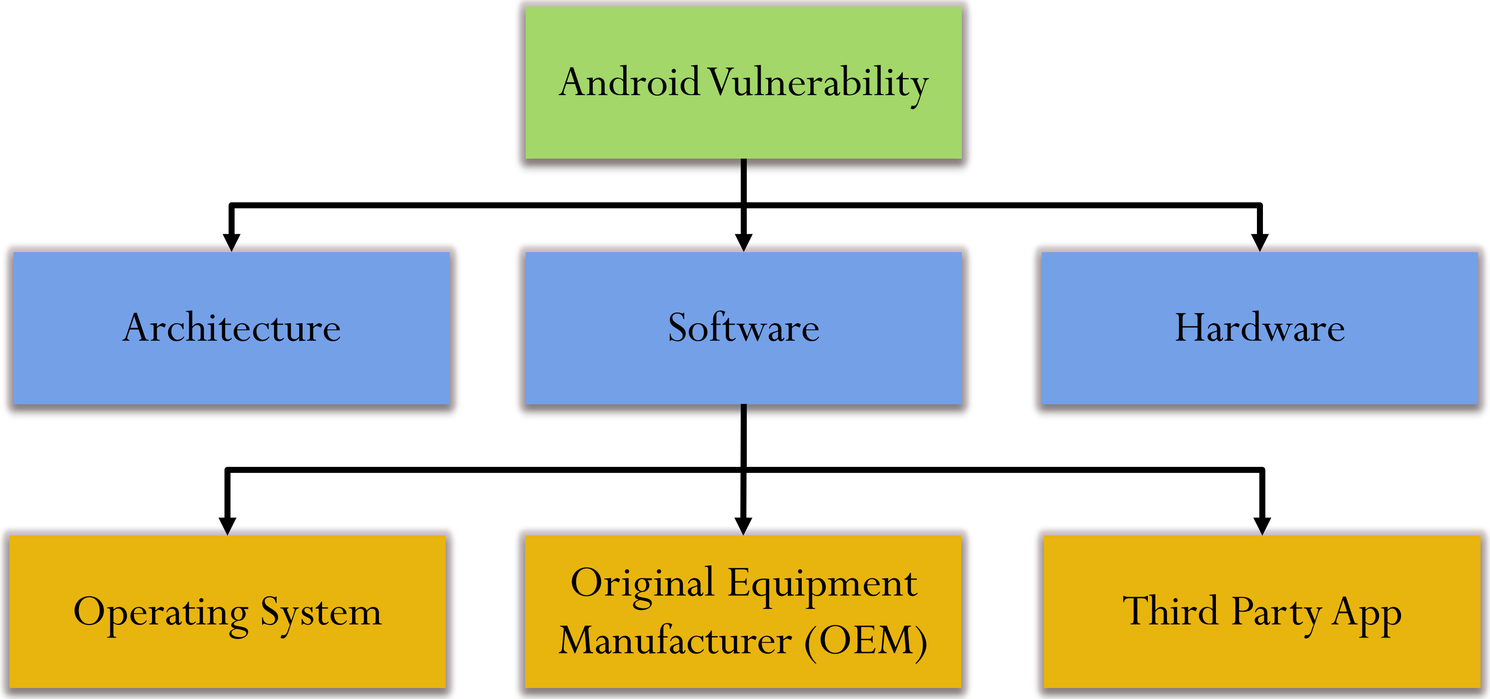
Paytm

Camera	●
Contacts	●
Location	●
SMS	●
Storage	●
Telephone	●

Android Vulnerabilities

- Architecture Based
- Software Based
- Hardware Based

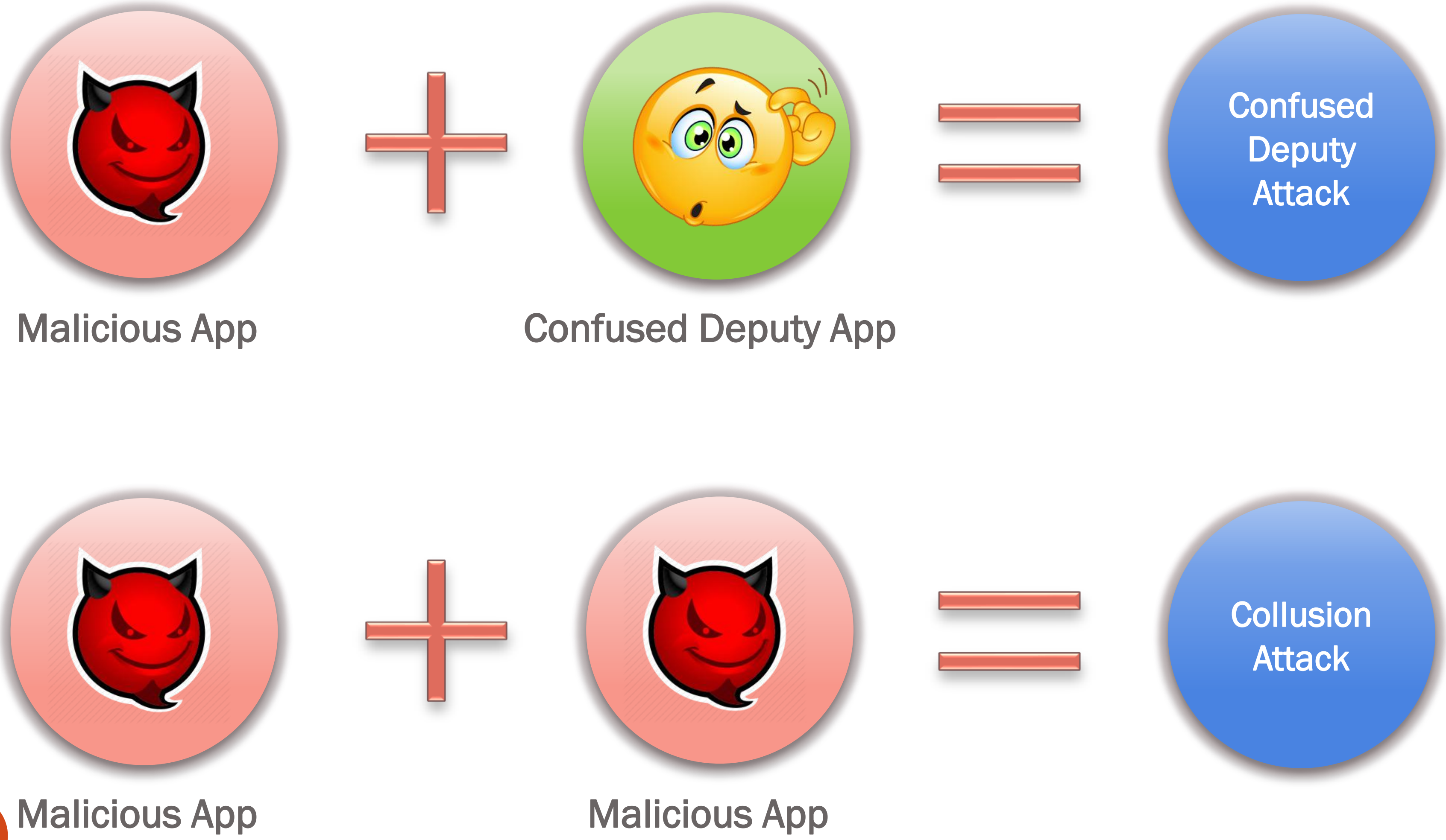
Vulnerability Classification



Android Vulnerability

- Architecture Based
- Software Based
- Hardware Based

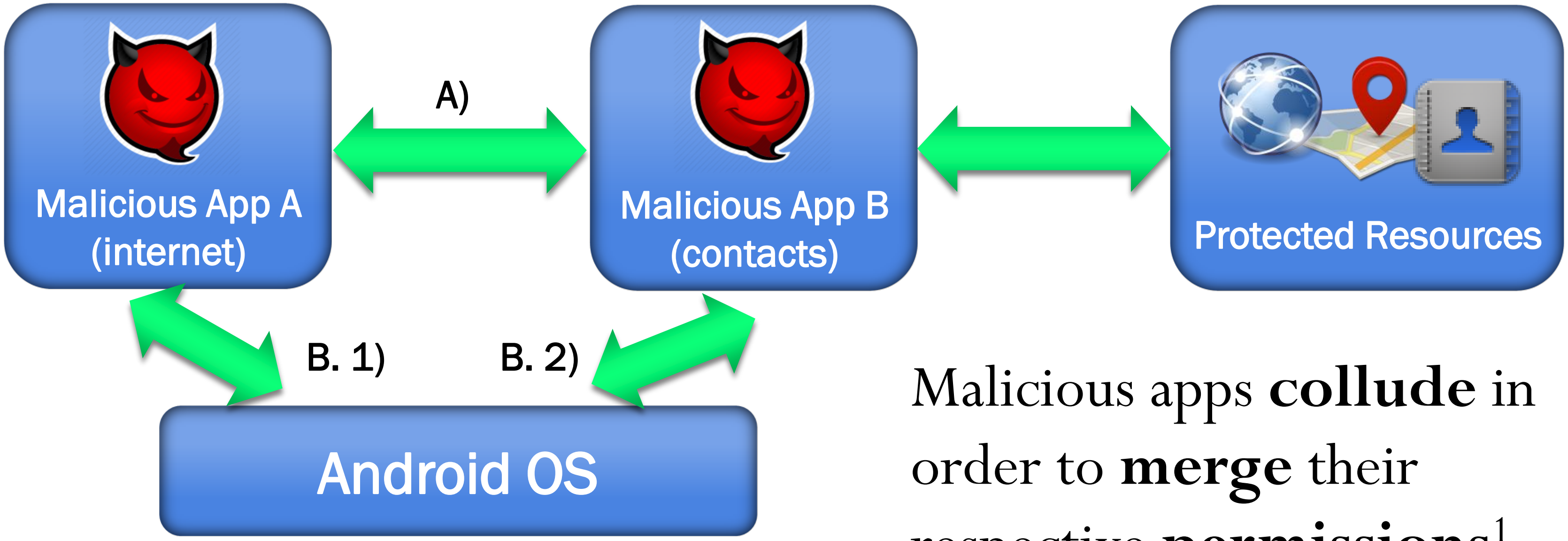
Application-Level Privilege Escalation Attack



Malicious App + Confused Deputy App = Confused Deputy Attack

Malicious App + Malicious App = Collusion Attack

Collusion Attack



Malicious apps **collude** in order to **merge** their respective **permissions**[1]

- Variants:
  - Apps communicate directly
  - Apps communicate via covert[2] channels in Android

[1] S. Karthick et al. "Android security issues and solutions," ICIMIA'17

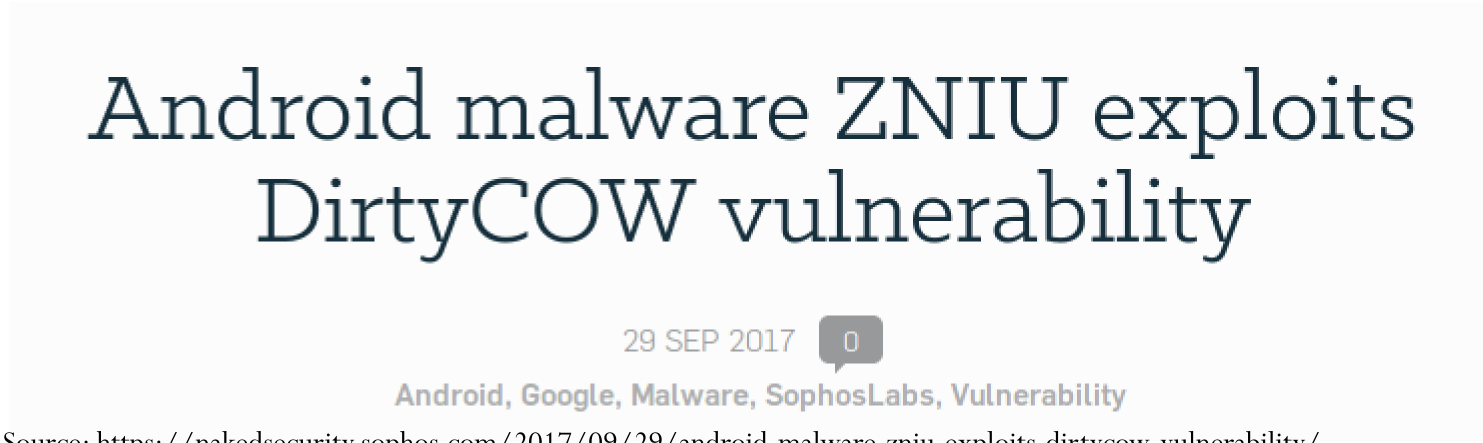
[2] C. Marforio et al. , "Analysis of the communication between colluding applications on modern smartphones," ACSAC'12

Android Vulnerability

- Architecture Based
- Software Based
- Hardware Based

DIRTY COW

- Existed in the Linux Kernel for **9 years**
- A **local** Privilege Escalation Vulnerability
- Exploits a race condition in the implementation of the **copy-on-write** mechanism
- Turns a **read-only** mapping of a file into a writable mapping



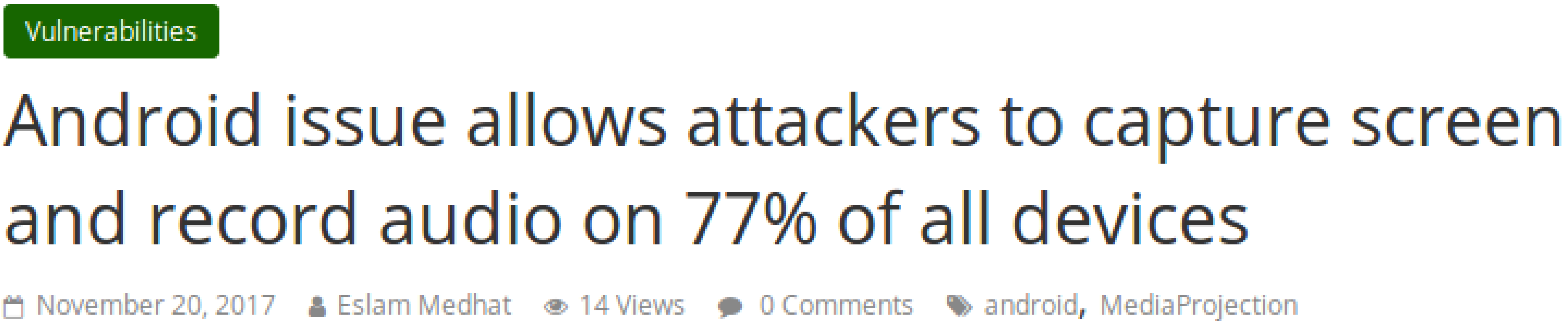
Android malware ZNIU exploits DirtyCOW vulnerability

29 SEP 2017 0

Android, Google, Malware, SophosLabs, Vulnerability

Source: https://nakedsecurity.sophos.com/2017/09/29/android-malware-zniu-exploits-dirtycow-vulnerability/

Media Projection Service Issue



Vulnerabilities

Android issue allows attackers to capture screen and record audio on 77% of all devices

November 20, 2017 Eslam Medhat 14 Views 0 Comments android, MediaProjection

Source: https://latesthackingnews.com/2017/11/20/android-issue-allows-attackers-to-capture-screen-and-record-audio-on-77-of-all-devices/

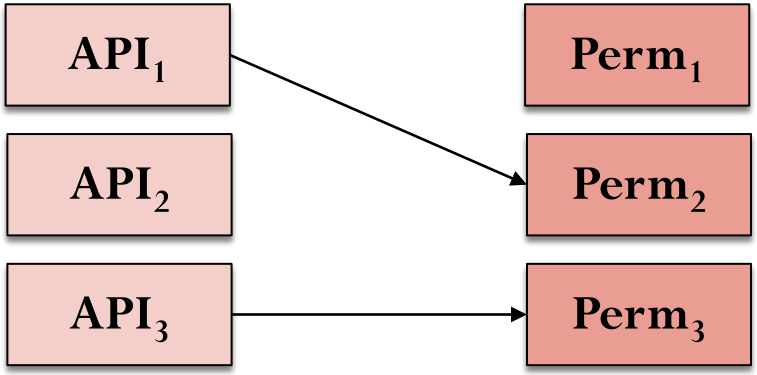
DYNAMIC PERMISSION[1]

- Is the context of the permission request **better recognizable**? ✗
- Invisible Permissions: 75.1%
  - Screen off (60%)
  - Invisible service (14.4%)
  - Background app (0.7 %)
- Non-indicative indicators: Location icon is **visible** for only **0.04%** of all **accesses** to location
- Around **8 requests/min**
  - Location: 10,960 / day
  - Reading SMS: 611 / day
  - Browser history: 19 / day

[1] P. Wijesekera et al., "Android permissions remystified: A field study on contextual integrity," SEC'15

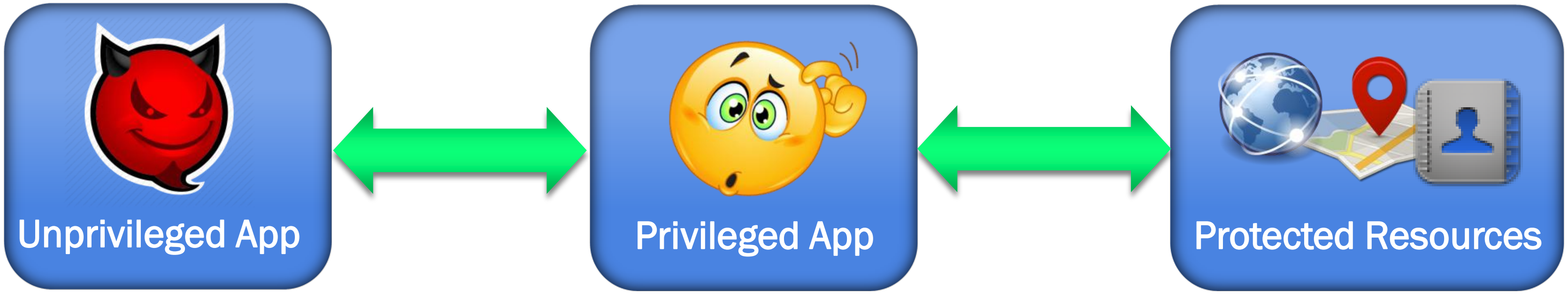
Over-privileged Apps[1]

- Many apps request permissions that their **functionality** does not **require**
- Suspected root cause: API **documentation/naming** convention
  - Solution: API Permissions Maps
    - **Can be integrated into lint tools**



1. M. Backes et al., "On Demystifying the Android Application Framework: Re-Visiting Android Permission Specification Analysis," SEC'16

Confused Deputy Attack



- A privileged app is fooled into **misusing** its privileges on behalf of another (malicious) **unprivileged app**[1]

- Example:
  - **Unauthorized** phone calls[2]
  - Various confused deputies in **system apps**[3]

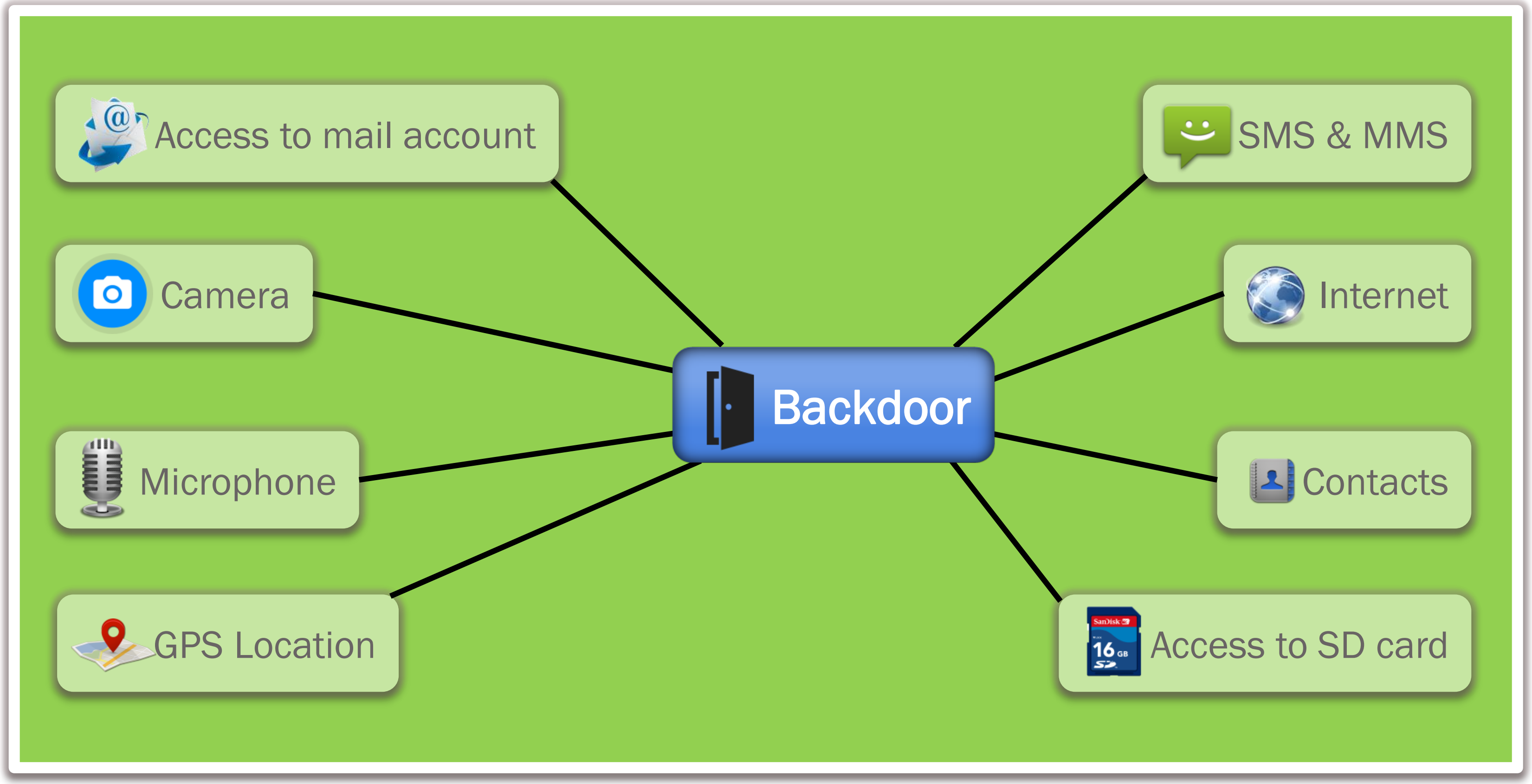
[1] S. Karthick et al. "Android security issues and solutions," ICIMIA'17

[2] W. Enck et al., "On lightweight mobile phone application certification," CCS'09

[3] A. Porter Felt et al., "Permission re-delegation: Attacks and defenses," SEC'11

Confused Deputy Introduce by OEMs[1]

- Several **confused deputies** found in Samsung devices' **firmware**
  - One deputy running with system privileges provided **root shell service** to any app

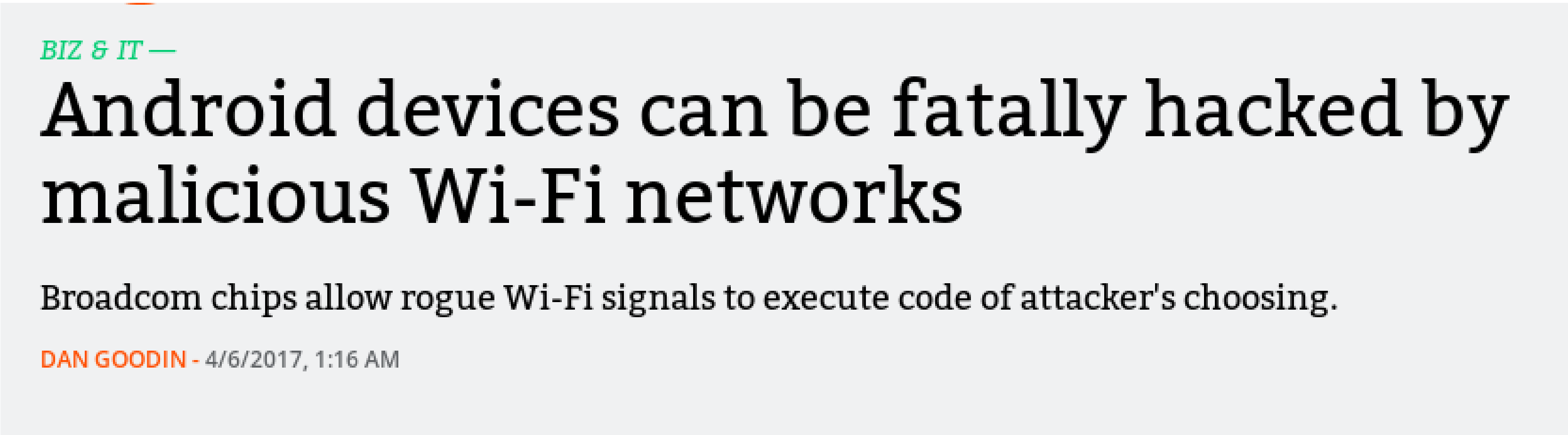


[1] A. Moulo, "Android OEM's applications (in)security and backdoors without permission"

Android Vulnerability

- Architecture Based
- Software Based
- Hardware Based

Broadcom Wi-Fi SoC Flaw



BIZ & IT —

Android devices can be fatally hacked by malicious Wi-Fi networks

Broadcom chips allow rogue Wi-Fi signals to execute code of attacker's choosing.

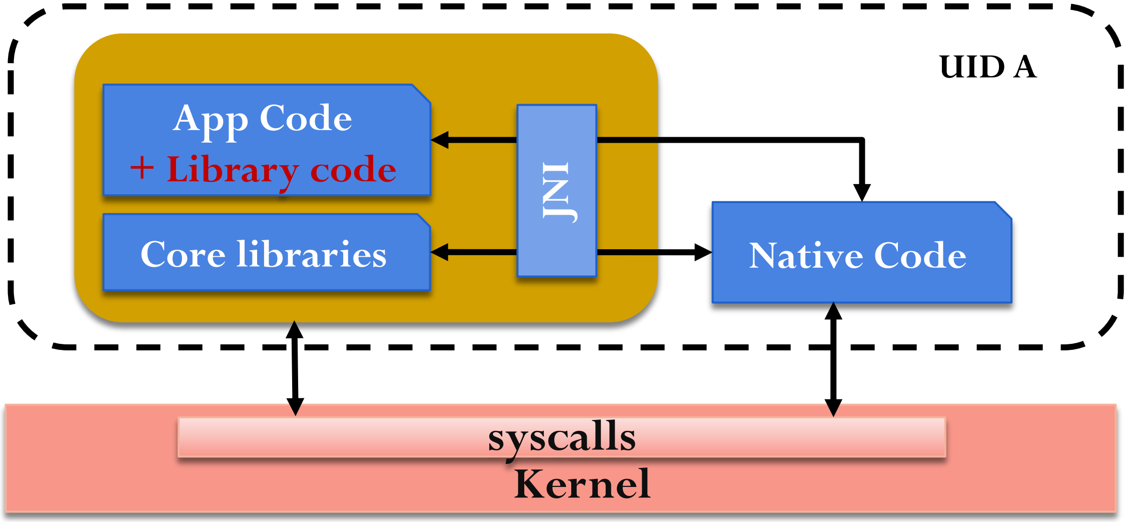
DAN GOODIN - 4/6/2017, 1:16 AM

Source: https://arstechnica.com/information-technology/2017/04/wide-range-of-android-phones-vulnerable-to-device-hijacks-over-wi-fi/

Advanced Threat

RISK OF 3RD PARTY LIBRARIES

- Have to be **included** in every app **package** that wants to use the lib
- Average **13 libs** per app in top **3000 apps** on Play[1]
- Library code, executed within the application process (same UID), **inherits** the host app's **privileges**
  - **nosecurity boundary!**



[1] M. Backes et al., "Reliable third-party library detection in android and its security applications," CCS'16

RISK OF 3RD PARTY LIBRARIES[1,2]

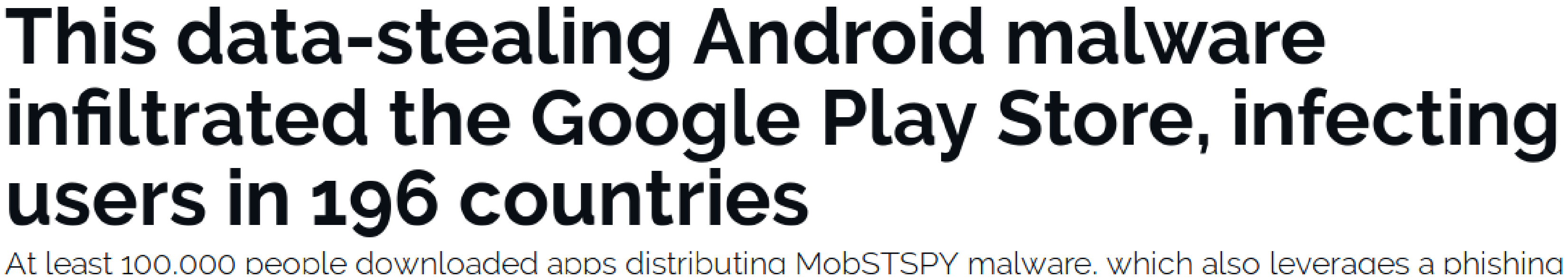
- Increase the host app’s **attack** surface
- **C**ompromise the device or violate the **user’s privacy**
- De-anonymization risks through quasi-identifiers
  - Has **access** to host app’s **local files** and **external files**
  - Can collect clear picture about the user
    - Gender, age, browsing history, user trajectories, etc.

1. S. Demetriou et al., ”Free for all! assessing user data exposure to advertising libraries on android,” NDSS’16, The Internet Society, 2016
2. S. Son et al., “What mobile ads know about mobile users,” NDSS’16, The Internet Society, 2016

MALWARE ANALYSIS

- Analysis Techniques and its Limitations

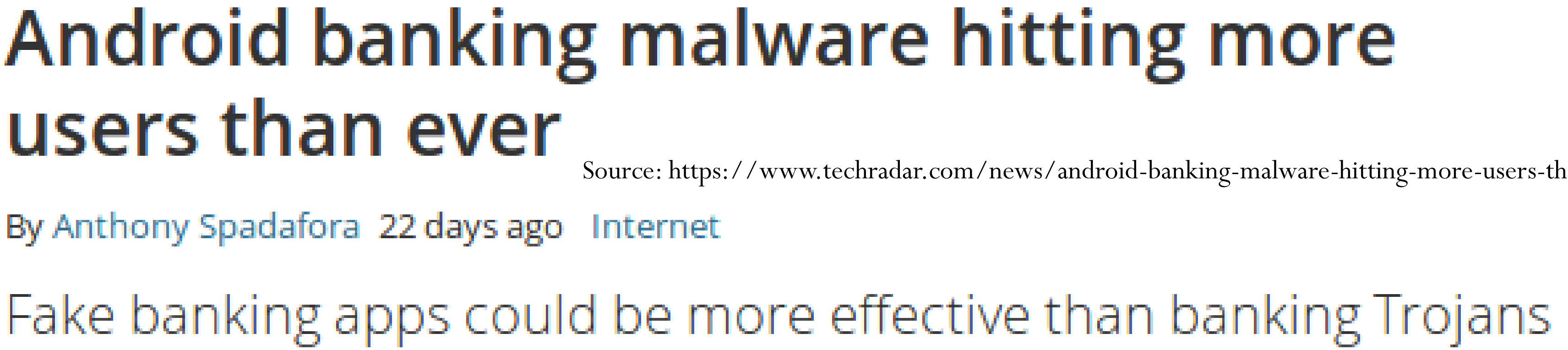
WHY MALWARE ANALYSIS?



This data-stealing Android malware infiltrated the Google Play Store, infecting users in 196 countries

At least 100,000 people downloaded apps distributing MobSTSPY malware, which also leverages a phishing

First Android Clipboard Hijacking Crypto Malware Found On Google



Android banking malware hitting more users than ever

Source: https://www.techradar.com/news/android-banking-malware-hitting-more-users-than-ever

By Anthony Spadafora 22 days ago Internet

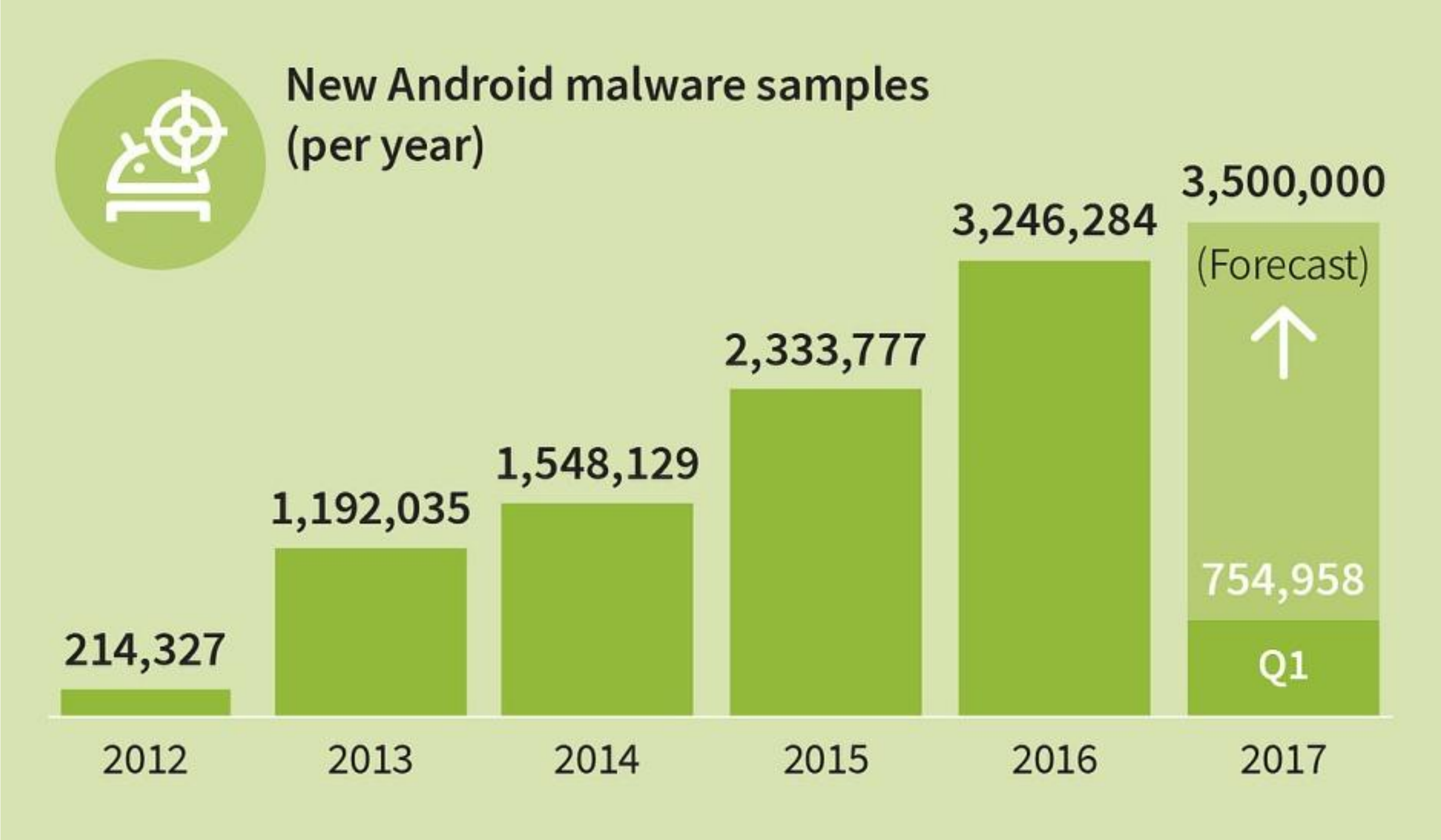
Fake banking apps could be more effective than banking Trojans

Several Popular Beauty Camera Apps Caught Stealing Users' Photos

February 04, 2019 Swati Khandelwal

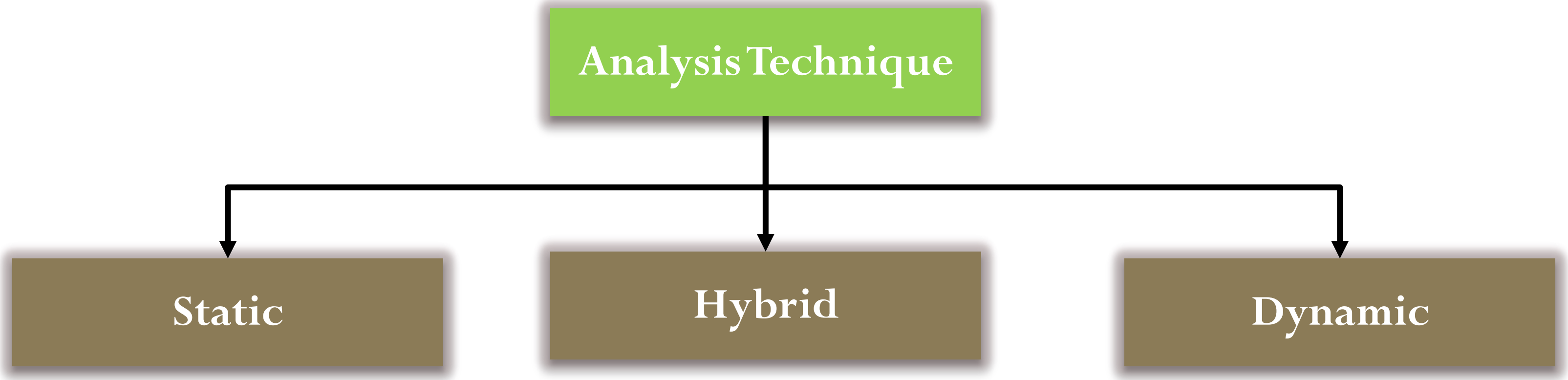
Source: https://thehackernews.com/2019/02/beauty-camera-android-apps.html

Malware Statistics

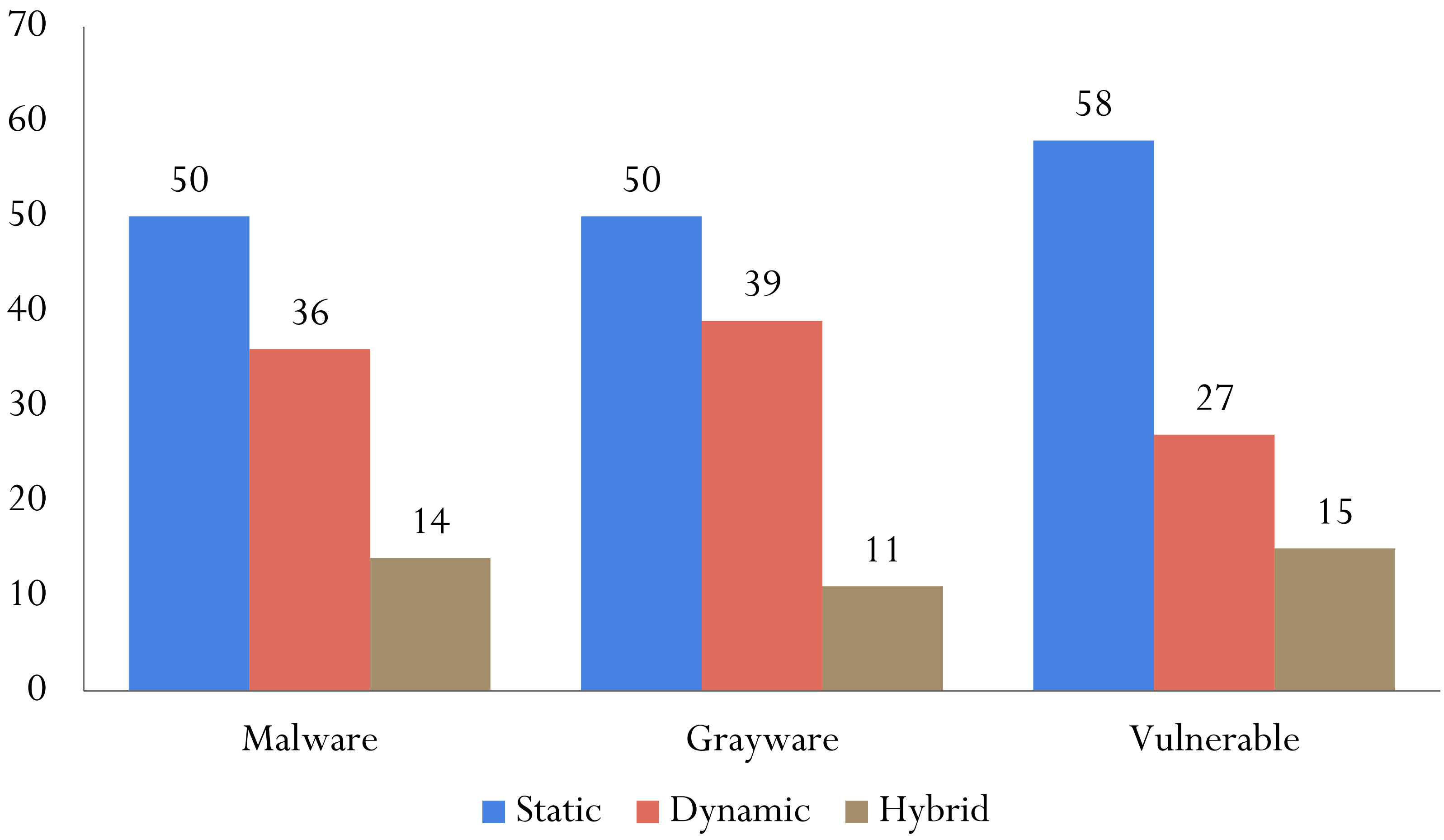


Source: https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day

Malware Analysis Techniques



Analysis Techniques used in Different Area[1]



[1] A. Sadeghi et al., "A Taxonomy and Qualitative Comparison of Program Analysis Techniques for Security Assessment of Android Software," in IEEE Transactions on Software Engineering, June 1 2017

https://github.com/skmtr1/techkriti-2019-CS-workshop-Android/

Questions..

Thank

You