Android Security Symposium
PhD School

Secure Copy Protection for Mobile Apps

Technische Universität München (TUM)
Faculty of Informatics (CS)

Chair for Operating Systems (F13)
Prof. Dr. Uwe Baumgarten
Boltzmannstr. 3
85748 Garching, Munich, Germany

About me

Nils T. Kannengiesser, M.Sc.
http://www.os.in.tum.de/personen/kannengiesser/

- **Research Associate** at the division for **Operating Systems / Prof. Dr. Uwe Baumgarten**
  - since March 2011
    - Computersysteme 2 (training, each summer)
    - Android Praktikum SS2011, WS11/12, SS12, WS12/13, SS13, WS12/13, […]
    - Advisor of lots of bachelor‘s and master‘s theses (see website for details)
    - **Guest speaker** at Texas A&M University (“Android security”) in April 12’ 13’ 14’

- Assistant lecturer of Android intensive courses (Kiel & College Station/US; English)
- Student assistant at Texas A&M University for Cisco Systems/Dallas, USA
- Student worker (Administrator, Student assistant, etc. […])
- I studied **Information Technology** in Kiel, Germany & College Station, USA (2 sem.)
- I‘m interested in Android, JavaME, Sensor networks and IT security
- My research area is **Android Security (copy-protection)**

Email: nils.kannengiesser@tum.de

Secure Copy Protection for Mobile Apps

Nils T. Kannengiesser
Technical University of Munich
Chair for Operating Systems (F13)
Munich, Germany
Nils.Kannengiesser@tum.de

Supervisor

Uwe Baumgarten
Technical University of Munich
Chair for Operating Systems (F13)
Munich, Germany
baumgaru@tum.de

Mentor

Sejun Song
University of Missouri-Kansas City
Computing and Engineering
Kansas City, USA
sjsong@umkc.edu

Secure Copy Protection for Mobile Apps

Students' contribution to my research topic by

Marius Muntean
Magnus Jahnen
Michael Bichlmeier
Patrick Bernhard
David Ellermann
Norbert Schmidbartl
Janosch Maier
Philipp Schreitmueller
Ioana Negoita
Patrick Bernardt
David Ellermann
Ozan Pekmezci
[…]

A huge
„Thank you"!

Content

- Introduction
- Proposals
- Findings
- Ongoing research
- Temporary results
  - Conclusion/Future Work
- Related work

Introduction

- I started to **look for a dissertation topic in 2012**, while supervising student projects on data security using secure elements for Android

- In **2013 Android's security came to my focus**, while discovering initial issues in regard to the easy reengineering possibilities of apps in conjunction with copyright protection by other researchers/hackers (**L**icense **V**erification **L**ibrary hacking)

Example: LicenseValidator.smali

```
[…] .field private static final LICENSED:I = 0x0
.field private static final LICENSED_OLD_KEY:I = 0x2
.field private static final NOT_LICENSED:I = 0x1
[…]
.sparse-switch
0x0 -> :sswitch_d3
0x1 -> :sswitch_de
[…]
```

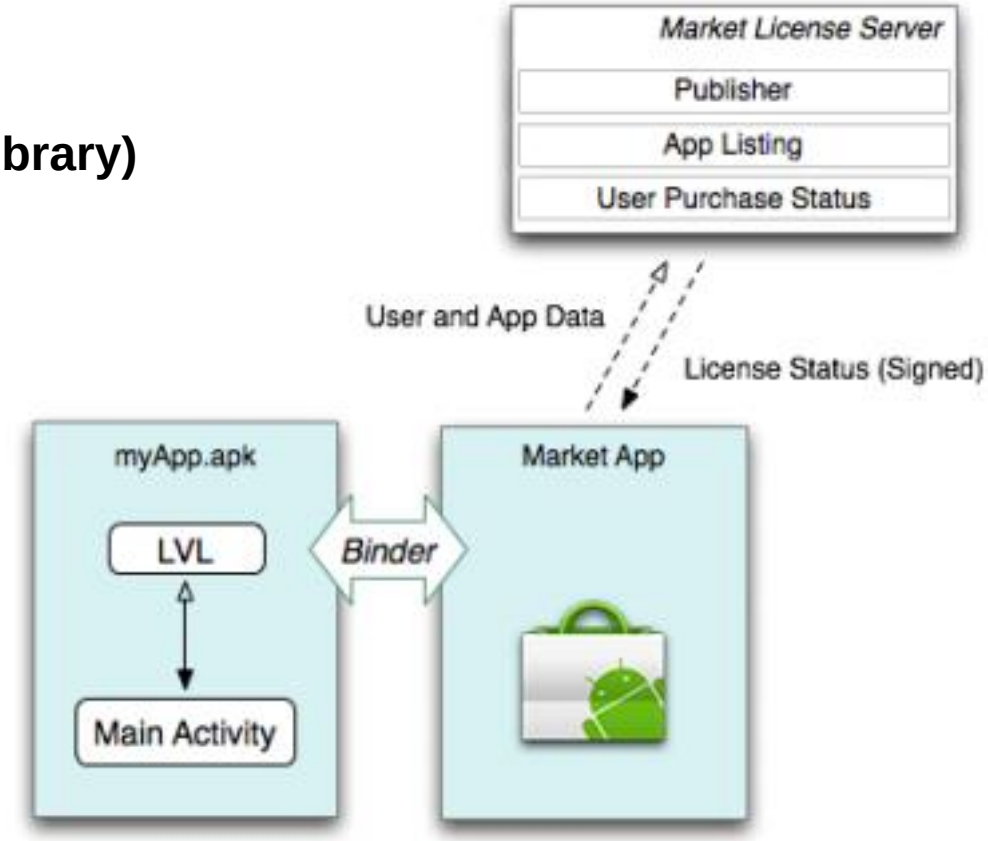
Ref. Example Source Code - http://www.androidpolice.com/2010/08/23/exclusive-report-googles-android-market-license-verification-easily-circumvented-will-not-stop-pirates/

Introduction

Excursus: LVL

(License Verification Library)

Google provides example code that can be integrated into an application to include basic license management.



Ref. http://developer.android.com/google/play/licensing/overview.html

Introduction

- **Why are copyright protection and obfuscation techniques actually important?**
  - Developers face the issue of **lost revenue**
  - **Others might earn money** with it by repacking the app and exchanging ad-IDs
  - Customers might **get infected** by repackaged apps with trojans etc.
  - The discovered techniques and **proposed methods may protect other technologies like In-App-Billing**, too. (Notice: In-App-Billing is vulnerable to presented techniques, too. We even discovered further possibilities that cannot be disclosed right now.)
  - **[…]**

Ref. MA Patrick Bernard (not published) ; W. Zhou, Y. Zhou, X. Jiang and P. Ning, "Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces"

Introduction

- We investigated the reengineering issues further and **found similar problems in recent LVL versions** that use even signed replies for additional security.

- We found a way to manipulate LVL's communication on the fly to trigger a valid license, by **intercepting calls** to exchange parameters **used for the license verification**. We **faked public and private keys** and used the **Xposed framework** for this purpose. It works with any application using the **default examples codes on rooted devices**.

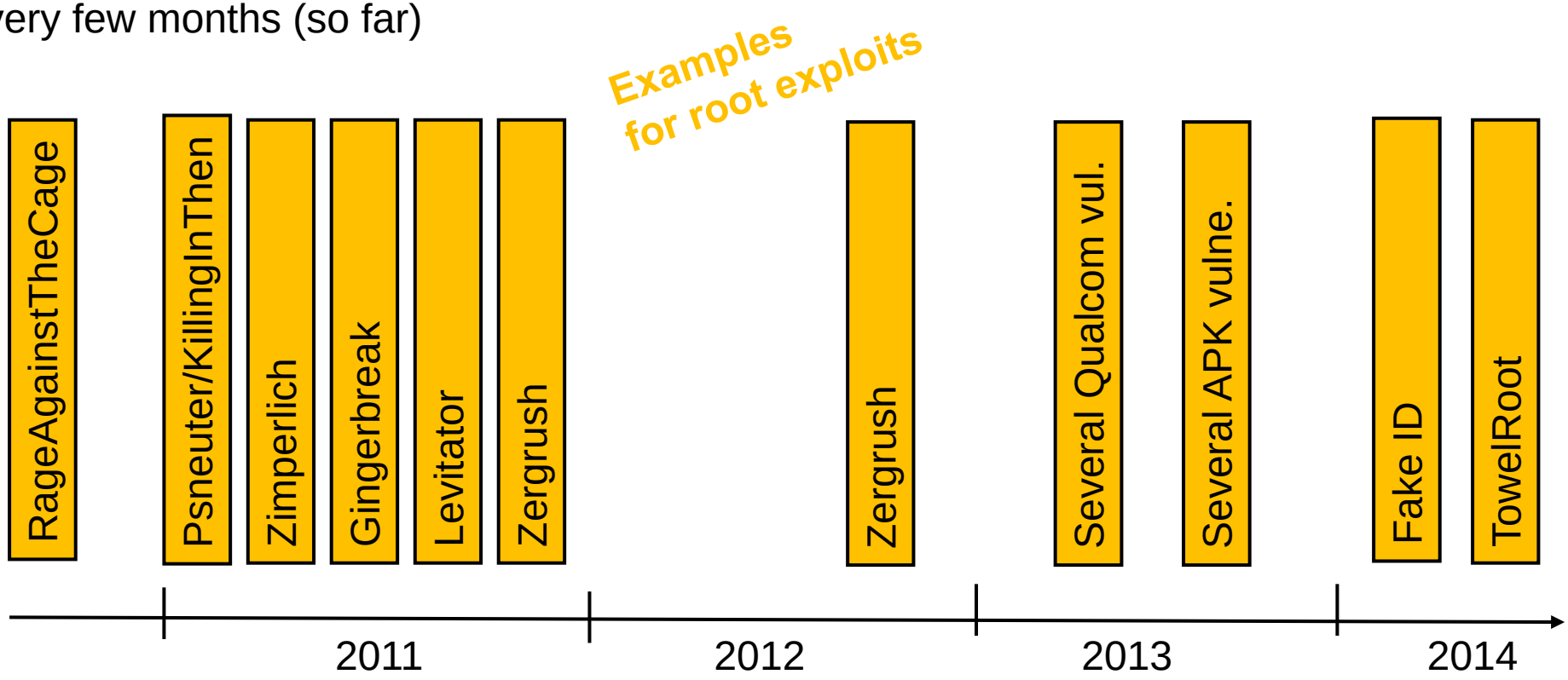
- Google Android Security was notified about this issue by Sep. 5th 2014 and classified it in a response "as a low severity issue since it requires the device to be rooted"

Notice: Almost any Android device may be rooted and one can discover root exploits every few months!

Ref. Marius Muntean / discussions / master's thesis (not published)

Introduction

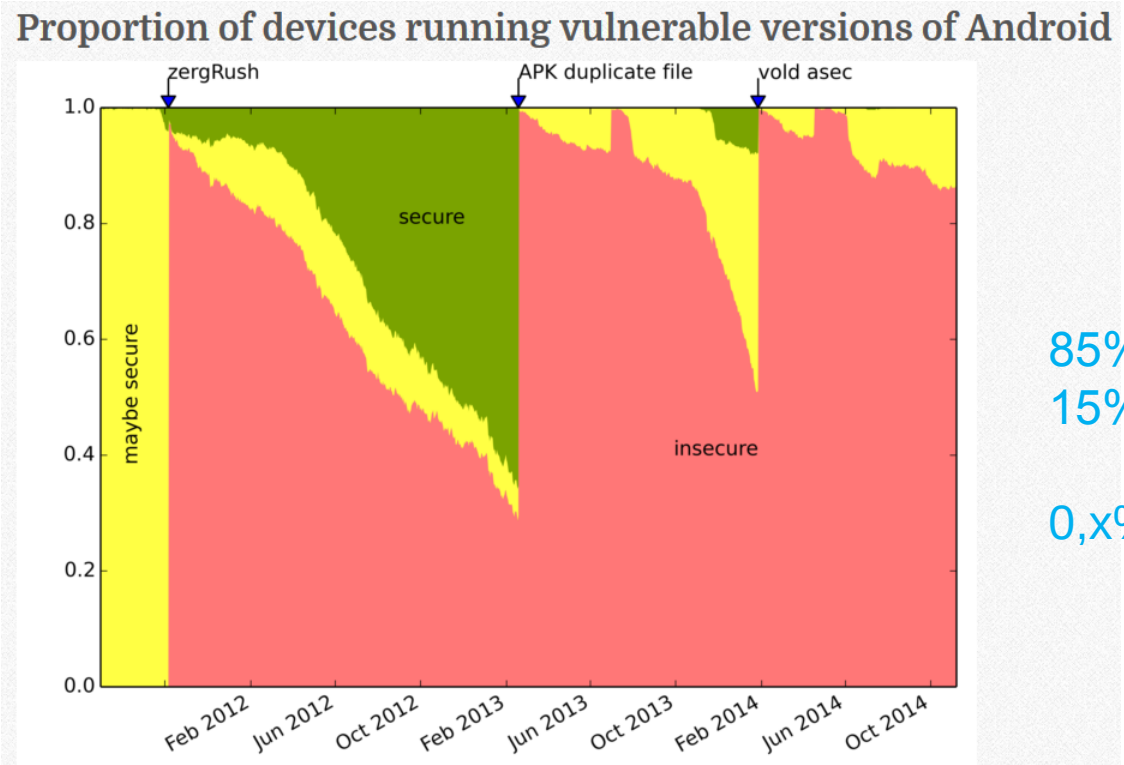
Notice: Almost any Android device may be rooted and one can discover root exploits every few months (so far)



Ref. AP WS Kirschner, Schleemilch, Smarzly (not published) ; BA Janosch Maier

Introduction

Notice: Almost any Android device may be rooted and one can discover root exploits every few months (so far)



85% “insecure”
15% “maybe secure”

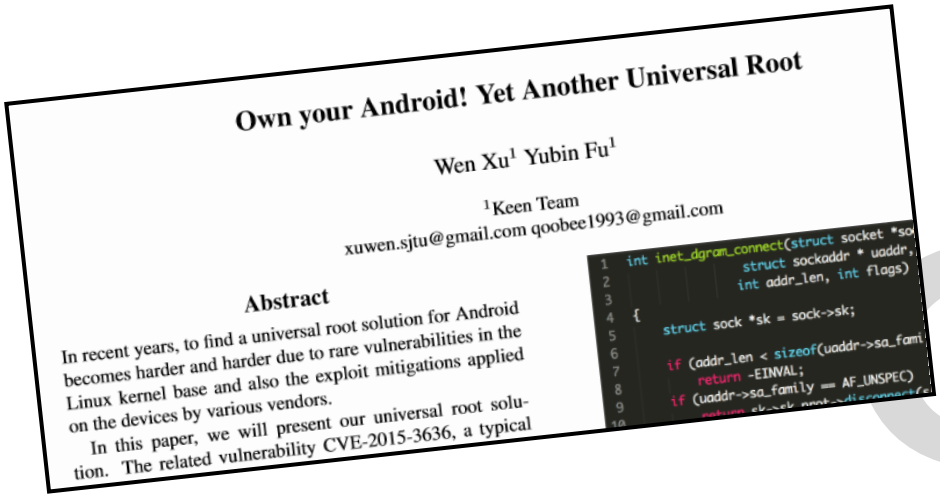
0,x% secure ?

Ref. http://www.androidvulnerabilities.org/index.html

Introduction

Notice: Almost any Android device may be rooted and one can discover root exploits every few months (so far)

2015 ? **“Stagefright”**



Own your Android! Yet Another Universal Root

Wen Xu[1] Yubin Fu[1]

[1]Keen Team

xuwen.sjtu@gmail.com qoobee1993@gmail.com

Abstract

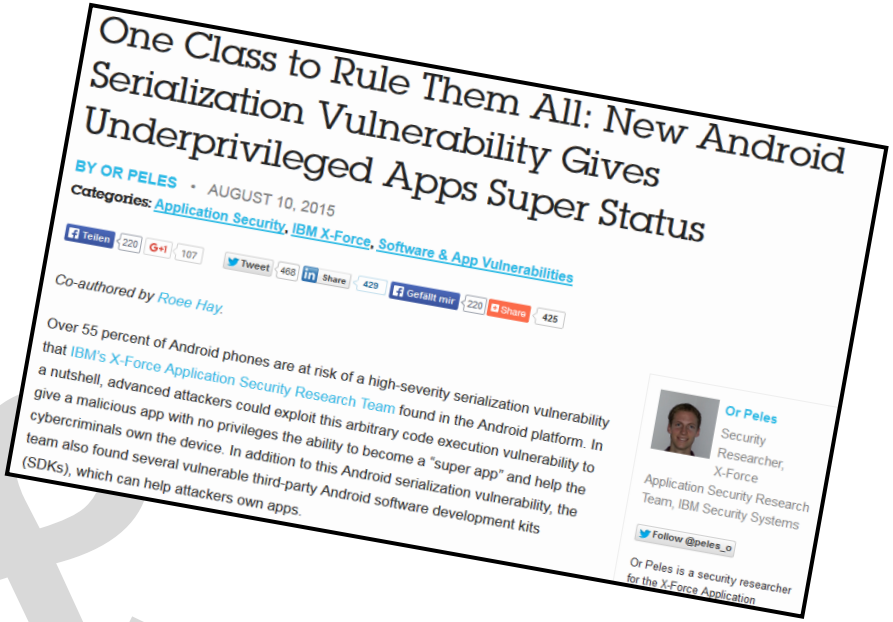
In recent years, to find a universal root solution for Android becomes harder and harder due to rare vulnerabilities in the Linux kernel base and also the exploit mitigations applied on the devices by various vendors.

In this paper, we will present our universal root solution. The related vulnerability CVE-2015-3636, a typical

```
int inet_dgram_connect(struct socket *so
                       struct sockaddr * uaddr,
                       int addr_len, int flags)
{
    struct sock *sk = sock->sk;

    if (addr_len < sizeof(uaddr->sa_fami
        return -EINVAL;
    if (uaddr->sa_family == AF_UNSPEC)
```

&



One Class to Rule Them All: New Android Serialization Vulnerability Gives Underprivileged Apps Super Status

BY OR PELES • AUGUST 10, 2015

Categories: Application Security, IBM X-Force, Software & App Vulnerabilities

Teilen 220 G+1 107 Tweet 468 Share 429 Gefällt mir 220 Share 425

Co-authored by Roee Hay.

Over 55 percent of Android phones are at risk of a high-severity serialization vulnerability that IBM's X-Force Application Security Research Team found in the Android platform. In a nutshell, advanced attackers could exploit this arbitrary code execution vulnerability to give a malicious app with no privileges the ability to become a "super app" and help the cybercriminals own the device. In addition to this Android serialization vulnerability, the team also found several vulnerable third-party Android software development kits (SDKs), which can help attackers own apps.

Or Peles
Security Researcher, X-Force Application Security Research Team, IBM Security Systems

Follow @peles_o

Or Peles is a security researcher for the X-Force Application

Ref. https://www.blackhat.com/docs/us-15/materials/us-15-Xu-Ah-Universal-Android-Rooting-Is-Back-wp.pdf
https://securityintelligence.com/one-class-to-rule-them-all-new-android-serialization-vulnerability-gives-underprivileged-apps-super-status/#.VctEZfntlBd

Introduction

Summary:

It's fair to assume that **many devices may be rooted "legally" or by exploit** and therefore reveal access to APKs' private files, encrypted communication by "method call interception" etc.

This is a **big issue for copyright protection**, since **it's about hiding (license) information somehow from users/attackers**.

Initial Research question:
How can be achieved that an app can only be used on "valid devices" (cf. license)?

Sub-questions:
Is there any solution for securing data, communication etc.?

Ref. Marius Muntean's master's thesis (not published)

Introduction

Is there any solution for securing data, communication etc.?

Yes, with different security levels:

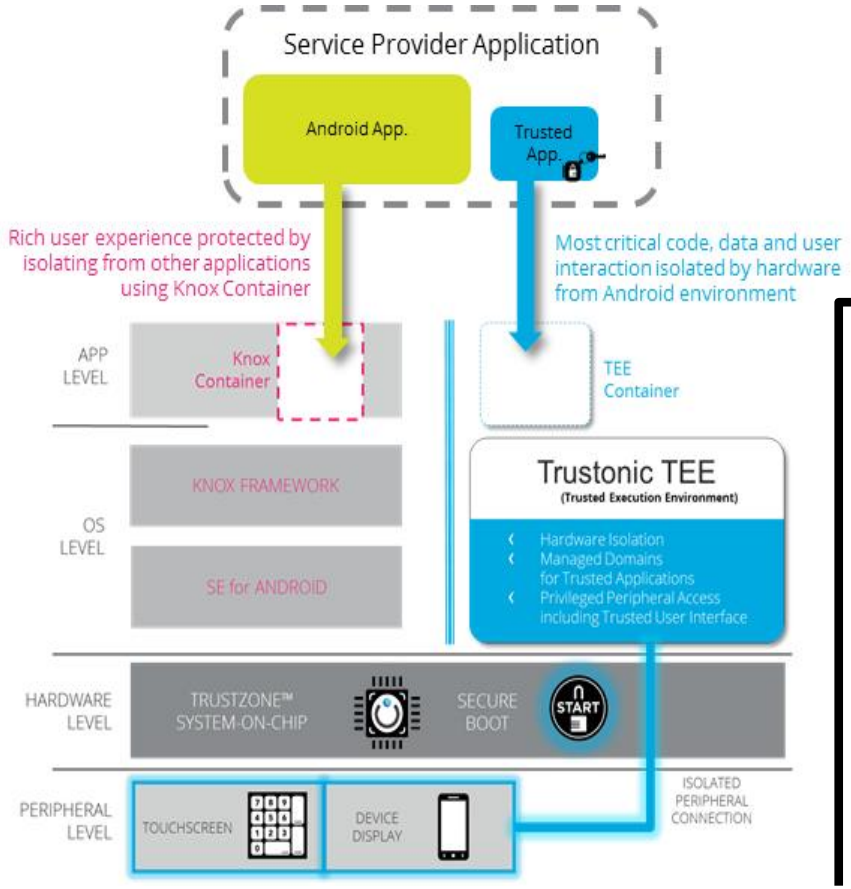
new hardware	plugin devices	software modification
TEEs	**SEs**	**Enhan. OS**
ARM's TrustZone	G&D's MSC	NSA's SEAndroid
Samsung's Trustonic for KNOX	Swissbit's PS-100u	Fraunhofer's TrustDroid

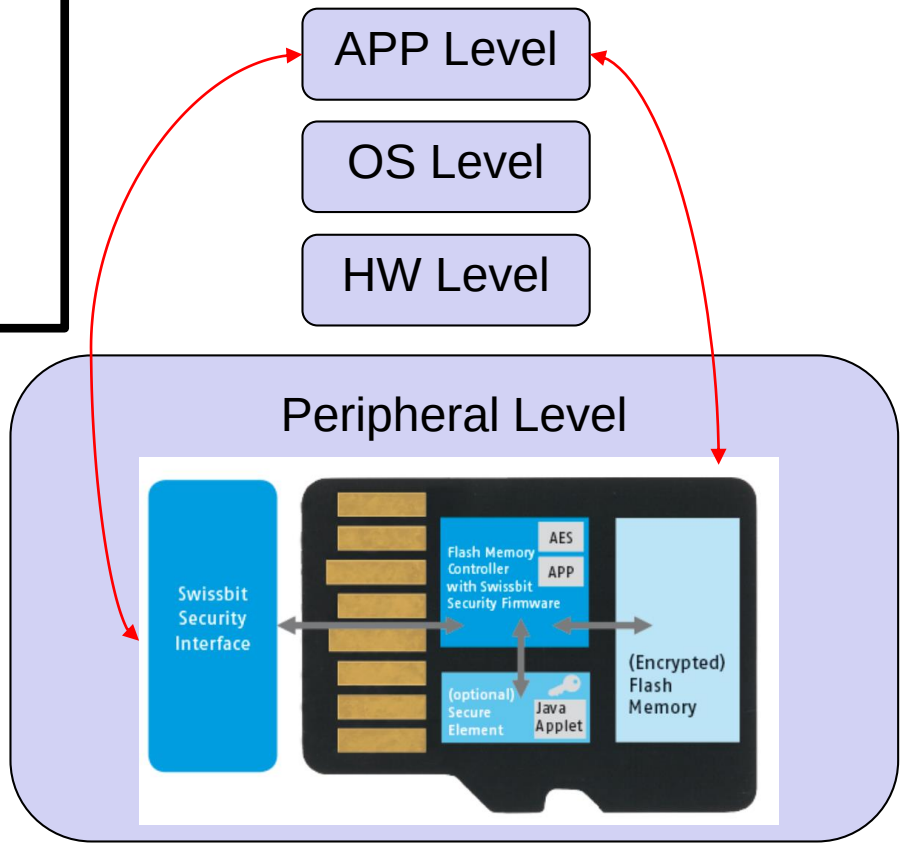
Ref. AmiEs Symposium

Introduction

Excursus TEEs and SEs "in a nutshell"

Trustonic for Samsung KNOX





Ref. Graphics http://www.swissbit.com/products/security-products/overwiev/security-products-overview/ and https://www.trustonic.com/products-services/trustonic-for-samsung-knox

Introduction

Is there any solution for securing data, communication etc.?

available to many devices / existing cooperations

Yes, with different security levels:

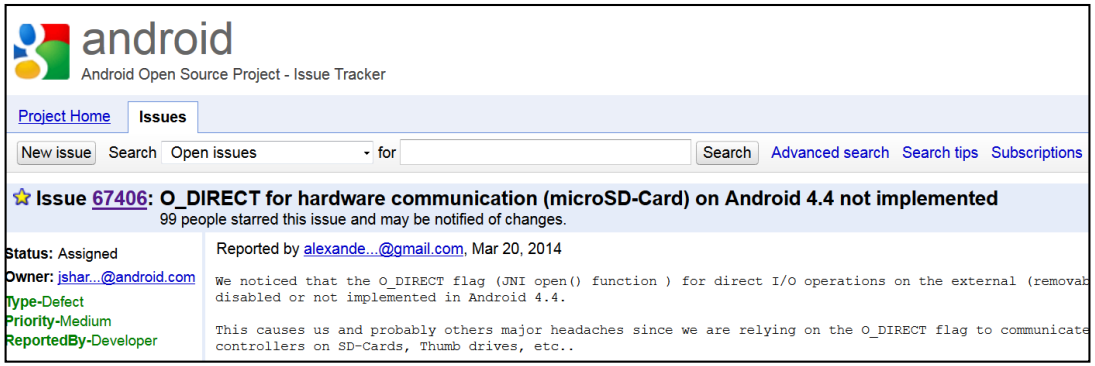
new hardware	plugin devices	software modification
TEEs	**SEs**	**Enhan. OS**
ARM's TrustZone	G&D's MSC	NSA's SEAndroid
Samsung's Trustonic for KNOX	Swissbit's PS-100u	Fraunhofer's TrustDroid

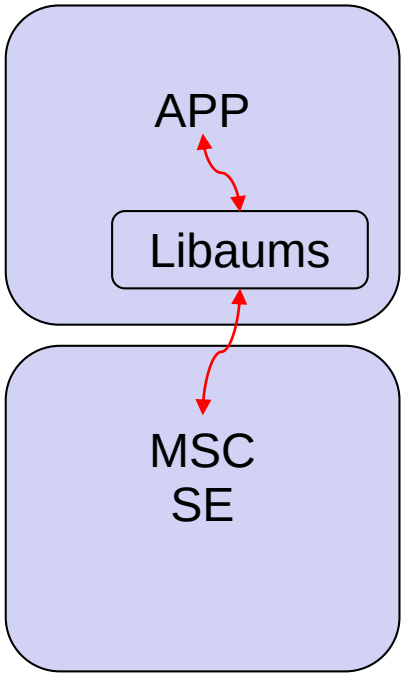
Ref. AmiEs Symposium

Introduction

→ available to many devices / existing cooperations

→ USB-OTG required (Android 3.x / MicroSD-MicroUSB-Adapter)
→ O_DIRECT required (issues below)
→ Mounting USB storage devices required (root rights necessary)



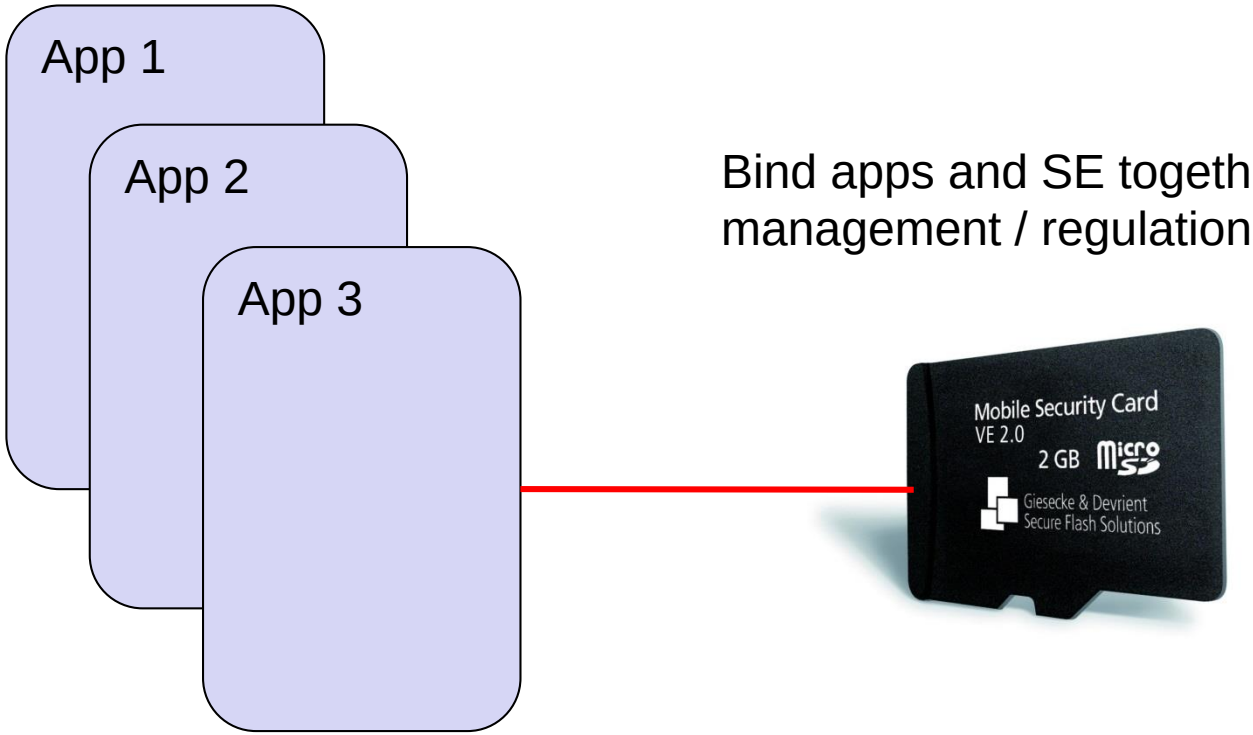


For this reason **we developed Libaums** (Library to access USB Mass Storage Devices) that enables O_DIRECT access for files, while requiring no root rights. It allows the communication with SE's applets finally. https://github.com/mjdev/libaums

Ref. https://code.google.com/p/android/issues/detail?id=67406 ; BA Magnus Jahnen

Proposals

cf. “Dongle” / Desktop World

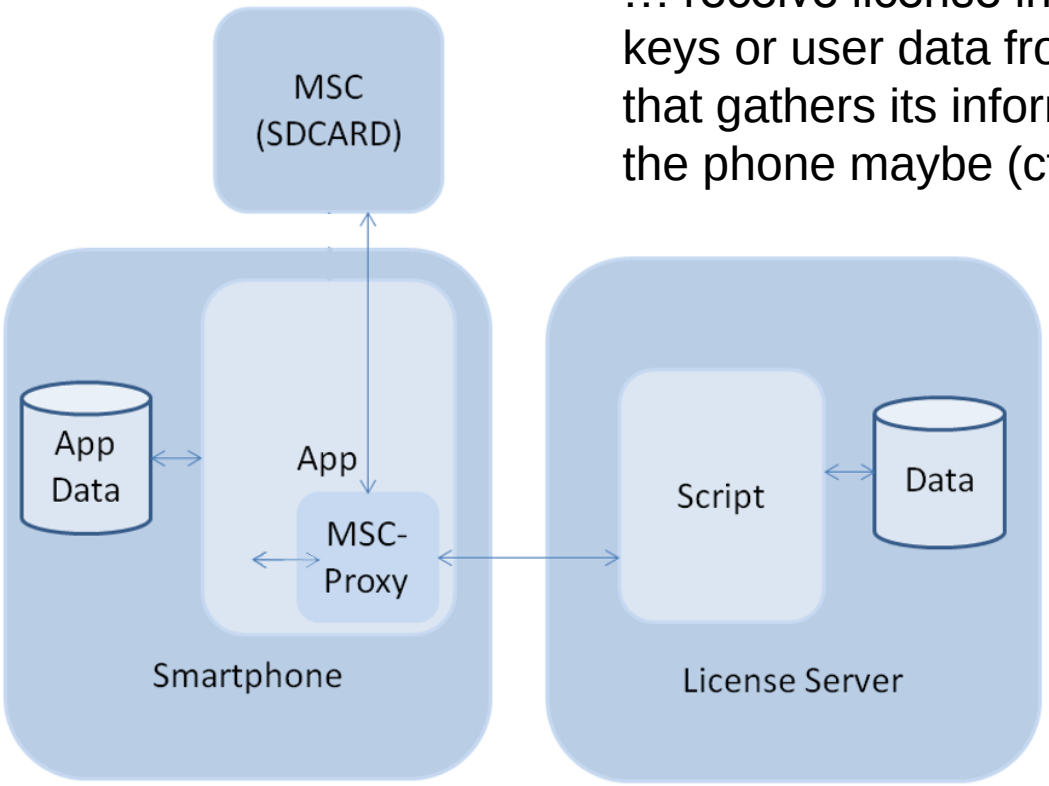


Bind apps and SE together to ensure license management / regulations

Ref. Graphic http://www.gi-de.com/de/about_g_d/press/press_releases/global_press_release_7234.jsp

Proposals

… receive license information, decryption keys or user data from the MSC/SE that gathers its information from a remote server or the phone maybe (cf. device information)

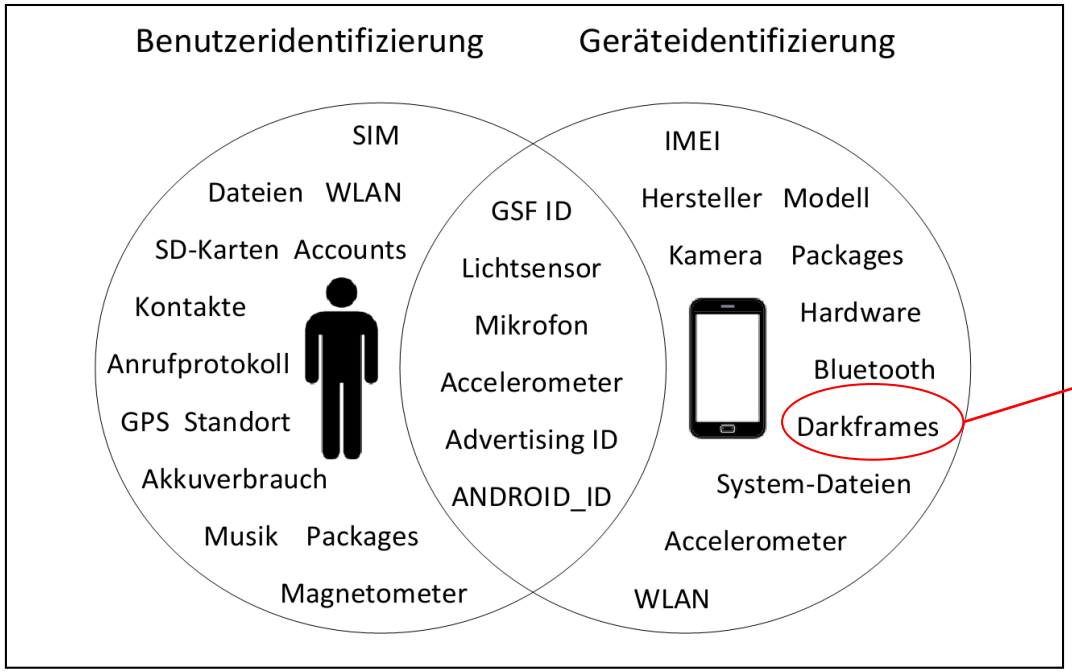


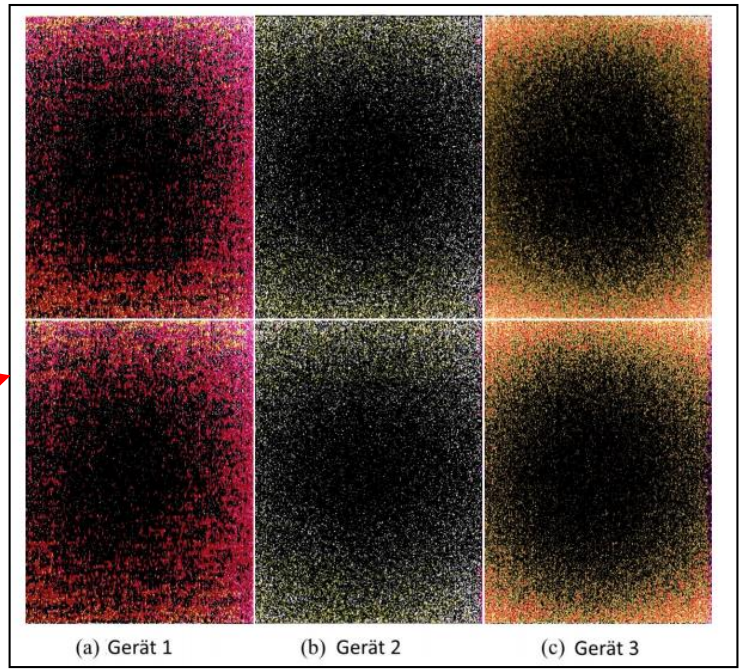
Ref. AmiEs Symposium

Proposals

… using a secure element and an app.

- **Device and User identification** and act according to **predefined rules** (cf. license server) that are mirrored by the SE (cf. advantage: no internet conne.)





Ref. AmiEs Symposium ; MA Norbert Schmidbartl / http://www.os.in.tum.de/fileadmin/w00bdp/www/Lehre/Abschlussarbeiten/Thesis_Final._Schmidbartl.pdf

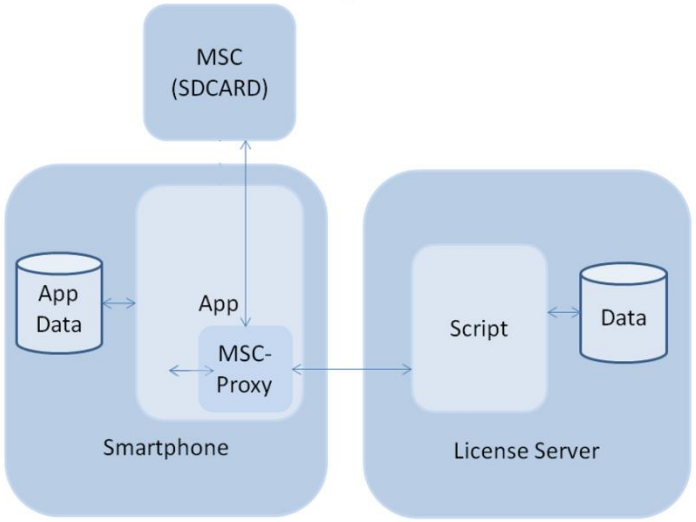
Proposals

… using a secure element and an app.

- **Content protection** using **keys received by the SE** (server) that differentiate per version
- **String protection** using keys received by the SE (server) that differentiate per version
- **URL/API protection** by using One-Time-URLs that are generated by the server/SE on request and loose validity after a certain time
- **Execution obfuscation by using reflections in conjunction with the SE** to receive the actual “execution roadmap” during runtime. A static analysis (reengineering) will not work anymore.

Ref. AmiEs Symposium

Findings



Performance tests of a sample implementation for testing purposes reveal a limited usage possibility due to the slow performance of SEs (10khz CPU).

We were required to adapt some ideas to lower the SE requests, while speeding up the app this way.

Performance information:
APP <> SE: 250 Bytes / req. / 200ms

Operation	Description	Time
SEtoServerCommunicationAsyncTask	Authentication of server	11.2 s
ReadDataAsyncTask	Read key from SE	0.9 s
WriteDataAsyncTask	Write key on SE	1 s
GetAppDataIP method	Get an IP that is saved in a variable on SE	0.2 s
GetLinksAsyncTask	Download related links from server	1.9 s
Decrypt method	Decrypts a String	0.001s
All operations before original app	Time until original app main activity	17.8 s

Table 2. The duration of some Android operations are shown.

File size	Time to write to the SE	Time to read from the SE
1 kB	2.4 seconds	1.5 seconds
2 kB	4.4 seconds	3.7 seconds
7 kB	14 seconds	12.4 seconds
10 kB	~ 20 seconds	~ 18 seconds
20 kB	~ 40 seconds	~ 36 seconds
35 kB	~ 70 seconds	~ 63 seconds

Ref. AmiEs Symposium ; BA Michael Bichlmeier (not published) ; BA Ozan Pekmezci (not published)

Findings

Security benefit? Sure !

Using secure elements increases the security and combining further methods (obfuscation etc.) adds additional security layers against reengineering attacks.

Nevertheless there are still ways for further improvement that we are analyzing these days.

Native LVL?

TEEs?

Reinvent G-Play?

Ref. AmiEs Symposium ; MA Patrick Bernard (not published)

ongoing research

C

vs.

Java

We are investigating the **additional security of native code** these days.

Current assumptions show that it might be beneficial to develop as much as possible code (especially license-related code) as native code, since it’s harder to reengineer.

Of course, this is not recommend by Google these days and according to them using C increases the app complexity. Different Android versions have different NDK versions, too.

Ref. MA Patrick Bernard (not published) ; https://developer.android.com/tools/sdk/ndk/index.html

ongoing research

Android 5.x?

Nevertheless Google **does not provide real solutions** here. Our recent research shows that DEX files are still included in OAT files used by the ART VM of newer Android version. There is no security benefit with the newest Android versions on this matter.

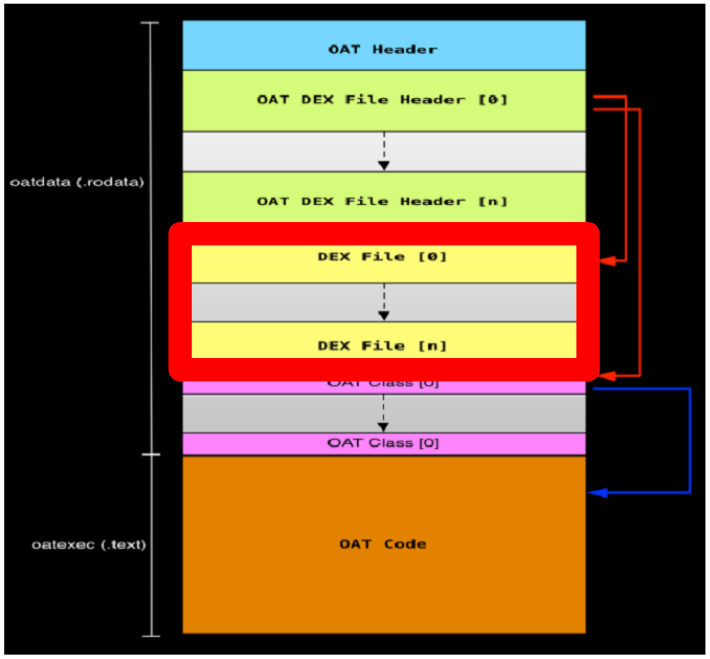
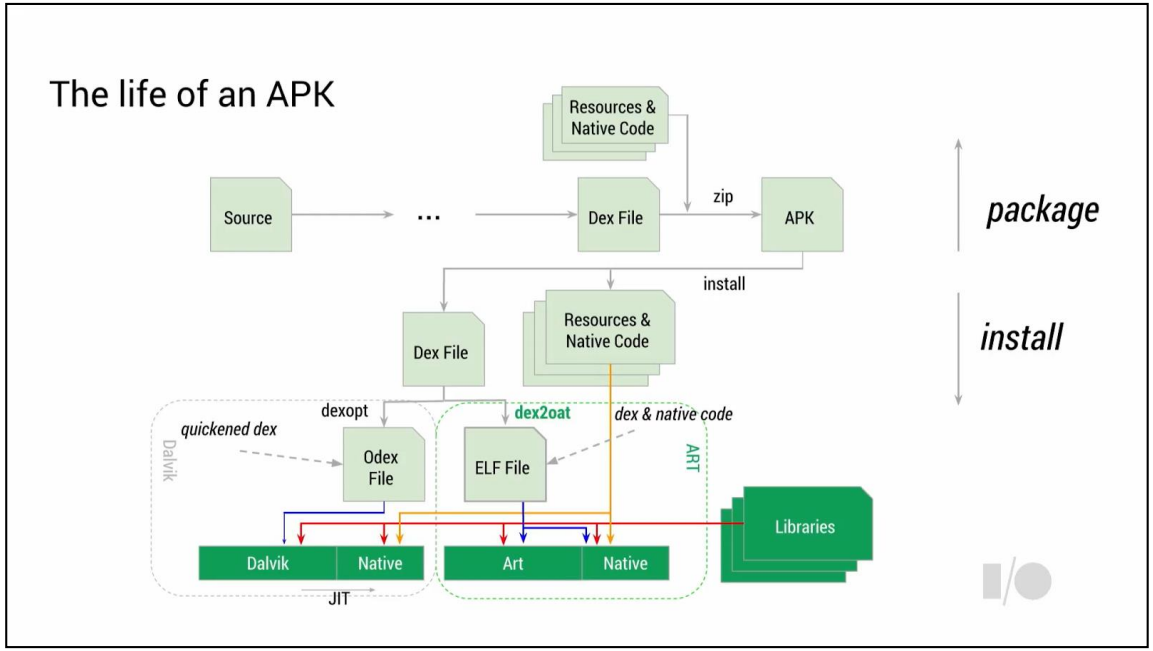


Figure 5 – oat file format (taken from [20 p. 30])

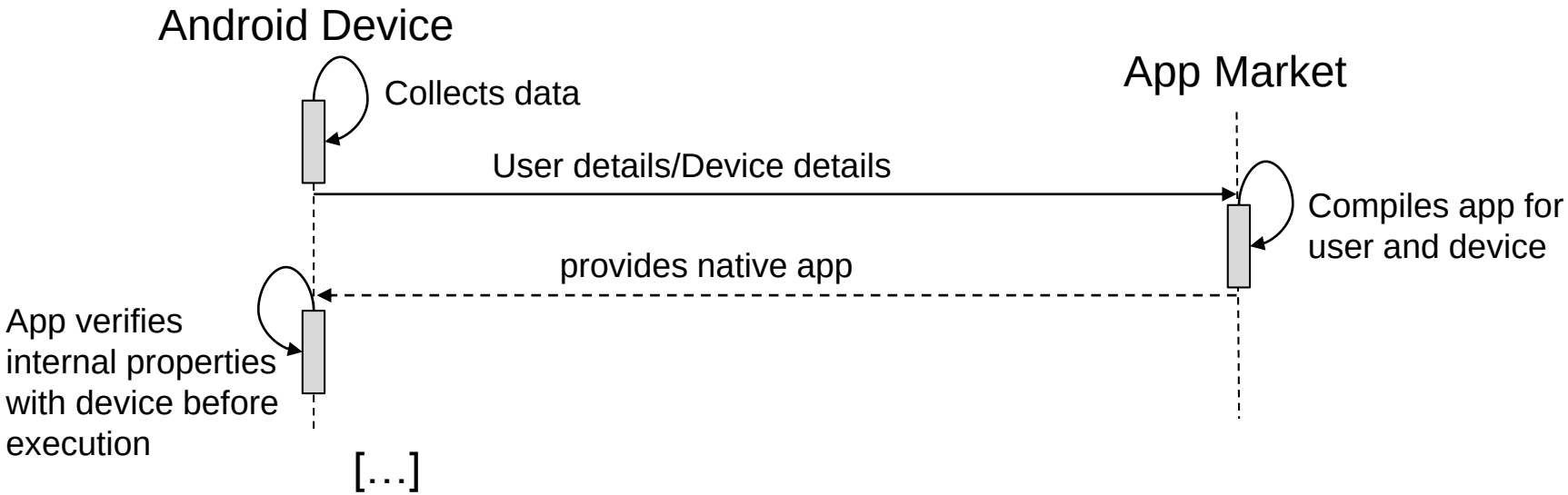


Ref. MA Patrick Bernard (not published) ; http://anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l/

ongoing research

Reinvent G-Play?

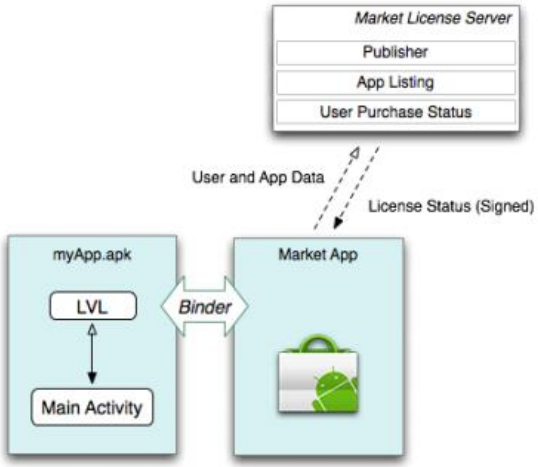
For increasing e.g. copyright security we propose a new app market that compiles applications up front based on the hardware and with user/device attributes included to be executed by exactly that user/device only. There is native code only and it's assumed to be more difficult to copy it.

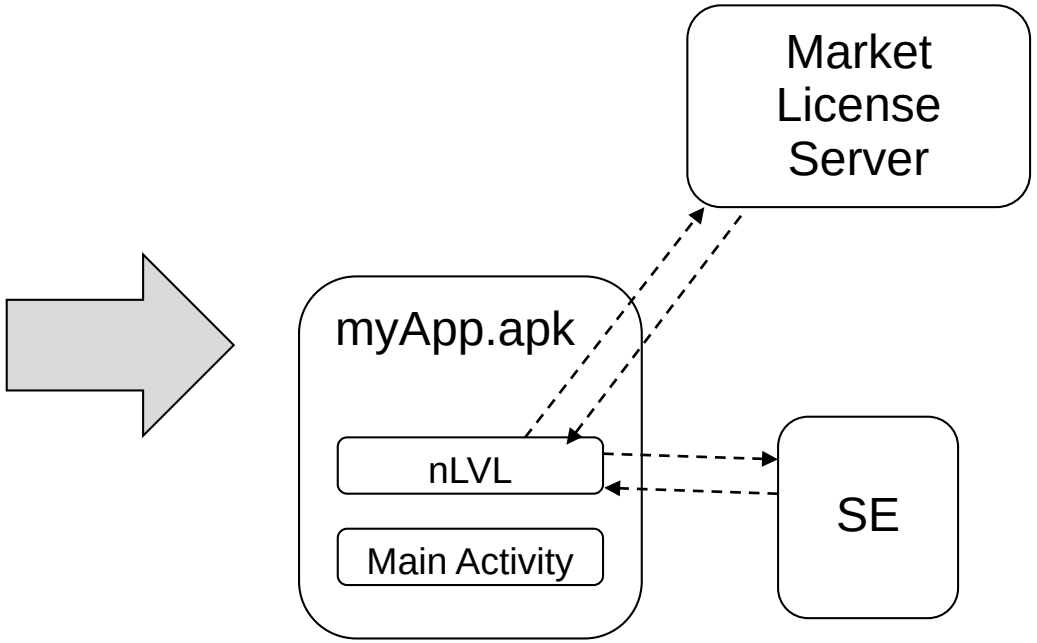


ongoing research

nLVL?

Due to our current results that native code is more secure, we are developing a **pure native LVL** similar to Google's LVL, while storing license data on a SE for offline purposes. Maybe it's possible to communicate with Google's server even directly.





Ref. http://developer.android.com/google/play/licensing/overview.html

Temporary results (Conclusion)

Current solutions provided by Google (LVL) or Amazon (Amazon DRM) do not provide sufficient copyright protection and may be easily cracked (cf. automated tools).

Increasing the complexity of applications with obfuscation and encryptions methods, increases the security against reengineering already.

Using secure elements (SE) provides interesting solutions for offline-purposes that could have been realized with server solutions in the past only. Nevertheless SE do not have sufficient performance to process huge amounts of data. In fact we are limited to a few kilobytes here.

Ref. MA Patrick Bernard (not published) ; BA Bichlmeier (not published) ; BA Pekmezci (not published)

Temporary results (Future work)

Nevertheless secure elements do not provide sufficient performance and we are required to process lots of data within Android (insecure!) or on a remote server.

Future projects might concentrate on developing a performant SE.

On the other hand this gap might be solved by trusted execution environments (TEE) that seem to share the hardware with the host system, while running on a dedicated operating system.

Related work (most important ones)

Recently **Google announced** the future release of **Project Vault** at Google IO 2015. A secure element to be used and programmed by users.

Almost a decade ago, **Thomas Aura et al.** created a solution for license management using smartcards. Their statements still apply today:

> *"there are always ways to work around the protection mechanisms [and only] […] the time to market for pirated copies [may be increased] and that pirated products cannot be sold as authentic"*

> *"copy-protection is always to some extent security by obscurity"*

Wu Zhou et al. developed the "first **VM-based protection** system for Android". Unfortunately we need to assume that their approach is not useable with Android 5.x anymore due to ART and its precompilation.

Ref. Thomas Aura, D. Gollmann "Software license management with smart cards" ;
Google IO https://www.youtube.com/watch?v=mpbWQbkl8_g&t=2940 ;
W. Zhou, Z. Wang, Y. Zhou and X. Jiang, "DIVILAR: Diversitying Intermedia Language for Anti-Repackaging on Android Platform"

Related work (most important ones)

Furthermore all **papers related to malware and copyright protections** are of interest, since malware's stealth techniques may be of interest for our copy protection, while other methods may be improved by our approach of secure elements maybe.

Examples that should be mentioned are a paper by Thansis Petsas et al. about malware's techniques to avoid dynamic analysis as well as papers about "Online Execution Class" and "Encryption-based Copyright Protection" by Sung Ryul Kim et. al.

Of course, **papers about TEEs** like Trustonic or Samsung's Trustonic for Knox are of interest in our future work section, too. Currently the usage of TEEs is limited to some vendors that own the keys and SDKs.

Ref. . Petsas, G. Voyatzis and E. Athanasopoulos, "Rage Against the Virtual Machine: Hindering Dynamic Analysis of Android Malware" ;
S. R. Kim, J. H. Kim and H. S. Kim, "A Hybrid Design of Online Execution Class and Encryption-based Copyright Protection for Android Apps" ;
https://www.trustonic.com/products-services/trustonic-for-samsung-knox

Android Security Symposium

Secure Copy Protection for Mobile Apps

Thanks for your attention

Comments? Questions?

Nils.Kannengiesser@tum.de

Copyright

Android Logo used in this presentation

Portions of this presentation (Android image) are modifications based on work created and shared by Google and used according to terms described in the
Creative Commons 3.0 Attribution License.

The logos by our partners are copyrighted by those.

Some images might belong to other owners and might be copyrighted. See references.

All other content is copyrighted by the university / division.