Data traces and protection functions

Understanding Privacy and its Risks

WSTNet Web Science Summer School 2016
July 1, 2016
University in Koblenz

Prof. Dr. Rüdiger Grimm
Universität Koblenz-Landau
Fraunhofer SIT Darmstadt
IT Risk Management
Your Google cardiac pacemaker has informed us that we will be needed here very soon.

Internet 4.0
Claude Monet: Le déjeuner. Public and private: the first “facebook scandal”
Content

1. Computer science’s understanding of „privacy“
2. Data sources and traces
3. Risk analysis of privacy
4. Limits of PET
Privacy

Public Government

Business Township Purchase

Family Friends

Myself

classical data protection; against government (esp. USA)

business secret; workers and customers as citizens

| employees customers | private sphere |

R. Grimm, July 2016: PET and Privacy
Privacy as legal concept

- The right to be left alone (USA, 1890)
- Personality (German constitution, GG Art. 1 u. 2)
- Post and telephone secret (Art. 10)
- Integrity of domicile (Art. 13)
- Informational self-determination
  - Census BVerfG 1983
  - Online-Surveillance BVerfG 2008
  - Preventive data storage EuCoJ 8.4.2014
- Data protection: right of confidentiality and integrity
Operational privacy principles

1. Purpose binding, data minimization
2. Consent, legal permission
3. Transparency, notice
4. Personal control
   • Notice, choice, correction, deletion
5. External control (DS-Beauftragte)
6. Confidentiality, onward transfer

Can be supported by automatic measures
Self and system data protection

- **Self**
  - Awareness
  - Abstinence
  - Choice (Telegram instead of WhatsApp)
  - NoScript, Ghostery, Ad-blocker
  - E2E encryption

- **System**
  - Data minimization
  - Deletion of purposeless data
  - Anonymization
  - Information
  - IPSec, SSL
PET: Privacy Enhancing Technology

- Contrary of PIT – Privacy Invading Technologies
- EPIC (1994) – Electronic Privacy Information Center (Marc Rotenberg)
- All IT,
  - that supports system and self data protection
  - that fights PIT
PET Typology

1. Communication with users
2. Tools for encryption
3. Tools for anonymity and pseudonymity
4. Filter tools
5. Policy tools


6. Rights management
PET type 1: communication with users

Transparence: privacy declaration before service
Choice/consent: user consent before service
User control: be informed, change, delete at any time during usage

Service to users: accounting info (and more)
Users to service: payment (and more)
PET type 2: tools for encryption

- Digital signature
- Encryption
- Steganography
- IPSec
- SSL
- S/MIME
- PGP
PET type 3: tools for anonymization

- Anonymizer
- Dining Cryptographer (DC) networks (academic)
- Mix (see extra lecture on crypto)
- Remailer, anonymous Web browsing (see extra crypto)
- Blind signature
- Id management

- Anonymous payment (Paysafecard, Bitcoin)
- Pseudonyms for e-commerce (eBay)
PET type 4: filter tools

- Cookie Cutter
  - CookieCooker, WebWasher
  - Browser settings

- Filter of Referer data field
- PICS: content label, content filter
  (pornography, violence, predefined content)
- PC firewalls
- Malware scanner
  - Viruses, spies, adware
Spybot Search & Destroy

Dies ist die Haupt-Suchseite von Spybot-S&D. Hier können Sie Ihr System untersuchen lassen (Schaltfläche "Überprüfen") und aufgetretene Probleme beheben (Schaltfläche "Markierte Probleme beheben"). Tipp: wenn Sie es noch nicht getan haben, lesen Sie doch mal die Einführung (siehe Hilfe-Menü).

Blende diese Information aus

<table>
<thead>
<tr>
<th>Problem</th>
<th>Art</th>
<th>Einträge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verfolgender Cookie (Internet Explorer: hundacker)</td>
<td>Cookie</td>
<td>1 Einträge</td>
</tr>
<tr>
<td>Verfolgender Cookie (Internet Explorer: hundacker)</td>
<td>Cookie</td>
<td>1 Einträge</td>
</tr>
<tr>
<td>Verfolgender Cookie (Internet Explorer: hundacker)</td>
<td>Cookie</td>
<td>2 Einträge</td>
</tr>
<tr>
<td>Verfolgender Cookie (Internet Explorer: hundacker)</td>
<td>Cookie</td>
<td>1 Einträge</td>
</tr>
<tr>
<td>Windows Security Center.AntiVirusOverride</td>
<td>Registrierungsdatum</td>
<td>1 Einträge</td>
</tr>
<tr>
<td>Einstellungen</td>
<td>nbank-And...</td>
<td></td>
</tr>
</tbody>
</table>

15 Probleme gefunden (05:39)
PET type 5: policy tools

- P3P
- Infomediaries
  - search engines?
  - route control?

PET type 6: rights management

- DRM – copy and usage rights
- Mobile apps permission (see extra lecture)
Content

1. Computer science‘s understanding of „privacy“
2. Data sources and traces
3. Risk analysis of privacy
4. Limits of PET
Motivation of data collection

- Law enforcement, fighting crime
- Predictive policing
- Scoring, insurances, ...
- Customer relation
- Human resource management
- User specific advertisement
- Political persecution

but ?
Three types of data sources

1. One‘s own action
   - I agree and I provide
   - I act and I am observed

2. Without one‘s own help by third parties
   - Alumni lists, schools, clubs
   - Credit reports (Schufa)
   - Newspapers

3. Environments
   - Habitation, occupation, education
   - Hobby, sports, arts, music
   - Shopping, travelling, visiting
   - Gender, nation

Not „myself“, but trans-individual cluster.
After just one match
⇒ vector in cluster
⇒ precise scoring
Traces by own activity

- Web – surfing
- Web – application
- E-mail
- Social networks
- „Telephone“
... and traces by mobility! (by consent!)

- Full Internet access
- Read contact data
- Read telephone status
- Read geo data
- Call installed applications
- Read running applications
- Receive/send SMS
- Use telephone
- Switch on camera, micro
- ... and a lot more
Content

1. Computer science’s understanding of „privacy“
2. Data sources and traces
3. Risk analysis of privacy
4. Limits of PET
Reference model of „trust“

- Ability
- Benevolence
- Integrity

Perceived Trustworthiness of Trustee

Perceived Risk

Risk Taking in Relationship

Outcomes

Trust

Trustor’s Propensity

[Mayer/Davis/Schoorman, 1995]
Trust Model Element M3. Risk (taken for Relation)

Applied to data protection principles

- Subject of trust:
  - Target of relation
- Risk:
  - What can go wrong? Estimation of damage
- Measure:
  - Limitation of risk
- Remaining risk:
  - Trust area
- Reason of trust:
  - Exchange (tit for tat), common value, contract, interest
Risk with …

- Purpose binding, data minimization
- Consent, legal permission
- Transparency, notice
- Personal control
- Notice, choice, correction, deletion
- External control (DS-Beauftragte)
- Confidentiality, onward transfer

explicated in in Bräunlich/Grimm, DuD 5/2015
Risk with confidentiality (risk, measures)

- **Risk:**
  - R1: Trustee transfers to others (online transfer without consent)
  - R2: Trustee is careless, applies no or wrong measures
    - e.g., no server protection
  - R3: Protection measures are vulnerable → encryption, access control

- **Measures:**
  - R1, External control, IT forensics
    - research gap: new proof methods, watermarks, „sticky logs“
  - R2, Quality requirement for trustee, education, legal requirements, customer pressure
  - R3, Quality requirements for technology, updates, additional self data protection (tracker blocker, e2e-encryption)
Risk with confidentiality (remaining risk)

- Remaining risk: lack of...
  - ...integrity and benevolence of trustee
  - ...technical and organizational integrity of medium

- **Working hypotheses:**
  - Institutional trust in medium strengthens trust in trustee
  - Medium security must work automatically with no burden on user
  - Does technical knowledge strengthen or weaken trust in medium? (Hypothesis: there is an upper limit)
Content

1. Computer science‘s understanding of „privacy“
2. Data sources and traces
3. Risk analysis of privacy
4. Limits of PET
Purpose binding, data minimization

- Reasons for distrust:
  - Personal data have financial value
  - Origin cannot be identified
  - Ads finance services
  - Better service by data analysis (e.g., LBS)
  - Esp: mobile apps!

- Measures require:
  - Benevolence of service providers
  - External control
  - Working hypothesis:
    - Data minimization belongs to system data protection
    - Data minimization cannot be enforced by users
Consent

- **Reasons for distrust:**
  - Trustee hides other purposes (see above, purpose binding)
    - Ads finance services
    - Better service by data analysis (e.g., LBS)
  - Trustee requires too many permissions (see above, purpose binding)
    - Critical: mobile Apps!

- **Measures require:**
  - User competence
  - Choice of alternative services
  - Therefore often: just click away
  - Working hypothesis:
    - Users are overburdened
Transparency, notice

- For example, published policies
  - Long, obscure
  - Surprisingly open, hidden tricks
  - E.g., [Google 2014]
- Example P3P
  - Euphoria 2000
  - W3C standard 2000
  - [Grimm/Roßnagel 2000]
  - AT&T Privacy Bird 2004 (Lorrie Cranor)
  - Policy Generator for Rheinland-Pfalz 2006
  - Meaningless P3P clones [Lämmel/Pek 2010]
Transparency, notice

- Reasons for distrust:
  - Provider hides real purpose (see above, purpose binding)
  - Provider avoids effort
  - Provider dislikes user control (deletion, correction)

- Measures require:
  - Understanding interrelations
  - Communication effort
  - Competence for decision
  - Alternatives of services
  - Working hypothesis:
    - Users are overburdened
External control

- Reasons of distrust:
  - Control target hides data (is better equipped than controller)
  - Controller is not neutral

- Measures require:
  - Political ambition
  - Well equipped controller
  - Good IT forensics (note research gaps)
  - Working hypotheses:
    - Control unburdens users
    - Control must not be a bureaucratic bond
Confidentiality (see above!)

- Reasons of distrust:
  - Business with personal data is profitable (onward transfer)
  - Server security is hardly detectable (external control, research gap)
  - Internet it a paradise for hackers

- Measures require:
  - Personal integrity and procedural benevolence of partners
  - Technical and organizational integrity of medium
  - Working hypotheses:
    - Combination of network and e2s encryption
    - Benevolence by ethics (awareness) and control
    - Purpose binding measures required (research gap)
Working hypotheses

Hyp 1 Individual actions do not lead to individual disadvantages; but they lead to social damage

Hyp 2 Individual consent and transparency overload users and remain toothless without external control

Hyp 3 Users must be unburdened of data protection work

Hyp 4 Data networks should be anonymous and personification goes only with content relationships (e2e) (notice: Facebook is both, data and content provider)

Hyp 5 System data protection must be extended
References


Security for Mobile Applications

SM14: Mobile Apps Security and Privacy

R. Grimm
Institute for Information Systems Re
University Campus Koblenz

areamobile, AreaDigital AG Berlin, 2015
Content

1. **Mobile and Distributed Apps Security Problem**
2. Apps Architecture
   - Example Android
3. Apps Security Architecture
   - Permission Model
4. Access Permission Groups
5. Users’ View: Self Data Protection for Apps
Mobile Smartphones Security Problem

- Mobile Smartphones are (also) computers
- They can attack, and can be attacked just like any other computer in a network
  - Man-in-the-middle
  - Virus distribution
  - Back doors, penetration, bots
  - “C.I.A”:: Eavesdropping, changing/deleting data, denial of service
  - Fraud, e.g. buying tickets with stolen credit cards
    (but easier to hide with prepaid SIM cards!)

- But mobiles provide some specific security challenges (mobility!), see next slides…
McAfee Labs 2015 Threats Predictions

Global Internet-Connected Devices

50 Billion


Personal Computers

Mobile Devices

Internet of Things

Sources: McAfee, based on research by BI Intelligence, IDC, and Intel.
Mobile and Distributed Apps Security Problem

- persons
devices

- apps

- mobile resources:
  - other mobile apps
  - other mobile services

- Internet resources:
  - other Internet services
  - other Internet users

1. attack
2. attack
Apps attack mobile and Internet resources (1)

Example: DB Navigator:
  Ticketing
  Timetables
  Delay Information Service
  …
Apps attack mobile and Internet resources (2)

- App
- Unauthorized distribution
- Also as bots controlled by external attackers
- Pictures
- Video recording
- Audio recording
- ...
2 Apps are attacked by externals

- External apps/services/resources attack apps and their users
- Apps are special in that their mobile devices
  - have sensors for mobile action
  - include mail, contacts, planner
    ... and many other apps
  - provide data about motion and apps usage
- Mobiles are attractive targets
  - (1) to listen from outside
  - (2) to control/operate from outside
Smartphone sensors

More Input:
• Micro
• Switches
• Keyboard
• Received data

More Output:
• Screen
• Speaker
• Vibration

smartphoneworld.me/hello-world-2 [download 14.06.2016]
Traces by mobility

- Access to Internet
- Read
  - contact data
  - calendar
  - telephone status
  - geolocation data
  - running apps
  - mail
- Invoke apps
- Deactivate stand-by-mode
- Receive/send SMS
- Use telephone
- Switch on micro, camera
- ... and more
Rights/permission management

- Global permission before installation (Android 5) vs. Explicit permission before usage (Android 6, iOS)
- Android, iOS, Windows Mobile permission models
- Protection mechanisms, e.g.
  - App Permission Watcher – makes rights/permissions visible
  - SRT AppGuard – active rights management via app
- Research challenge:
  - New model required
  - Fair
  - Balance risks and opportunities
  - Unburden users

(See discussion of *permission model* below)
Summary – Mobile and Distributed Apps Security Problem

- Apps are in danger like any other computer
- Apps have special mobile security challenges:

1. Apps attack Internet and mobile resources
   1. unauthorized access to service APIs
   2. unauthorized distribution of content

2. Apps are attacked by externals
   - Mobiles have sensors and PIM data
   - Mobiles are attractive targets
     1. to listen from outside
     2. to control/operate from outside
Content

1. Mobile and Distributed Apps Security Problem
2. Apps Architecture
   – Example Android
3. Apps Security Architecture
   – Permission Model
4. Access Permission Groups
5. Users’ View: Self Data Protection for Apps
Android

• Example for Smartphone OS
  – Others: iOS (Apple); Windows Mobile (MS), BlackBerry

• Open Source Development
  – Open Handset Alliance, led by Google

• Android SDK available since 2007

• First Android G1 phone available since Oct 2008

• T-Mobles manufacturer HTC sells > 1 Mio phones by end of 2008

• Market leader today (in Germany: > 75%)

• Android 6 Marshmallow since fall 2015
Android

Kantar Worldpanel ComTech: Smartphone OS sales market share
http://www.kantarworldpanel.com/global/smartphone-os-market-share/intro,
download 14.06.2016
Android Software Development

• Easy application development through SDK on top of Java
  – Special Java middleware APIs
  – Each application has its own user id (Linux thread)
  – Controlled applications interaction via special API
  – Simple permission label assignment model (MAC!)
    (however, somewhat perforated by extensions) --- see below!

• Seamless interaction with Google services
  – Gmail, Calendar, Contacts
Android 5 Architecture

Android 6 Architecture

Applications
- Home
- E-Mail
- S-planner
- SMS
- Browser
- Phone
- Contacts
- WhatsApp
- Camera
- Media Player
- ...

Application Middleware
- Activity Manager
- Content Providers
- Window Manager
- View System
- Package Manager
- ICC MAC Enforcer
- Telephony Manager
- Resource Manager
- Location Manager
- Notification Manager

Libraries (SDK for developers)
- Surface Manager
- Media Framework
- SQLite
- OpenGL
- libc (Gnu C)
- SGL
- SSL
- FreeType
- Android Runtime
  - Core Libraries
  - Dalvik Virtual Machine

HAL - Hardware Abstraction Layer
- Audio
- Bluetooth
- Media
- DRM
- External Storage
- Graphics
- Input
- Camera
- Sensors
- TV

Linux Kernel
- Binder (Inter Process Communication) Driver
- Camera Driver
- Display Driver
- Flash Memory
- Keypad Driver
- WLAN Driver
- Audio Drivers
- Power Management

Linux kernel

• Linux - Linux 2.6 with (more than 100) patches
• Basic system functionality
  – process management
  – memory management
  – device management like display, keypad, micro, speaker, camera, etc.
• Networking (WLAN, UMTS)
• Device drivers (interfacing to peripheral hardware)
Libraries

including

- Web browser engine WebKit
- GNU C Library libc
- SQLite database (storage and sharing application data)
- Play and record audio and video
- OpenGL for 2D/3D graphics
- SSL libraries

- Android Runtime (see next slide)
Android Runtime

• Key component: Dalvik Virtual Machine
  – Java Virtual Machine specially designed and optimized for Android
  – for memory management
  – for multi-threading (typical for Java)
  – to run in its own process, with its own instance of the Dalvik virtual machine.

• An additional set of core libraries
  – for Android application developers
  – to write Android applications using standard Java
Application Framework (or “Application Middleware”):

- Higher-level services to applications in the form of Java classes
- For direct use of application developers
- ICC: Inter Component Communication
  - Message exchange and action invocation
  - Security/permission rules (see below)

Applications:

- All Android applications, like
- Contacts, e-mail, s-planner, messages, telephone, …
- Browser, games etc.
- Self-written and imported apps
Android Application Architecture

• Applications consist of components
• Components interact through ICC
  – ICC: Inter Components Communication
  – Like IPC: Inter Process Communication
  – ICC respects permission model (security rules, see below)
• Four component types
  – Typical for their functionality

Learnt from Enck/Ongtang/McDaniel 2009
Four Android Component Types

- Activity
- Service
- Content Provider
- Broadcast Receiver
Android Component Type “Activity”

- Activity
  - “Presentation layer of an application”
  - One Activity per screen
  - Defines an API for itself
  - Activities start one another, passing and returning values
  - One activity on system has keyboard and processing resource, all others are suspended
    - Usually entry point to the application (like “main”)
    - to start the user interface
    - Usually same name as application

- Service
- Content Provider
- Broadcast Receiver
Android Component Type “Service”

- Activity

- Service
  - Background processing
  - Continue running after user interface of it disappears
  - Example: download files, play music, send mail
  - Through RPC interface: services wait for calls
  - Other components call services to send commands, retrieve data, …

- Content Provider
- Broadcast Receiver
Android Component Type “Content Provider”

- Activity
- Service

- Content Provider
  - Provide relational database interface
  - Store and share data
  - Access controlled by an “authority”
  - “Authority” describes content type
  - Other components perform SQL queries on Content Provider, like
    - SELECT, INSERT, DELETE, ...

- Broadcast Receiver
Android Component Type “Broadcast Receiver”

- Activity
- Service
- Content Provider

- Broadcast Receiver
  - Mailbox for messages from other applications
  - Two ways to receive message:
    1. explicit
      - applications address Broadcast Receivers explicitly by including their assigned namespace
    2. implicit:
      - applications send messages to “implicit destinations”
      - Broadcast Receivers subscribe to “destinations” in order to receive those messages
Android Manifest XML-File

- Developer creates Manifest file *Manifest.xml*
- Manifest file …
- … specifies all components
- … defines *permission policy* *(see below)*
- … marks specific *Activity* to start an application
Example Application FriendTracker (Enck et al.)

FriendTracker Application

- BootReceiver
- FriendTracker
- FriendTrackerControl
- FriendProvider
- Broadcast receiver
- Service
- Activity
- Content provider

Learnt from Enck/Ongtang/McDaniel 2009
Example Application FriendTracker, Components

- **Activity FriendTrackerControl**
  - Entry point (main)
  - Core application logic

- **Service FriendTracker**
  1. Invokes *external Web Service* that collects locations of external friends
  2. Bind to system *LocationManager* service:
     2.a "Give me my current position in order to …“
     2.b "…. to check if I am near a friend“
     2.c "…. and to inform Web Service about my position”

- **Content Provider FriendProvider**
  - Stores geographic coordinates of friends from *FriendTracker* service

- **Broadcast Receiver BootReceiver**
  - Waits for notification from system if system services (*LocationManager* etc) are available
  - *FriendTrackerControl* activity starts on this notification *FriendTracker* service

Learnt from Enck/Ongtang/McDaniel 2009

© R. Grimm
Example Application FriendTracker and FriendViewer (Enck et al.)

FriendTracker Application

- Boot Receiver
- Broadcast receiver
- FriendTracker Control
- Activity
- FriendTracker
- Service
- FriendTracker Control
- Content provider
- Read/write
- Start/stop
- FriendProvider

FriendViewer Application

- FriendReceiver
- Broadcast receiver
- FriendViewer
- Activity
- FriendMap
- Start
- Activity
- Start
Example Application FriendViewer, Components

- **Activity FriendViewer**
  - Entry point (main)
  - Reads from *FriendTracker* app all friends and their coordinates
  - Starts *FriendMap* activity

- **Activity FriendMap**
  - Displays info from *FriendViewer* an a map
  - Displays “near friend” from *FriendReceiver*
  - Uses system service *ViewContact* for display

- **No Content Provider**
  - (*FriendViewer* reads from CP *FriendProvider* of *FriendTracker* app)

- **Broadcast Receiver FriendReceiver**
  - Waits for message from *FriendTracker* service (2.b): “I am near a friend <name>”
  - *FriendMap* activity displays “near friend” on this notification

Learnt from Enck/Ongtang/McDaniel 2009
Example Application FriendTracker+FriendViewer +System (Enck et al.)

Learnt from Enck/Ongtang/McDaniel 2009
Inter-Component Interaction (ICC)

- ICC by messages
- Messages are called "intents":

<table>
<thead>
<tr>
<th>destination component address</th>
<th>payload data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>............</td>
</tr>
<tr>
<td></td>
<td>............</td>
</tr>
<tr>
<td></td>
<td>............</td>
</tr>
<tr>
<td></td>
<td>............</td>
</tr>
<tr>
<td></td>
<td>............</td>
</tr>
</tbody>
</table>

- Android API methods accept intents, like
  - startActivity(Intent1)
  - startService(Intent2)
  - sendBroadcast(Intent3)
  - readData(Intent4)
- The *intent object* defines the “intent” to perform an “action” on the destination component, specified by the payload data
- ICC conforms to explicit *security rules* (see below)
Android Intent addressing mechanism

- Flexible: two ways of addressing
  - Firstly, directly addressing a target component by using its application’s namespace
  - Alternatively, implicit names!!

- Implicit name = “action string”
  - For example, VIEW, CONTACT, PLAY
- System determines appropriate target application (by intent filters)
- Target components use “intent filters” to accept action string addresses
  - (Developers of components specify the intention filters in Manifest file)
- Typical use for action strings: address a group of broadcast receivers
Intents in Application Example (Enck et al.)

Learnt from Enck/Ongtang/McDaniel 2009
Component types support intent types

- **Services support**
  - Start, stop, bind (for RPC commands)

- **Broadcast receivers support**
  - Explicit target address
  - Or implicit: BR “subscribes” to an action filter
  - Execution specified by intent data
  - BR *FriendReceiver* subscribes to *FRIEND_NEAR*

- **Content providers don’t accept intents**
  - Explicit *authority string* in URI, like
    `Content://<authority id>/<SQL query on specific table records>`
Content

1. Mobile and Distributed Apps Security Problem
2. Apps Architecture
   – Example Android
3. Apps Security Architecture
   – Permission Model
4. Access Permission Groups
5. Users’ View: Self Data Protection for Apps
Teaching focus: permission model

• This presentation does not address
  – communication confidentiality
  – data integrity
  – denial of service
  – fraud prevention
  – information flow control
  – accountability

• This presentation focusses on
  – access control
  – the apps permission model
  – Android as an example
The mobile apps access problem

• Apps process resources
• Apps communicate across
  – other apps
  – other components of the device
  – other devices
  – other domains
  – worldwide

• Who is allowed to access my resources?
• How do I gain access to others‘ resources?
Android applications access model

- Two levels of access control
  - DAC (discretionary access control) on Linux level
  - MAC (mandatory access control) on Android Middleware level

- Linux DAC:
  - All apps run as a separate Linux process
  - Each app has its own UserId / GroupId

- Android Middleware MAC:
  - Components own permissions in order to access other components
  - Components require permissions, i.e. allow access only to those other components that have the required permissions
  - ICC reference monitor checks if required permissions and owned permissions match
Two levels of Android security enforcement

Figure 1.2: Two levels of Android security enforcement

Cited from Yury Zhauniarovich (2014), section 1.2, p.8
MAC – mandatory access control by labels (like Bell-LaPadula (*)

- Components have labels for access control
- Components interact with (have access on) one another
- A **MAC matching rule** specifies which combination of labels yield acceptance or denial of access
- A **MAC Reference Monitor** enforces the matching rule system-wide

MAC – Example

- Matching Rules
  - Read access allowed iff *reading* component has **no lower label numbers** than target component
  - Write access allowed iff *writing* component has **no higher label numbers** than target component

```
Component C1
Code
Start;
Read f;
If() then... else...
Labels
L1
L2

Component C2
Code
Start;
Open g;
If() then... else...
Labels
M1
M3

Component C3
Code
Start;
Write h;
If() then... else...
Labels
N3
N4
```
Android MAC with “Permissions”

- MAC labels are “Permissions”:
- Components *own permissions* in order to access other components:
  - Their Manifest file contains “use-permission” elements
- Components *require permissions* in order to allow access by other apps:
  - Their Manifest file declares “permission” elements for the whole app;
  - and in their Manifest file the components have “permission” attributes for those permissions that they require
- ICC reference monitor checks if *required permissions* and *owned permissions* match:
  - App A is allowed access on component B1 of app B iff A has exactly all “use-permissions” that are required by B1’s “permission” attributes
Android MAC with Permissions – Example

• App whoop wishes to have access on Activity blah1 in app blah, i.e. whoop wishes \textit{(at some point)} to start the program code of blah1

• In blah’s Manifest.xml file
  – blah creates permissions required by its components
  – Components of blah specify those permissions that allow other apps to gain access to the component, e.g.
  – Activity blah1 specifies a permission with the name “blubb” that other apps must have if they want access to blah1

• In whoop’s Manifest.xml file
  – whoop creates “permissions owned”
  – whoop specifies in “permissions owned” that it will use the permission “blubb” (that it “owns blubb”)
Android MAC with Permissions – Example

```
<manifest package="pac">
  <uses-permission android:name="p1" />

  <activity android:name="pac.Main"
            android:label="@string/app_name">
    <intent-filter>
      ...
    </intent-filter>
  </activity>

  Component c2
  etc...
</manifest>

<manifest package="blah">
  <permission android:name="blubb"
              android:label="Use the results of blah"
              android:protectionLevel="dangerous"
              android:permission.Group="... one of the 17+ groups..."

  <activity android:name="blah1"
            android:label="@string/app_name"
            android.permission="blubb">
    <intent-filter>
      ...
    </intent-filter>
  </activity>

  Component blah2 (no permission attribute)
  etc...
</manifest>
```
Android MAC with Permissions – Example

```
<manifest package="whoop">
  <uses-permission android:name="blubb" />
  <activity android:name="whoop.Main"
    android:label="@string/app_name">
    <intent-filter>
      ...<intent-filter>
  </activity>
</manifest>

<manifest package="blah">
  <permission android:name="blubb"
    android:label="Use the results of blah"
    android:protectionLevel="dangerous"
    android:permission.Group="... one of the 17+ groups..."
  >
  <activity android:name="blah1"
    android:label="@string/app_name"
    android.permission="blubb">
    <intent-filter>
      ...<intent-filter>
  </activity>
</manifest>

Component c2
etc...

Component blah2 (no permission attribute)
etc...
```

© R. Grimm
1. App *blah* creates a *permission name=*"blubb"* element (and more permissions)

```xml
<manifest package="blah">
  <permission android:name="blubb"
              android:label="Use the results of blah"
              android:protectionLevel="dangerous"
              android:permission.Group=" ... one of the 17+ groups..."
  />
</manifest>
```

This and following code examples are learnt from [Jeff Six 2012]
How does an app define its protection domain?

1. App *blah* creates a *permission name="blubb"* element (and more permissions)
2. Activity *blah1* requires other apps to have this permission in order to start activity *blah1*: it defines *permission="blubb"* attribute

```xml
<manifest package="blah">
  <permission
    android:name="blubb"
    android:lable="Use the results of blah"
    android:protectionLevel="dangerous"
    android:permission.Group=" ... one of the 17+ groups..."
  />

  <activity
    android:name="blah1"
    android:label="@string/app_name"
    android:permission="blubb">
    <intent-filter>
      ... 
      </intent-filter>
  </activity>

  ...
</manifest>
```
How does another app gain access on a protected activity?

3. App *whoop* defines a *uses-permission name="blubb"* element

```xml
<manifest package="whoop"> /

<uses-permission android:name="blubb" />

<activity android:name="...whoop.MainActivity" android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>

<activity> ... more activities ... </activity>
<contentprovider> ... </contentprovider>
<broadcastreceiver> ... </broadcastreceiver>
<service> ... </service>
</manifest>
```
How does a client allow an app to access its resources? (until Android 5)

- Once app \textit{whoop} is installed on a device with THIS permission \textit{blubb}, it will be allowed to start component \textit{blah1} (to execute \textit{blah1}'s code)
- \textbf{At installation time}, before installation of whoop,
  - Google Store reads the "uses-permission" attribute in \textit{whoop}'s Manifest file and asks the downloading client, if it would allow access to \textit{blah1}
  - If client says "yes", \textit{whoop} is installed
  - If client says "no", \textit{whoop} is \textit{not} installed
How does a client allow an app to access its resources? (Android 6/Marshmallow)

• Permission at *execution time* for every single usage.
• If the app *whoop* is installed on a device with THIS permission *blubb*, then the permission is not yet active, but…
• … when client starts *whoop*,
  – at the point where whoop calls the component *blah1*,
    Android checks if "uses permissions" and "required permissions" match.
  – If they do not match, *blah1* is not executed.
  – If they match, the client asks if it allows access on *blah1*:
    • if user says "yes", *blah1* is executed,
    • if user says "no“, *blah1* is not executed.
Access permission in Android 6 / Marshmallow

Permissions – you decide:

No continuous access by apps: under Android Marshmallow you can decide yourself, when and on what apps on your device have access. Moreover, you can deactivate permissions at any time.

https://www.android.com/intl/de_de/versions/marshmallow-6-0/ [14.6.2016]
How does Android middleware enforce MAC matching rule?

- Let apps `whoop` and `blah` be installed on a client device.
- If `whoop` runs and tries to start `blah1` (by `whoop`'s running code):
  - The Android middleware (ICC MAC enforcement module) checks if "permission names" in `blah1`'s Manifest file match the "uses-permission names" in `whoop`'s Manifest file
  - If "yes", `blah1` is started
  - If "no", the start is "thrown" and `whoop` continues without start of `blah1`
Alternative method to define protection domain – by program code

Instead of

```
<activity android:name="blah1" android:label="@string/app_name"
    android.permission="blubb">
```

in `blah`’s Manifest file,

the program code of `blah1` could check the "uses-permission name“ in any calling app:

```
int check_blubb = checkCallingOrSelfPermission("blubb");
if (check_blubb != PERMISSION_GRANTED)
    throw new SecurityException();
```

In this case, the *program controls* the access,

not *Android middleware* through Manifest files.

This way to manage permissions is harder,
because for any update, program code must be updated.
Services are protected against access…

- … exactly like activities by permission="blubb" attributes
Content Providers are protected against access…

- Content Providers (CP) are standard methods to store and share data.
- CP allow to specify **read** or **write** permissions separately:

```xml
<provider android:name="blah.DataProvider"
          android:authorities="blah.app1.dataprovider"
          android:readPermission="blubb_1"
          android:writePermission="blubb_2">
</provider>
```

- If app *whoop* wants to **read** from *blah.DataProvider*, it must have the element
  ```xml
  <uses-permission android:name="blubb_1" />
  ```
  in its Manifest file

- If app *whoop* wants to **write** into *blah.DataProvider*, it must have the element
  ```xml
  <uses-permission android:name="blubb_2" />
  ```
  in its Manifest file
Broadcast Intents are protected against access...

- Broadcast Receivers (BR) are “mailboxes” to exchange *Intents*
- Intents are secured via program code of calling component
- Intents can be protected at both ends, at the **sending and receiving end**, separately:

  - Sending intents from a component of an app to a BR can include a *permission parameter*, that a *receiving BR* needs to have to register to this service:

    ```java
    sendBroadcast(Intent1,"blubb.ALLOWED_TO_RECEIVE_FROM_ME");
    ```

  - Receiving intents by a BR can include a *permission parameter*, that a *sending component* needs to have in order that the BR accepts this intent:

    ```java
    registerReceiver(rcv, intentFilter"blubb.ALLOWED_TO_SEND_TO_ME", null);
    ```
Unprotected broadcast intents endanger privacy

- Consider *FriendTracker* service: *FriendTracker* broadcasts an intent to the “FRIEND_NEAR” action string (implicit addressing)
- All installed *FriendViewer* apps (which include the *FriendReceiver* BR) would be informed of the identified near friend

![Diagram showing FriendTracker Application broadcasting to FriendReceiver]

- *FriendTracker* should require a permission by all BR to receive this intent:
  
  ```
  sendBroadcast(Intent1,"blubb.ALLOWED_TO.Receive_FROM_ME");
  ```

- Without this permission *FriendReceiver* BR wouldn’t be allowed to register to this *FriendTracker* activity
Public and private app components

• Components can be declared as “private”
  – In Manifest file declare `exported="false"` attribute
  – This component would deny any access from outside
  – No need to protect it by further permission requirements

• All other components are “public”
  – For access control explicit permission requirements must be declared (see above)
Typical system app permissions

<uses-permission android:name= "..." />

"android.permission.ACCESS_NETWORK_STATE"
"android.permission.INTERNET"
"android.permission.WRITE_EXTERNAL_STORAGE"
"android.permission.WRITE_SETTINGS"
"android.permission.CAMERA"
"android.permission.VIBRATE"
Four permission protection levels: “normal”, “dangerous” etc.

```
<manifest package="blah">
  <permission android:name="blubb"
               android:label="Use the results of blah"
               android:protectionLevel="dangerous"
               android:permission.Group="... one of the 17+ groups... "
  ...
</manifest>
```

**old**
- **“system”:** permission is granted only to apps that are installed in `data/system`.
- **“application”:** permission is granted to any app asking for it (they are installed in `data/app`).

**new**
- **“normal”:** permission is granted to any app asking for it (like the old “application” permission).
- **“dangerous”:** user has to confirm permission at installation time.
- **“signature”:** permission is granted only to apps signed by the same developer key as the granting app.
- **“signature or system”:** like “signature”.

© R. Grimm
Permission required by the FRIEND_NEAR broadcast intent is “dangerous”!
However, this permission is checked by the *FriendTracker* service on a code-level (see above):

```java
sendBroadcast(Intent1, "blubb ALLOWED_TO_RECEIVE_FROM_ME");
```
1. Mobile and Distributed Apps Security Problem
2. Apps Architecture
   – Example Android
3. Apps Security Architecture
   – Permission Model
4. Access Permission Groups
5. Users’ View: Self Data Protection for Apps
User oriented grouping of permissions (Android)


• “Before you download an app on Google Play, you may need to give the app permission to access specific capabilities or information on your device, known as permissions groups”

• Permissions are organized into permissions groups (group icons) in order to “make an informed decision more easily on whether you would like to install the app” [Google]

• Users confirm permission groups
  – Once accepted, there is no confirmation for update
  – “All-or-nothing” decision

• For criticism on Android permission handling, read [Nerney 2014] and Gerber/Volkamer 2015
  – “too coarse”, “all-or-nothing decisions”, “hard to understand”, “hidden conformations”
Android permission groups (1-3)

1. In-app purchases
   An app can ask you to make purchases inside the app

2. Device & app history
   Can use one or more of the following:
   - Read sensitive log data
   - Retrieve system internal state
   - Read your web bookmarks and history
   - Retrieve running apps

3. Cellular data settings
   Can use settings that control your mobile data connection and potentially the data you receive
4. **Identity**
   Can use your account and/or profile information on your device, e.g.:
   - Find accounts on the device
   - Read your own contact card (e.g., name and contact information)
   - Modify your own contact card
   - Add or remove accounts

5. **Contacts**
   Can use your device's contacts (read and modify)

6. **Calendar**
   Can use your device's calendar information, e.g.:
   - Read calendar events plus confidential information
   - Add or modify calendar events and send email to guests without owners' knowledge
Android permission groups (7-8)

7. Location
   Can use your device's location, e.g.:
   - Approximate location (network-based)
   - Precise location (GPS and network-based)
   - Access extra location provider commands
   - GPS access

8. SMS
   Ability to use text, picture, or video messages, SMS and MMS
   - Receive text messages (SMS or MMS)
   - Read your text messages (SMS or MMS)
   - Edit your text messages (SMS or MMS)
   - Send SMS messages; this may cost you money
   - Receive text messages (WAP)
Android permission groups (9-10)

9. Phone
   Can use your phone and/or its call history (note: possible extra charges), e.g.:
   – Directly call phone numbers; this may cost you money
   – Write call log (example: call history)
   – Read call log
   – Reroute outgoing calls
   – Modify phone state
   – Make calls without your intervention

10. Photos/Media/Files
    Can use files or data stored on your device:
    – Read the contents of your USB storage (example: SD card)
    – Modify or delete the contents of your USB storage
    – Format external storage
    – Mount or unmount external storage
Android permission groups (11-13)

11. Camera
   - Take pictures and videos
   - Record video

12. Microphone
   - Record audio

13. Wi-Fi connection information
   - access your device's Wi-Fi connection information, like if Wi-Fi is turned on and the name(s) of connected devices
   - View Wi-Fi connections
Android permission groups (14-16)

14. Bluetooth connection information
   – Can broadcast to or get information about nearby Bluetooth devices

15. Wearable sensors/activity data
   – Access data from wearable sensors, such as heart rate monitors
   – Can receive periodic updates on physical activity levels

16. Device ID & call information
   – Can access
     • your device ID(s) and phone number
     • whether you're on the phone
     • and the number connected by a call
   – Can read phone status and identity
17. Other

- Can use custom settings provided by your device manufacturer or application-specific permissions
- If an app adds a permission that is in the "Other" group, you'll always be asked to review the change before downloading an update

“Other” access may include the ability to:
- Read your social stream (on some social networks)
- Write to your social stream (on some social networks)
- Access subscribed feeds

When you review individual permissions, all permissions, including those not displayed in the permissions screen, will be shown in the "Other" group.
Criticism on Android‘s permission policy

[Nerney 2014]:

Google [...] permissions groups, along with a catch-all “other,” including “in-app purchases,” “device & app history,” “cellular data settings,” and “identity.”

Let’s take “identity” as an example: [...] if you approve an app that requests permission to “find accounts on the device” (first function listed under “identity”) you also are permitting the other “identity” functions: “read your own contact card,” “modify your own contact card,” and “add or remove accounts.”

Under “contacts/calendar”, a user who approves “read your contacts” also unknowingly approves “modify your contacts,” “read calendar events plus confidential information,” and “add or modify calendar events and send email to guests” without owners’ knowledge.

I bet it doesn’t even feel like your phone anymore!

A third-party developer named SnoopWall last December released an app that allowed Android users granular control over apps permissions. It’s worked very well for me, but I’m not sure if it will be able to detect and alert users to permissions coming in on apps updates.
Content

1. Mobile and Distributed Apps Security Problem
2. Apps Architecture
   – Example Android
3. Apps Security Architecture
   – Permission Model
4. Access Permission Groups
5. Users’ View: Self Data Protection for Apps
Self Data Protection

- Check permission requests carefully
  - see, e.g., Gerber/Volkamer 2015
- Refrain from data-hungry apps, select privacy-friendly alternatives
  - see, e.g., "Privacy friendly QR Scanner" by P. Rack (TU Darmstadt)
- Check permissions at every update version
- Consider the use of protection apps
Exercise: Study “Privacy friendly QR Scanner” by P. Rack (TU Darmstadt)

- Download the app
- Notice the user confirmation of permissions
- Read the Manifest.xml file (included in lecture material)
- Interpret the permission elements
- Interpret the activity elements
- Which components are defined?
- What are they supposed to do?
- Which system activities are expected to be used?
Technical support of self data protection

1. Protection apps, e.g.,
   - App Permission Watcher – makes rights/permissions visible
   - SRT AppGuard – active rights management via app

2. Change source code of data hungry apps
   - Could make app useless
   - Respect copyright!
   - Open source license ok

3. Change operating system to have better permission control
   - Could have unwanted side effects (slow down apps, denial of access)
   - Google extension function warns against dangerous apps (jelly-bean)
   - Try TaintDroid (www.appanalysis.org [14.6.2016])
   - Android is open source ⇒ no license problem

[Bodden et al. 11/2013, Brummund 2014]
Summary: What we’ve learnt

- Apps interact across domains, devices, worldwide
- Apps security model addresses AC only
  - no classical “C.I.A.” security mechanisms
- Apps are protected by Permissions (per component)
- Separation of app code from Manifest.xml files
- Android MAC
  - by matching permissions rule
  - enforced by ICC MAC reference monitor in Android middleware
  - supported by Linux Kernel DAC and SELinux-MAC
- User handling
  - permission groups too coarse
  - user unfriendly permission confirmation model
  - PEM support available, but hard to use
  - Android 6.0 (Marshmallow) and iOS-Permissions are better, more precise
**Self Data Protection (Selbstdatenschutz):**


**Criticism on Permission Policy of Android:**


**Mobiles Identification:**
**Android Security:**

**iOS Security:**

**general:**
Questions to check your knowledge

1. Summarize the mobile and distributed apps security problem (apps attack... and are attacked...); give examples for each problem you mention.
2. Which layers does the Android architecture consist of?
3. Describe the Android architecture layers: specify their roles and give important examples of their elements (3-5 examples for every layer).
4. Describe the Android application component types: what is their function and intended relationship between one another?
5. Describe the intent mechanism of inter-component interaction (ICC), give examples.
6. Describe the intent addressing mechanism, explicit and implicit.
7. Describe the general mandatory access control (MAC) by labels (like Bell-LaPadula).
8. On which levels does Android enforce its application access security?
9. What are the MAC labels, and what is the matching rule in Android?
10. How does an Android app package specify its protection domain?
11. How does a developer of an Android activity/service/content provider specify the permissions that another app needs to have in order to get access to it?
12. How are broadcast intents protected against unauthorized access?
13. How does an Android app gain an access permission for a component or intent of another app? In what way are users involved in this process?
14. Identify (and shortly explain) the Android permission groups.
15. Read and interpret concrete Android Manifest.xml files.