esse

Security for Systems Engineering – VO 06: Mobile Security

Clemens Hlauschek
Lukas Brandstetter
Christian Schanes

Agenda

News in Mobile Security

Android Threat- und Sicherheitsmodell

Reversing Android Applications

Angriffe Berechtigungen, Intents

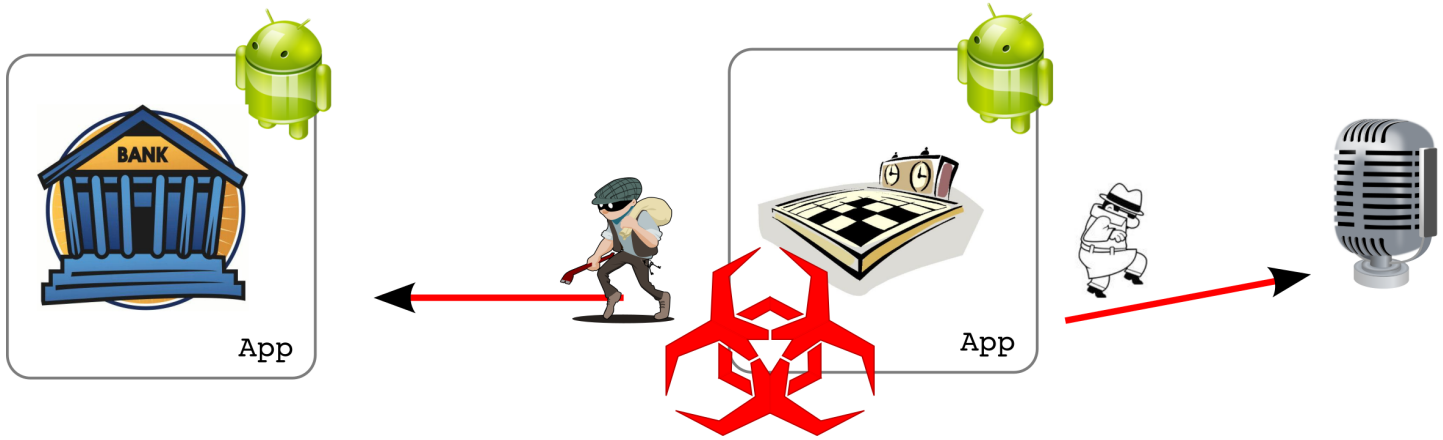
News in Mobile Security

CLEAR: Crypto Wars 2.0

- US-Regierung will Zugriff auf verschlüsselte Smartphones. (Siehe, zB., Cracking the Crypto War, Wired 2018)
- Key Escrow Scheme mit Master-Keys für jedes Device
- Probleme mit Sicherung der Master-Keys. (Siehe, z.B., Matt Green's Blog, 2018))
- Privacy Implikationen?
- Sollen Staaten diese Möglichkeit haben?

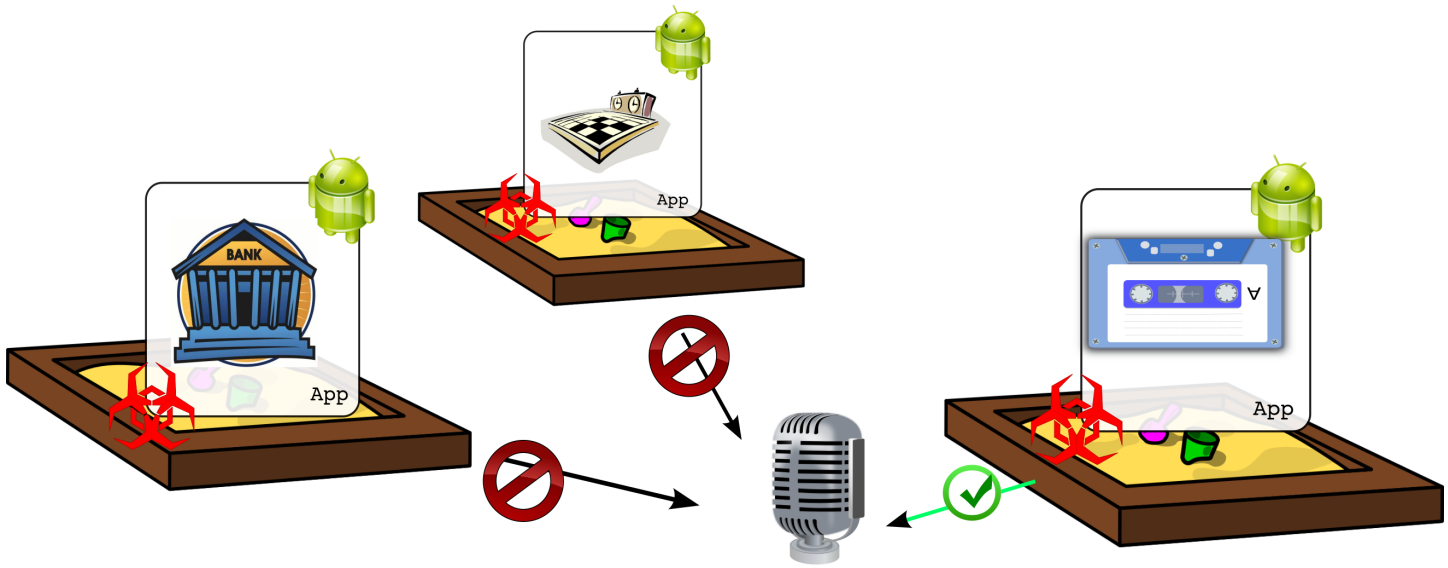
Android Threat- und Sicherheitsmodell

Android-App Threat Model: Motivation



- User hat unterschiedliches Vertrauen in verschiedene Apps
- Game-App soll nicht auf Bankdaten zugreifen können
- Nicht jede App soll auf alle OS-Funktionen (z.B. Mikrofon) zugreifen können)

Android-App Threat Model: Umsetzung



- Jede App potentiell maliziös
- Inter-App Isolation über Sandbox
- Protected System API via Berechtigungen

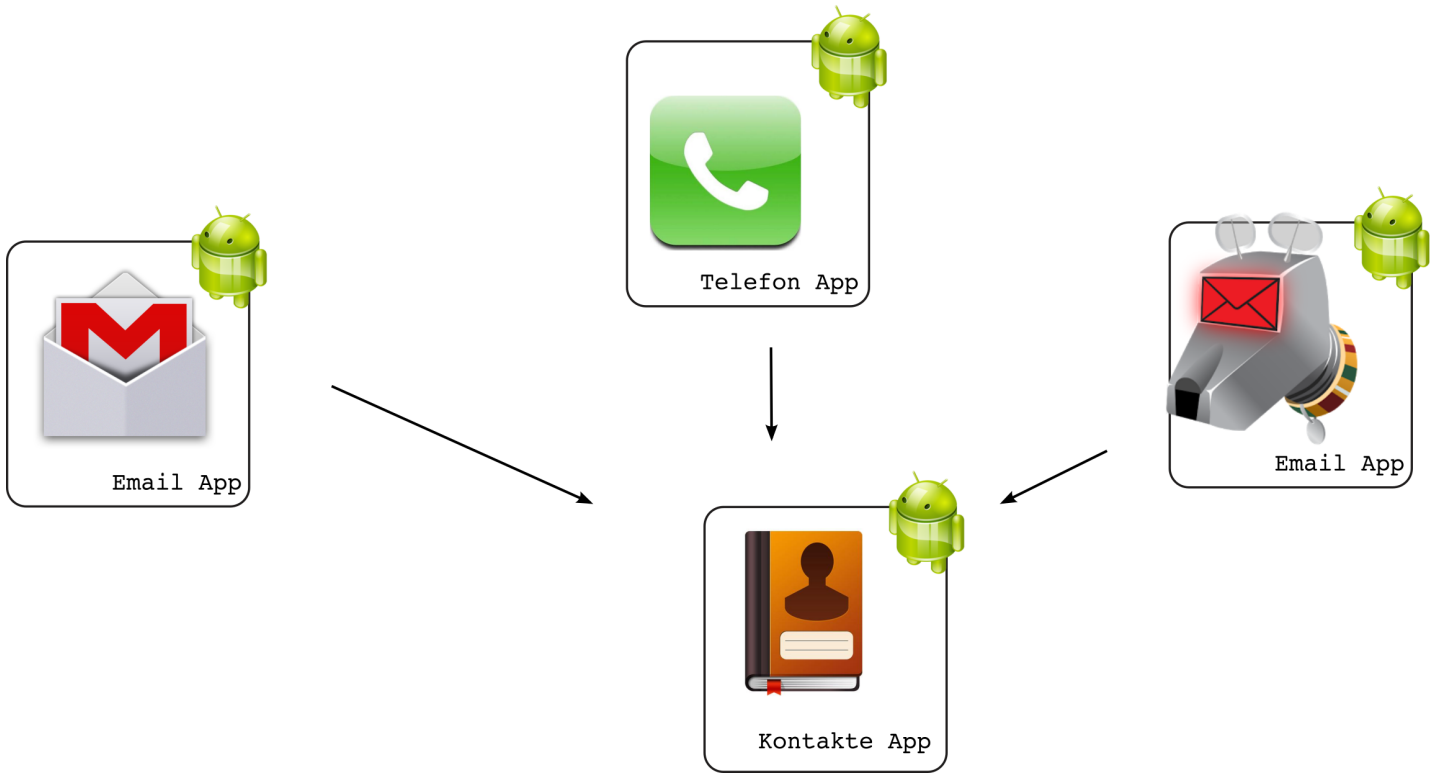
Sandboxing

- Apps laufen in eigener VM isoliert durch OS
- Jede App bekommt eigenen Linux User
  a<number>_a<packagenumber>
- Verzeichnis in /data/user/<usernumber>/<packagename>
- Durch Linux-Berechtigungen geschützt

Android Berechtigungen

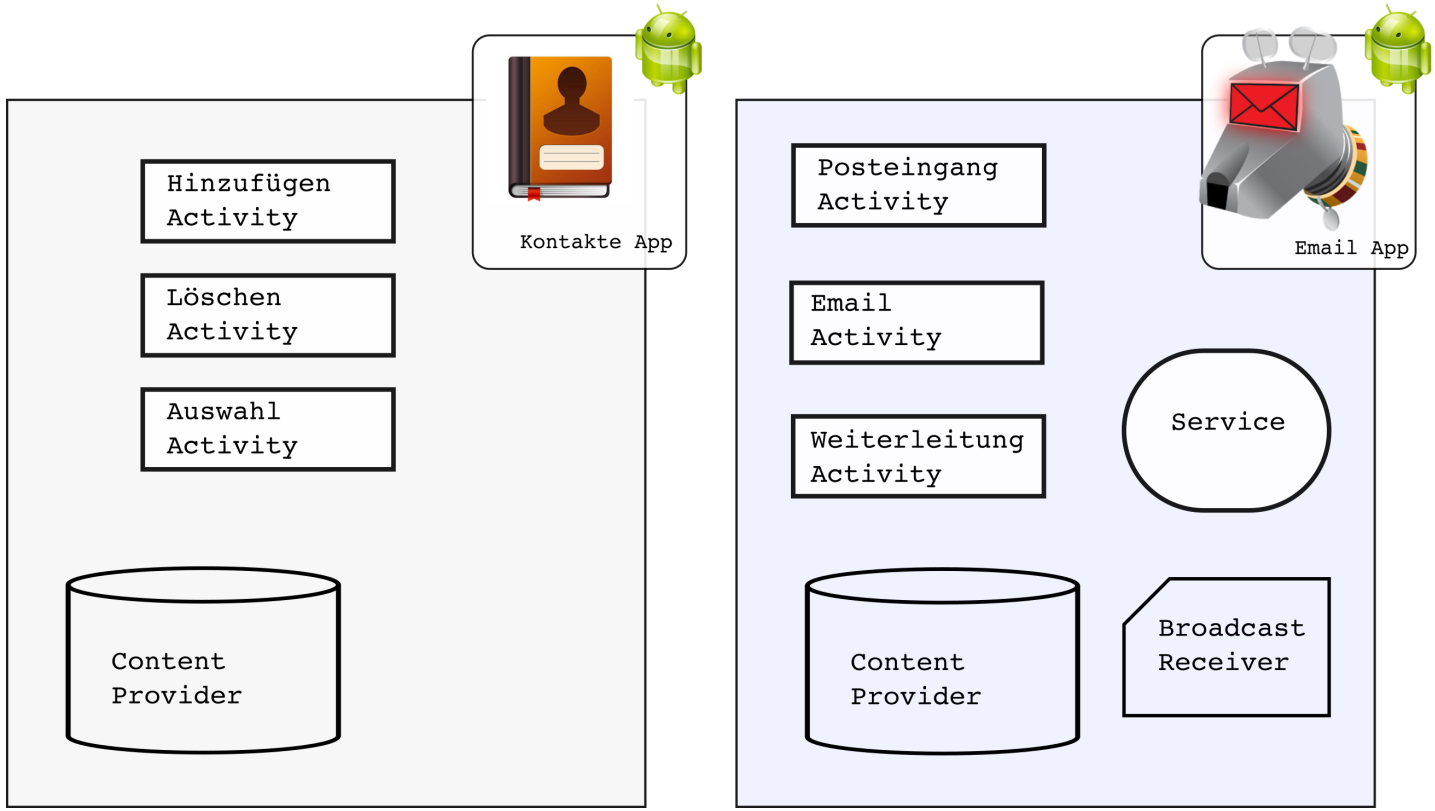
- Berechtigungen steuern Zugriff auf Ressourcen
  - Kamera and Mikrofon
  - Kontakte and Telefonie
  - Location Services
  - ...
- Explizites Einverständnis
- Einhaltung durch OS gegeben
- Feingranularer und widerrufbar ab Android 6

Android-App Ökosystem



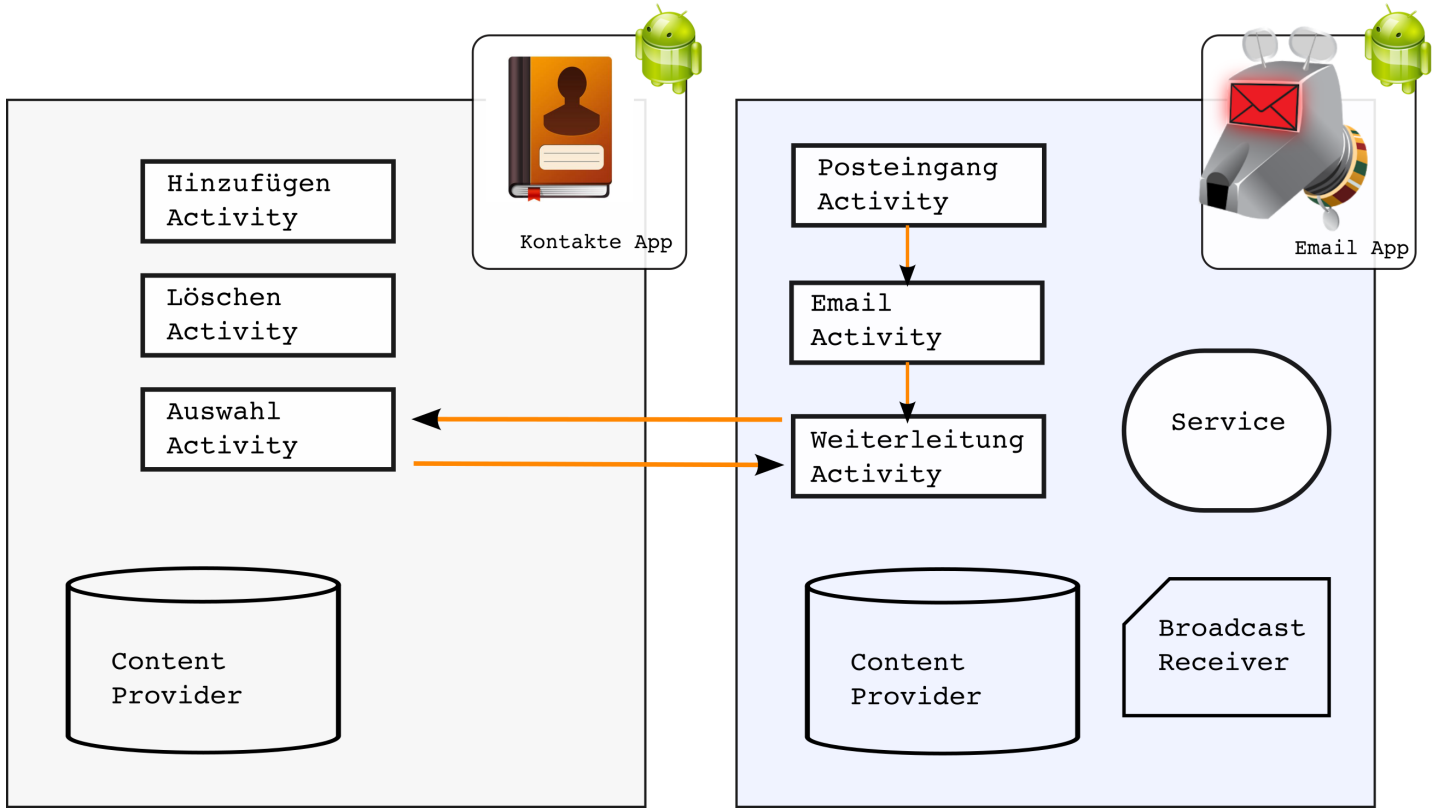
- Unterschiedliche Apps miteinander integriert: z.B. Email-Apps und Telefon-App verwenden Kontakte-App

Android-App Komponenten



- App besteht aus Komponenten: `Activity`, `Service`, `BroadcastReceiver`, `ContentProvider`

Inter Component Communication (ICC): Intents



- Asynchronous message passing System
- Intra- und Interapp Kommunikation

Empfangen von Intents

- Explicit Intent vs Implicit Intent
  - Explicit Intent: An konkrete Komponente addressiert
  - Implicit Intent: An Komponente die spezielle Action erfüllt addressiert (z.B.: `action.SEND`)
- Komponente ist public wenn
  - EXPORTED Flag in AndroidManifest gesetzt, oder
  - Mind. 1 Intent Filter deklariert

Intent Gefahren

- Große Angriffsfläche für maliziöse Apps
- Intents müssen als untrusted Input behandelt werden

- Potenzielle Probleme
  - Informationen auslesen
  - (Arbiträre) Eingaben
  - Schlechte Filterung/Validierung

Reversing Android Applications

Reverse Engineering Android/Java

- Java/Android decompilers
  - Leicht verfügbar, meistens FOSS oder gratis
  - Produzierter Source ist lesbar und nah am Original

- Achtung: Client side code ist nicht von selbst geschützt
- Logik des Programms relativ leicht verständlich

Warum Reverse Engineering?

- App verbessern
- An andere Use-Cases anpassen
- Wissen darüber was die App macht
- Spaß!

Tools

- smali/baksmali
  - Assembler/Disassembler für Android
  - Hiermit kann Code verändert und rekompiliert werden.
- apktool
  - Disassembliert apks zu smali
  - Reassemblieren von funktionierender App möglich
- Decompilers (jadx, Bytecode-Viewer, . . .)
  - Dekompilieren jars und apks zu Java
  - Kein kompletter Source, aber hilfreich für Verständnis der Logik

Smali/Baksmali

```
.method public static a([B[B)[B
    .locals 3
    new-instance v0, Ljavax/crypto/spec/SecretKeySpec;
    const-string v1, "AES/ECB/PKCS7Padding"
    invoke-direct {v0, p0, v1}, Ljavax/crypto/spec/SecretKeySpec
        ;-><init>([BLjava/lang/String;)V
    const-string v1, "AES"
    invoke-static {v1}, Ljavax/crypto/Cipher;->getInstance(Ljava/
        lang/String;)Ljavax/crypto/Cipher;
    move-result-object v1
    const/4 v2, 0x2
    invoke-virtual {v1, v2, v0}, Ljavax/crypto/Cipher;->init(
        ILjava/security/Key;)V
    invoke-virtual {v1, p1}, Ljavax/crypto/Cipher;->doFinal([B)[B
    move-result-object v0
    return-object v0
.end method
```

Jadx

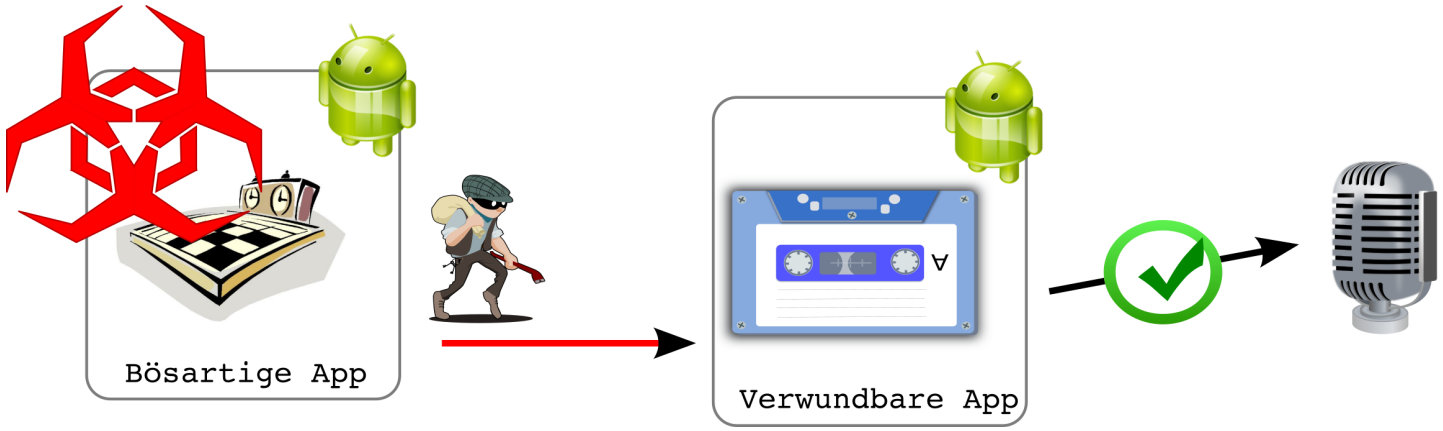
```
public static byte[] a(byte[] bArr, byte[] bArr2) {
    Key secretKeySpec = new SecretKeySpec(bArr, "AES/ECB/
        PKCS7Padding");
    Cipher instance = Cipher.getInstance("AES");
    instance.init(2, secretKeySpec);
    return instance.doFinal(bArr2);
}
```

Hilfestellungen

- Hooking, Instrumentation und Debugging zur Laufzeit
- Sniffen von Netzwerkverkehr
  - Transport Layer Protection muss umgangen werden
  - Schwieriger mit Certificate Pinning
- File I/O
- API und System Calls
- Logs, Debug Output

Angriffe Berechtigungen, Intents

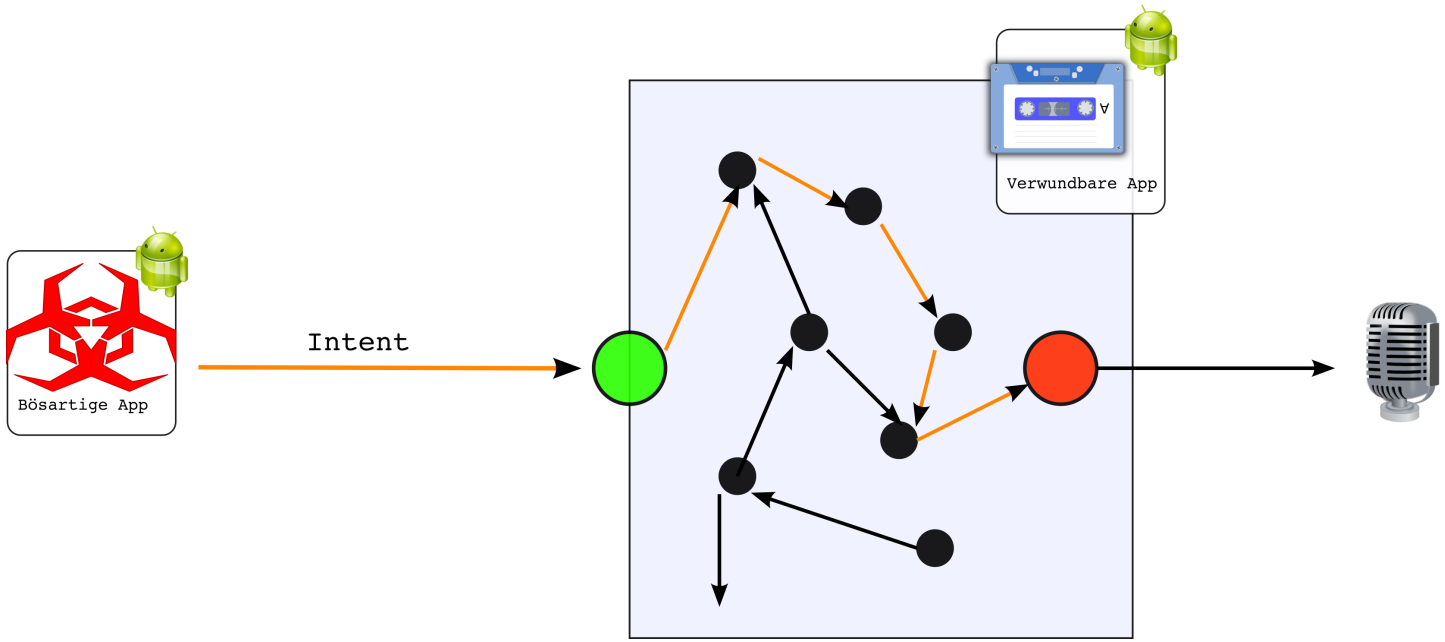
Confused Deputy Attack: Privilege Escalation



- Game-App mit Schadcode, ohne MICROPHONE permission
- Verwundbare Trusted-App (Confused Deputy), mit MICROPHONE Berechtigung
- Game-App sendet Intent an exportierte Komponente in Trusted-App, um Zugang zum Mikrofon zu erhalten.

Voraussetzungen: Confused Deputy Attack, Privilege Escalation

- Verwundbare, aber harmlose App mit mehr Permissions
- Verwundbare App exportiert Komponente
- Code path von public entry point zu Ziel-Resource



Beispiel: Facebook Information Disclosure (Takeshi Terada)

- Ziel
  - Angreifer liest private Daten aus Facebook App
- Facebook App verwundbar, da
  - `LoginActivity` wird von Facebook App exportiert
  - Intent nach `LoginActivity` kann weiteren Intent `continuation_intent` enthalten
  - `continuation_intent` wird von `LoginActivity` im Facebook-App-Kontext weitergesendet

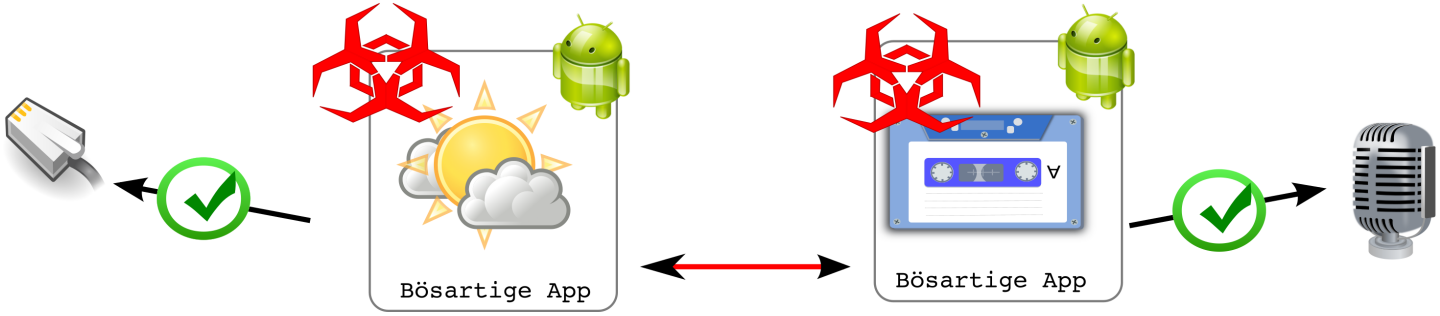
Content Leak und Content Pollution

- Android $<$ 4.2: `ContentProvider` EXPORTED-Flag gesetzt per default
- Zahlreiche verwundbare Apps mit Problemen wie:
  - Auslesen von SMS Nachrichten (z.B. GoWidget), Contacts, User Credentials, Browser History, Call Log
  - Veränderung von Blacklists $\rightarrow$ Blockiere Anrufe, SMS (DoS)

Gegenmaßnahmen

- Nur Komponenten exportieren wenn absolut notwendig.
- Custom permission, Permission Level “dangerous” oder “signature”
- Wenn Komponente intern wie extern zugänglich sein muss: Aufteilen
- Saubere Programmierung

Collusion-Angriffe



- Angreifer installiert mehrere Apps
  - Apps kommunzieren miteinander, oder
  - Sind installiert unter der selben UID (`sharedUserId` Attribute)
- Z.B. Recorder-App, Zugang zu Mikrofon; und Wetter-App, Zugang zum Netzwerk
- Angreifer erhält so die Vereinigung der Berechtigungen

Zusammenfassung

- Android-Architekur
- Verschiedene Aspekte des Android Sicherheitsmodells
- Angriffe, um das Sicherheitsmodell zu umgehen
  - Confused Deputy Attack
  - Collusion-Angriffe
- Häufige Fehler bei der Entwicklung

Interesse geweckt?

- Advanced Security for Systems Engineering WS2018
- Projekte (PR)
- Bakkalaureatsarbeiten
- Diplomarbeiten, Dissertationen
- Mögliche Themengebiete
  - Automatisierung von Security-Tests für Android/iOS
  - Android/iOS Exploitation
  - Ubuntu Phone

Literatur und Links – 1/3

- https://developer.android.com/guide/
- https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- Elenkov (2014): Android Security Internals
- Drake, et al. (2014): Android Hacker's Handbook
- Zhou, et al. (2012): Dissecting Android Malware: Characterization and Evolution
- Felt, et al. (2011): A Survey of Mobile Malware in the Wild
- Felt, et al. (2011): Android Permissions Demystified

Literatur und Links – 2/3

- Enk, et al. (2009): Understanding Android Security
- Bugiel, et al. (2012): Towards Taming Privilege-Escalation Attacks on Android
- Zhang, et al. (2014): Appsealer
- Chin, et al. (2011): Analyzing Inter-Application Communication in Android
- Zhou, et al. (2013): Detecting Passive Content Leaks and Pollution in Android Applications
- Jana, et al. (2012): Memento: Learning Secrets from process footprints

Literatur und Links – 3/3

- Schlegel, et al. (2011): Soundcomber: A Stealthy Context Aware Sound Trojan for Smartphones
- Deshotels (2014): Inaudible Sound as a Covert Channel in Mobile Devices
- Fahl, et al. (2013): Why Eve and Mallory Love Android. An Analysis of Android SSL (In)security
- Bianchi, et al. (2015): What the App is that? Deception and Countermeasures in the Android User Interface
- `https://ibotpeaches.github.io/Apktool/`
- `https://github.com/JesusFreke/smali`
- `https://github.com/skylot/jadx`

Vielen Dank!

https://security.inso.tuwien.ac.at/