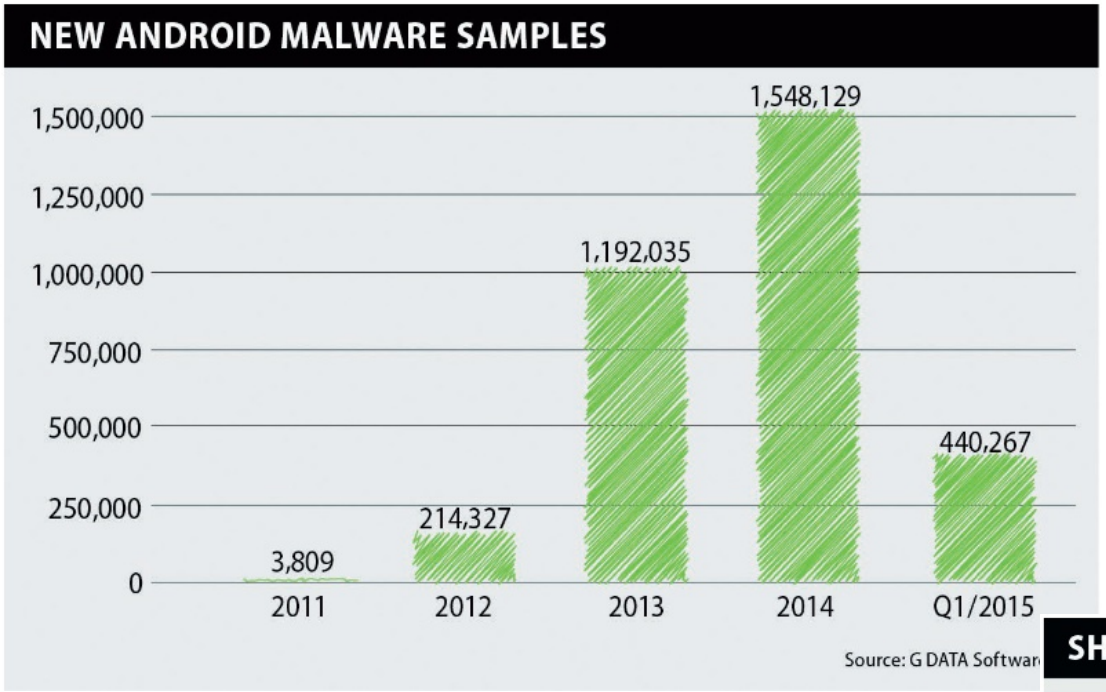
Common Weaknesses of Android Malware Analysis Frameworks

Lars Richter

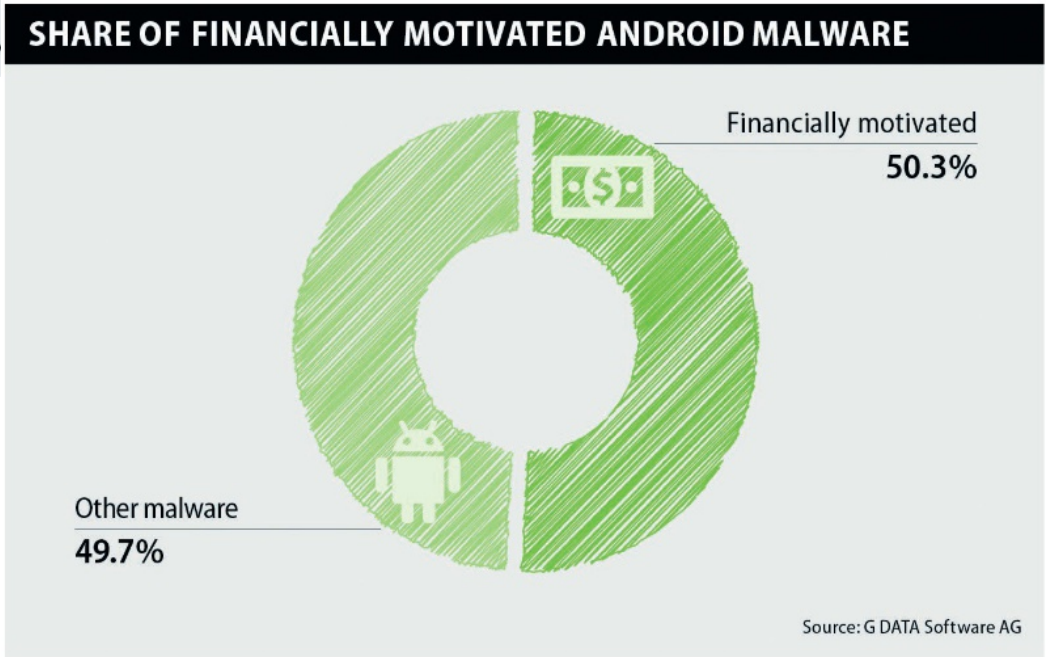
Aufbau

- Android Malware
- Analysis Frameworks
- Schwachstellen
  - Code Verschleierung
  - Fingerprinting
  - Verdeckte Kommunikation
  - Unerwartete Ereignisse
- Fazit

Android Malware



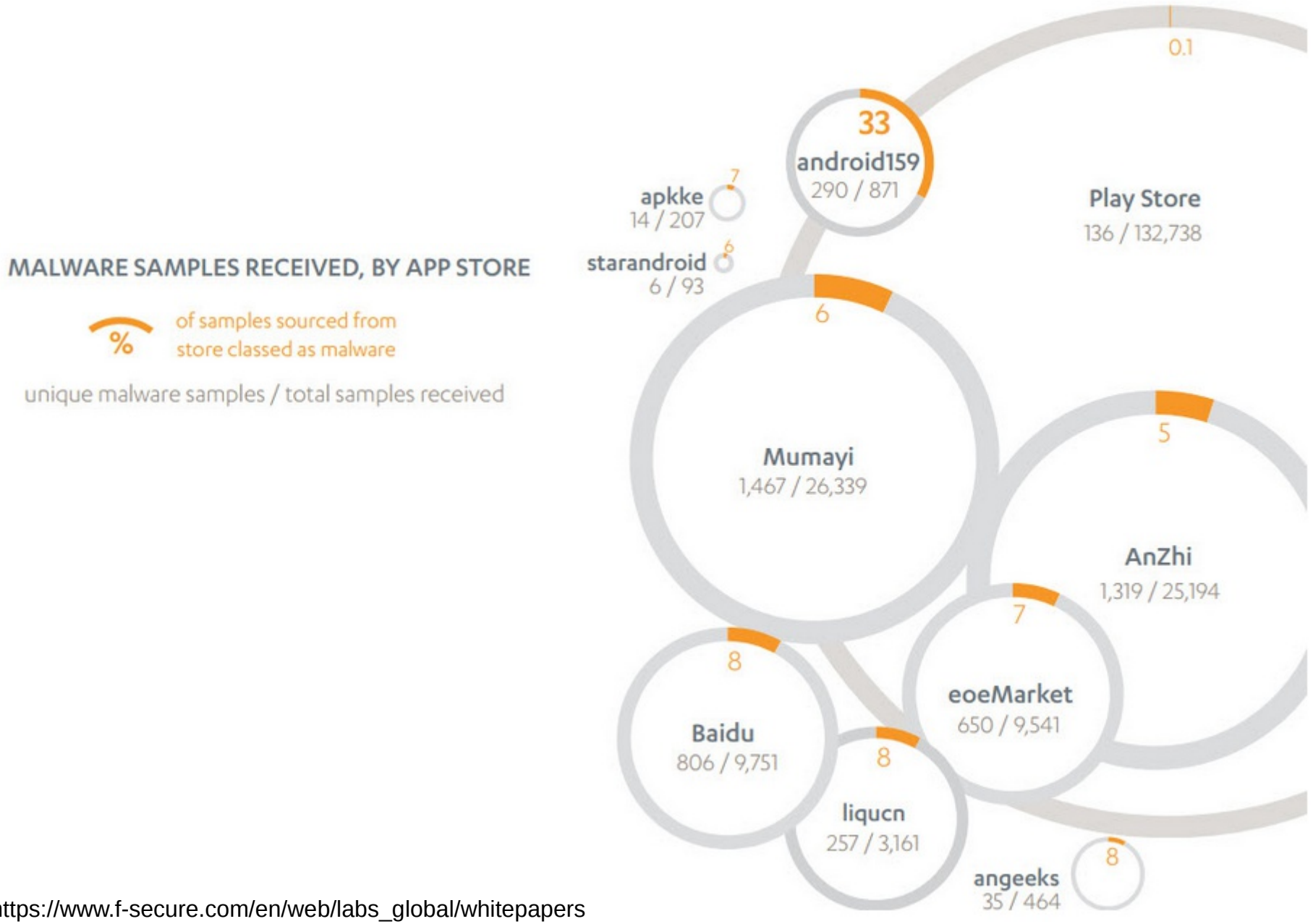
- 1Q15 >440.000 malware Proben bei GDATA
- 21% mehr als im 1Q14



- >50% finanziell motiviert
  - e.g. Bank-, SMS-Trojaner
- andere Malware meist Spyware, Adware, Cryptolocker

Source: https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/G_DATA_MobileMWR_Q1_2015_US.pdf

Malware Proben nach AppStore



Source: https://www.f-secure.com/en/web/labs_global/whitepapers

Google Bouncer

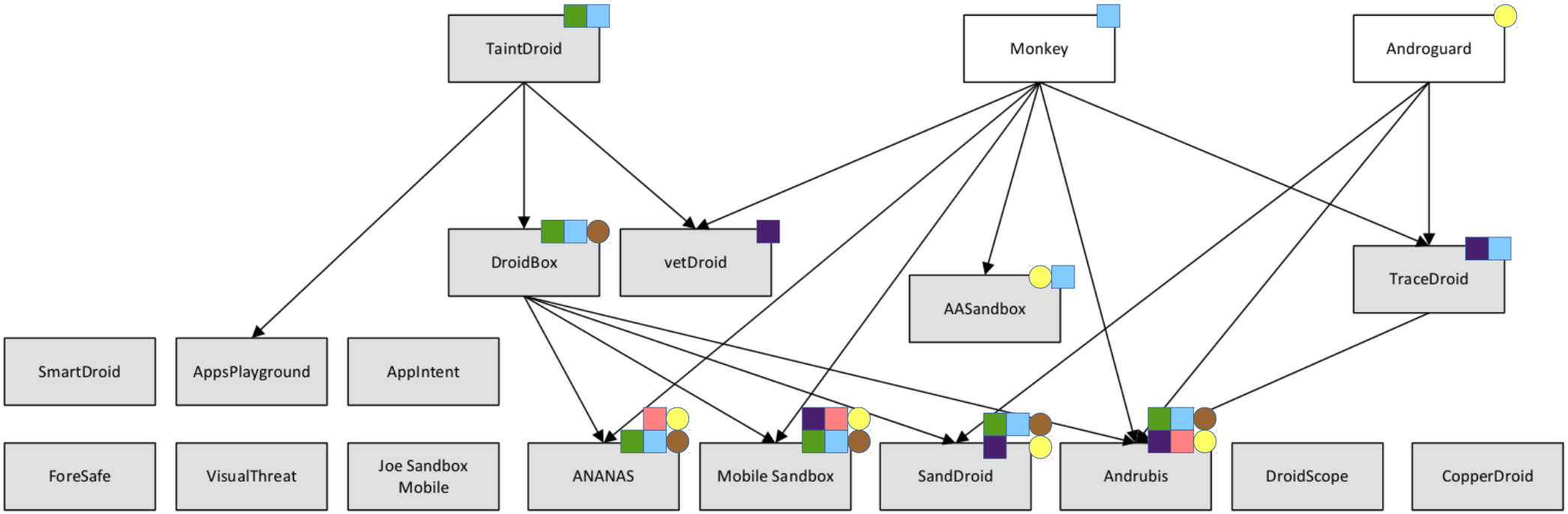
- Aktiv seit 2012
  - 40% Rückgang von Malware im Play Store
- Statische + Dynamische Analyse
- Fingerprinted:
  - Standard QEMU emulator
  - 5min Test
  - Vorhersehbare Eingaben
- Reicht aus für normale Malware

Quelle: https://jon.oberheide.org/files/summercon12-bouncer.pdf

Malware Analysis Frameworks

- Typen
  - Statisch
    - Analyse von Meta-Informationen (Manifest, Services, etc)
    - Byte Code Weaving (Trace-Funktionen einbetten)
    - Decompilieren
  - Dynamisch
    - Taint Tracking
    - Virtual machine introspection
    - Method tracing
  - Hybrid
    - Statische Analyse zur Unterstützung der Dynamischen

Android Sandbox Landschaft

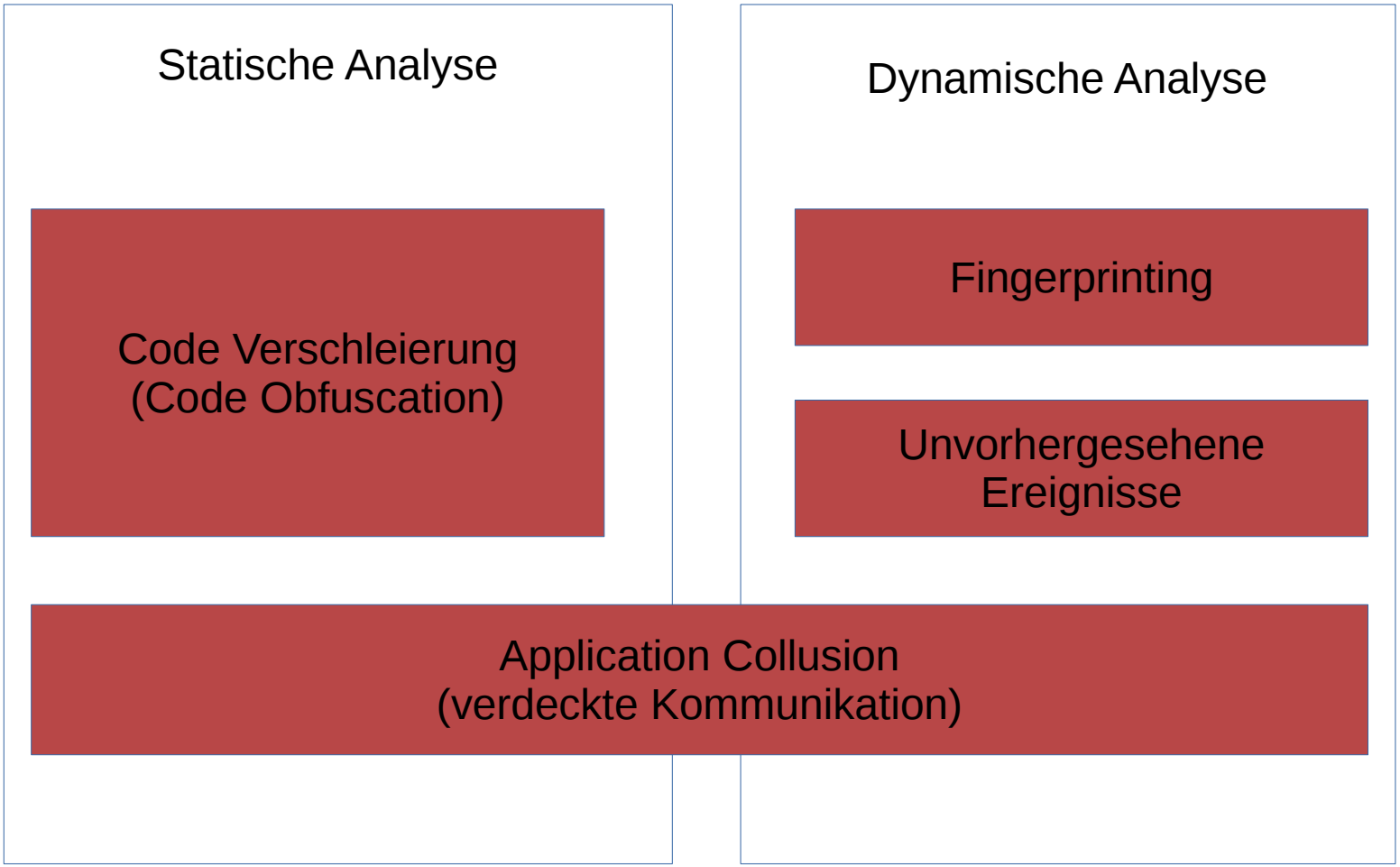


Abhängigkeiten der genutzen Tools und Dienste

- Statische Analyse
- ByteCode Weaving
- Dynamische Analyse
- Native Code
- Taint Analyse
- Tracing

Quelle: Sebastian Neuner, et al. Enter sandbox: Android sandbox comparison.

Schwachstellen



Code Obfuscation / Verschleierung

- Triviale Transformationen
  - Neu Erstellen der APKs, Änderungen im Manifest
  - Gegen Erkennung basierend auf Hashwerten
- Erkennbare Transformationen
  - Umbenennung von Klassen / Methodennamen
  - Änderung des Datenflusses
  - Ausführen von verschlüsseltem Native Code
- Nicht Erkennbare Transformationen
  - Java Reflection API
  - Laden von verschlüsselten externen Anwendungsteilen
    - Dynamisches Nachladen zur Laufzeit möglich

Code Obfuscation - Beispiel

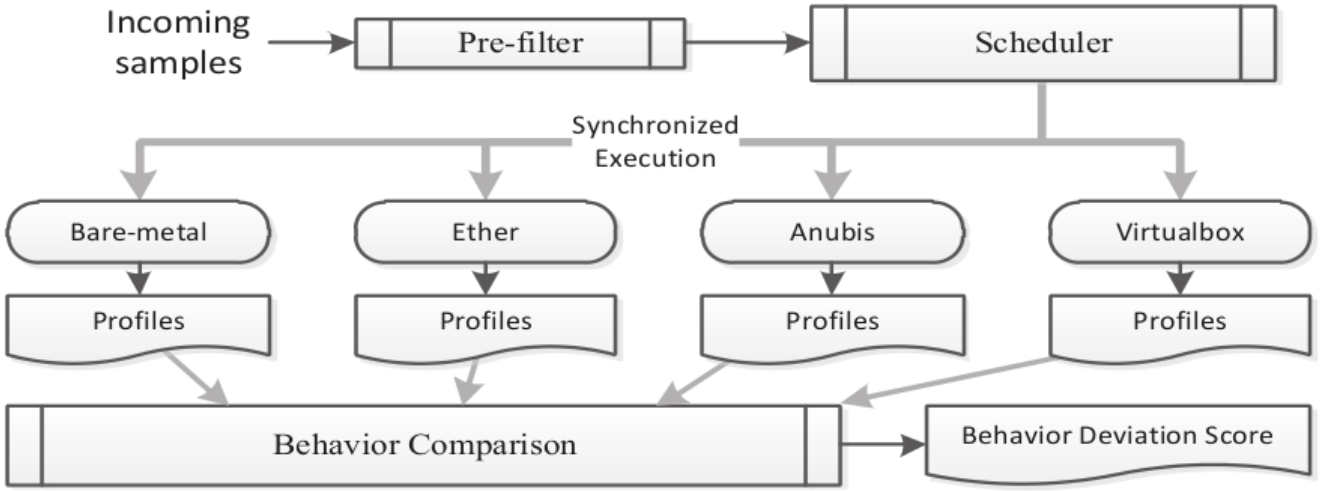
- Malware Android/BadAccents
  - Versand von Informationen via Email
  - Auffälliger API Call: Versand von Email
  - Nützliche Infos: Empfänger, Email Body
  - Obfuscation:
    - sensitive Informationen in Native Code verschlüsselt
    - Nicht auffindbar mit statischer Analyse

Fingerprinting

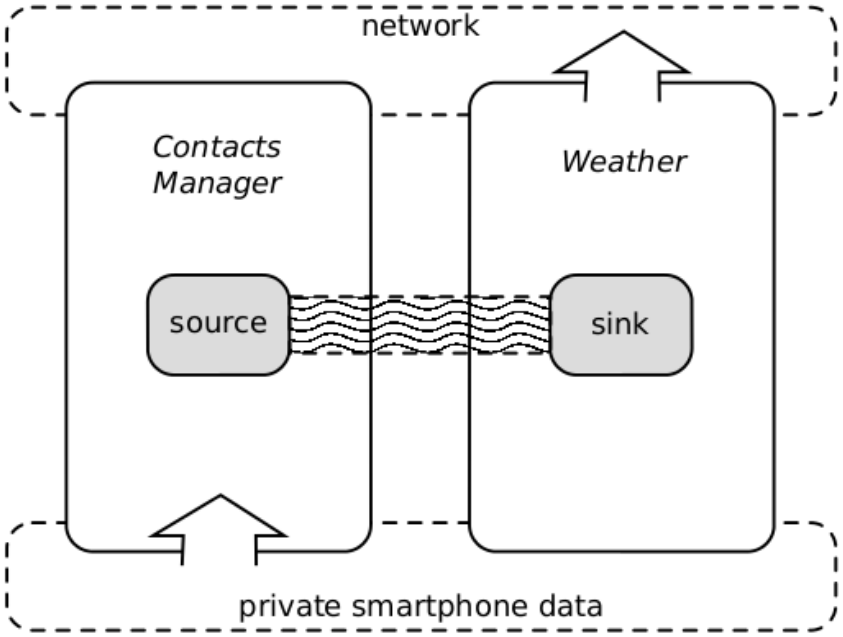
- Problem
  - Im Gegensatz zum PC Sensoren nicht modular
    - GPS, Helligkeits-, Beschleinigungs-, Temperatursensor, ...
    - Müssen ansprechbar sein!
- Fingerprinting Möglichkeiten
  - Auffälligkeiten der Sensoren
  - Virtualisierte Smartphones
  - Nutzerinteraktion
  - Protokoll Implementierungen
  - persönliche Informationen
  - ...

Fingerprinting - Gegenmaßnahmen

- Bestimmte, bekannte Instruktionen ersetzen
- Simulieren normaler Benutzung
- BareCloud Ansatz
  - Kombination aus „bare-metal“, transparenter und einfacher Analyse
  - Erkennen von Verhaltensunterschieden



Verdeckte Kommunikation



Hauptproblem: Analyse einzelner Apps, abgeschirmt von ein ander

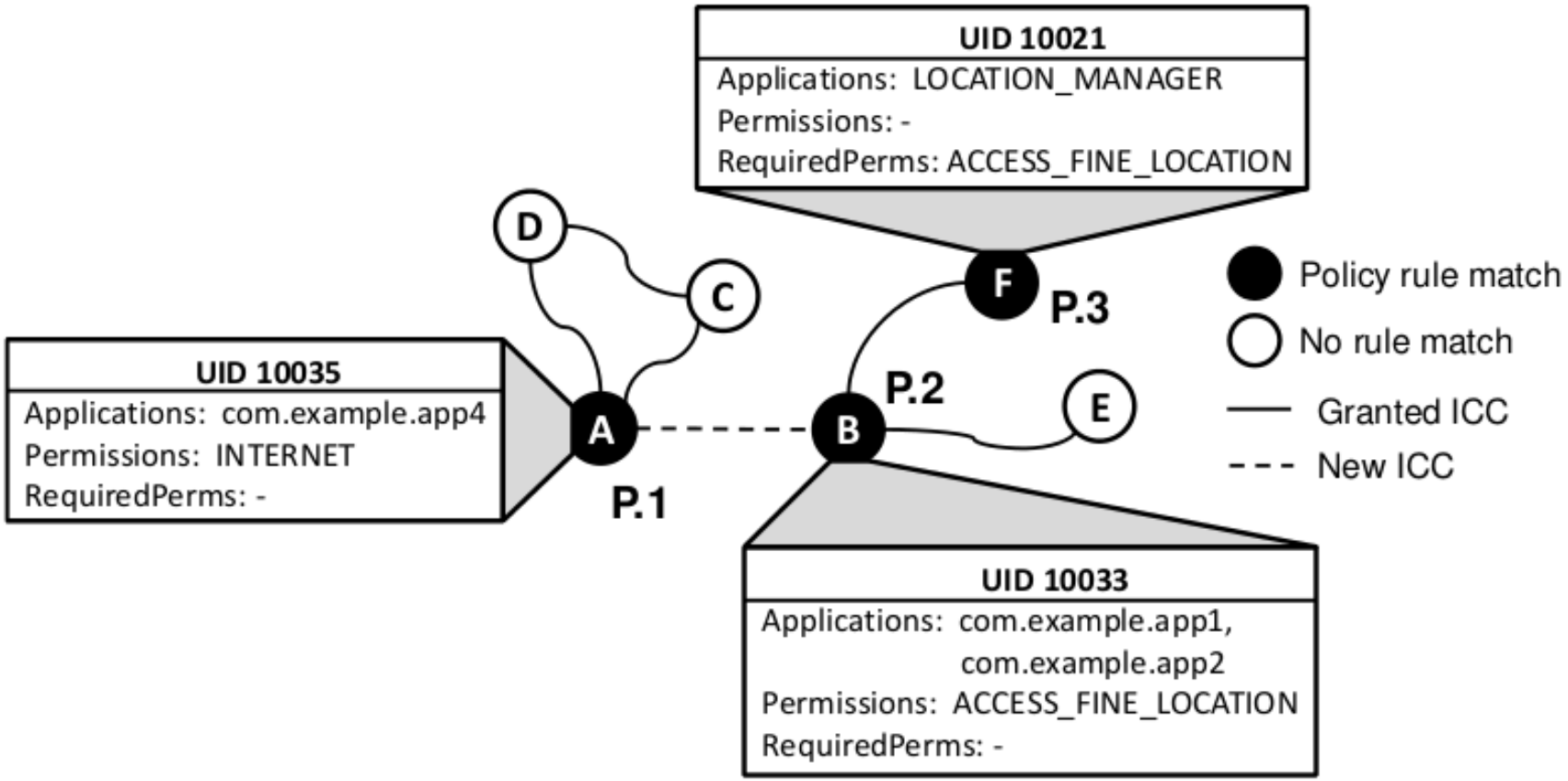
Quelle: Claudio Marforio et al. Analysis of the communication between colluding applications on modern smartphones. In Proceedings of the 28th Annual Computer Security Applications Conference, pages 51–60. ACM, 2012.

Verdeckte Kommunikation

- Offene Kanäle
  - Shared configuration
  - Broadcast Intents
- Verdeckte Kanäle
  - Lautstärke Einstellungen (150bps)
  - Datei Sperren (685bps)
  - Intent-Typen (4324bps)
  - Nachteil: Synchronisationszeit, bei Intent-Typen 716ms

Verdeckte Kommunikation - XManDroid

- Erweitert Berechtigungs Framework von Android
- Erkennt Privileg Eskalation auf App Ebene



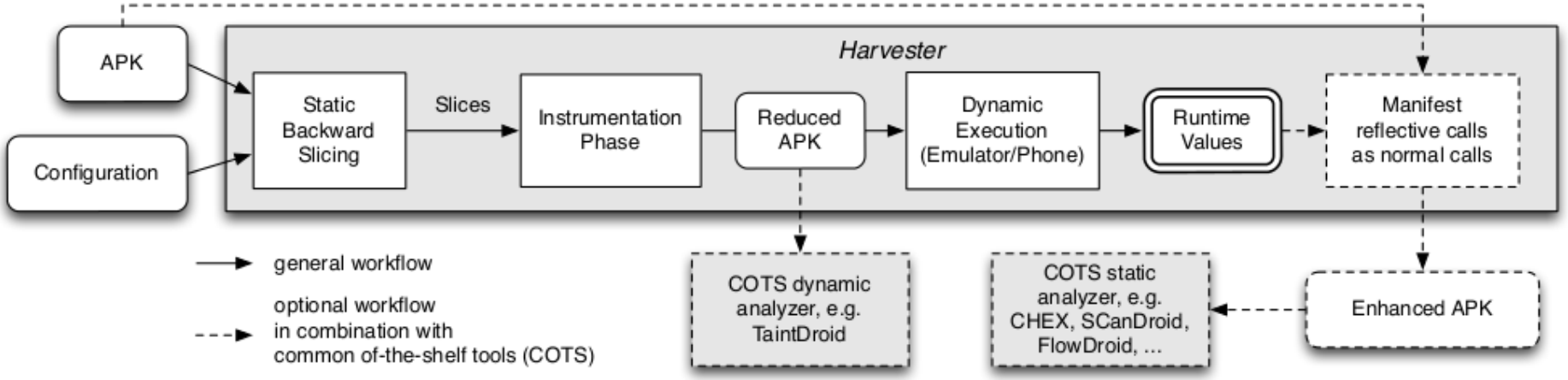
Quelle: Sven Bugiel et al. Technische Universität Darmstadt, Technical Report TR-2011-04, 2011

Unvorhergesehene Ereignisse

- Externe Ereignisse
  - Intent-Filter auf ankommende SMS, Anruf, ...
  - Können Malware aktivieren
  - Verdeckte Kanäle
- Zeitliche Ereignisse
  - Aktivierung der Malware nach bestimmter Zeit
    - Überlistet z.B. Google Bouncer
  - Bsp.: Durak Game – 5-10Mio. Installationen
    - Bösartiges Verhalten nach 30 Tagen
- Statische Analyse kann Intent-Filter und Timer erkennen

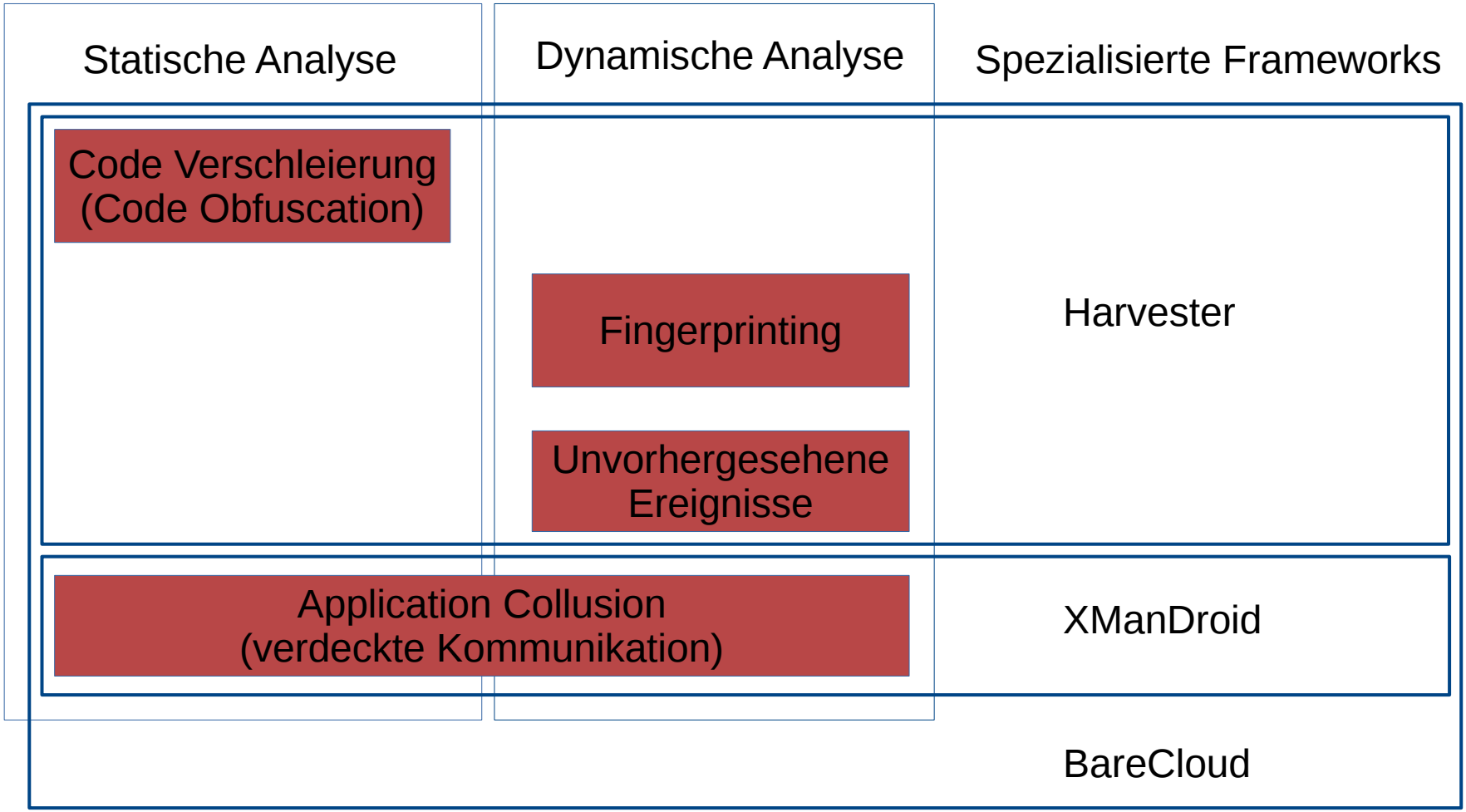
Unvorhergesehene Ereignisse - Harvester

- Robust gegen unvorhergesehene Ereignisse und nicht erkennbare Code Transformationen
- Zerteilt Programm in Abschnitte
- Simuliert alle möglichen Zustände
- Erzeugt analysierbare APK ohne Anti-Analysis Methoden



Quelle: Siegfried Rasthofer et al. Harvesting runtime data in android applications for identifying malware and enhancing code analysis. 2015.

Fazit


Statische Analyse
Dynamische Analyse
Spezialisierte Frameworks
Code Verschleierung (Code Obfuscation)
Fingerprinting
Unvorhergesehene Ereignisse
Harvester
Application Collusion (verdeckte Kommunikation)
XManDroid
BareCloud