Android Security

Christian Küster <c.kuester@tarent.de>

↪ tarent

FrOSCon 2009

Agenda

- **Sicherheitskonzept** der Android-Plattform (eine Bestandsanalyse)
  - Einführung - Was ist Android?
  - (Interessante) Komponenten von Android
  - Binäre Einschlüsse
  - Berechtigungskonzept und Durchsetzung
  - Zugriffsschutz und Prozess-Isolation
- **Erweiterung** um Sicherheitsfunktionen
  - Virtual Private Network
  - Dateiverschlüsselung
  - SmartCards

Allgemein

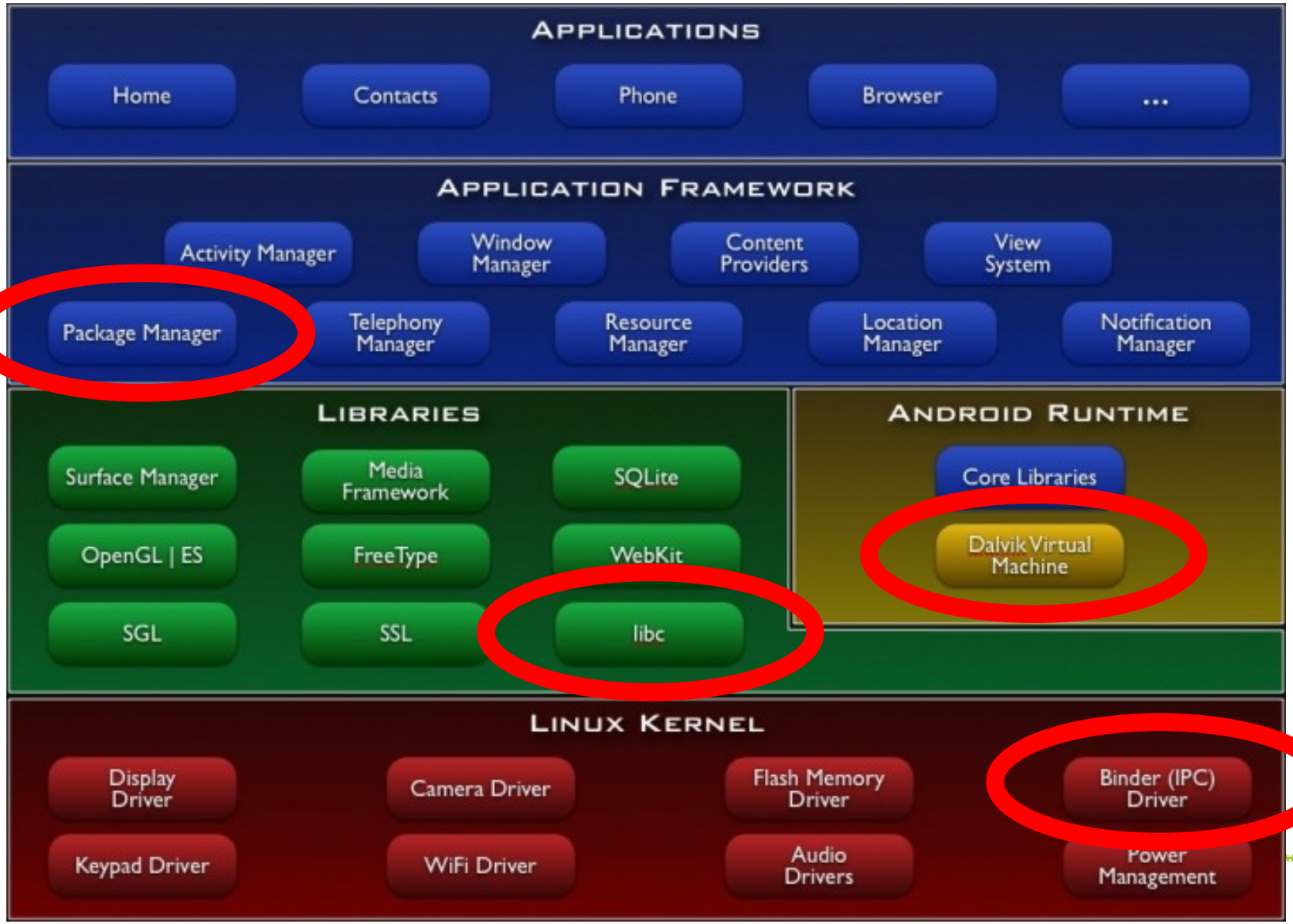
- **Android** ist eine (offene?) Plattform für mobile Geräte
  - PDAs, Netbooks, mobile Telefone
  - Wir haben nur ein Entwicklermodell (G1)
  - Retail-Geräte sind sehr geschlossen
- Plattform ist ein **Gesamtkonzept**
- **Betriebssystem** bis Telefon/SmartPhone **Framework**
  - Basiert auf Linux
  - Basisset von Telefon/SmartPhone Applikationen
- **Sicherheit** der Plattform offensichtlich Entwurfsziel
  - Im Sinne eines durchplanten Sicherheitsmodells

Motivation – The Neverending Why

↪ tarent

- **Malware** ist auch für Telefone eine Gefahr
- **SmartPhones** enthalten eine Reihe von **sensiblen Gütern**
  - Kontaktdaten
  - Adressen von Remote-Peers
  - Browser-Historie
  - Historien von Anrufen/Short-Messages
  - ...
- Wussten Sie, dass Ihr iPhone **Screenshots** von laufenden Applikationen ablegt? (vgl. „Mobile Malware“, Dunham, 2009, p. 314)
- Android ist **quelloffen** und es weckt Neugier

System-Architektur



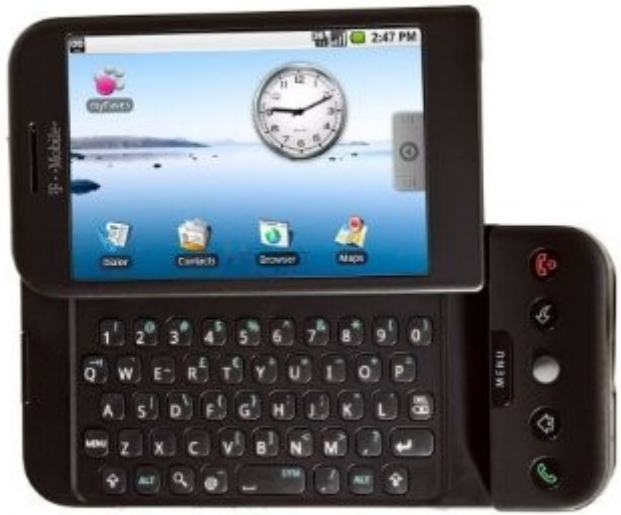
Komponenten von Android .. ? (eine Auswahl)

↪ tarent

- **Kernel** mit Android-Spezifischen Anpassungen
  - Ashmem
  - Logger
- **OpenBinder** (oder Binder) ist verwendetes IPC-Framework
  - System-V-IPC wurde unter Sicherheitsbedenken entfernt
- **Bionic** ist ein Subset der NetBSD-Standard-C-Bibliothek
- **Framework**
- Virtual Machine: **Dalvik**
- Ein Set von **externen Bibliotheken** (OpenSSL, webkit, SQLite)

Wo kann man sich Android ansehen?

- Ein Retail-Gerät kaufen?
- Entwicklertelefon G1 (HTC Dream)
  - Radio Layer Userspace-Bibliothek
  - Wlan Kernel-Modul
  - GPS-Treiber
  - Audio-Treiber
  - Kamera-Treiber
  - Flashbar (auch Kernel!)
- Angepasster Qemu
  - Native Ausführung der Plattform



Applikationen

↪ tarent

- Android bringt einen **Paketmanager** mit
  - Einzige Möglichkeit Pakete zu installieren (Retail)
- **Applikationspakete** sind immer vom Entwickler **signiert**
  - Entscheidet nicht, ob Applikation überhaupt installiert werden darf
  - Vertrauensbeziehung zwischen Applikationen
- Applikationsdeployer weißt automatisch neue Linux Benutzer-Identität (UID) zu
  - UID-Sharing nur bei gleicher Entwicklersignatur
- Bionic bietet spezielle Funktionen dafür
  - kein /etc/passwd bzw. shadow
- Applikationen bestehen zur Zeit nur aus Dalvik-Byte-Code
- Dalvik-Byte-Code wird aus Java-Byte-Code transformiert

Berechtigungssystem

↪ tarent

- **Systemkritische Aktionen** benötigen Berechtigungen
- **Deny-All Ansatz** für zu installierende Applikationen
- Berechtigungen werden statisch vergeben
  - AndroidManifest.xml
  - Eine vordefinierte Menge steht bereit: INTERNET_ACCESS, READ_CONTACTS, ...
  - Benutzer entscheidet bei der Installation
  - Werden während der Laufzeit geprüft
- **Selbstdefinierte Berechtigungen** möglich
  - Bspw. für den Schutz eigener Services
- Jede Berechtigung hat ein **Risk-Level**

Durchsetzung von Berechtigungen

↪ tarent

- Anhand von **Linux-Gruppen** und **Linux-ID**s
  - GID werden beim App-Start mitgegeben
  - Mapping GIDs: /etc/permission/platform.xml
  - z.B. Content-Provider prüft UID
- **OpenBinder** IPC-Implementierung
  - Prüft Zugriffsberechtigungen auf (Binder-)Objekten
- **Kernel**
  - Beispiel INTERNET_ACCESS
  - Der Kernel wurde erweitert
  - Überprüfung der Gruppenzugehörigkeit “inet”
- **Java-Security-Manager**?
- **Eskalation** der Zugriffsverletzung ins Framework (SecurityExceptions)

Durchsetzung von Berechtigungen – Beispiel Netzwerk

↪ tarent

Aus dem Kernel: ./net/ipv4/af_inet.c

```
#ifdef CONFIG_ANDROID_PARANOID_NETWORK
static inline int current_has_network(void) {
    return (!current_euid() || in_egroup_p(AID_INET) ||
        in_egroup_p(AID_NET_RAW));
}
static inline int current_has_cap(int cap) {
   if (cap == CAP_NET_RAW && in_egroup_p(AID_NET_RAW))
       return 1;
   return capable(cap);
}
# else
static inline int current_has_network(void) {
   return 1;
}
static inline int current_has_cap(int cap) {
    return capable(cap);
}
#endif
```

Prozess-Isolation

- Android ist ein **Multiprozess-System**
- Linux-Prozess als **Vertrauenseinheit**
- **Eigener Benutzer** für **jede** Android-Applikation
  - Paketmanager weißt diese zu
- UID-Sharing möglich (vgl. Vertrauensbeziehung)
- **System-V-IPC** entfernt
  - Interprozesskommunikation durch Binder
  - Kein einfaches Shared-Memory
- **Virtualisierung** durch Dalvik-VM **Verteidigungslinie**?
  - Nein
  - JNI für Applikationen ist Entwurfsziel der Plattform
  - Möglich mit Android Development Kit

Zugriffsschutz

↪ tarent

- Basiert auf Linux **Discretionary Access Control**
  - -rw-r--r-- 1 christian christian 248K 20. Jun 18:42 android-security.odp
- Android-Applikationen legen Dateien ausschließlich unter ihrer Benutzer-ID ab
- Android bietet eigene Konzept zur **Datenkapselung** an: Content-Provider
  - Zugriff kann öffentlich gemacht werden
  - Kann mit Zugriffsberechtigung versehen werden!
  - Beispiel: Kontaktdaten sind durch Content-Provider gekapselt (READ_CONTACTS)
  - Daten sind auch abgelegt unter eigener Benutzer-ID
- WORLD_{WRITEABLE|READABLE} Berechtigung möglich
- /data/data/<paketname>/

Zugriffsschutz - Partitionsebene

↪ tarent

- /, /system, /data, /cache, ...
- Schreibbar ist grundsätzlich nur
  - /data und /cache Partition
  - Und Mount-Point /sdcard
- Alle Systemprogramme und Bibliotheken sind nach /system verlagert und nur lesbar
  - /system/bin, /system/lib, /system/etc, ...
- Root (/) ist nur lesbar
- SD-Karten werden „noexec“ eingehangen
- Updates von Systemprogrammen (Beispiel: Dialer) werden durch ein Overlay in /data ermöglicht

Erweiterung um Sicherheitsfunktionen

- Geht es überhaupt?
  - Die hier vorgestellten Sicherheitsfunktionen benötigen Änderungen bzw. Erweiterungen auf Kernel und Systemebene
  - Daher nur auf einem Entwicklertelefon
  - Bedingte Flash-Möglichkeit von Retail-Geräten
  - Kernel muss meistens geflasht werden
- Was ist interessant?
  - Virtual Private Network Clients
  - Dateiverschlüsselung
  - SmartCards als Vertrauensanker
  - Sicherheitspolicies

Portierung von Software für Android

↪ tarent

- Cross-Compiling-Vorgang
- Standard Lib-C „bionic“ ist nur ein Subset der GLIBC oder NetBSD Libc
  - Thread Cancellation
  - Shared Memory
  - Mlock() Funktionsfamilie
  - ...
- Für Android-Buildprozess müssen eigene Makefiles (Android.mk) geschrieben werden (von Hand!)
- Set von externen Bibliotheken klein

Virtual Private Network

- Sehr interessant, weil **viele Einsatzgebiete**
  - Lösungen stehen anscheindend bereit
  - Problem: Kernel-Config Änderungen nötig
    - tun/tap für OpenVPN
    - IPsec/l2tp
    - Strongswan setzt auf Funktionen auf, die nicht in bionic vorhanden sind (thread cancellation)
- Google Clients arbeiten alle auf **https** Basis
  - Zum Beispiel Kontakte synchronisieren
- Keine Unterstützung für **System-Policies**
  - Verhindere Nicht VPN-Traffic
  - Iptables vorhanden

Virtual Private Network

- OpenVPN ist portierbar
  - Etwas tricky mit der Konfiguration
  - Kein Privilege-Dropping (via setuid(..))
- Steuerung und Start
  - Start Schwierig direkt aus Android (braucht root)
  - Als Background-Prozess im Management-Modus starten (braucht exec im init)
  - Steuerung durch Config-File
  - SIGHUP über Management-Console
- Geht bisher nur auf Entwicklergerät (Kernel-Config)
- BSD-Like ifconfig bei Android -> Busybox ifconfig

Dateiverschlüsselung

↪ tarent

- Einige OSS-Alternativen aus dem Linux-Umfeld
  - Dm-crypt
  - Loop-AES
  - eCryptFS, ...
- Dm-crypt und Loop-AES setzen Container ein,
  - Wir haben jedoch YAFFS2 als Dateisystem wegen eingebauten NAND-Speicher
  - Außer DM-Crypt bisher nichts im Standard-Kernel
- EcryptFS braucht in gewissen Fällen Shared-Memory
  - Userland Werzeuge haben eine Reihe Abhängigkeiten
  - Sys-Dateisystem nötig

Dateiverschlüsselung (2) - eCryptFS ↪ tarent

- Macht **Overlay** über bestehendes Dateisystem
- Interessane Mount-Points: **/cache**, **/data**
- Portierte **Abhängigkeiten**:
  - libgpg-error/Libgrypt (teil von GnuPG)
  - Keyutils
  - Ecryptfs-utils (Library + mount.ecryptfs)
- **Blocker**: Was geht nicht (aktuell)?
  - Daemon -> Muss auf Binder/Ashmem umgestellt werden
  - Public-Key-Verschlüsselung (openssl, pkcs11)
- Braucht **Erweiterung** der init-Sprache um „exec“ bzw. „exec serial“
- Dateinamen-Obfuscation erst ab 2.6.29

Dateiverschlüsselung (3) - Tücken

- Telefone bergen ihre ganz eigenen Tücken
- „Cold-Boot“-Angriffe
  - PCs macht man irgendwann aus
  - Mobile Telefone meistens nicht
  - Sie kommen häufig weg, wenn sie noch angeschaltet sind
  - Nützt mir dann Verschlüsselung überhaupt was?
- Initialisierung der Container
- RNG auf Embedded Devices tricky
- SmartCard als Vertrauensanker
- Wie macht man die Passwort-Abfrage zur Entschlüsselung und wann?

SmartCards unter Android

↪ tarent

- Zwei Probleme: Hardware und Software
- Kein CardReader
  - USB On-The-Go via USB-Port? -> Schwierig
  - Bluetooth wie z.B. Blackberry BT SC-Reader (Treiberstand unbekannt)
  - MicroSD Karte der Fa. CertGate
    - Eigenschaften: Eigener RNG, eigene API, kein Quellcode des IFD-Handlers
- Keine PKCS#11 API in Android (wie z.B. in der Sun JVM-API)
- Auch native Komponenten benötigt (z.B. SmartCard Middleware)

Artefakte - Zusammenfassung

↪ tarent

- SmartCard Middleware: pcsc-lite/pcscd (Ungetestet)
  - Jemand hier, der eine Idee hat?
- GnuPG (Version 1)
  - Random Number Generator?
  - libgcrypt, libgpg-error (GnuPG 2)
- eCryptFS-utils (libecryptfs, mount.ecryptfs)
  - Keyutils
  - Libgcrypt, libgpg-error
- Loop-AES
  - Losetup und linux-utils mount
- OpenVPN
  - Busybox ifconfig

Add: Wie kann man bei Android mitmachen?

↪ tarent

- Code aus dem öffentlichen Repository
- Man kann alles im Emulator austesten -> kein Entwicklergerät nötig
- Eigenes Review Portal
- Gerrit: https://review.source.android.com/
- Contributor License Grant unterzeichnen
- Loslegen und Patches zur Review vorlegen
- Project Lead stimmt ab ;-)

Zusammenfassung

- Android hat ein gut durchdachtes Sicherheitsmodell
- Berechtigungssystem ist eine Kompromiss zwischen Benutzerfreundlichkeit und Sicherheitsmaßnahmen
- Erweiterungen bedingt möglich
  - Benötigt Entwicklertelefon
  - Binäre Einschlüsse die Regel
  - Hoher Portierungsaufwand für Abhängigkeiten von Software und Subset der Standard-C-Bibliothek
- Hoffentlich kommen bald mehr offene Geräte :-)

Fragen ...