Communitygetriebene Android Systemerweiterungen

Marc Seeger
Computer Science and Media
Hochschule der Medien Stuttgart
e-mail: mail@marc-seeger.de

17. Januar 2010

1. Einleitung

Die Android Plattform hat seit der Veröffentlichung der ersten auf Android basierten Geräte eine große Entwicklercommunity um sich gebildet. Diese Community erstellt nicht nur eine Fülle an Anwendungen, sondern erweitert auch die Plattform und das darunter liegende Betriebssystem selbst. Die Quelloffenheit der Plattform ermöglicht es Entwicklern mit tiefgreifenden Änderungen zu experimentieren und diese einer breiten Öffentlichkeit zur Verfügung zu stellen. Durch den Linux Kern der Android Plattform wird es auch Entwicklern ohne Erfahrungen im Embedded-System Umfeld ermöglicht, einen einfachen Einstieg in die Systementwicklung für Mobiltelefone zu finden.
Die Systemerweiterungen ermöglichen risikobereiten Endnutzern das Hinzufügen von Features, welche im Auslieferungszustand des Mobilgerätes nicht vorgesehen sind. Diese Features erstrecken sich von der Unterstützung neuer Audioformate bis zur Erhöhung des CPU Takts. Die von den Entwicklern bereitgestellten Modifikationen lassen sich mit unterschiedlichem Aufwand auf die Endgeräte einspielen. Android ermöglicht somit als erste Mobilplattform eine Grundlegende Änderung des vom Hersteller vorgegebenen Betriebssystems. Eine interessante Fragestellung in diesem Bereich sind auch die rechtlichen Hintergründe bei der Verbreitung und Nutzung der Modifikationen.
Dieses Paper erläutert zuerst die Voraussetzungen für die Modifikationen. Danach folgt eine Definition der unterschiedlichen Bereiche in welche die Modifikationen gegliedert werden können. Jeder Bereich wird hierbei durch entsprechende Beispiele verdeutlicht. Schlussendlich wird die lizenzrechtliche Situation zur Verbreitung der Modifikationen betrachtet.

2. Installation der Modifikationen

Das Verändern und Hinzufügen von Systembestandteilen erfordert sowohl unter Linux im Desktop-Umfeld, als auch auf der Android Plattform entsprechende Privilegien. Diese Privilegien lassen sich unter Android auf zwei Arten erreichen. Entweder man hat die Möglichkeit Programme mit Rechten des root-Benutzers zu starten um im laufenden System Modifikationen vornehmen zu können, oder man besitzt Zugriff auf den Boot-Prozess und kann somit direkt ein entsprechend erweitertes Betriebssystem einspielen.
Im Normalfall unterliegt der Boot-Prozess dem Hardware-Hersteller und der Zugriff wird durch diesen entsprechend eingeschränkt. Da Android-Endgeräte jedoch während ihrer Lebenszeit einige offizielle Softwareupdates erfahren, besteht in allen Fällen zumindest die Möglichkeit entsprechend kryptografisch signierte Updates einzuspielen.
Am Beispiel des HTC Dream können die typischen Sicherheitsvorkehrungen von Android Mobiltelefonen gut beobachtet werden. Dies erlaubt auch die Erläuterung der Schritte, welche von der Entwicklercommunity unternommen wurden, um diese Vorkehrungen auszuhebeln.

2.1. Root-Zugriff

Der Root-Zugriff auf dem Mobiltelefon hat die selbe Bedeutung wie auf auf Desktop Rechnern. Er ermöglicht die Ausführung von Programmen als privilegierter Systemuser. Diese Privilegien erlauben unter anderem den unbeschränkten Zugriff auf Hardware und Dateisystem des Systems.
Um die in diesem Paper vorgestellten Erweiterungen nutzen zu können, ist es größtenteils notwendig Zugriff auf diese Systembereiche zu haben. Es

existieren heutzutage viele Android Distributionen der Community, welche einen Root-Zugriff von Haus aus bereitstellen. Es ist jedoch nicht möglich diese auf einem Telefon im Auslieferungszustand einfach einzuspielen. Die Recovery Partition welche für den Flashprozess zuständig ist, überprüft die kryptografische Signatur des zu flashenden Images. Das System weigert sich daher eine Datei einzuspielen welche nicht von Google beziehungsweise dem Netzbetreiber kryptografisch signiert wurde.
Glücklicherweise ist es möglich die Recovery Partition aus dem laufenden System heraus zu ersetzen, falls es dem Benutzer gelingt Root-Rechte zu erlangen. Der erste Angriffspunkt ist also das erreichen von Root-Rechten um dadurch entweder direkt Erweiterungen einzuspielen oder die Recovery Partition zu ersetzen um ein Flashen mit unsignierten Images zu ermöglichen.
Der zweite Angriffspunkt ist der Bootloader des Gerätes. Der sogenannte "Engineering Bootloader", welcher sich auf Entwicklergeräten befindet, erlaubt Entwicklern einen Systemzugriff via USB schon während der Bootloader-Phase des Systemstarts. Mit dem Entwicklertool *Fastboot* lassen sich so in Kombination mit dem Engineering Bootloader beliebige Partitionen mit unsignierten Images über USB flashen. In Endbenutzer-Geräten ist die USB Kommunikation mit dem Bootloader deaktiviert. Gelingt es einem Benutzer im laufenden System Root-Rechte zu erlangen, kann der aktuelle Bootloader durch sein Engineering Pendant ersetzt werden und erlaubt somit beliebig erweiterte System-Images einzuspielen.
Die nächsten 2 Abschnitte erläutern wie die Umgehung der Signaturmechanismen in der Praxis umgesetzt wurde.

2.1.1 Die erste Lösung

Als das HTC Dream als erstes Android-Endgerät auf den Markt kam, besaß die installierte Android Version eine Schwachstelle welche unglaubwürdiger nicht sein konnte. Das Gerät interpretierte Texteingaben in jedem beliebigen Teil des Systems als Kommando. Besonders Kritisch wurde dieser Fehler dadurch, dass jegliche Eingabe auch noch mit Root-Rechten ausgeführt wurde.
Der Grund für dieses Verhalten lag in einer Kombination vieler Konfigurationseinstellungen. Ein Teil davon ist die init.rc Datei, welche beim Systemstart ausgeführt wird und folgende Zeile enthält:

```
service console /system/bin/sh
```

Da der init Prozess als Standard Input (stdin) das /dev/console Device gesetzt hat, wird nach diesem Befehl jegliche Eingabe an /dev/console mit Root-Privilegien ausführt.
Dies ist im Normalfall ungefährlich, da der Benutzer keinen direkten Zugriff auf /dev/console besitzt. Zu dieser Zeit wurde dem System beim booten jedoch ein "console="-Parameter übergeben welcher /dev/console auf mehrere Devices lauschen lässt. Im Normalfall sollte /dev/console nur auf den seriellen Port lauschen um ein Debugging (z.B. durch den Kundendienst) zu ermöglichen. Im Falle des G1 lauschte /dev/console jedoch auch auf das virtuelle Konsolen Device "tty0". Die von Linux bekannten virtuellen Konsolen sind bei einem System wie Android eigentlich nicht nötig, da der Benutzer nur über eine grafische Oberfläche (auf Framebuffer Basis) mit seinem System interagiert und keine Konsolen als Eingabemöglichkeit benötigt. Der Kernel des HTC Dream wurde jedoch mit gesetztem "CONFIG_VT" Parameter kompiliert und bot deshalb Unterstützung für die virtuellen Konsolen. Alle diese einzelnen Probleme laufen zusammen wenn der Benutzer eine Taste auf der Hardwaretastatur des Gerätes drückt. Der Tastaturtreiber leitet jeglichen Tastendruck auf vorhandene virtuelle Konsolen weiter. Dies bedeutet dass tty0 jegliche Eingabe erhält. Von dort gelangt es durch den gesetzten Boot Parameter an /dev/console, welches die eingehenden Kommandos mit Root-Rechten ausführt.
Um von außen Root-Zugriff auf das Gerät zu erlangen genügt somit schon die Eingabe von "telnetd" und das drücken der Enter Taste. Dadurch wird der im System mitgelieferte Telnet Daemon gestartet. Mit laufendem Telnet Daemon kann dann über das Netzwerk auf das Gerät zugegriffen werden. Dieser Zugriff kann dazu benutzt werden den Bootloader zu überschrieben. Das führt dazu, dass auch unsignierte Systemimages eingespielt werden können.
Da signierte Images mit dieser unglücklichen Kombination von Fehlern in Umlauf sind können Besitzer des G1 ein Downgrade durchführen falls die Sicherheitslücke in späteren Versionen geschlossen wird. In diesem Fall gibt es für HTC und die Netzbetreiber keine einfache Lösung mehr diese Lücke zu schließen.

2.1.2 Die aktuelle Situation

Neuere Handsets als das G1 wurden mit Versionen der Android Plattform ausgeliefert auf denen dieser Fehler behoben wurde. Da für diese Telefone keine kompatiblen "fehlerhaften" Images im Umlauf sind, muss der Root-Zugriff auf eine andere Art hergestellt werden.
Das Tool "FlashRec"[1] nutzt dazu mehrere Exploits im Linux Kernel. Durch diese Exploits gelingt es einen Flashvorgang des Recovery Images mit den erforderlichen Root-Privilegien zu starten.
Für Firmware-Images welche vor August 2009 veröffentlicht wurden wird von FlashRec eine Sicherheitslücke genutzt welche unter dem Namen *ADV-2009-2272* in der National Vulnerability Database veröffentlicht[2] wurde. Dieser Exploit nutzt eine Schwachstelle des Kernels bei Socket Operationen um beliebigen Code mit Root-Privilegien ausführen zu können. Die genaueren Interna des Exploits würden den Umfang des Papers sprengen, können jedoch auf der Seite der National Vulnerability Database nachgelesen werden. Das Realisieren des Exploits über Socket Verbindungen wird auch beim Installieren des Tools erkennbar. Der Benutzer muss dem Tool Zugriff auf das Bluetooth Subsystem genehmigen. Über das Bluetooth Subsystem kann vom Programm der Socket erstellt werden welcher den Exploit ermöglicht.
Für Firmware-Images die zwischen August 2009 und November 2009 veröffentlicht wurden benutzt das Tool die Kernel Sicherheitslücke CVE-2009-3547[3]. Diese Lücke erlaubt Angreifern durch race-conditions im pipe Systemcall die Ausführung von beliebigem Code mit Root-Privilegien. In unserem Fall leitet der Code das Flashen des Recovery Images ein.

3. Gruppierung der Modifikationen

Die von unabhängigen Entwicklern in die Android Plattform integrierten Erweiterungen wurden lassen sich grob in drei Bereiche gliedern:

- Kernel
- System
- Applikationen

Der Kernelbereich befasst sich mit Erweiterungen die aus der offiziellen Linux Kernelentwicklung stammen. Zum Zeitpunkt ihrer Veröffentlichung sind diese Erweiterungen im Normalfall jedoch auf keinem offiziellen Android Endgerät im Einsatz. Hier wäre als Beispiel die Integration des "Brainfuck Scheduler" in den Android Kernel zu nennen.
Der Systembereich besteht aus Erweiterungen bestehender Systemkomponenten z.B. durch Plugins zur Wiedergabe von bisher nicht unterstützten Medienformaten oder der Implementation von Funktionalität welche die aktuelle Hardware zwar bieten könnte, jedoch in der offiziellen Entwicklungsversion von Android so nicht zur Verfügung steht. Ein Beispiel wäre die Implementation des Bluetooth Object Exchange Protokolls (OBEX) welches den Dateitransfer zwischen Endgeräten über Bluetooth ermöglicht.
Der Applikationsbereich konzentriert sich auf die Erweiterung vorhandener Applikationen oder die Portierung von Applikationen neuerer Endgeräte auf ältere Android-Versionen. Hier wäre z.B. die Integration von Multitouch in den Android Webbrowser zu nennen.
Im Folgenden werden die einzelnen Bereiche über Beispielprojekte verdeutlicht.

4. Erweiterungen auf Kernel Ebene

Die Erweiterungen auf der Kernel Ebene dienen zum Großteil der Erhöhung der Performanz des Gesamtsystems. Einige von ihnen legen jedoch gleichzeitig auch den Grundstein für andere Modifikationen im System- oder Anwendungsbereich.

4.1. BFS - Der "Brainfuck Scheduler"

In heutigen Betriebssystemen kämpfen zu jedem Zeitpunkt mehrere Prozesse darum, ihren Programmcode auf dem Prozessor ausführen zu dürfen. Die grundlegende Instanz des Betriebssystems, welche sich mit dieser Ressourceneinteilung beschäftigt wird "Scheduler" genannt. Insbesondere bei den begrenzten Ressourcen von Mobilgeräten muss er für eine Balance aus Hintergrundprozessen (z.B. die Aktualisierung der Location-Daten über Google Latitude) und einem flüssigen Ablauf der Benutzeroberfläche sorgen.
Ursprünglich konnte man im Android Quellcode, wie auch im regulären Linux Kernel den Completely Fair Scheduler (kurz: CFS) in der Datei sched_fair.c vorfinden. Dieser Scheduler ist im Gegensatz zu seinen Vorgängern (z.B. dem O(1)-Scheduler) weitgehend frei von komplexen Heuristiken und Statistiken. Es findet keine Priorisierung von

einzelnen Tasks statt, stattdessen werden alle Prozesse gleich behandelt. Der Prozess welcher schon am längsten auf CPU Zeit warten musste wird als nächster aktiviert. Ergänzend muss noch angemerkt werden, dass es in neueren Kernel Versionen die Möglichkeit gibt Prozesse zu "Task Groups" zu bündeln. Die Auswahl erfolgt dann primär auf Gruppenebene um zu verhindern dass einzelne Benutzer über eine große Anzahl an Prozessen das gesamte System blockieren.
Mitte 2009 entwickelte Con Kolivas jedoch eine Alternative zum CFS Scheduler da dieser, in seinen Augen, für interaktionsbasierte Endanwendersysteme ungeeignet sei. Ein Zitat aus seiner BFS-FAQ[4] zum "hakeligen" Verhalten des CFS auf seiner neuen Hardware lautete wie folgt:

> It's not a profound effect in CFS and that's admirable. It just doesn't behave the way I feel the scheduler should being forward looking only (not calculating sleep) and it doesn't really make the most of a relatively lightly loaded machine without many many cpus. So I threw it all out and wrote exactly the opposite.

Um die Gegenläufigkeit zur aktuellen Schedulern zu verdeutlichen nannte er seine Kreation BFS, den "Brainfuck Scheduler" und veröffentlichte die bereits erwähnte FAQ Datei[4], welche seine Motivation und die Grundidee verdeutlichte.
Er charakterisierte das Verhalten des Schedulers darin mit folgenden Worten:

> It was designed to be forward looking only, make the most of lower spec machines, and not scale to massive hardware. ie [*sic*] it is a desktop orientated scheduler, with extremely low latencies for excellent interactivity by design [...]

Durch die Optimierung auf Interaktion und leistungsschwache Systeme ergibt sich eine interessante Einsatzmöglichkeit in Mobiltelefonen. Dies erkannte auch Steve Kondik, der Community besser als "Cyanogen" bekannt. Er ist Autor der wohl beliebtesten Android-Distribution "Cyanogenmod". Er war auch der Erste der BFS auf Android portierte und ein lauffähiges Image veröffentlichte, welches die Endnutzer mit recht wenig Aufwand auf ihre Telefone einspielen konnten. Das Userfeedback attestierte dem neuen Scheduler eine subjektiv flüssigere Bedienbarkeit des Gesamtsystems. Genauere Details des Schedulers würden den Rahmen des Papers sprengen, können jedoch bei Bedarf in Con Kolivas Dokument zum BFS Design[5] nachgelesen werden.
Ob es in direktem Zusammenhang steht ist nicht bekannt, jedoch gab es einige Wochen später einen commit[6] welcher BFS dem offiziellen Android Sourcecode hinzufügte.

4.2. Das Compcache Modul

Eines der größten Probleme im täglichen Einsatz von mobilen Endgeräten ist der begrenzte Arbeitsspeicher. Besonders in Systemen (wie z.B. Android) in denen der Endnutzer mehrere Programme parallel laufen lassen kann wird Arbeitsspeicher schnell zur knappen Ressource. Die normale Herangehensweise auf Desktop Systemen ist, Teile der Festplatte quasi als "Überlauf" zu nutzen und aktuell nicht genutzte Programmteile vom RAM auf die Festplatte zu verschieben. Mobile Endgeräte besitzen jedoch Flashspeicher der in der Anzahl von Schreibzugriffen begrenzt ist und über die Zeit degradiert. Neben diesem "wear-leveling" Problem verlangsamt eine Auslagerung auf den langsameren Flashspeicher das gesamte System und sollte vermieden werden.
Hier ist der Einsatzort des Compcache Moduls[10]. Anstatt den "Überlauf" des Arbeitsspeichers auf die Festplatte bzw. den Flashspeicher zu verlagern, nimmt das Modul einen Teil des Arbeitsspeichers dafür. Dieser Teil wird als Blockdevice mit dem Namen "ramzswap" zur Verfügung gestellt. Der Vorteil dabei ist, dass die Daten welche auf das Blockdevice geschrieben werden, transparent komprimiert werden. Dies vergrößert die Menge an Daten die (gleichzeitig) im schnellen RAM gehalten werden können und ist zugleich bei Weitem schneller als der Zugriff auf einen entsprechenden internen Flashspeicher des Mobilgerätes.
Die Community selber ist sich jedoch noch unschlüssig ob der zusätzliche Arbeitsspeicher das Gesamtsystem wirklich beschleunigt oder ob der zusätzliche Overhead sich negativ auf die Gesamtperformance für ein normales Benutzungsschema auswirkt.

4.3. Erhöhung des CPU Taktes

Das Übertakten von Computern ist in der PC Community seit Jahren schon ein beliebtes Mittel um gekaufter Hardware mehr Leistung zu entlocken. Die Geschwindigkeit einer CPU wird durch zwei Merkmale bestimmt:

- den Bustakt
- den Multiplikator

Multipliziert man diese 2 Angaben erhält man die Nenngeschwindigkeit des Prozessors(z.B. 12 * 133.33 MHz = 1600 MHz).
Während in normalen Heimcomputern die Prozessoren immer mit der vom Prozessorhersteller vorgegebenen maximalen Geschwindigkeit getaktet werden, sieht die Situation für Mobilgeräte anders aus. Um potentielle Überhitzung oder zu großen Stromverbrauch zu vermeiden werden Prozessoren künstlich gedrosselt.
Ein Beispiel dafür ist das HTC Dream, auch bekannt als "Google Dev Phone 1" oder "T-Mobile G1". Es besitzt einen 528 MHz Qualcomm MSM7201A ARM11 Prozessor, welcher jedoch im Auslieferungszustand des Gerätes auf 384 MHz begrenzt ist.
Die Begrenzung ist jedoch rein softwareseitig und kann, entsprechende root-Rechte vorausgesetzt, aufgehoben werden.
Programme wie setcpu[7] erlauben die Auswahl des CPU Governor und die Anpassung der minimalen und maximalen Frequenzen.
Durch die erhöhte Leistung entsteht auch ein erhöhter Stromverbrauch, jedoch nur zu den Zeitpunkten in denen die CPU wirklich rechnen muss. Bei normaler Interaktion mit dem Gerät wird somit zwar mehr Strom verbraucht, jedoch nur über kürzere Intervalle. Somit kann das Gerät flüssiger bedient werden. Insbesondere Spiele, die im Normalfall eine andauernde Auslastung der CPU garantieren werden dem übertakteten System wohl mehr Energie entlocken als einem Äquivalenten System mit Begrenzung des CPU Takts.

5. Erweiterungen auf System Ebene

Die Erweiterungen auf Systemebene verschaffen dem Enduser neue Features welche Unzulänglichkeiten der vom Hersteller zur Verfügung gestellten Firmware beheben sollen.

5.1. Integration des OBEX Protokolls

Die Integration des OBEX Protokolls[11] ist Beispielhaft für eine Erweiterung, bei der die Hardware des Endgerätes entsprechende Möglichkeiten bietet, jedoch von Softwareseite eine Einschränkung herrscht. Im Falle von OBEX war die Hardware das im HTC Dream verbaute Bluetooth Modul. Im Bluetooth Standard sind neben den Grundlegenden Eigenschaften wie z.B. Frequenzen und Sicherheitsmodelle auch sogenannte "Profile" festgelegt. Diese Profile stehen für eine entsprechend standardisierte Funktionalität wie z.B. das Hands-Free Profile (kurz: HFP), welches jegliche Details bei der Kommunikation mit Auto-Freisprechanlagen festlegt.
Für den Endanwender ist das OBject EXchange Protokoll (kurz OBEX) interessant. Es dient als Grundlage für viele andere Profile wie z.B. Object Push Profile oder Generic Object Exchange Profile. Diese Profile ermöglichen es z.B. Visitenkarten elektronisch zwischen Endgeräten zu übertragen oder Fotos an ein anderes Bluetooth-Gerät (z.B. den eigenen Laptop) zu senden.
Das HTC Dream konnte im Auslieferungszustand zwar Bluetooth-Freisprecheinrichtung benutzen, jedoch existierte keinerlei Unterstützung für das OBEX Protokoll um Dateitransfers über Bluetooth abzuwickeln.
Dies ist insbesondere deswegen verwunderlich, weil bereits einige Open-Source Implementationen des OBEX Protokolls für Linux existieren. Diese Implementationen bildeten die Grundlage für einige erfolgreiche Kommandozeilen Experimente. Die funktionierenden Tools stehen unter dem Namen "android-obex"[8] als Sourcecode zur Verfügung.
Eine entsprechende grafische Endbenutzer-Applikation stand kurze Zeit später im Android Market zur Verfügung. Die einzige Einschränkung zur Nutzung dieser Applikationen ist, dass man Root-Zugriff auf sein Gerät benötigt um die entsprechenden Kommandos an das Bluetooth Subsystem abgeben zu können.
Als Beispiel für eine der ersten grafische Applikationen welche Dateitransfers über OBEX vornehmen konnten sei "Bluex" genannt. Der Preis der Applikation fällt mit 2 USD recht gering aus.

5.2. Unterstützung von FLAC

Eine weitere Systemerweiterung die sich großer Beliebtheit erfreut ist die Unterstützung des FLAC Formats für das Multimedia-Subsystem.
FLAC ist der de-facto Standard für verlustfrei komprimierte digitale Audiodateien. Es ist ein offenes Format und findet als Archivformat bei audiophilen Menschen großen Anklang. Es eignet sich auf Grund seiner geringen CPU Anforderungen beim Decodieren für vergleichsweise langsame CPUs wie sie in Smartphones und anderen mobilen Systemen vorkommen.

Das Schöne an dieser Erweiterung ist, dass sie auf das OpenCORE Media Framework aufsetzt. Dieses Framework bildet die Grundlage für die erweiterbare Multimedia-Unterstützung in Android. Der Android Quellcode bindet OpenCORE mit den Formaten aac, gsm_amr, mp3 und sbc ein[9]
Bereits im Dezember 2008 wurde im offiziellen Android Bugtracker unter der Ticketnummer 1461[13] der Wunsch nach Unterstützung des FLAC Formates deutlich. Leider hat es ein Jahr später noch keine offizielle Änderung des Tickets gegeben. Es existieren Ende Dezember 2009 jedoch schon über 100 Kommentare die diesen Wunsch bekräftigen.
Zum 31. Juli 2009 wurden jedoch von Kenny Root 3 Patches eingereicht, welche den FLAC Support in Android nachrüsten. Um Patches zum offiziellen Android Repository hinzuzufügen muss ein Review Prozess angestoßen werden. Dieser Prozess besteht aus drei Schritten. Der erste Schritt ist ein Upload der Patches in das "Gerrit" genannte Code Review System unter `http://review.source.android.com`. Schritt zwei ist eine Zustimmung durch berechtigten Personen. Jedes Projekt im Android Source Tree hat eine Liste von Leuten die einer Änderung zustimmen können. Je nach Projekt müssen ein oder mehrere berechtigte Personen der Änderung zustimmen. Sollte dies geschehen sein, sind die Patches bei Schritt drei angekommen und werden in den offiziellen Android Source Tree gemerged.
Von den FLAC Patches befinden sich zwei aktuell im Zustand "Review in Progress", der dritte Patch wurde schon in das offizielle Repository gemerged.
Um dem langsamen offiziellen Prozess zu entgehen, wurde das Addon schon vor längerer Zeit in die "Cyanogen" Android Distribution übernommen und kann damit schon auf aktuellen Geräten genutzt werden.
Die offizielle Webseite [12] des Projektes macht nur auf eine Limitierung aufmerksam: Es wird nur FLAC mit einer Sampletiefe von 16 Bit unterstützt. Dies sollte jedoch kein großes Problem darstellen, da FLAC Dateien zum Großteil von Audio CDs erzeugt werden, welche selber nur 16 Bit Sampletiefe besitzen.

5.3. Apps2SD

Die Möglichkeit der Auslagerung von Programmen auf die Speicherkarte ("Apps2SD") ist ein sehr beliebtes Thema in der Android Community.
Das HTC Dream war das erste Android Smartphone welches von einem Mobilfunkanbieter verkauft wurde. Dieses von T-Mobile "G1" getaufte Endgerät besitzt einen internen Speicher von 256 Megabyte. Um mehr als eine Hand voll Musik, Bilder oder Videos auf dem Gerät vorhalten zu können, besitzt es wie die meisten Smartphones eine Möglichkeit den internen Speicher durch den Einsatz einer Speicherkarte (meist im Micro SD Format) zu erweitern. Leider hilft diese Erweiterung dem Endkunden nicht, wenn ihm durch große Applikationen der interne Speicher knapp wird. Gizmodo(.com) berichtete, dass als Grund für diese Praxis die Angst vor Raubkopien genannt wurde. Es wird laut Aussage von einem Google Mitarbeiter jedoch für kommende Android Versionen an einer Lösung gearbeitet, die Applikationen verschlüsselt auf der SD Karte abzulegen.
Der aktuelle Stand ist jedoch, dass es offiziell nicht möglich ist. Da Android ein Linux basiertes System ist, wird die SD Karte jedoch intern nur als weiterer Ordner in das Dateisystem "gemounted". Dem laufenden System steht die Karte unter dem Pfad */sdcard* zur Verfügung.
Da die interne Systempartition mit dem Dateisystem YAFFS2 formatiert ist, beherrscht sie das erstellen von "symbolischen Links". Diese stellen eine Indirektion dar um einen Verweis auf einen anderen Ordner bereitzustellen. Will man nun ein Programm auf die SD Karte auslagern, kann man einfach den Ordner des Programmes vom internen Flash auf die SD Karte verschieben und einen entsprechenden symbolischen Link zurücklassen. Da Programme nur über die üblichen Dateisystem APIs auf Dateien zugreifen und diese APIs mit symbolischen Links umzugehen wissen, ändert sich für die Anwenderprogramme selber nichts.
Die erste Vorbereitung welche der Benutzer für diesen Prozess selbst treffen werden muss ist, auf der SD Karte eine Ext3 bzw Ext2 formatierte Partition einzurichten. Leider unterstützt das normalerweise für SD Karten vom System eingesetzte FAT32/vfat die Posix Access Control Listen zur Rechteverwaltung von Dateien nicht und ist damit nicht mit dieser Modifikation kompatibel.
Die zweite Voraussetzung ist ein Kernel der mit Unterstützung für das Ext Dateisystem kompiliert wurde. Im Auslieferungszustand bietet Android nur Unterstützung für das YAFFS und FAT32 (aka vfat) Dateisystem. Da jedoch bei Android ein normaler Linux Kernel zum Einsatz kommt, ist die EXT Unterstützung nur eine Konfigurationseinstellung vor dem Kompilieren und benötigt keinen weiteren Aufwand.
Vorteile dieser Erweiterung sind der dadurch quasi unbeschränkte Platz für Programme und die

Möglichkeit nach einer Systemwiederherstellung die vorher installierten Programme mit einem Kommando zurück zu bekommen.
Diese Modifikation hat jedoch nicht nur Vorteile. Sie nimmt dem Benutzer die Möglichkeit die SD Karte im laufenden Betrieb entfernen zu können. Dadurch dass im System ein plötzliches fehlen von Programmen nicht vorgesehen ist, könnten unvorhersagbare Probleme auftreten. Des Weiteren ist die Setup Prozedur recht komplex und sollte mit Vorsicht durchgeführt werden. Schlussendlich ist es, je nach SD Karte, auch möglich dass eine Verlangsamung der Programme durch den permanenten SD Karten Zugriff auftritt. Die Automatisierung des kompletten Prozesses ist heute jedoch bei vielen Android Distributionen (z.B. der "Cyanogen" Distribution) gegeben bzw. über ein erweitertes Recovery Image realisierbar.
Um den USB-Modus des Mobiltelefons weiterhin auch auf Systemen nutzen zu können welche keine Unterstützung für das EXT Dateisystem bieten ist es ratsam eine seperate FAT32 formatierte Partition auf der SD Karte zu erstellen um weiterhin einen einfachen Zugriff auf Bilder und Musikdateien zu ermöglichen.

5.4. Funktionsumfang des "Recovery Image"

Die Erhöhung des Funktionsumfangs des "Recovery Image" ist quasi ein Muss für den Hobby-Entwickler. Die bisher auf dem Markt befindlichen Android-basierten Endgeräte haben einen mehrstufigen Systemstart.
Zu Beginn der Startsequenz wird überprüft ob gewisse Tastenkombinationen gedrückt sind. Eine dieser Kombinationen (beim G1 z.B. Home-Key und die rote Hörertaste) führt dazu, dass das Mobiltelefon nicht das volle Android Betriebssystem startet sondern das System die "Recovery" Partition bootet. Diese Partition beinhaltet ein (Linux) Minimalsystem welches nur zum Update/der Wiederherstellung der eigentlichen Systempartition gedacht ist.
Erweiterungen dessen Funktionsumfanges beginnen beim Einspielen eines neuen Kernels mit einer größeren Dateisystemunterstützung (z.B. wichtig für "Apps2SD").
Eine der attraktivsten Erweiterungen, genannt "Nandroid", verschafft dem Benutzer die Möglichkeit, den gesamten NAND Flash des Systems auf die SD Karte zu sichern und bei Bedarf auch wieder zurückzuspielen. Dies ermöglicht das Erstellen von kompletten System Backups, was insbesondere für Entwickler von großer Bedeutung ist.
Teil der erweiterten Recovery Images sind auch Tools welche z.B. die Partitionierung von SD-Karten ermöglichen. Die im vorherigen Kapitel besprochene "Apps2SD" Erweiterung kann somit vollständig über die neuen Bordmittel durchgeführt werden.
Die beliebteste Distribution des Recovery Image ist das sogenannte "Pimped out recovery image with new features"[14] welches von Steve 'Cyanogen" Kondik erweitert wurde. Die Installation kann aus dem laufenden System erfolgen und beschränkt sich auf die Eingabe eines simplem Kommandos in einem Terminal-Emulator:

```
flash_image recovery
/sdcard/cm-recovery-1.4.img
```

5.4.1 APN Listen

Die Konfiguration der Datenverbindung eines Mobilgerätes ist sicherlich eine der komplexeren Aufgaben mit denen sich ein Smartphone-Besitzer konfrontiert sehen kann.
Im Normalfall sind auf Mobilgeräten welche durch einen Mobilfunkprovider verkauft werden diese Daten automatisch hinterlegt. Das Hauptaugenmerk liegt hierbei auf dem Access Point Name (kurz: APN). Beim Aufbau einer Datenverbindung wird der im Telefon hinterlegte APN mit an das Netz übergeben. Das Netz ermittelt daraufhin über eine DNS Anfrage an welchen Gateway die Anfrage weitergeleitet werden muss. Dieses Gateway leitet die Anfragen des Nutzers aus dem Netz des Mobilfunkproviders in das Globalen Internet weiter. Sollte ein falscher APN an das Netz übergeben werden, findet keine Weiterleitung der Pakete ins Internet statt.
Die Mobilgeräte verfügen im Auslieferungszustand über eine Liste der APNs von Vertragspartnern. Kleinere Provider oder lokale Provider bei Importgeräten bleiben in desen Listen jedoch oft außen vor. Die Zusammenarbeit der Community beim Sammeln von entsprechenden Netz und APN Kombinationen führt zu umfangreichen APN Listen welche eine manuelle Einrichtung meist überflüssig macht.

6. Erweiterungen auf Applikations Ebene

Auf Applikationsebene sind in diesem Umfeld selten komplette Neuentwicklungen zu finden. Es sind meist kleine Patches bzw Konfigurationsänderungen an vorhandenen Anwendungen.

6.1. Better Browser

Der "Better Browser"[15] ist eine Erweiterung des Standardwebbrowsers um u.a. Multitouch-Gesten. Da der Webkit-basierende Android Browser unter Open-Source Lizenz steht, konnten zusätzliche Features mit vergleichbar geringem Aufwand implementiert werden.
Die zusätzlichen Features welche im originalen Browser nicht vorhanden sind erstrecken sich von einfachen kosmetischen Änderungen bis hin zu neuer Funktionalität. Einer der größten Unterschiede ist der Support für Multitouch. Die vom iPhone üblichen "pinch" Gesten um Ausschnitte von Webseiten zu vergrößern werden dadurch bei entsprechendem Kernel (mit Multitouch Support) unterstützt. Kosmetische Änderungen wie das Deaktivieren der Zoom Buttons gehen Hand in Hand mit neuen Gesten wie z.B. die Double-tap Geste um die Zoomstufe zu erhöhen oder die Triple-tap Geste um die Zoomstufe zu verringern. Beachtlich ist hierbei dass die "Triple-Tap" Geste im Android SDK nicht vorgesehen ist sondern von Hand implementiert wurde.
Die letzte größere Erweiterung des Browsers ist das "Mobile Web Settings" Feature. Es ermöglicht die Übertragung eines beliebigen HTTP User Agents. Dieser von vielen Seiten ausgewertet um zu entscheiden ob eine für Mobilgeräte optimierte Version der Seite ausgeliefert wird. Durch den geänderten User Agent wird es dem Android Gerät ermöglicht eine z.B. auf iPhones spezialisierte Ansicht bestimmter Webseiten zu erhalten falls der Webseitenbetreiber die Android Plattform bisher noch nicht als Mobilgerät erkennt.

6.2. Backports

Die Community hat auch eine Reihe von "Backports" hervorgebracht. Dies sind Softwarekomponenten welche auf neueren Handymodellen erschienen sind und von Entwicklern auf älteren Hardwaremodellen lauffähig gemacht werden.

Da Google dem Android Open-Source Projekt in regelmäßigen Abständen neue Releases des Systems beisteuert bieten die offiziellen Android Sourcen oft zusätzliche Features welche in älteren Geräte nicht existieren.
Dies konnte insbesondere beim Erscheinen der Android Releases mit den Codenamen "Donut" bzw "Eclaire" und deren konsequenter Portierung auf das HTC Dream beobachtet werden. Hierbei wurden z.B. Features wie PPTP/L2TP VPN und WPA Enterprise Support hinzugeügt, welche zu dieser Zeit nur im experimentellen "Donut" Release zu finden waren.
Ein weiteres Beispiel sind oft einzelne Widgets welche in späteren Handymodellen beziehungsweise Systemversionen aufgetaucht sind. Widgets sind kleine Programme welche direkt auf dem Homescreen des Geräte laufen und nicht explizit gestartet werden.
Hierzu sind z.B. das Power Widget (zusammen mit dem "App Fuel Gauge" Systemfeature) oder das HTC Twitter-Widget zu nennen. Diese Widgets erlauben eine einfache Aktivierung von Services (Bluetooth, GPS, ...) respektive die Interaktion mit dem Micro-Blogging Dienst Twitter.
Insbesondere bei den Widgets von HTC stellt sich jedoch oft die Frage nach der Legalität der weiteren Verbreitung der Widgets, da diese nicht unter offenen Lizenzen stehen.

7. Lizenzfrage

Bei der Verbreitung von "flash-fertig" Image Dateien über Entwicklerforen stellt sich natürlich noch die Frage der Lizenzen der Einzelkomponenten. Während Großteile des Systems unter einer freien Lizenz stehen, sind einige Anwendungen (z.B. Google Maps) proprietär und deren Verbreitung ohne Zustimmung des Rechteinhabers ist umstritten. Hier ist es interessant zu sehen, welche Systembestandteile der Android Smartphones unter welche Lizenz fallen.
Die im offiziellen Android Repository (`http://android.git.kernel.org/`) bereitgestellten Komponenten befinden sich alle unter der Apache v2 Open Source Lizenz.
Einige der Hauptkomponenten (neben den üblichen Linux Systembibliotheken) sind:

- Das OpenCORE Media Framework
- Die Tesseract Open Source OCR Engine
- Der radio interface layer ("ril")
- Andwendungen wie z.B. Alarm Clock, Browser, Calendar, Calculator, Camera, Contacts, Email, IM, Music, Voice Dialer, Sound Recorder

Während diese Komponenten ausreichen um ein Lauffähiges System herzustellen, gibt es jedoch noch einige Bestandteile welche sich auf aktuellen Endgeräten finden, jedoch nicht unter einer freien Lizenz verfügbar sind:

- Google Maps
- YouTube
- Google Voice
- Google Talk (ein Instant Messaging Client für Googles XMPP basierende Chat Platform)
- Google Android Market
- Google Search
- Gmail (ein spezialisierter eMail client für Googles Mailservice)
- Der Google Calendar Provider (um eine synchronisation zu Googles Online-Kalenderdienst bereitzustellen)

Diese Software ist Teil der sogenannten "Google Experience" und wird von Google an die Hersteller der Endgeräte bzw. an Provider verkauft.
Interessant ist hierbei die anfängliche Argumentation von Steve Kondik (besser bekannt unter seinem Pseudonym"cyanogen"). Er verbreitete die "Google Experience" Anwendungen als Teil seiner Images. Darin sah er jedoch kein Problem, da die Images nur auf "Google Experience" Geräten lauffähig waren. Der Einsatz der Anwendungen erfolgte also stets von Leuten die durch den Kauf des Telefons dazu berechtigt waren.
Während diese Argumentation intuitiv recht nachvollziehbar ist, hat sich Steve Kondik darauf nicht verlassen und vertreibt seine Android Distribution fortan ohne die entsprechenden Applikationen. Dies bedeutet jedoch nicht dass die Benutzer der Distribution darauf verzichten müssen. Da die offiziellen Firmware Images von HTC (dem Telefon-Hersteller) die Anwendungen enthalten, wird die Installation der Distribution quasi als Update über ein vorher geflashtes offizielles Image durchgeführt.
Somit verbreitet er selbst nur unbedenkliche Komponenten auf seinen Images und erhält die "Google Experience" aus den offiziellen Images.
Eindeutig ist die Lage bei Widgets oder anderer zusätzlicher Software, welche auf neueren Geräten erscheint und nicht unter einer freien Lizenz steht. Hier wird durch die Verbreitung als Backport oder als Teil eines flashbaren Images klar das Copyright missachtet.

8. Fazit

Die Beliebtheit der einzelnen Modifikationen spricht für sich. Es gab in der Geschichte der Mobiltelefone noch keine anderen Zeitpunkt an dem eine größere Anzahl Endbenutzer tiefgreifenden Änderungen an ihrem Mobiltelefon durchgeführt haben.
Insgesamt kann man sagen dass sich die Bereitschaft von Endbenutzern stark erhöht hat, ihr Endgerät eigenständig zu modifizieren. Benutzer des Apple iPhone konnten sich über einen "Jailbreak" schon lange vor dem offiziellen Marktstart des Appstore zusätzliche Applikationen installieren. Benutzer des Palm Pre wird es durch "Preware" ermöglicht ihr Gerät mit zusätzlichen Programmen und Erweiterungen zu versehen.
Durch das Aufkommen von unix- bzw linuxbasierten Endgeräten wird es nun auch Programmierern ermöglicht Erweiterungen zu schreiben, welche vorher keinerlei Erfahrung mit Embedded Systems hatten.
Trotz der relativ großen Verbreitung der Modifikationen gibt es bisher keine Hersteller die versucht haben rechtliche Schritte gegen eben jene Softwareprojekte zu unternehmen. Palm CEO Josh Rubenstein ging auf der Consumer Electronics Show 2010 sogar soweit, die Ergebnisse der Entwickler Community lobend in seiner Keynote zu erwähnen.
Einzig bei der Verbreitung von bestehendem intellektuellem Eigentum ohne Zustimmung des Herstellers wurde eine Unterlassungsanordnung von Google gegen den Entwickler der beliebten Cyanogen Distribution ausgesprochen. Jedoch erfolgte auch hier die Kommunikation sehr verständnisvoll und hatte keine finanzielle Schädigung des Entwicklers zur Folge.
Für die Zukunft scheint sich eine weitere Popularisierung der Modifikationen durch Endbenutzer abzuzeichnen. Ob diese Zukunft für Hardwarehersteller ähnliche Symptome zeigen wird, wie sie auch schon die Content Anbieter beim Wechsel des Web 1.0 zum Web 2.0 mitgemacht haben, wird sich zeigen.

Literatur

[1] "Tool to flash the recovery image on Android phones":
http://zenthought.org/content/project/flashrec

[2] Vulnerability CVE-2009-2692:
http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-2692

[3] Vulnerability CVE-2009-3547:
http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-3547

[4] Con Kolias, "FAQS about BFS":
http://ck.kolivas.org/patches/bfs/bfs-faq.txt

[5] Con Kolias, Design Übersicht für BFS:
http://ck.kolivas.org/patches/bfs/sched-BFS.txt

[6] Commit des BFS Schedulers in den offiziellen Android Sourcecode:
http://android.git.kernel.org/?p=kernel/experimental.git;a=commit;h=c8fa3555b59a200f9a5b1098f17c0333b9529258

[7] SetCPU: Tool zum Anpassen der Taktfrequenz auf dem HTC Dream: http://www.pokedev.com/setcpu/

[8] android obex - provide obex file sharing in Android:
http://gitorious.org/android-obex

[9] Audio Support im Git Repository des OpenCORE Frameworks:
http://android.git.kernel.org/?p=platform/external/opencore.git;a=tree;f=codecs_v2/audio

[10] Compcache Module, compressed in-memory swap device for Linux:
http://code.google.com/p/compcache/

[11] Erin Yueh,"how to have OBEX function in Android?":
http://i-miss-erin.blogspot.com/2009/10/how-to-have-obex-function-in-android.html

[12] Kenny Root, FLAC on Android:
http://the-b.org/FLAC_on_Android

[13] Bugtracker-Ticket zur Erweiterung des Android Sourcecode um FLAC:
http://code.google.com/p/android/issues/detail?id=1461

[14] Steve "Cyanogen" Kondik: Pimped out recovery image with new features:
http://forum.xda-developers.com/showthread.php?p=3915123

[15] Loccy: Better Browser:
http://forum.xda-developers.com/showthread.php?t=551119