Learning from Android

Parallelen der Sicherheit von mobilen und automotiven Systemen

Dipl.-Inf. **F. Schaub**, Dipl.-Inf. **B. Könings**, Prof. Dr. **M. Weber**
Institut für Medieninformatik, Universität Ulm

Kurzfassung

Durch die steigende Verbreitung von Smartphones erhöhen sich auch die Anforderungen an die Sicherheitsmechanismen mobiler Plattformen. Am Beispiel von Android, eines der zur Zeit populärsten Betriebssysteme für mobile Geräte, werden Sicherheitsmechanismen sowie existierende Sicherheitsprobleme diskutiert und auf den automobilen Kontext übertragen. Aus dieser Übertragung werden schließlich Erkenntnisse zur Risikominimierung bei der Entwicklung von Sicherheitsmechanismen für automobile Anwendungsplattformen abgeleitet.

1. Einleitung

Smartphones erfreuen sich steigender Beliebtheit. Gerade die einfache Erweiterbarkeit durch Drittanbieteranwendungen über App Markets ist ein wichtiges Kaufkriterium, da Kunden ihre bevorzugten Dienste und Services auch mobil nutzen wollen. App Markets bieten aber auch eine zusätzliche Einnahmequelle für Plattformanbieter und Gerätehersteller durch Beteiligung an Einnahmen von Anwendungsverkäufen über diese Kanäle. Auch im Automobil-Bereich zeichnet sich Interesse an Anwendungen im Fahrzeug ab. Neben Navigationssystemen werden immer mehr, zunehmend auch externe, Dienste in die Headunit bzw. das In-Vehicle Infotainment (IVI)-System integriert. Die Audi connect Dienste [1] integrieren beispielsweise lokale Suche, Satellitenbilder, aktuelles Wetter und Nachrichten in das IVI-System. Andere Fahrzeughersteller nutzen Terminal Mode [2], um Anwendungen und Daten von Smartphones über die Headunit nutzbar zu machen. Ford SYNC bietet neben dem Zugriff auf Smartphone-Daten auch Zugang zu externen Diensten wie den iTunes Store oder das Erstellen von „Vehicle Health Reports“ [3]. Die GENIVI Alliance will mit MeeGo-IVI ein eigenständiges, hersteller-unabhängiges Ökosystem für IVI-Anwendungen schaffen [4].

Generell besteht der Vorteil von integrierten Anwendungen darin, dass sie im Gegensatz zu Smartphone-Anwendungen Fahrzeug-eigene Sensorik und Systeminformationen nutzen können und nahtlos in die Headunit integrierbar sind. Wenn zunehmend Drittanbieteranwendungen in die Headunit und IVI-Systeme integriert werden, z.B. auch über App Markets

für automobile Anwendungen, wird die technische Absicherung von Fahrzeuganwendungsplattformen eine wichtige Rolle spielen, um den Safety-Anforderungen der Fahrzeugbranche gerecht zu werden. Hinzu kommen neue Herausforderungen durch den Umgang mit Personenbezogenen Daten und der weitreichenderen Öffnung der Headunit für Kommunikation mit externen Geräten und Diensten.

Ähnlichen Herausforderungen im Bereich Sicherheit und Privatsphäre musste sich auch der Smartphone-Markt stellen. Daher liegt eine Untersuchung von Parallelen zwischen den beiden Domänen und die Prüfung der Übertragbarkeit von Konzepten und Risiken nahe. Android, als eine sehr erfolgreiche, gleichwohl noch relativ junge, herstellerübergreifende Plattform soll hierfür als Fallstudie dienen. Im Folgenden werden Konzepte für Sicherheit und Schutz der Privatsphäre auf mobilen Plattformen anhand von Android skizziert, bestehende Probleme und Herausforderungen diskutiert und Parallelen zu automobilen Anwendungsplattformen gezogen. Daraus lassen sich eine Reihe von Erkenntnissen und Empfehlungen für die Realisierung von sicheren automobilen Anwendungsplattformen ableiten.

2. Android Systemarchitektur und Ökosystem

Die Android-Architektur lässt sich in vier Schichten aufteilen [5]. Für eine Sicherheitsbetrachtung ist es darüberhinaus sinnvoll auch das umliegende Ökosystem mit zu betrachten, wie in Bild 1 veranschaulicht.

Hardware. Die Hardware-Plattform umfasst die physische Ausprägung eines Mobilgeräts und seine Leistungsmerkmale. Hierzu zählen auch verbaute Kommunikationstechnologien und Sensorik. Die Hardware-Plattform dient typischerweise als Differenzierungsmerkmal zwischen Herstellern und ist somit Hersteller-abhängig.

Betriebssystem. Als unterliegendes Betriebssystem kommt bei Android Linux zum Einsatz. Neben dem Linux Kernel sind hier Treiber für die unterliegende Hardware und allgemeine Software-Bibliotheken enthalten. Ebenfalls auf dieser Ebene angesiedelt sind die Dalvik Virtual Machine, die Ausführungsumgebung von Android, sowie Smartphone-spezifische Bibliotheken, z.B. für Telefonie und Multitouch-Eingabe. Unter Umständen nehmen Hersteller auf dieser Ebene Anpassungen für ihre Geräte vor.

Anwendungsframework. Das Anwendungsframework ist der Kern von Android. Hier werden Android-spezifische Dienste und Schnittstellen zur Anwendungsentwicklung geboten und die graphische Benutzerschnittstelle (GUI) definiert. Beispiele sind Softwarepaket-Management, Ressourcen-Management, Multitasking-Management, Lokalisierungsdienste, Notification-Management sowie Fenster- bzw. Activitiy-Management. Speziell GUI-Komponenten und -Funktionalität werden oft von Herstellern angepasst,

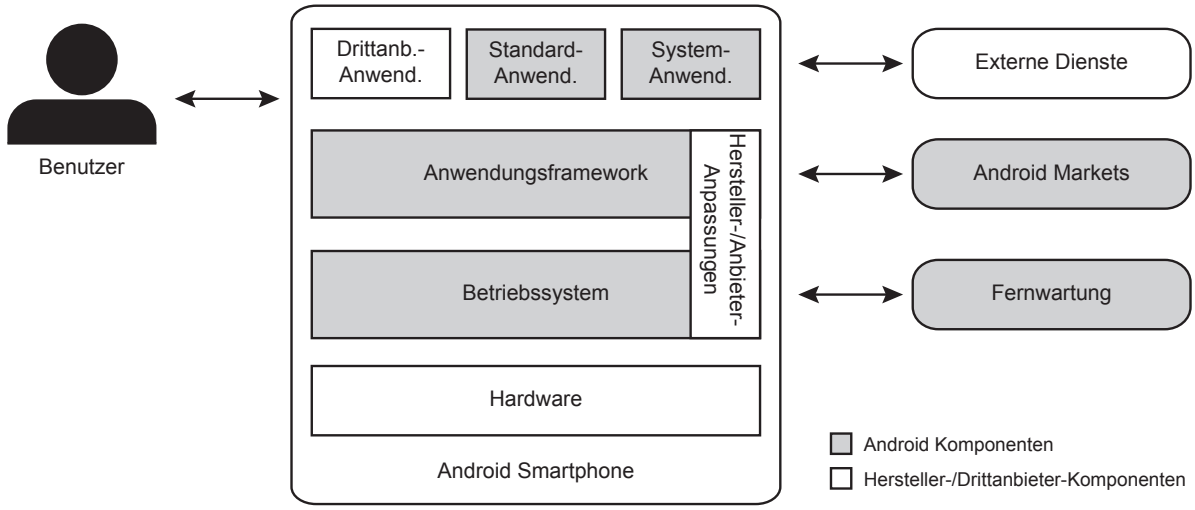

Bild 1: Überblick Android Systemarchitektur und Ökosystem.

um ein markenspezifisches Look&Feel zu erzielen. HTC Sense und Samsung TouchWiz sind Beispiele für Hersteller-eigene Android GUIs. Neben Geräteherstellern nehmen auch Mobilfunkanbieter Anpassungen vor.

Anwendungen. Im Hinblick auf Sicherheitsaspekte kann man bei Android verschiedene Anwendungsarten unterscheiden. System-Anwendungen (z.B. Einstellungen oder Telefon) sind auf dem Gerät vorinstalliert und lassen sich nicht entfernen, sie werden bei Systemupdates aktualisiert. Standard-Anwendungen (z.B. Kalender oder Kontakte) sind ebenfalls vorinstalliert und nicht entfernbar, bieten aber für den System-Betrieb keine essentiell notwendige Funktionalität. Darüber hinaus gibt es Drittanbieter-Anwendungen, die vom Benutzer beliebig installiert werden können, entweder über einen Android Market oder von Drittanbieterseiten.

Android Markets. Über Android App Markets können Benutzer bequem Anwendungen suchen und installieren. Updates für installierte Anwendungen werden ebenfalls über diese Markets verteilt. Android ist nicht an einen einzelnen Market (z.B. Googles) gebunden, vielmehr können Kunden Anwendungen über unterschiedliche Markets beziehen, z.B. von Mobilfunkbetreibern, Geräteherstellern oder anderen Firmen. Markets können entweder über eine eigene Android-Anwendung oder eine Webseite zugegriffen werden.

Externe Dienste. Viele Anwendungen sind direkt mit Internetdiensten verknüpft und tauschen mit diesen Informationen aus. Weiterhin besteht bei kostenlosen Anwendungen häufig eine Verbindung zu Werbediensten, um die Anwendungen zu finanzieren. Die

Standard-Anwendungen von Android sind sehr eng mit den entsprechenden Google-Diensten integriert, z.B. Google Calendar, Mail, Contacts oder Picasa.

Fernwartung. Viele sicherheitskritische Wartungsfunktionen laufen bei Android über die Luftschnittstelle. Neue Android-Versionen und Updates werden über das Mobilfunknetz *Over-The-Air* (OTA) an Geräte verteilt. Weiterhin verfügt Google über die Möglichkeit Anwendungen aus der Ferne von Kundengeräten zu entfernen.

3. Sicherheitskonzepte von Android

Die Sicherheitskonzepte von Android sind auf die verschiedenen Architekturschichten verteilt und lassen sich in drei Kategorien unterteilen [6], die im Folgenden vorgestellt werden.

Linux-Mechanismen

Zu den Linux-Mechanismen gehören die Verwaltung von Benutzer-IDs sowie der darauf basierte Zugriff auf Prozesse und Dateien. Jeder Android-Anwendung (apk-Paket) wird bei der Installation eine eindeutige Benutzer-ID zugewiesen, mit der dann der dazugehörige Anwendungsprozess gestartet wird. Dies führt dazu, dass eine Anwendung nicht auf andere Anwendungsprozesse oder deren Code-Bestandteile zugreifen kann.

Neben dieser einfachen Art des „Sandboxings“ bietet Linux mit der Zugriffskontrolle auf das Dateisystem ein weiteres wichtiges Sicherheitskonzept. Gruppen und Benutzern zugewiesene Lese-, Schreib- und Ausführungsrechte werden bei Zugriffsversuchen durch den Linux-Kernel kontrolliert. Bei Android sind System-Dateien entweder dem Benutzer „system“ oder „root“ zugeordnet. Anwendungsbezogene Dateien sind der Benutzer-ID der jeweiligen Anwendung zugeordnet, sodass eine Anwendung nur auf eigene Daten und nicht auf System-Dateien oder Dateien anderer Anwendungen zugreifen kann.

Mechanismen der Ausführungsumgebung

Die Ausführungsumgebung von Android enthält ebenfalls verschiedene Sicherheitskonzepte. Die *Memory Management Unit* (MMU) ist beispielsweise eine Hardware-Komponente, die kontrolliert, dass Prozesse nicht auf den Speicherbereich anderer Prozesse zugreifen können. Dadurch sollen *Privilege Escalation* Angriffe, also das Ausführen von Code mit den Rechten eines anderen Prozesses, erschwert werden.

Der Einsatz von Java als Programmiersprache bei Android garantiert die Typ-Sicherheit von Variablen und schützt vor *Buffer-Overflow*-Angriffen. Die Typ-Sicherheit von Binder, dem Android-spezifischen Mechanismus zur Inter-Prozess-Kommunikation, wird durch die Verwendung der *Android Interface Definition Language* (AIDL) während der Code-Übersetzung

garantiert. Allerdings besteht für Android-Entwickler auch die Möglichkeit native C-Bibliotheken zu integrieren, auf die über das *Java Native Interface* (JNI) zugegriffen werden kann. Dies bietet auf der einen Seite zwar mehr Freiheiten für Entwickler, stellt auf der anderen Seite aber auch ein erhöhtes Sicherheitsrisiko dar.

Android-spezifische Mechanismen

Das Android-Anwendungsframework verfügt über drei wichtige Sicherheitsmechanismen: *Anwendungs-Rechte*, *Komponenten-Abschirmung* und *Anwendungs-Signierung*.

Die Rechte (*Permissions*) einer Anwendung beschränken den Zugriff auf geschützte Funktionen der Android-API zur Laufzeit. Die benötigten Rechte werden durch den Anwendungs-Entwickler in einer *Manifest*-Datei spezifiziert. Bei der Installation einer Anwendung müssen diese Rechte explizit angefordert und durch den Benutzer bestätigt werden. Android spezifiziert über 100 verschiedene Rechte, die z.B. den Zugriff auf Funktionen für Anrufe, Lesen von Kontaktdaten oder Internet-Zugang regeln.

Durch die beschriebene Verwendung verschiedener Benutzer-IDs pro Anwendung werden Anwendungs-Komponenten effektiv von anderen Anwendungen abgeschirmt. Ein Entwickler kann aber eine Komponente explizit für andere Anwendungen freigeben. Eine Möglichkeit ist die explizite Verwendung derselben Benutzer-ID (*sharedUserID*) in verschiedenen Anwendungen. Dies ist allerdings nur möglich, wenn beide Anwendungen vom gleichen Entwickler signiert wurden. Eine andere Möglichkeit ist die Angabe der *exported*-Eigenschaft im Manifest, welche die Freigabe von Komponenten an beliebige Anwendungen ermöglicht.

Für die Aufnahme in einen App-Market muss eine Anwendung signiert sein. Die Anwendungssignatur dient der Bestätigung der Integrität und Herkunft einer Anwendung. Die Verifikation der Anwendungs-Herkunft ist insbesondere wichtig, um ein Vertrauensbezug zwischen mehreren Anwendungen desselben Entwicklers herzustellen. Die zur Signierung nötigen Zertifikate können, zumindest für Googles Android-Market, durch Entwickler selbst erstellt werden und unterliegen somit keiner Überprüfung durch Zertifizierungsstellen.

Ein weiteres Sicherheitskonzept ist das Einspielen von System-Updates. Android ermöglicht dies auf Betriebssystem-Ebene durch OTA-Updates. Sobald ein Update zur Verfügung steht, wird der Benutzer automatisch darüber informiert und muss lediglich die Installation bestätigen. Die Verbreitung dieser Updates geschieht durch unterschiedliche Instanzen (Google, Gerätehersteller oder Mobilfunkanbieter), je nach dem wie das Mobilgerät bezogen wurde.

Neben kompletten System-Updates, können bei Android bestimmte Sicherheitsprobleme durch Remote-Eingriffe behoben werden. Jedes Android-Gerät besitzt dazu eine persistente

GoogleTalk-Datenverbindung [7], über die Google aus der Ferne bösartige Anwendungen entfernen, sowie bestimmte Konfigurations-Parameter anpassen, kann.

4. Sicherheitsprobleme bei Android

Trotz der aufgeführten Sicherheitsmechanismen existieren dennoch einige Probleme, die die Sicherheit von Android beeinträchtigen können. Potentielle Angriffe können sich gegen das Betriebssystem, das Anwendungsframework oder einzelne Anwendungen richten.

Sicherheitsprobleme des Betriebssystems

Angriffe auf der System-Ebene verfolgen meist das Ziel Root-Rechte und somit vollen Zugriff auf das System zu erlangen (auch *JailBreak* oder *Rooting* genannt). Dabei können Schwachstellen in Geräte-Treibern, System-Bibliotheken oder im Linux-Kernel ausgenutzt werden. Schwachstellen dieser Art sind nicht immer auf Android beschränkt, sondern können teilweise auch bei anderen Linux-Distributionen ausgenutzt werden. Sobald Sicherheitslücken in einer Kernel-Version bekannt werden, erscheint jedoch meist in kürzester Zeit ein Kernel-Update, das die Lücke schließt.

Um eine System-Schwachstelle auf einem Android-Gerät zu beheben, muss ein komplettes System-Update (OTA) durchgeführt werden. Dies kann aber je nach Vertreiber unterschiedlich lang auf sich warten lassen. Während Google meist zeitnahe Updates für eigene Geräte (z.B. Nexus One) bereitstellt, kann sich bei anderen Geräteherstellern die Bereitstellung durch weitere Anpassungen um teilweise mehrere Monate verzögern. Wurden Geräte „gebrandet“, d.h. durch einen Mobilfunkanbieter ebenfalls angepasst, kann sich die Bereitstellung nochmals verzögern. Teilweise werden Updates für ältere Geräte auch überhaupt nicht mehr bereitgestellt. Bis Juli 2011 wurden mehr als 80% der im Umlauf befindlichen Geräte wurden noch nicht auf Android-Version 2.3.3 oder höher aktualisiert [8]. 59% der Geräte verwenden noch Version 2.2 mit Linux-Kernel 2.6.32, für den seit über einem Jahr Sicherheitsupdates existieren die auf diesen Geräten aber noch nicht verfügbar sind. Die Problematik der Update-Verzögerung erhöht das Sicherheitsrisiko also erheblich.

Sicherheitsprobleme des Anwendungsframeworks

Schwachstellen des Anwendungsframeworks können von Anwendungen ausgenutzt werden, um Rechte-Beschränkungen zu umgehen. Zum einen scheitert Androids Rechte-Management im Umgang mit transitiven Rechten [9]. Eine Anwendung mit geringen Rechten könnte also auf Komponenten einer Anwendung mit höheren Rechten zugreifen und somit indirekt Funktionen unautorisiert aufrufen.

Das Rechte-Management-Konzept weist außerdem die Nachteile einer zu geringen Rechte-Granularität und fehlender Selektierbarkeit auf. Anwendungen, die beispielsweise Werbung einblenden wollen, benötigen die Berechtigung INTERNET. Diese Berechtigung erlaubt der Anwendung dann aber vollen Internet-Zugriff, auch wenn der Zugriff auf einen einzelnen Werbeserver ausreichen würde [10]. Des Weiteren hat ein Benutzer während der Installation einer Anwendung nur die Möglichkeit allen geforderten Rechten zuzustimmen oder die Installation abzubrechen. Zustimmen bzw. Ablehnen einzelner Rechte ist nicht möglich. Da Anwendungen nicht angeben müssen wozu sie bestimmte Rechte benötigen, kann ein Benutzer die Legitimität einzelner Berechtigungen auch nur schwer einschätzen und muss letztlich dem Entwickler vertrauen. Durch die fehlende Identitäts-Verifizierung selbst-signierter Entwickler-Zertifikate fehlt aber die technische Basis für dieses Vertrauen.
Die Art der Anwendungs-Installation stellt eine weitere Schwachstelle dar. Bei einer Installations-Anfrage sendet der Market-Server über die persistente GoogleTalk-Verbindung eine Installations-Anweisung an das Gerät, das daraufhin die entsprechende Anwendung herunterlädt und installiert. Kann ein Angreifer diese Anweisung fälschen, so kann er beliebige Anwendungen auf Fremdgeräten installieren. Dass derartige Angriffe möglich sind, wurde kürzlich mit Hilfe einer XSS-Lücke in Googles Web Market demonstriert [11].

Sicherheitsprobleme von Anwendungen

Auf Anwendungsebene existieren verschiedene Sicherheitsprobleme in Form von Malware, durch persistente Datenspeicherung sowie die Kommunikation mit externen Diensten.
Android bietet durch die Offenheit bei der Anwendungsinstallation zahlreiche Möglichkeiten für Malware. Ein Angreifer kann harmlos wirkende Anwendungen in einem Android Market einstellen und warten bis ein Benutzer diese installiert. Um benötigte Berechtigungen zu erlangen, kann ein Angreifer entweder hoffen, dass ein Benutzer diese bei der Installation ignoriert, oder durch geschickte Wahl der Anwendungsfeatures die Berechtigungen legitimieren. So können beispielsweise SMS, E-Mails, oder Kontaktdaten ausspioniert werden. Mit der Möglichkeit nativen Code in eine Anwendung zu integrieren, kann Malware durch das Ausnutzen von System-Schwachstellen potentiell auch Root-Rechte erlangen. Die erste derartige Malware mit dem Namen *DroidDream* wurde Anfang 2011 entdeckt [12].
Die Kontrolle von neu eingestellten Anwendungen in Googles Android Market scheint sehr gering [13]. Eventuell existierende Sicherheits-Kriterien des Kontrollprozesses sind weder für Entwickler noch für Benutzer transparent nachvollziehbar. Somit ist es für Angreifer relativ leicht Malware zu verbreiten. Einige OEMs, wie z.B. Archos, verweigern allerdings ihren Ge-

räten den Zugriff auf Googles Android Market und betreiben stattdessen einen eigenen Market mit strikteren Kontrollen, aber meist auch deutlich weniger Anwendungen.
Wurde eine Malware in Googles Android Market entdeckt, verschickt Google über die GoogleTalk-Verbindung eine Deinstallations-Anweisung an alle Geräte. Die betroffene Anwendung wird daraufhin automatisch entfernt. Der Benutzer hat auf diesen Vorgang keinen Einfluss und wird lediglich von einer Statusmeldung über die Deinstallation informiert. Falls die Malware allerdings schon das System infiziert und eventuell weitere Schadsoftware nachinstalliert hat, bleibt dieser Eingriff wirkungslos.
Die persistente Datenspeicherung durch Anwendungen ist ebenfalls ein Sicherheitsproblem, insbesondere wenn ein Gerät in falsche Hände gelangt. In entsprechenden Datenbanken sind zahlreiche unverschlüsselte Informationen enthalten, wie z.B. die Browser-Historie, gespeicherte Passwörter, oder Authentifizierungstokens für Google-eigene Dienste. Zudem speichern Anwendungen häufig ihre Daten auf der externen SD-Karte. Dies ist insbesondere bei sensiblen Daten ein Sicherheitsrisiko, da der Inhalt der gesamten SD-Karte grundsätzlich von jeder Anwendung ausgelesen werden kann.
Ein weiteres Sicherheitsrisiko ist die Kommunikation mit externen Diensten. Findet diese Kommunikation unverschlüsselt statt, können Daten unter Umständen mitgelesen werden, z.B. in öffentlichen WLANs. Tatsächlich kommunizierten selbst Google-eigene Dienste zur Synchronisierung des Kalenders und der Kontakte bis Android-Version 2.2 unverschlüsselt. Ein Angreifer konnte so sensible Daten mitlesen und durch Abfangen der ebenfalls nicht verschlüsselten Authentifizierungstokens Daten manipulieren [14]. Durch eine Anpassung der Anwendungs-Konfiguration über die GoogleTalk-Verbindung wurde diese Lücke bei allen Android-Geräten geschlossen. Über die Anpassung wurden Benutzer allerdings mit keiner Statusmeldung informiert. Die mangelnde Transparenz bei derartigen Eingriffen kann unter Umständen das Nutzervertrauen negativ beeinflussen.

5. Übertragung auf den automobilen Kontext

Um die erwähnten Sicherheitsprobleme und -risiken auf den automobilen Kontext zu übertragen, gilt es zunächst Unterschiede zwischen beiden Domänen zu identifizieren. Als Hauptunterschied dient das Fahrzeug primär dem sicheren Fahren, während Smartphones direkt keine sicherheitskritische Funktion erfüllen. Anwendungen müssen sich daher im Fahrzeug immer Safety-Anforderungen unterordnen. Fahrzeug-Software und -Anwendungen werden momentan typischerweise von OEMs oder Zulieferern entwickelt und vorinstalliert. Dabei kommt häufig ein Echtzeitbetriebssystem eines Zulieferers zum Einsatz, z.B. Microsoft Windows Embedded Automotive oder QNX. Darauf aufsetzende Hersteller-spezifische HMI

bzw. das Headunit-Design dienen als Differenzierungsmerkmale. Zukünftige Fahrzeugsysteme, wie MeeGo-IVI [4] oder das von Continental auf Basis von Android entwickelte AutoLinQ [15], werden es darüber hinaus Benutzern wahrscheinlich ermöglichen Drittanbieteranwendungen über App Markets zu installieren.

Sicherheitsprobleme auf Betriebssystem-Ebene können in Fahrzeugen drastischere Konsequenzen haben als ein *JailBreak* bei Smartphones, da Vollzugriff auf das System unter Umständen Zugriff auf Safety-kritische Bus-Systeme ermöglichen könnte [16]. Auch im Automobilbereich besteht die Gefahr, dass sich Sicherheits-Updates aufgrund von OEM-spezifischen Anpassungen verzögern. Gleichzeitig muss bei Updates sichergestellt sein, dass sie ohne Fehler ablaufen und ein Fahrzeug bzw. dessen Headunit nicht in einen unbenutzbaren Zustand versetzen können.

Im Fahrzeug könnten Anwendungen potentiell Zugriff auf Displays, Lautsprecher, GPS-Position, Sensorik, Fahrzeug-Merkmale, Kommunikationsmedien wie Bluetooth oder WLAN, Internetdienste über Mobilfunknetze, aber auch auf andere Anwendungen, z.B. das Radio oder das Navigationssystem, haben. Das Anwendungsframework muss den Zugriff auf diese Ressourcen regeln, z.B. über Berechtigungen. Allerdings darf der Fahrer während der Fahrt durch Berechtigungsanfragen nicht abgelenkt werden. Durch transitive Rechteübertragung, zu geringe Granularität und fehlende Selektierbarkeit könnten Anwendungen nicht nur Zugriff auf Informationen erlangen, der ihnen eigentlich nicht zusteht, sondern der Fahrer könnte auch negativ in seinem Fahrverhalten beeinträchtigt werden, z.B. wenn eine Anwendung das Headunit-Display hochfrequent blinken lässt oder kontinuierlich Audiomeldungen produziert.

Weiterhin entstehen durch die Öffnung nach außen, z.B. durch App Markets, Internet-Konnektivität sowie über WLAN-, Bluetooth- oder USB-Schnittstellen, neue Angriffsvektoren über die potentiell Malware in das Fahrzeugsystem eingeschleust werden kann.

In den Fahrzeugsystemen werden zunehmend mehr Daten gespeichert, z.B. Telefonbucheinträge, favorisierte und häufige Navigationsziele, und unter Umständen auch Authentisierungsdaten für externe Dienste. Malware könnte derartige Daten sowie Informationen über Art und Zustand des Fahrzeugs, an externe Dienste weiterleiten oder nutzen, um das Fahrverhalten des Fahrers zu analysieren und Bewegungsprofile zu erstellen. Hinzu kommt, dass Fahrzeugsysteme typischerweise nicht auf mehrere Benutzer ausgelegt sind, und somit jeder Fahrer auf alle im Fahrzeug gespeicherten Daten und Dienste zugreifen kann.

Werden Fernwartungs-Kanäle genutzt, um vermeidliche Schadsoftware zu entfernen, Konfigurationen zu ändern oder Updates zu installieren, ohne dass der Benutzer darüber informiert wird, kann daraus ein Vertrauensverlust resultieren. Auch wenn diese Mechanismen

zur Erhöhung der Sicherheit gedacht sind, könnten sie als unerwünschtes Eingreifen in das private System des Benutzers aufgefasst werden.

6. Erkenntnisse für Sicherheitsmechanismen automobiler Anwendungsplattformen

Aus den vorherigen Abschnitten lassen sich nun Erkenntnisse für die Gestaltung von Sicherheitsmechanismen für automobile Anwendungsplattformen ableiten. Diese Erkenntnisse sind allerdings nicht als vollständiger Maßnahmenkatalog zu verstehen, sondern sollen auf Aspekte hinweisen, die über die Grundsicherung des Systems hinausgehen. Denn das Beispiel Android zeigt, dass in einem komplexen Ökosystem auch eine unter Berücksichtigung von Sicherheitsaspekten konzipierte Basisarchitektur Spielraum für Missbrauch bieten kann.

Betriebssystem-Ebene

Eine strikte **Trennung von Anwendungsplattform und Bus-Systemen** findet sich bereits in aktuellen Fahrzeugsystemen. Nichtsdestotrotz bietet die Nutzung von Fahrzeuginformationen und Sensorwerten einen Mehrwert gegenüber Smartphones und kann neue Anwendungen im automobilen Umfeld ermöglichen, z.B. zur Unterstützung von ökologischerem Fahren. Weiterhin lassen sich durch GPS und Internetanbindung ortsbasierte Dienste realisieren. **Definierte Schnittstellen** können abstrakten (Lese-)Zugriff auf Sensorwerte ermöglichen ohne kritische Fahrzeugsysteme zu beinträchtigen.
Weiterhin sollte auf System-Ebene durch **Ressourcenkontrolle** sichergestellt werden, dass Anwendungen voneinander isoliert ausgeführt werden und nicht durch übermäßige Ressourcenbelegung das Gesamtsystem ausbremsen können.

Updates

Bei Android verhindert die Durchmischung von System-Schicht und Anwendungsframework mit Gerätehersteller-Anpassungen oft **zeitnahe Verteilung von Updates**. Wenn quelloffene Systeme wie Android oder MeeGo als Basis dienen, sind zeitnahe Updates aber wichtig, da die Offenlegung der Quellen ein schnelleres Finden und Beheben von Fehlern durch eine größere Community ermöglicht, entsprechende Lücken dann aber auch bekannt sind. Geschlossene Systeme sind im Gegensatz dazu schwerer zu analysieren. Es ist aber bei ihnen nicht auszuschließen, dass existierende Lücken ausgenutzt werden ohne dass dies bekannt wird. Eine klare **Trennung von OEM-Anpassungsschicht und System** könnte derartige Verzögerungen verhindern. Weiterhin sollte eine **Aktualisierung von Teilkomponenten** und Standardanwendungen möglich sein, ohne das Gesamtsystem aktualisieren zu müssen.

Komponenten-basierte Entwicklungsplattformen, wie OSGi, ermöglichen modulare Updates und kommen bereits bei der Entwicklung von Telematiksystemen zum Einsatz [17].
Mechanismen zur **effizienten Verteilung von Updates** sind notwendig. Eine OTA-Schnittstelle wie bei Android wäre auch für Fahrzeugsysteme denkbar, wenn Updates entsprechend durch den Hersteller zertifiziert werden. Eine andere Möglichkeit, die auch heute bereits eingesetzt wird, sind Updates die vom Benutzer heruntergeladen und dann über eine USB-Schnittstelle im Fahrzeug installiert werden. In jedem Fall können Updates nur im Stillstand und nicht während der Fahrt durchgeführt werden.
Weiterhin ist **Transparenz bei Sicherheitslücken** sehr wichtig. Benutzer sollten frühzeitig gewarnt und über Gefahren sowie verfügbare Updates aufgeklärt werden. Wird die transparente Kommunikation vernachlässigt oder werden Sicherheitslücken bewusst verschwiegen, kann dies der Wahrnehmung der Marke erheblich schaden. Insbesondere dann, wenn durch Ausnutzen solch einer Lücke ein Schaden beim Kunden entsteht.

Berechtigungsmodell

Da Anwendungen in Fahrzeugen potentiell auf viele Ressourcen zugreifen können, bietet sich ein deklaratives Berechtigungsmodell ähnlich zu Android an. Allerdings sollten Schwächen des Android-Modells vermieden werden. Zunächst werden Mechanismen zur **Einschränkung der transitiven Nutzung von Berechtigungen** anderer Anwendungen benötigt. Rechtedeklarationen sollten außerdem in ihrer Granularität an die entsprechende Ressource angepasst werden, um effektive Kontrolle zu bieten. Beispielsweise könnte statt einer allgemeinen Berechtigung „INTERNET“, eine **feingranulare und parametrisierbare Berechtigungsdeklaration** die Angabe von gezielten URIs fordern. Die Kommunikation dieser Anwendung könnte dann durch eine Firewall-Komponente auf diese URI beschränkt werden.
Weiterhin könnten Berechtigungen um eine **Zweckangabe** erweitert werden, um Benutzern bei der Installation zu vermitteln wofür die Berechtigung benötigt wird. Da diese Information aber durch den Anwendungsentwickler bereitgestellt wird, ist dies nur im Zusammenhang mit Anwendungsüberprüfungen sinnvoll (siehe nächster Abschnitt).
Bei der Anwendungsinstallation sollte die **Deaktivierung von einzelnen Berechtigungen** möglich sein. Blackberry OS bietet diese Möglichkeit im Gegensatz zu Android [18]. Durch selektierbare Berechtigungen steigen aber auch die Anforderungen an Anwendungen, da sie robust genug sein müssen, um mit reduzierten Berechtigungen umgehen zu können.
Im Gegensatz zu Mobilgeräten können Sicherheitsmechanismen in automobilen Systemen nicht jeder Zeit nach Entscheidungen fragen. Das System muss darauf ausgelegt sein, dass nachdem eine Anwendung installiert wurde (während das Fahrzeug steht) keine weitere In-

teraktion notwendig ist, besonders nicht während der Fahrt. Im Stand sollte aber die Möglichkeit gegeben sein Anwendungsberechtigungen zu prüfen und anzupassen.

Anwendungsüberprüfung

Um die Gefahren von Drittanbieter-Anwendungen einzuschränken sind **strikte Richtlinien zur Anwendungsentwicklung** notwendig, deren Einhaltung beim Einstellen in einen App Market überprüft werden. Derartige Richtlinien sollten prinzipiell festlegen wie sich Anwendungen verhalten dürfen und wie mit persönlichen und Personen-bezogenen Daten umzugehen ist, z.B. Datenverschlüsselung. Darüber hinaus müssen auch Richtlinien zur Verhinderung der Fahrerablenkung vorgegeben werden, um sicherzustellen, dass die Anwendungen für den Einsatz im Fahrzeug geeignet sind.

Für den automobilen Bereich ist also ein restriktiverer Ansatz, wie ihn z.B. Apple mit iOS verfolgt, dem offenen Ansatz von Android vorzuziehen. In so einem *WalledGarden* Modell [13], müssen alle Anwendungen einen **Überprüfungsprozess** durchlaufen bevor sie freigeschaltet werden. Besteht die Anwendung diesen Prozess, wird sie vom Betreiber des App Markets signiert und damit freigeschaltet. Fahrzeugsysteme würden dann nur die Installation von durch den Market-Betreiber signierten Anwendungen zulassen.

Fernwartung

Um bei Sicherheitsproblemen zeitnah reagieren zu können, sollten **Fernwartungsmöglichkeiten** auch bei automobilen Anwendungsplattformen integriert sein. Von derartigen Mechanismen sollte aber nur sparsam Gebrauch gemacht werden, da sie vom Benutzer als Eingriff in seine Privatsphäre aufgefasst werden könnten und somit zu Vertrauensverlust führen können. Bei Eingriffen sollte der Benutzer deshalb immer informiert werden. Werden, wie bei Android, Konfigurationsänderungen teilweise ohne Mitteilung an den Benutzer durchgeführt [14], wissen Benutzer sonst nicht ob ihr Fahrzeug noch verwundbar ist oder nicht. Falls Änderungsanfragen über die Fernwartungsschnittstelle während der Fahrt empfangen werden, sollten diese erst umgesetzt werden wenn das Fahrzeug abgestellt wird [19], da sonst ein Systemausfall während der Fahrt nicht ausgeschlossen werden kann. Alternativ könnte Benutzern auch über eine Webseite die Möglichkeit gegeben werden Änderungen an ihrem Fahrzeug zuzustimmen oder neue Anwendungen zu installieren.

Schutz und Respekt der Privatsphäre

Das Fahrzeug wird von Fahrern teilweise als privater Raum wahrgenommen [20]. Dies gilt es auch bei der Integration von Anwendungen, der Kommunikation mit externen Diensten und

der Verarbeitung von personenbezogenen Daten zu berücksichtigen. Im Sinne der **minimalen Datenhaltung**, sollten Informationen möglichst direkt im Fahrzeug verarbeitet werden. Externe Dienste sollten beispielsweise nicht ohne Einwilligung des Benutzers in der Lage sein die zurückgelegte Fahrstrecke nachzuvollziehen. Generell sollte respektvoll mit Benutzerdaten umgegangen werden, d.h. der Benutzer sollte immer informiert werden ob Daten gesammelt werden und zu welchem Zweck [21]. Persönliche Daten, wie Telefonbücher, die letzten Reiseziele und Account-Passwörter sollten **sicher gespeichert** werden und Zugriff auf diese Daten sollte nur über explizite Berechtigungen gewährt werden.
Die **Authentisierung von individuellen Fahrern**, z.B. durch Schlüssel mit unterschiedlichen IDs, könnte zusätzlich Datenzugriff eingrenzen, auch ohne dass man sich explizit am Fahrzeug anmelden muss. Darüber hinaus ermöglicht die Identifikation einzelner Fahrer eine Personalisierung von Headunit und Anwendungen.

7. Fazit

In diesem Paper haben wir einen Überblick über Sicherheitsmechanismen und -probleme von Android gegeben und diskutiert wie diese sich auf den automobilen Kontext übertragen lassen. Die gesammelten Erkenntnisse können als Ansatz dienen, um Sicherheitsrisiken bei der Entwicklung von Anwendungsplattformen zu adressieren. Einige der gesammelten Punkte werden teilweise bereits in Fahrzeugsystemen umgesetzt. Uns ist jedoch kein System bekannt, das alle Punkte abdeckt. Gerade die Entwicklung von benutzbaren Sicherheitsmechanismen, die nicht zur Ablenkung des Fahrers führen sowie die Schaffung von feingranularen aber handhabbaren Berechtigungsmodellen sind Herausforderungen von großem Forschungsinteresse.

Literatur

[1] Audi AG: Audi connect Dienste. Webseite. 2011 http://www.audi.de/de/brand/de/neuwagen/kommunikation/audi_online_dienste.html

[2] Car Connectivity Consortium: Terminal Mode. Webseite. 2011 http://terminalmode.org/

[3] Ford Motor Company: Ford SYNC. Webseite. 2011 http://www.ford.com/technology/sync/

[4] Smethurst, G.: Changing the In-Vehicle Infotainment Landscape. GENIVI Alliance Whitepaper. 2010

[5] Google Inc.: Android Developer's Guide. 2011 http://developer.android.com/guide/

[6] Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Doley, S. u. Glezer. C.: Google Android – A Comprehensive Security Assessment. IEEE Security & Privacy, 8(2), 2010, S. 35-44

[7] Cannings, R.: Exercising Our Remote Application Removal Feature. Android Blog. 2010 http://android-developers.blogspot.com/2010/06/exercising-our-remote-application.html

[8] Google Inc.: Android Platform Versions (5. Juli 2011). Webseite. 2011 http://developer.android.com/resources/dashboard/platform-versions.html

[9] Davi, L., Dmitrienko, A., Sadeghi, A.-R. u. Winandy, M.: Privilege Escalation Attacks on Android. Information Security, vol. 6531, Springer, 2011, S. 346-360

[10] Barrera, D., Kayacik, H. G., van Oorschot, P. u. Somayaji, A.: A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android. ACM Conference on Computer and Communications Security (CCS). 2010, S. 73-84

[11] Oberheide, J.: How I Almost Won Pwn2Own via XSS. Jon Oberheide Blog. 2011 http://jon.oberheide.org/blog/2011/03/07/how-i-almost-won-pwn2own-via-xss/

[12] Mahaffey, K.: Security Alert – DroidDream Malware Found in Official Android Market. Lookout Blog. 2011 http://blog.mylookout.com/2011/03/security-alert-malware-found-in-official-android-market-droiddream/

[13] Barrera, D., van Oorschrot, P.: Secure Software Installation on Smartphones. IEEE Security&Privacy, 9(3), 2011, S. 42-48

[14] Könings, B., Nickels, J. u. Schaub, F.: Catching AuthTokens in the Wild – The Insecurity of Google's ClientLogin Protocol. Online Report. 2011 http://www.uni-ulm.de/in/mi/mitarbeiter/koenings/catching-authtokens.html

[15] Continental Automotive GmbH: AutoLinQ. Webseite. 2011 http://www.autolinq.de/

[16] Hoppe, T. u. Dittmann, J.: Vortäuschen von Komponentenfunktionalität im Automobil: Safety-und Komfort-Implikationen durch Security-Verletzungen am Beispiel des Airbags. Sicherheit. 2008, S. 341–354

[17] Ai, Y., Sun, Y., Huang, W. u. Qiao, X.: OSGi Based Integrated Service Platform for Automotive Telematics. IEEE Conference on Vehicular Electronics and Safety. 2007

[18] Kirkup, M.: Application Control: An Expert's Guide to Interacting With Your Users. Blackberry DEVCON Asia. 2011

[19] Zhang, D., Wang, X. H. u. Hackbarth, K.: OSGi Based Service Infrastructure for Context Aware Automotive Telematics. IEEE Vehicular Technology Conference (VTC-Spring). 2004, S. 2957-2961

[20] Fraine, G., Smith, S. G., Zinkiewicz, L., Chapman, R. u. Sheehan, M.: At Home on the Road? Can Drivers' Relationships with their Cars be associated with Territoriality? Journal of Environmental Psychology. 27(3). 2007, S. 204-214

[21] European Parliament and the Council of the European Union: Directive on Privacy and Electronic Communications. 2002/58/EC. 2002