TRUST CENTER CONTROLLED CODE ACCESS SECURITY (TC^3AS)

Grischa Schmiedl, Kerstin Blumenstein

Aus Sicherheitsgründen ist der Zugriff auf lokale Ressourcen für Webanwendungen eingeschränkt. Vor allem mobile Websites [1]) können die erweiterten Fähigkeiten mobiler Plattformen nicht nutzen. Das Konzept der Trust Center Controlled Code Access Security (TC^3AS) soll einerseits die sichere Aufhebung von Beschränkungen von Web Applikationen gewährleisten und andererseits dem User eine verständliche und einfache Konfiguration der Sicherheitseinstellungen ermöglichen.

1 Einleitung

Nachdem der Mobilfunkmarkt in Bezug auf Neukundengewinnung mittlerweile als gesättigt gilt, wird das mobile Web als die Hoffnung der Mobilfunkindustrie gehandelt. Web Applikationen haben aus Sicherheitsgründen aber nur eingeschränkten Zugriff auf die Ressourcen sowie die Hardware des lokalen Systems (Sandbox-Prinzip). Die Notwendigkeit dieser Einschränkungen ergibt sich dadurch, dass Webapplikationen nicht am lokalen System installiert werden, sondern nur durch das Aufrufen einer Webseite „gestartet" werden und somit „fremder" und damit potentiell gefährlicher Code auf dem lokalen System ausgeführt wird.

Obwohl Aufweichungen dieses Sicherheitskonzeptes existieren [2]), sind viele Applikationen mit Web Technologien derzeit nicht realisierbar. Dies betrifft sowohl Webanwendungen am gewöhnlichen PC als auch mobile Applikationen, bei denen z.B. der Standort [3]), aber auch die in den Geräten verbaute Hardware (Kamera, Gyrometer etc.) oft einen entscheidenden Mehrwert für Applikationen bringt.

---

[1] Herkömmliche Webanwendungen können auf mobilen Geräten weitgehend problemlos angezeigt werden. Für eine optimale Bedienbarkeit sollten diese zumindest „mobile aware" ausgeführt werden. Eine Alternative ist eine die normale Website ergänzende „mobile optimized" Version für Smartphones, deren Presentation Layer auf die Besonderheiten der mobilen Geräte optimiert ist [2] und idealer Weise die technischen Fähigkeiten der Geräte nutzen sollte (Lage- und Positionssensorik, Kamera etc.).

[2] Aufweichungen des Prinzips existieren bereits im Bereich der RIA-Plugins, z.B. bei Adobe Flash (Zugriff auf Kamera, Mic) und bei Silverlight (Protected local Storage). Für Java Applets kann durch das Signieren des Codes der Zugriff auf das gesamte System gewährt werden. HTML5 bringt ebenfalls einen erweiterten aber dennoch nach wie vor beschränkten Zugriff auf lokale Ressourcen.

[3] Der Zugriff auf die Location-Daten (z.B. GPS-Position) ist per HTML5 möglich, die Methode der Zugriffsbeschränkung durch Nachfragen beim User für jede einzelne Applikation aber nicht zufriedenstellend.

Um diese Einschränkungen zu umgehen, bieten viele Hersteller von Webapplikationen [4]) als Alternative gerätespezifische *Native Applications* für diverse Betriebssysteme im Mobilfunkbereich an, die diesen Beschränkungen nicht unterliegen. Der Download sowie die Installation dieser Anwendungen aus den bei mobilen Plattformen mittlerweilen gängigen *Application Stores* [5]) des Betriebssystem-Anbieters wird dem User extrem einfach gemacht.

Native Applications haben vollen Zugriff auf das System. Ihre Inbetriebnahme ist aber kaum komplizierter als das Aufrufen einer Webseite – und die AnwenderInnen zeigen nur wenig Scheu, Programmen, deren Vertrauenswürdigkeit sie genau so wenig beurteilen können wie die von Web Applikationen, unlimitierten Zugriff auf ihr Gerät zu geben.

2 Hintergrund und Zielsetzung

In diesem Paper beschreiben wir das Konzept einer Methode, mit der Web Applikationen über ein normiertes API Zugriff auf alle Bereiche des Systems bekommen können. Die tatsächlichen Rechte einer Applikationen werden durch ein userfreundliches Rechtemanagement-System festgelegt. Die Problematik der Konfiguration von Sicherheitsrichtlinien besteht aus Usabilitysicht darin, dass die Konfiguration der Sicherheitseinstellungen der BenutzerIn einerseits keine direkt nutzbaren Vorteile bringt (die Applikation funktioniert dadurch nicht „besser“) und andererseits viele BenutzerInnen mit der Aufgabe schon auf Grund mangelnden Verständnisses für die Gefahren überfordert sind. In [3] analysierten Braz und Robert diverse IT-Sicherheitssysteme und mussten sehr oft mangelhafte Usability diagnostizieren. Tatsächlich bricht bereits ein typischer Login-Dialog sechs der acht von Shneiderman vorgeschlagenen „goldenen Regeln“ des User Interface Design [15].

Das Ziel dieses Rechtemanagementsystems ist es, eine für die AnwenderIn rasche und sichere Konfigurationsmöglichkeit anzubieten, mit welcher der Zugriff von Applikationen auf das eigene Gerät gesteuert und überwacht werden kann. Die notwendigen Konfigurationsschritte für die AnwenderIn sollen minimiert werden, ohne die Möglichkeit zur detaillierten Steuerung zu verweigern. Damit soll das System eine Lösung zu dem bereits in [11] beschriebenen Problem bieten, dass AnwenderInnen auf Grund der hohen Komplexität bestehende Sicherheitssysteme nicht richtig konfigurieren können. Dieses Rechtemanagementsystem soll vor allem im Bereich von Webapplikationen zum Einsatz kommen. Es lässt sich aber im Prinzip auch auf Desktop Applikationen, die einem Codesicherheitssystem unterliegen (z.B. Java, .Net), anwenden.

3 Anforderungen an das System

Folgende Anforderungen werden an das System gestellt:

- Das System soll für die BenutzerInnen einfach verständlich und einfach konfigurierbar sein. Es soll verhindert werden, dass die BenutzerIn für jede Applikation den Zugriff auf diverse Funktionen einzeln konfigurieren muss.
- Die Freigabe soll nicht für jede Applikation einzeln getroffen werden müssen. Sie ist mit einem Zertifikatssystem eines globalen oder lokalen (Intranet-Sicherheit) Trust Centers verbunden, das die Rechte vergibt.

[4] U.a. Google, Youtube, eBay sowie diverse Communityportale

[5] Application Stores existieren mittlerweile für alle modernen Smartphones z.B. für das iPhone (App Store), Android-Geräte (Market), Windows Phone 7 (Marketplace), BlackBerry (BlackBerry App World) und für Nokia (Ovi Store).

- Die Rechte sollen für Gruppen gesetzt werden, in denen ähnliche Applikationen zusammengefasst sind. Die Gruppierung soll nicht durch die AnwenderIn selbst erfolgen müssen.
- BenutzerInnen sollen vor Start der Applikationen sehen, welchen Zweck ein Programm verfolgt und welche Zugriffe es benötigt. Das System soll nur genau diese Zugriffe ermöglichen. Es muss gewährleistet sein, dass ein Programm nur den genannten Zweck erfüllt.
- Die Bewertung der Vertrauenswürdigkeit eines Anbieters bzw. einer Applikation soll nicht der BenutzerIn allein überlassen werden. Eine zentrale Instanz (~Trust Center), die Applikationen hinsichtlich ihres Einsatzzwecks zertifiziert, soll die BenutzerIn dabei unterstützen.
- Das Trust Center unterstützt die BenutzerIn bei der Vergabe der notwendigen Rechte, zeigt Gefahrenpotentiale auf und kann nachträglich Applikationen Rechte wieder entziehen.
- Die Rechte, die einer Applikation oder Applikationsgruppe von der BenutzerIn zugestanden werden, können für zentral administrierte Geräte (z.B. Company-Smartphones) eingeschränkt werden (Company-Policy merged with User Policy).
- Der Zugriff auf die geschützten Funktionen erfolgt über ein genormtes Application Programming Interface (API). Web Applikationen müssen anmelden, welche Rechte sie (wofür) benötigen, bevor Sie mit Hilfe dieses API Zugriff auf ein lokales System bekommen. Die BenutzerIn kann diese Berechtigung für diese Session oder generell für diese Applikation bzw. Applikationsgruppe erteilen oder verweigern. Das System soll den User bei einer Automatisierung dieser Rechtevergabe unterstützen.
- Wird eine Berechtigung einmal dauerhaft erteilt, kann der User diese jederzeit einsehen, widerrufen oder ändern. Außerdem wird die Verwendung der Schnittstelle der AnwenderIn generell angezeigt, auch wenn nicht nachgefragt wird.
- Die Zugriffe auf das API durch eine Applikation werden vom API geloggt. Ein Missbrauch kann an das Trust Center berichtet werden, welches das Zertifikat entziehen kann.
- Die Einstellungen einer BenutzerIn können im Trust Center abgelegt und daher auch auf andere Geräte der NutzerIn mitgenommen werden.

4 Trust Center Controlled Code Access Security

Im Folgenden wird das Konzept eines Trust Center-gesteuerten Rechtemanagementsystems für Web Applikationen vorgestellt.

4.1 Bestehende Ansätze zur Codesicherheit

Es gibt diverse Ansätze, die Rechte einer Applikation einzuschränken. Im Bereich der Web Applikationen können die Sicherheitsrichtlinien im Browser entweder sehr grob (z.B. IE-Sicherheitsstufe) oder sehr (zu) detailliert vergeben werden. Die Rechte sind meist für alle Applikationen einer Sicherheitsklasse (Intranet, Internet, Local Zone etc.) identisch. Bestimmte Möglichkeiten können für eine Sicherheitsklasse meist nur komplett an- oder abgeschaltet werden (z.B. JavaScript deaktivieren). In anderen Systemen können manche Rechte (z.B. Zugriff auf das lokale System durch signierte Java-Applets) pro Applikation (identifiziert durch URL) oder pro Hersteller (identifiziert durch den in der Signatur festgelegten Hersteller) eingestellt werden.

Jeweils eigene Ansätze für Code Access Security existieren für Java / Java Applets, das .Net Framework, diverse Browser (Firefox, IE ...) getrennt sowie für diverse Browser Plug-Ins (Flash, Silverlight).

Die Konfiguration der Sicherheitsrichtlinien wird vom User selbst durchgeführt oder kann im Unternehmensbereich durch den Systemadministrator zentral vorgegeben werden. In vielen Fällen wird der User vom System zu einer sicherheitsrelevanten Entscheidung gezwungen, um eine Anwendung verwenden zu können. So muss bei den gängigen Implementationen von HTML5 (z.B. im Safari-Browser des iPhones) der Zugriff auf die Position vom User für jede einzelne Applikation freigegeben werden. EndbenutzerInnen sind mit der Konfiguration der Sicherheitsrichtlinien oft nicht nur überfordert, die Sicherheitsnachfragen werden meist auch als lästig empfunden.

4.2 Der Trust Center Ansatz

Der Begriff Trust Center wird im Bereich der IT-Security für das Schlüsselverwaltungszentrum einer PKI (Public Key Infrastructure) eingesetzt. Zertifikate bzw. Signaturen einer PKI garantieren die Identität einer Person oder Organisation bzw. die Authentizität von Dokumenten, Webseiten, Programmen etc. [5]

Das Konzept der Trust Center Controlled Code Access Security geht über diese Aufgaben hinaus und basiert auf folgenden Ideen:

- Die Applikationen diverser Hersteller werden in vordefinierte Applikationsklassen eingeteilt. Für diese *Purpose Classes* sind die für den Zweck der Software benötigten Rechte allgemein gültig definiert. Die Zuteilung einer Applikation zu einer Purpose Class ergibt sich aus dem „Zweck“ der Applikation.
- Die Purpose Classes für eine Applikation werden zentral zugewiesen. Dies muss nicht von der EndanwenderIn entschieden werden.
- Ein Trust Center im Trust Center Controlled Code Access Security-Konzept kann von unterschiedlichen Institutionen und Organisationen intern oder öffentlich betrieben werden.
- Das Trust Center bestätigt nicht nur die Identität des Herstellers und die Authentizität der Software, sondern legitimiert auch den Zugriff der Software auf das lokale System

4.3 Funktionen des Trust Centers

Das Trust Center vereint zwei Funktionen. Einerseits stellt es die Identität eines Anbieters bzw. dessen Angebotes an Applikationen durch Zertifikate sicher. Andererseits realisiert es eine Art Gütesiegel für zertifizierte Anwendungen, das die Applikation einer bestimmten Purpose Class zuordnet und die Einhaltung der Kriterien dieser Purpose Class garantiert.

Die Zuordnung einer Applikation zu einer Purpose Class erfolgt immer durch den Anbieter der Applikation, der seine Anwendung bei einem Trust Center anmeldet. Die Anmeldung beinhaltet neben der URL der Anwendung und der Identifikation des Anbieters auch ein *Mission Statement* des Anbieters, in dem er den genauen Zweck der Applikation erklärt. Je nach Zertifikatsgüte kann die Evaluation der Zuordnung entweder durch das Trust Center durchgeführt und die Zugehörigkeit damit formal bestätigt werden oder die formale Überprüfung unterbleibt. In beiden Fällen ist die Zuordnung einer Applikation zusätzlich einer durch das Trust Center realisierten *Community-Based Evaluation* unterworfen.

Um sicherzustellen, dass eine Applikation nach Vergabe des Zertifikats nicht verändert wird, können herkömmliche kryptologische Verfahren (Code-Hashes, Code-Signing) angewendet werden. Damit kann allerdings der serverseitige Teil der Anwendung nicht überprüft werden.

Abbildung 1 zeigt die Kommunikation der Komponenten, bestehend aus dem *Extensible Device API* am Client, dem Applikationsanbieter und dem Trust Center.

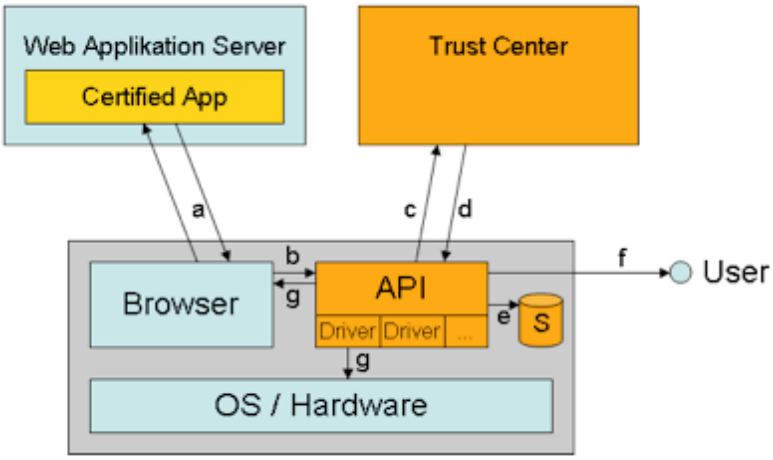

Abbildung 1 - Trust Center Architektur

Die Kommunikation erfolgt in folgender Weise:

a) Die AnwenderIn navigiert auf die Seite des Anbieters einer Applikation, welche das Extensible Device API einsetzt.
b) Um das API aufrufen zu können, muss sich die Applikation bei dem API mit Ihrer Zertifikatskennung anmelden.
c) Das API evaluiert die Gültigkeit des Zertifikats beim Trust Center, das dieses Zertifikat ausgestellt hat. Das Trust Center muss im Browser als vertrauenswürdig eingestuft sein.
d) Das Trust Center bestätigt die Authentizität des Zertifikats und übermittelt dem API:
    - Die URL der Anwendung, anhand der das API erkennen kann, ob die Anwendung vom richtigen Anwender stammt
    - Das Mission Statement des Anbieters zu der Anwendung; der User kann sich daher anhand der Beschreibung des Anbieters vorab über die Applikation informieren
    - Allfällige Userbewertungen aus der Community-Based Evaluation
    - Die Purpose Class für die diese Applikation angemeldet ist, sowie die Rechte, die sich aus der Purpose Class ergeben, wenn diese im API nicht vorhanden sein sollten
    - Die Art des Zertifikats (self-paced oder Trust Center evaluated)
e) Das API überprüft auf Basis der lokalen Einstellungen, wie mit der Applikation weiter verfahren werden soll. Die Entscheidung basiert im Normalfall auf den Usereinstellungen in Bezug auf die Purpose Class und der Art des Zertifikats. Denkbar ist auch den Zertifikaten eines bestimmten Trust Centers oder einem bestimmten Anbieter generell höheres Vertrauen entgegenzubringen. Die Möglichkeiten sind:
    - Zugriff ohne Nachfragen erlauben
    - Zugriff mit Nachfragen erlauben
    - Zugriff verweigern
f) Im Falle von „Zugriff mit Nachfragen erlauben“ wird der AnwenderIn das Zertifikat mit sämtlichen Daten vor allem aber auch dem Mission Statement und allfällige Bewertungen aus der Community-Based Evaluation angezeigt. An dieser Stelle könnte die AnwenderIn

Applikationen dieses Trust Centers, dieses Anbieters oder nur dieser Applikation für die Zukunft erweiterte Rechte einräumen, z.B. „Allen Applikationen der Purpose Class „POI“ ohne Nachfragen vertrauen, wenn sie vom Unternehmen XY stammen“.

g) Der Applikation wird vom API ausschließlich auf die Ressourcen Zugriff gewährt, die der Purpose Class entsprechen.

4.4 Purpose Classes

Applikationen werden in *Purpose Classes* eingeteilt. Diese Einteilung richtet sich nach dem Zweck, den eine Applikation verfolgt und legt die für diesen Zweck benötigten Zugriffe auf Ressourcen des lokalen Systems fest. Der Zweck bezieht sich nicht nur darauf, welche Daten benötigt werden, sondern auch, wie bestimmte Daten verwendet werden.

So könnten Applikationen, welche die Location der AnwenderIn nur dafür verwenden, um dieser nahe gelegene *Points of Interest (POIs)* [6]) anzuzeigen, in einer Purpose Class „POI“ zusammengefasst werden. Ein gänzlich anderer Anwendungszweck der Location (also der gleichen Ressource) ist aber, wenn die Location anderen BenutzerInnen weitergegeben wird (z.B. Buddydienste, Fleet Management …).

Dieses Bespiel zeigt, dass die Zusammenstellung der Purpose Classes nicht nur als Kombination verschiedener Zugriffsrechte gesehen werden darf. Es wirft gleichzeitig die Frage auf, wie gewährleistet werden kann, dass Applikationen die erworbenen Daten nur klassenkonform verwenden.

Statt der Konfiguration der Rechte zum Zugriff auf einzelne lokale Ressourcen soll die Vergabe der Rechte durch die BenutzerIn auf Basis der *Purpose Classes* erfolgen. In einer Purpose Class ist genau festgelegt, welche Rechte benötigt werden. Gibt ein User eine Purpose Class frei, so können alle Applikationen, die zu dieser Klasse gehören, auf alle Features zugreifen, die dieser Klasse angehören. Beim ersten Aufruf der Applikation meldet das System dem User, zu welcher Klasse die Applikation gehört und fragt nach, ob der Zugriff auf die Klassenfeatures dieser Applikation gewährt werden soll.

4.4.1 Definition der Purpose Classes

Eine große Herausforderung ist mit Sicherheit die Erstellung sinnvoller Purpose Classes. Die Anzahl an Purpose Classes muss gering gehalten werden, um der EndanwenderIn nicht erneut Probleme bei der Konfiguration zu bereiten. Die Vereinfachung der Konfiguration ist eines der Ziele zur Schaffung der Purpose Classes. Es ist aber nicht anzunehmen, dass jede nur denkbare Anwendung perfekt in eine der Purpose Classes passen wird. Die Klassen haben einen gewissen Spielraum und beschränken sich darauf, die Zugriffe auf mehrere Ressourcen sowie die datenschutzrelevante Verwendung einzuschränken.

Als Ausweg für nicht einteilbare Applikationen bleiben folgende Varianten:

a) Individual-Applikation ohne Purpose Class: Solche Applikationen müssen im Vorhinein bekannt geben, welche Systemzugriffe sie benötigen. Im Mission Statement wird der

[6] Bsp. für POIs sind Apotheken, Geldautomaten, Restaurants,... in der Umgebung des Anwenders. So die Anwendung nicht auch einen „Routenfinder“ mit genauen Anweisungen, wie man zu dem POI kommt, enthält, ist die per GPS realisierbare metergenaue Position nicht notwendig. Die den Applikationen der Klasse POI zur Verfügung gestellte Position könnte also absichtliche – vom Anwender konfigurierbare – Abweichungen enthalten.

Verwendungszweck der Daten deklariert. Der User entscheidet anhand der einzelnen benötigten Rechte, ob er die Applikation zulässt. Dieses für die AnwenderIn umständliche „Fall Back Szenario“ ist derzeit bei jenen Systemen im Einsatz, die eine feine Konfiguration der Sicherheit eines Systems unterstützen.

b) Vollzugriff für die Applikation: Diese scheinbar undenkbare Variante ist die Realität bei den Native Applications, die als Alternative zu Web Applikationen derzeit angeboten werden.

4.5 Bedeutung des Mission Statements

Das Mission Statement ist eine vom Hersteller der Software verfasste Erklärung, welchen Zweck die Anwendung erfüllt und was mit den übertragenen Daten passiert (z.B. eventuelle Speicherung oder Weitergabe von Location-Daten). Eine technische Überprüfung, ob der Anbieter sein Mission Statement einhält, ist durch die hier vorgestellte Methode nicht möglich. Das Mission Statement ist daher eher als Vertrag zu sehen, den der Anbieter allen AnwenderInnen anbietet. Die Nichteinhaltung des Mission Statements ist demnach als Vertragsbruch zu bewerten. Das Mission Statement ist als Verfeinerung der Regeln, die für eine Purpose Class gelten, zu sehen. Es darf den Regeln der Purpose Class nicht widersprechen.

4.6 Promoter Zertifikate

Im bisherigen Konzept wird das Zertifikat immer von der Zertifikatsstelle allein ausgestellt. Dieses Zertifikat ist dann prinzipiell weltweit gültig. Möchte eine Organisation bzw. Unternehmen für die eigenen Mitglieder / MitarbeiterInnen nur bestimmte Applikationen freigeben (d.h. diese Applikationen sollen auf die für sie relevanten Funktionen der Schnittstelle zugreifen können), kann diese Organisation eine eigene Zertifikatsstelle implementieren – analog zu einem Zertifikatszentrum in einer unternehmensinternen PKI.

Alternativ kann sie aber auch als Promoter dieser Applikationen bei einer öffentlichen Zertifikatsstelle auftreten. Als Promoter übernimmt sie die Aufgabe eine Applikation und deren Hersteller zu überprüfen. Ein Zertifikat einer öffentlichen Zertifikatsstelle kann einen oder mehrere Promotoren haben. Am Gerät der EndanwenderIn kann eingestellt werden, Zertifikaten immer zu vertrauen, die einen bestimmten Promoter aufweisen. Unternehmen können diese Vertrauensstellung auf den Geräten ihrer MitarbeiterInnen einmal einrichten und danach beliebig viele Zertifikate (und damit Applikationen) promoten.

Die Methode eignet sich besonders für Unternehmen, die Applikationen von Application Service Providern (ASP) zukaufen, diese also nicht selbst betreiben. Hier hält der Provider die Lizenz für die erforderlichen Systeme und Software [16].

Die Authentizität eines Promoters in einem Zertifikat wird durch herkömmliche digitale Zertifikate gewährleistet.

5 Extensible Device API

Derzeit zeichnet sich nur im Bereich der Geolocation eine Standardisierung für APIs ab, die im Browser angesprochen werden können. Für andere Ressourcen gibt es noch keine Standards.

Da die Code Access Security nicht nur bestehende Ressourcen sondern auch erst zukünftig eventuell zur Verfügung stehende Ressourcen (z.B. RFID-Reader im Handy) behandeln soll, muss es zumindest den Ansatz einer erweiterbaren Schnittstelle für diverse Ressourcen geben.

5.1 Bestehende Ansätze

Es gibt bereits mehre Ansätze zur Erweiterung der Fähigkeiten einer Browserapplikation. Dazu zählen die erweiterten Möglichkeiten von Flash (Zugriff auf Kamera, Mikrophon) und Silverlight (local storage, local database).

Bemerkbar ist auch der Trend zu APIs, auf die im Browser über JavaScript zugegriffen werden können:

- Das W3C hat im September 2010 die Candidate Recommendation der „Geolocation API Specification" freigegeben. Die *privacy considerations* besagt, dass der User sein Einverständnis zur Nutzung der Schnittstelle geben muss. Diese Einwilligung muss für die AnwenderInnen widerrufbar sein. [10]
- Das Google Gears Framework realisiert u.a. Offline Access, Local Storage und Geolocation. Die Rechtevergabe im Framework basiert auf Freigaben auf Einzelapplikationsebene bei jedem Zugriff. Aktuell weist Google die Gears Ressourcen als überholt (deprecated) aus [8]. Grund ist das Ziel die Ressourcen in den HTML5-Standard einzuarbeiten [9]. Ansätze dafür sind das eben beschriebene Geolocation API sowie FileAPI, Web Storage und Offline Web Application [6, 10, 14, 17].
- Fire Eagle ist ein serverbasierter Ansatz, bei dem die Location von einer Anwendung am Gerät des Users zum Fire Eagle Service geschickt wird und anschließend von Web Applikationen abgefragt werden kann. Die Information wird immer über das Fire Eagle Service verwaltet. Der Zugriff des Service erfolgt über http(s). [7]
- Geode: Die Firefox Extention für Geolocation implementiert das W3C Geolocation API. Seit Version 3.1 von Firefox ist das API ein fester Bestandteil des Browsers [13]. Die Schnittstelle wird ebenfalls in der mobilen Version des Firefox unterstützt.

Während sich im Bereich der Geolocation mit der Candidate Recommendation der Geolocation API Specification in [10] eine Standardisierung abzeichnet, ist im Bereich anderer Erweiterungen (local file access, Kamera/Mic-Support) bisher kein Standard zu erkennen.

Im Bereich der Sicherheit sehen zwar alle Ansätze vor, die erweiterten Fähigkeiten nur dann einer Web Applikation zur Verfügung zu stellen, wenn dies durch den User legitimiert wird, die damit verbundene Komplexität für den User wird aber kaum diskutiert.

5.2 Extensible Device API für Web Applikationen

Bei der Definition einer Schnittstelle, die auf möglichst vielen Endgeräten zum Einsatz kommen soll, ist es wichtig, bestehende Implementationen und Spezifikationen nicht zu ignorieren.

Das API soll daher einer modularen Architektur entsprechen, bestehend aus dem generischen Interface für die darauf basierenden Applikationen und hersteller- bzw. gerätespezifischen Treibern.

Folgende API-Teile sind von den spezifischen Treibern und Implementationen unabhängig und daher immer vorhanden:

- Authorization-Interface: Funktionen, die zur Anmeldung der Applikation an das API verwendet werden.

- Supported Device Interface: ermöglicht der Applikation herauszufinden, welche der für diese Applikation prinzipiell autorisierten Devices auf dem konkreten Gerät auch tatsächlich verfügbar sind.

Es ist möglich, über das Extensible Device API auch bestehende APIs einzubinden (z.B. die HTML5-Extentions für Geolocation). Der Zugriff auf die Ressourcen wird dann nicht mehr auf die vorgesehene Weise kontrolliert (z.B. durch Nachfragen beim User), sondern dem Extensible Device API überlassen. Sollen die bestehenden APIs nicht verändert werden, kann das Extensible Device API auch als Ergänzung zu bestehenden Systemen eingesetzt werden. Wird das bestehende API auf gewohntem Weg (nicht über das Extensible Device API) aufgerufen, so kommt deren normales Sicherheitssystem zum Einsatz. Die Kompatibilität zu bestehenden Applikationen bleibt also erhalten. Erfolgt der Zugriff aber über das Extensible Device API, so soll keine weitere Überprüfung durch das Sicherheitssystem der bestehenden Schnittstelle erfolgen müssen.

6 Einsatzszenarien

Im folgenden Abschnitt werden ausgewählte Einsatzszenarien vorgestellt, in denen die Trust Center Controlled Code Access Security zum Einsatz kommen könnte.

6.1 Softwarehersteller für Web Applikationen

Der Softwarehersteller möchte Websites herstellen, die speziell den Mobilmarkt ansprechen und mobil einen Mehrwert gegenüber der Nutzung am PC versprechen. Für die volle Funktionalität wird direkter Zugriff auf das Telefon benötigt.

Die Erstellung von Native Applications für die diversen Plattformen ist keine Alternative, da der Aufwand durch die vielfältigen und völlig verschiedenen Development-Umgebungen am Handymarkt zu groß wäre und die MitarbeiterInnen des Herstellers nicht die benötigten Skills (C++, C#, Java, Objective-C usw.) aufweisen. Die Anwendung ist in erster Linie eine typische Webanwendung (Informationsanwendung), die mobile Szenarien optimal unterstützen soll, aber auch als Web Anwendung genutzt werden kann.

Um dies zu ermöglichen, verwendet der Anbieter die TC^{3}AS-Infrastruktur und bietet die erweiterten Funktionen an. BenutzerInnen, die das API aus der TC^{3}AS-Infrastruktur nicht installiert haben (oder dem Anbieter nicht vertrauen), können nur die erweiterten Funktionen nicht verwenden, der Rest der Applikation ist benutzbar.

6.2 Application Service Provider (ASP) / Unternehmenskunde

Ein ASP möchte branchenspezifische Lösungen für AußendienstmitarbeiterInnen von Unternehmen herstellen. Die MitarbeiterInnen / Unternehmen haben unterschiedliche Endgeräte. Die Unternehmen möchten die Lösung vom ASP betreiben lassen. Aus Sicherheitsgründen soll die Funktionalität der Software eingeschränkt sein. Außerdem benötigt die Applikation erweiterten Zugriff auf die Hardware. Eine normale Web Anwendung ist daher nicht ausreichend.

Ein Beispiel dafür wäre eine Versicherung, die ihre AußendienstmitarbeiterInnen zur Schadensaufnahme mit mobilen Geräten ausstattet. Fotos des Schadenfalls, werden gemeinsam mit den erhobenen Daten und dem Ort der Schadensaufnahme automatisch an den Versicherungsrechner geschickt. Die Software benötigt also folgende Zugriffe auf die eingebaute

Kamera des mobilen Gerätes, die Position (Location) und eventuell auf den lokalen Speicher (Zwischenspeicher bei schlechter Verbindung).

Der ASP verwendet die TC^{3}AS-Infrastruktur und erwirbt ein Zertifikat bei einer beliebigen Zertifikatsstelle. Das Versicherungsunternehmen testet die Software und tritt selbst als Promoter der Software auf. Die mobilen Geräte der MitarbeiterInnen werden so konfiguriert, dass sie allen Lösungen dieses Promoters (also der eigenen Firma) vertrauen. Damit sind alle Lösungen verschiedener Hersteller (oder auch des Unternehmens selbst) für die MitarbeiterInnen freigeschalten, die von dem Unternehmen promotet werden.

Das Einspielen von Updates erfolgt nur auf der Serverseite des Anbieters. Das Unternehmen kann jederzeit die Promotion für eine Software beenden. Damit kann sie von den MitarbeiterInnen nicht mehr eingesetzt werden. Die Software hat nur Zugriff auf die im Zertifikat angegebenen Funktionen. Sie ist also im Vergleich zu Native Applications eingeschränkter und damit sicherer.

6.3 Promoter: Interessensvertretung

Als Erweiterung des beschriebenen Szenarios kann auch eine Interessenvertretung (z.B. Verein, Fachverband, Verlag) als Promoter von Softwarelösungen auftreten. So könnte z.B. ein branchenspezifischer Fachverband Software für diese Branche (z.B. Ärztekammer, Fachverbände) oder ein Verlag, der Software testet und bewertet, diese Lösungen promoten (z.B. Ich vertraue jeder von meiner Interessensvertretung als gut befundene Site automatisch).

Denkbar wäre auch ein Geschäftsmodell, bei dem ein Promoter (z.B. ein Branchenspezialist) für seine Dienstleistung des Testens und Promotens vom Kunden bezahlt wird. Der Promoter tritt damit (rechtlich) als Vermittler zwischen den Anbietern und den Kunden auf, ähnlich eines Versicherungsmaklers, der die Angebote mehrerer Versicherungen prüft und den Kunden die beste Variante vermittelt.

6.4 Unternehmen mit eigener Infrastruktur

Ein Unternehmen betreibt seine eigenen Webanwendungen für seine MitarbeiterInnen. Die Anwendungen sollen auf die lokalen Ressourcen des mobilen Gerätes Zugriff haben. Jedes beliebige Unternehmen mit eigener Softwareinfrastruktur kommt in Frage. Die Einschränkung der Rechte der eigenen Software spielt innerhalb eines Unternehmens im Allgemeinen eine untergeordnete Rolle. Bei großen Organisationen mit dezentraler IT (z.B. Ämter, Großunternehmen mit mehreren Länderniederlassungen) kann die Relevanz allerdings gegeben sein. Erfolgt der Zugriff auf die Services von privaten Geräten der MitarbeiterInnen, ist eine Einschränkung der Rechte der Applikation in jedem Fall gerechtfertigt.

Als Beispiel könnte eine mobile Zeiterfassung dienen, bei welcher der Aufenthaltsort der Person (ungefähr) erfasst werden soll. Dieser muss aus Komfortgründen automatisch erkannt werden. Dazu betreibt das Unternehmen entweder ein eigenes Trust Center für seine Anwendungen, oder es erwirbt das Zertifikat bei einer öffentlichen Zertifikatsstelle. Die mobilen Geräte der MitarbeiterInnen vertrauen entweder allen Sites dieser Zertifikatsstelle (hausinterne Zertifikatsstelle) oder allen Sites dieses Herstellers (öffentliche Zertifikatsstelle).

6.5 Trust Center Betreiber: Mobile Internet Access Provider

Während der Hersteller eines bestimmten Gerätes die Entwicklung proprietärer Software für sein Gerät favorisieren wird, kann es für den Netzanbieter (Vodafone, T-Mobile etc.) durchaus interessant sein, den eigenen KundInnen auf ihrem Providerportal Dienstleistungen anzubieten, die über reine Webapplikationen hinausgehen. Der Einsatz der TC^{3}AS-Architektur ist aus Kostengründen relevant. Zusätzlich bietet es sich für einen Netz-Provider an, ein Trust Center für seine KundInnen zu betreiben.

Das *Extensible Device API* der TC^{3}AS-Architektur könnte bei Vertragshandys des Providers auf neuen Geräten sogar vorinstalliert werden.

7 Fazit

Die Konfiguration von Sicherheitseinstellungen ist für viele AnwenderInnen nicht zumutbar. Aus diesem Grund muss auf Webapplikationen, die Zugriff auf das lokale System benötigen, bisher verzichtet werden. Das System der „Trust Center Controlled Code Access Security“, das in diesem Paper vorgestellt wird, löst die Problematik:

- Die Konfiguration der Sicherheitseinstellungen muss nicht für einzelne Applikationen getroffen werden. Stattdessen werden die Einstellungen für alle Applikationen einer gemeinsamen Klasse getroffen.
- Die Einteilung konkreter Applikationen in diese Purpose Classes erfolgt nicht durch die AnwenderIn, sondern durch den Anbieter der Applikation. Diese Einteilung muss durch eine vertrauenswürdige Instanz bestätigt werden.
- Das Promoter-Prinzip befreit die EndanwenderInnen davon, einzelnen unbekannten Applikationen vertrauen bzw. diese konfigurieren zu müssen.

Die Konfiguration beschränkt sich für den User damit darauf, Applikationen bestimmter Purpose Classes zuzulassen oder abzulehnen bzw. zu bestimmen, welche Applikation oder welcher Promoter als vertrauenswürdig eingestuft wird.

Diese Konfiguration kann bequem am PC durchgeführt und in einem Trust Center hinterlegt werden. Durch die Auswahl eines Trust Centers sowie Eingabe der User Credidentials wird das Sicherheitsprofil am mobilen Gerät übernommen.

Besucht die AnwenderIn eine neue Webapplikation, die von den erweiterten Möglichkeiten das im Paper dargestellten „Extensible Device API“ Gebrauch machen möchte, so muss diese Applikation ein digitales Zertifikat aufweisen, das die Identität des Anbieters bzw. der Applikation bescheinigt. Bevor die Applikation auf das lokale System Zugriff bekommt, wird das Zertifikat beim Trust Center automatisch überprüft. Ein Programm bekommt im Allgemeinen aber nie Zugriff auf alle Funktionen der Schnittstelle, sondern nur auf jene, die zur Erfüllung des Zwecks der Applikation (bestimmt durch die Purpose Class) notwendig sind.

8 Weiterführende und zukünftige Arbeit

Das geschilderte System der Trust Center Controlled Code Access Security ist im Konzeptionsstatus. Derzeit sind einige Details noch ungeklärt bzw. dienen als Anhaltspunkte für weiterführende Diskussionen:

- In welcher Form können Purpose Classes am besten identifiziert werden? Soll die Klassifizierung durch eine zentral entworfene Taxonomie erfasst werden oder kann sie durch eine Folksonomie gefunden werden. In welcher Form ist eine nachträgliche Erweiterung der Klassifizierung vorzusehen?
- Die meisten Desktop Browser verwenden ein Plug-In Konzept, das die Erweiterung des Browsers um das Extensible Device API zulässt. Als Referenz kann hier das Gears-API dienen. Browser von Mobiltelefonen können derzeit meist nicht durch Plug-Ins erweitert werden. Browser von Drittherstellern, welche das Extensible Device API implementieren, könnten den Lizenzbedingungen der Plattformhersteller widersprechen.
- Im vorliegenden Konzept kann nur der clientseitige Teil des Codes einer Anwendung signiert werden. Der Serverteil bzw. was mit den Daten am Server passiert, kann nicht durch eine Signatur gesichert werden. Hier kann der Hersteller nur auf Basis des Mission Statements rechtlich belangt werden, wenn er gegen die Vereinbarungen in seinem Mission Statement verstößt. Die rechtlichen Hintergründe (Gesetzeslage) muss geklärt werden. Hier sind zudem regionale Unterschiede zu erwarten.

Unabhängig von den angeführten offenen Punkten bietet das Konzept der Trust Center Controlled Code Access Security für die potentiellen BenutzerInnen Erleichterung und Sicherheit beim Arbeiten mit Web Applikationen.

9 Literatur

[1] BARANAUSKAS, C. 2007. *Human-computer interaction - INTERACT 2007*. Springer.

[2] BLUMENSTEIN, K. und SCHMIED, G. 2010. Die vier Kernprobleme der mobilen Webentwicklung. *Proceedings of 3. Forum Medientechnik* (St. Pölten, Austria, 2010).

[3] BRAZ, C. und ROBERT, J.-M. 2006. Security and usability: the case of the user authentication methods. *Proceedings of the 18th International Conferenceof the Association Francophone d'Interaction Homme-Machine* (New York, NY, USA, 2006), 199–203.

[4] CRANOR, L.F. und GARFINKEL, S. 2005. *Security and usability: designing secure systems that people can use*. O'Reilly Media, Inc.

[5] ECKERT, C. 2009. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg Wissenschaftsverlag.

[6] File API: 2011. *http://dev.w3.org/2006/webapi/FileAPI/*. Accessed: 2011-06-06.

[7] Fire Eagle: 2007. *http://fireeagle.yahoo.net/*. Accessed: 2011-06-03.

[8] Gears API - Google Code: 2011. *http://code.google.com/intl/de-DE/apis/gears/*. Accessed: 2011-06-03.

[9] Gears API Blog: Hello HTML5: 2010. *http://gearsblog.blogspot.com/2010/02/hello-html5.html*. Accessed: 2011-06-06.

[10] Geolocation API Specification: 2010. *http://www.w3.org/TR/2010/CR-geolocation-API-20100907/*. Accessed: 2011-06-06.

[11] JENDRICKE, U. und GERD TOM MARKOTTEN, D. 2000. Usability meets security - the Identity-Manager as your personal security assistant for the Internet. *Proceedings of the 16th Annual Computer Security Applications Conference* (Washington, DC, USA, 2000), 344.

[12] KRIHA, W. und SCHMITZ, R. 2009. *Sichere Systeme: Konzepte, Architekturen und Frameworks*. Springer.

[13] Mozilla Labs » Blog Archive » Introducing Geode: 2008. *https://mozillalabs.com/blog/2008/10/introducing-geode/*. Accessed: 2011-06-03.

[14] Offline Web Applications: 2008. *http://www.w3.org/TR/2008/NOTE-offline-webapps-20080530/*. Accessed: 2011-06-06.

[15] SHNEIDERMANN, B. 1998. *Designing the user interface : strategies for effective human-computer interaction*. Addison-Wesley.

[16] VECCHIA, M.D. 2004. *Outsourcing, Managed Services, Application Service Providing*. BPX @ Dalla Vecchia GmbH.

[17] Web Storage: 2011. *http://dev.w3.org/html5/webstorage/*. Accessed: 2011-06-06.