

Sicherheit in Mobile Apps

Know-how Apps laufen auf unsicheren Plattformen, lassen sich in der Praxis aber dennoch genügend sicher umsetzen. Zentral dabei sind ein umfassendes Verständnis des Themas und die richtige Anwendung der Technologien und Prozesse.

Von Tom Sprenger und Sandro Antoniol

Mobile Apps sind omnipräsent und spielen in kritischen Geschäftsbereichen eine zunehmend wichtige Rolle. So dienen bei Finanzdienstleistern heute Mobile Apps als Kundenkanal oder bequeme Authentisierungslösung. Mittels Smartphone bezahlen Kunden an der Kasse oder im Online-Shop ihre Einkäufe. In der Industrie haben zum Beispiel Field Service Engineers über Mobile Apps Zugriff auf wichtige Informationen von Kundeninstallationen und -prozessen. Hohe Anforderungen an die Sicherheit bei der Implementation sind damit inhärent.

Implementation versus Plattform

Der Kern der Herausforderungen bei der Entwicklung einer sicheren Mobile App ist, dass hohe Anforderungen an die Sicherheit bei der Implementation bestehen, jedoch die Mehrzahl der heutigen mobilen Geräte als unsichere Plattformen zu betrachten sind. Zwar haben die Gerätehersteller in den letzten Jahren kontinuierlich nachgelegt, um das sichere Ausführen von Mobile Apps zu unterstützen. Sogenannte Trusted Execution Environments (TEE) werden aber noch lange nicht von allen Geräten und Betriebssystemen konsequent unterstützt. Aus der Theorie ist bekannt, dass sich auf unsicheren Plattformen keine 100 Prozent sicheren Lösungen bauen lassen. In der Praxis ist es aber so, dass man Apps mit den richtigen Massnahmen nichtsdestotrotz für ihren Bestimmungszweck genügend sicher umsetzen kann.

Zentral: Know-how der Entwickler

Geeignete Tools sind zwar wichtig, aber wer meint, dass es allein mit dem Einsatz eines der vielen App Hardening Tools getan sei, liegt falsch. Vielmehr bildet das Know-how der Mitarbeitenden im Secure Mobile Development Life Cycle (SMDLC) eine kritische Voraussetzung. Entsprechend lohnt es sich, in die Ausbildung der Mitarbeitenden zu investieren sowie deren Security-Bewusstsein zu schulen. Die Mitarbeitenden müssen in der Lage sein, Risiken und entsprechende Schutzbedürfnisse in einer App zuverlässig abzuschätzen und adäquate Schutzmassnahmen zu implementieren. Eine weitere Voraussetzung für die erfolgreiche Umsetzung des SMDLC ist eine sichere Entwicklungsumgebung. Dazu zählt neben den Entwicklungstools wie etwa dem Sourcecode Repository auch ein gemanagtes Repository der Dritartefakte (z.B. Libraries, GUI-Widgets), die in einer App verbaut werden. Ein durchgängiges Identitäts- und Accessmanagement stellt sicher, dass nur berechtigte Personen Zugriff auf die Entwicklung haben und dass in einem Audit Trail zu jedem Zeitpunkt nachvollziehbar dokumentiert ist, wer was gemacht hat. Basierend darauf lassen sich für besonders heikle Elemente einer sicheren Mobile App auch «Vier-Augen-Codereviews» umsetzen.

Im SMDLC lassen sich grob drei Phasen unterscheiden:

- Mobile-App-Architektur und -Design
- Mobile-App-Implementation (Coding und funktionale Isolation)

- Sicherhalten der Mobile App (Hardening, Auditing und Proactive Management)

Die Phasen bauen aufeinander auf. Eine sichere Mobile-App-Architektur und ein entsprechendes Design bilden damit die Basis für eine sichere Implementation.

Architektur und Design

Eine einfache, aber wirksame Grundregel liegt in der Aussage «Make everything as simple as possible, but not simpler» von Albert Einstein. Das Risiko von Sicherheitslücken steigt mit der Komplexität der Architektur. Es gilt, die Mobile App so zu designen, dass eine möglichst hohe Gewissheit über deren korrektes Funktionieren erreicht werden kann. Dazu gehört auch, dass die Grundmechanismen der Lösung einfach an Projektmitarbeitende vermittelt werden können. Ebenfalls wichtig ist, dass mittels automatisierter Tests einfach verifiziert werden kann, ob die sicherheitsrelevanten Mechanismen korrekt implementiert wurden.

Durch die Installation auf den mobilen Geräten entzieht sich die eigentliche Mobile App der Kontrolle durch den App-Anbieter. Die App läuft in einer nicht vertrauenswürdigen Umgebung und ist zusätzlichen Angriffsvektoren ausgesetzt. Es gilt daher, möglichst wenig kritischen Code oder kritische Daten in der App zu halten. Vielmehr sollen diese wo möglich serverseitig in einer State-of-the-art-Service-Architektur umgesetzt und über gesicherte APIs der App angeboten werden. Zwingend sind dabei die Authentisierung und Autorisierung auf dem API sowie eine sicher verschlüsselte Kommunikation zwischen der Mobile App und den serverseitigen APIs.

Für die Funktionalität, die als App auf dem mobilen Gerät ausgeführt wird, ist im Rahmen der technischen Architekturüberlegungen ein passender Technologieentscheid zu fällen. Dabei werden grundsätzlich drei Technologieansätze unterschieden: native, hybrid oder HTML5. Letztere sind keine Mobile Apps im eigentlichen Sinne, sondern Web Apps optimiert für mobile Geräte. Welcher Ansatz der geeignetste ist, hängt von den konkreten Anforderungen an die App ab (GUI-Komplexität, zu unterstützende Plattformen etc.). In Bezug auf Security haben native und hybride Apps gegenüber Web-Apps den Vorteil, dass sie auf den meisten mobilen Plattformen signiert

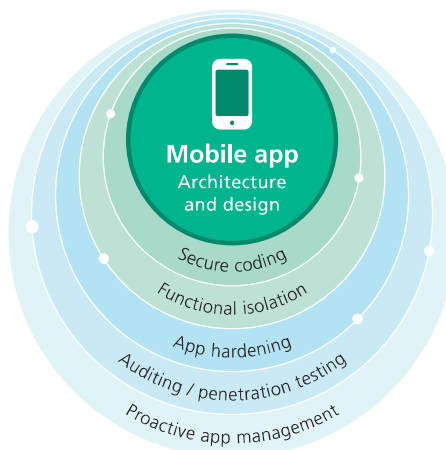
ausgeführt oder zumindest installiert werden, was der ungewollten Manipulation des Codes (Code Tampering) entgegenwirkt. Unabhängig von App-spezifischen Anforderungen gilt als Grundregel: für Funktionalität mit hohen Sicherheitsanforderungen ist die Implementation als nativer Code vorzuziehen. Dieser Code lässt sich deutlich besser schützen, weil er sich so kompilieren lässt, dass er keine Metadaten mitliefert, die das Reverse-Engineering als möglichen Angriffsvektor vereinfachen. Dabei ist wichtig zu wissen, dass eine native App nicht automatisch nativen Code impliziert. So wird beispielsweise unter Android eine App, die in Java geschrieben wird, als native App bezeichnet. Aber der Ausdruck «Native Code» bezieht sich auf Code, der in C/C++ geschrieben wurde. Native Code kann in eine native wie auch in eine hybride App integriert werden, nicht aber in eine mobile Web App.

Implementation: Secure Coding & Co.

Bei der eigentlichen Implementation geht es im ersten Schritt darum, die Grundregeln für sicheres Programmieren zu beachten. Als hilfreich erweisen sich zum Beispiel die plattformunabhängig formulierten OWASP Secure Coding Practices ergänzt durch die plattformspezifischen Secure Coding Guidelines, die für alle relevanten Plattformen verfügbar sind. Letztere helfen insbesondere bei der korrekten Verwendung der von einer Plattform angebotenen Funktion. Dies ist ein wichtiger Punkt, weil viele der Angriffsvektoren auf eine unsachgemäße Verwendung der Plattformfunktionen zurückzuführen sind. Ein typischer Fehler ist das unsichere Speichern von sensiblen Informationen (z.B. Passwörter) auf dem Gerät. Hier gilt es, wenn immer möglich plattformspezifische Funktionen (z.B. Secure Enclave) zu nutzen. Falls die Plattform selber keine solche Funktion bietet, ist alternativ der Einsatz von sogenannter White-Box-Kryptographie zu erwägen. Diese Technologie ermöglicht es, kryptographische Operationen durchzuführen, ohne vertrauliche Informationen wie etwa den kryptographischen Schlüssel preiszugeben.

Ein weiteres Thema bei der Implementation sind Benutzereingaben. Hier sind bei Mobile Apps dieselben Grundregeln zu beachten wie bei anderen Anwendun-

gen. Sämtliche Benutzereingaben müssen client- und serverseitig geprüft (validated) und von möglicherweise injiziertem Schadcode gesäubert werden (sanitized). Bei der Eingabe sensibler Informationen (z.B. PIN, Passwörter) gilt es, besondere Massnahmen zu beachten. Grundsätzlich sollen keine Third-Party-Keypads zugelassen werden, da diese Benutzer-Eingaben zum Teil zwecks Analyse und weiterer Services auf externe Server senden. Aus analogen Risikoüberlegungen sind Copy, Caching, Autokorrektur und -vervollständigung bei der Eingabe abzuschalten. Ebenso sollten für sensitive Eingaben keine Webviews verwendet werden, die es einer native App erlau-



Secure Mobile Development Life Cycle
(Bild: Adnovum)

ben, eingebettete Web-Technologien anzuwenden, und dadurch anfällig auf Hijacking beziehungsweise das Injizieren von Touch-Events sind. So könnte etwa eine Schadsoftware die Touch-Events einer Eingabe abfangen und dadurch Zugriff auf sensitive Eingabedaten erlangen. In kritischen Fällen kann es sich lohnen, in der App ein eigenes Keypad zu implementieren, das gegen unerwünschte Key Logger auf Systemebene immun ist.

Funktionale Isolation

Da das Gerät, auf dem die App läuft, bei der Entwicklung von Mobile Apps als grundsätzlich nicht vertrauenswürdig anzusehen ist, kann man auch einem Systemfunktionsaufruf aus der App nicht vertrauen. Ein Angreifer könnte den Aufruf abfangen und die übergebenen Parameter lesen oder modifizieren. Dies ist besonders dann kritisch, wenn es um kryptographische Funktionen geht, bei denen ein Teil der Parameter beziehungsweise Rückga-

bewerte typischerweise schützenswerte Daten sind.

Hier kommt das Konzept der sogenannten funktionalen Isolation zum Zug. Um sicherzustellen, dass sicherheitskritische Funktionen auch auf einem kompromittierten System korrekt funktionieren, werden diese dabei – obwohl grundsätzlich auch durch das OS bereitgestellt – als fester Bestandteil der Mobile App ausgeliefert. So empfiehlt es sich, für sichere Apps die Library für kryptographische Operationen in die App einzukompilieren. Ein weiterer Vorteil dieses Ansatzes ist, dass beim Bekanntwerden einer Sicherheitslücke die Library im Rahmen eines App Updates rasch aktualisiert werden kann.

Eine weitere Massnahme ist die applikatorische End-zu-End-Verschlüsselung sensibler Daten. Alle sensiblen Daten werden für die Übertragung zwischen Mobile App und Server basierend auf einem in die App einkompilierten Public Key verschlüsselt. Der Private Key wird dabei sicher auf der Serverseite verwaltet. Damit wird sichergestellt, dass nur die Applikation Zugriff auf die Daten hat. Man-in-the-middle-Angriffe, sprich ein unerwünschtes Mitlesen beziehungsweise Manipulieren auf der Ebene des möglicherweise kompromittierten Betriebssystems, können so verhindert werden.

Hardening der Mobile App

Im nächsten Schritt geht es darum, die Mobile App zu schützen. Man spricht hier vom Hardening der App. Dies beinhaltet zum einen den Schutz gegen Integritätsrisiken wie zum Beispiel die Modifikation oder das Einschleusen von Code oder die Manipulation von Ressourcen (z.B. Texte, URLs). Zum anderen sollen Vertraulichkeitsrisiken wie Reverse Engineering oder Codeanalyse gemindert werden.

Der Schutz gegen Integritätsrisiken kann durch das Injizieren von spezifischen Checkpoints in den Bytecode der App erreicht werden. Diese Checkpoints werden im Rahmen der normalen Ausführung der App für den Benutzer nicht feststellbar mitausgeführt und überprüfen die Integrität des Mobile App Codes. Als Teil davon wird beim Aufruf externer Libraries und Systemfunktionen validiert, ob diese Aufrufe nicht zum Beispiel durch Malware abgefangen oder umgelenkt werden. Eine weitere Art von Kontrolle besteht in der Überprüfung

«Swiss IT Magazine ist Pflichtlektüre für Manager in der IT-Industrie.»

Christian Schollenberger,
Leiter Informatik,
KIBAG Dienstleistungen AG

Jetzt drei Ausgaben
kostenlos zur Probe:
www.itmagazine.ch/probeabo
Versand nur in die Schweiz.

Mobile-App-Entwicklung 2018

der Ausführungsumgebung auf Jailbreaking oder Rooting des Geräts. Mit dieser Überprüfung kann sichergestellt werden, dass die Mobile App nicht in einer Umgebung zur Ausführung kommt, in der die systemseitigen Sicherheitsmechanismen ausgeschaltet sind oder einfach umgangen werden können.

Das Thema Vertraulichkeit und somit der Schutz des geistigen Eigentums sowie das Verhindern von Codeanalysen zur Ausnutzung von Verwundbarkeiten für mögliche Angriffe wird durch sogenannte Code und Control Flow Obfuscation sichergestellt. Dabei werden Variablen- und Methodennamen so umbenannt, dass sie möglichst keinen semantischen Rückschluss auf den Zweck der Funktion zulassen. Zudem wird der Kontrollfluss des Codes verschleiert, um die Interpretation des Programmablaufs zu erschweren. Als weitere Massnahme werden Zeichenketten verschlüsselt. Auch dies mit dem Zweck, Rückschlüsse für ein mögliches Reverse Engineering zu verhindern. Open Source und andere einschlägige Anbieter bieten Tools an, um das Hardening der App als mechanischen Task in den SMDLC einzubetten. Die Praxis zeigt allerdings, dass ein fortgeschrittenes Hardening einiges an Aufwand bedeutet. Die Schutzmechanismen haben ihren Preis. Kritische Grössen sind die Zunahme des Fussabdrucks der App (Grösse des App-Codes und Speicherbedarf) sowie mögliche Performance-Einbussen. Letztlich gilt es, im Rahmen einer Risikobetrachtung einen adäquaten Trade-off zwischen Schutz und Performance zu finden.

Auditing und Penetration-Testing

Als nächster Schritt im SMDLC gilt es, die Wirksamkeit der umgesetzten Sicherheitsmassnahmen zu validieren. Die ausgeklügeltsten Massnahmen nützen nichts, wenn etwa durch einen Konfigurationsfehler in der Build Pipeline die Mechanismen nicht oder nicht korrekt auf die Mobile App appliziert werden. Ein regelmässiges Auditing der Entwicklungsprozesse und der architektonischen Entscheide sowie ein möglichst unabhängiges Testing der korrekten Funktionsweise der Sicherheitsmassnahmen sind deshalb von fundamentaler Bedeutung. Um das Leben möglicher Angreifer weiter zu erschweren, empfiehlt sich eine periodische Anwendung des Hardenings und ein

Hochladen in den App Store auch dann, wenn sich an der eigentlichen App nichts geändert hat. Damit kann erreicht werden, dass Angreifer bei ihrer Bemühung, die App zu verstehen und Schwachpunkte zu finden, immer wieder von vorne beginnen müssen, da eine neu gehärtete App komplett anders aussieht. Server-seitig sollte zudem sichergestellt werden, dass die Services für ältere Versionen der App nicht mehr zur Verfügung stehen.

80 Prozent Handwerk

Zusammenfassend kann gesagt werden, dass man bei der Entwicklung einer sicheren Mobile App gut beraten ist, der Umgebung, in der sie ausgeführt wird, grundsätzlich nicht zu trauen und systemseitige Sicherheitsmechanismen kritisch zu beurteilen. Das Bauen einer sicheren App beginnt beim Design. Der Einsatz von Hardening Tools ist notwendig, kann aber Schwachstellen im Design oder der Implementation nicht wettmachen und ist darum definitiv nicht hinreichend. Eine sichere App zu entwickeln bedeutet zu 80 Prozent Handwerk und nur zu 20 Prozent Einsatz von Tools. Die Achillesferse eines erfolgreichen SMDLCs sind somit die involvierten Mitarbeitenden. Verfügen sie über das nötige Security-Wissen und verstehen ihr Handwerk, lassen sich auf heute gängigen Plattformen zwar keine zu 100 Prozent, aber für ihren Bestimmungszweck definitiv genügend sichere Mobile Apps entwickeln. ■

DIE AUTOREN

Tom Sprenger ist CTO des Software-Unternehmens Adnovum, das führende Schweizer Mobile-Banking- und Mobile-Payment-Lösungen entwickelt



hat. Er befasst sich seit vielen Jahren mit der Sicherheit mobiler Applikationen und zeichnet bei Adnovum für den Secure Mobile Development Life Cycle verantwortlich.

Sandro Antoniol ist Senior Software Architect bei Adnovum. Er hat das Design und die Integration diverser Software-Lösungen geleitet und dabei umfangreiches Expertenwissen in Mobile App Security aufgebaut.

