Mobile Security

Michael Fischer, ORACLE Deutschland B.V. & Co. KG

Mit der zunehmenden Nutzung mobiler Endgeräte sowohl im privaten als auch im beruflichen Umfeld und den täglichen Berichten über Daten-Missbrauch und neue Regularien wie das EU-Datenschutzgesetz steigt das Interesse an mobiler Security. Dieser Artikel zeigt eine End–to-End-Betrachtung von Konzepten und Lösungen für eine Absicherung bei der Verwendung von mobilen Devices wie Smartphones oder Tablets.

Wenn man nach „Mobile Security" im Internet sucht, erscheinen in den Trefferlisten hauptsächlich Lösungen für Anti-Virus- und Anti-Malware-Lösungen. Diese betrachten Mobile Security aus Sicht des Anwenders, der sich wenig darum kümmert, wie die Absicherung außerhalb seines Device aussieht. In etwa so, wie er es vom Desktop her kennt, und vielleicht ist er auch deshalb froh, dass er nicht mehr so viele Wahlmöglichkeiten hat, etwa automatisiertes Patchen zu aktivieren, Passwörter in einem Password-Safe abzuspeichern oder über die Desktop-Firewall selektiv Dienste abzuriegeln. Unternehmen stecken bei der Endgeräte-Absicherung in einem Zwiespalt, einerseits nur möglichst sichere und kontrollierte Endgeräte zuzulassen, andererseits ein attraktives Arbeiten auch mit eigenen Devices zu ermöglichen. Das Paradigma „work from anywhere – with anything" tut sein Übriges dazu.

Die Absicherung der Endgeräte, ob Smartphone, Tablet oder Desktop, betrachtet nur einen Teil der potenziellen Einfallstore. Studien von Analysten und Beratern beschäftigen sich bei Mobile Security oft lediglich damit, die Apps aus zugänglichen App Stores zu analysieren und zu bewerten. Dies deckt Probleme auf, wie eine implizite Weitergabe von Daten, betrachtet aber wieder nur einige Aspekte. Eine Lösung bedarf nicht nur Korrekturen im App-Design, sondern benötigt ein ganzheitliches Security-Konzept.

Der Aufbau

Mithilfe einer logischen Architektur werden die einzelnen Schutzmechanismen erläutert. Parallel dazu erfolgt eine Betrachtung aus der Sicht der Akteure – hier der Endbenutzer, Entwickler, Betreiber sowie Unternehmen. Über diese Zerlegung kann ein Mapping auf einzelne Produkte verschiedener Hersteller erfolgen oder eine bestehende Architektur lässt sich auf Lücken prüfen. Exemplarisch sind das Oracle-Portfolio gemappt und ein Kundenbeispiel aufgeführt. Werden bei einer Architektur zwischen diesen Schichten standardisierte Schnittstellen und Protokolle berücksichtigt, lassen sich fehlende Funktionen nachrüsten. Ein Vergleich aus Anbietersicht mit Komplettlösungen findet nicht statt, jedoch könnte eine Betrachtung über die vorhandenen Protokolle erfolgen. Komplettlösungen ergeben durchaus Sinn, wenn es um vollständig kontrollierbare und abgeschottete Einsatzszenarien geht. Wermutstropfen dabei ist die Nutzbarkeit; obwohl etwa das Phone der Kanzlerin eine solche abgesicherte Lösung bietet, erfolgt parallel die Nutzung als normales Smartphone.

Nicht alle Nutzungsszenarien sind gleich, daher sind die skizzierten Funktionen und Komponenten eine nicht vollständige Obermenge. Der Einstieg mit verschiedenen Layern und Funktionsbausteinen ist jedoch schon relativ früh sinnvoll, so kann eine Anmeldemöglichkeit an

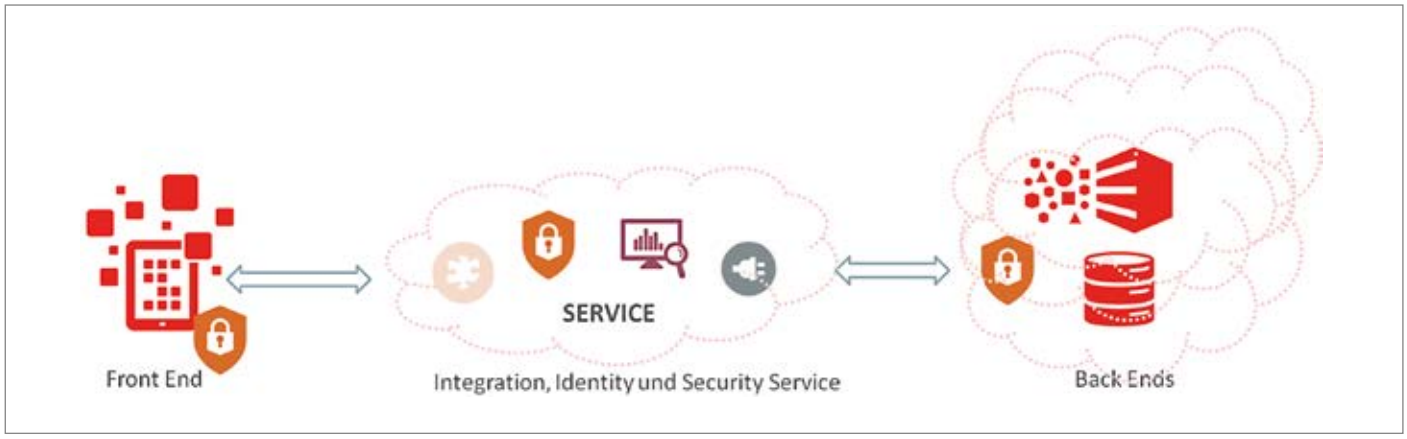

Abbildung 1: Architektur-Überblick Mobile Security

einer App oder das Übertragen von Bewegungsdaten bedingt durch das Bundes- oder EU-Datenschutzgesetz bereits einen Großteil dieser Funktionen impliziert fordern. Beispiele sind verschlüsselte Übertragung und Speicherung, Zugriffsbeschränkung für Administratoren, Nachweis für den Nutzer darüber, was gespeichert wurde, oder das Recht auf Löschungen.

Beteiligte Komponenten

Die beteiligten Komponenten lassen sich grob vereinfacht in einem Drei-Schichten-Modell darstellen *(siehe Abbildung 1)*. Das Frontend beschreibt das Device, unabhängig davon, ob ein Browser oder eine App genutzt wird. Service- und Back-End-Layer sind Cloud-artig dargestellt, da es ja keine Rolle spielt, woher sie kommen, aus dem eigenen Rechenzentrum und/oder in Kombination mit anderen Anbietern aus der Cloud.

Klassischerweise ist ein Back-End-Layer vorhanden, bei traditionellen Unternehmen sind das die bestehenden Rechenzentren, bei neuen disruptiven Firmen kann dieser auch beim gleichen Betreiber liegen wie der Service-Layer oder ein Mash-up aus vorhandenen verteilten Services sein.

Der Service-Layer kann verschiedene Back-Ends integrieren, etwa Google Maps zusammen mit einem Asset-System. Als Beispiel würde der Servicetechniker eines Unternehmens bei einer Reparatur von Aufzügen Informationen zum Aufzug aus dem Asset-System bekommen und über Google Maps zum Reparaturort oder Ersatzteilelager geleitet werden. Ein analoges Endkundenbeispiel ist die Suche nach einem Buchhändler, der ein bestimmtes Buch vorrätig hat, inklusive einer Lotsenfunktion unter Berücksichtigung der Erreichbarkeit und der Öffnungszeiten.

Über den Service-Layer erfolgt auch eine Integration, die den Benutzer identifizieren kann und personalisierte Zugriffe auf Services erlaubt. Der Benutzer kann dabei entscheiden, wer bei Bedarf seine Identität verifiziert (etwa Facebook, Microsoft oder ein Unternehmen) und welche Daten an wen weitergegeben werden dürfen.

In der folgenden Unterteilung sind die Security-relevanten Funktionen aufgeführt. Funktionen zum Absichern des Front-Ends sind:

- *Gesicherte Ablauf-Umgebung*
  Basis ist der Schutz des Device, etwa mit PIN und/oder Fingerprint. Zudem erfolgt durch Anti-Virus- und Anti-Malware-Lösungen eine bestmögliche Sicherung der Ablauf-Umgebung. Die geprüften Apps an sich und der Benutzer bleiben ohne größere Einschränkungen.

- *Sichere Ablauf-Umgebung/Container*
  Eine sichere Ablauf-Umgebung, auch Container, verhindert den Zugriff von anderen Apps auf die laufende App oder deren Daten und steuert zudem die Möglichkeit, Daten zu exponieren, wie Drucken oder das Speichern in eine Cloud-Umgebung. Bei der Definition werden Device-seitig zulässige Apps festgelegt und zur Erkennung des Geräts in der Regel Zertifikate ausgerollt. Der Speicherbereich ist verschlüsselt, um die Daten auch offline beziehungsweise auf dem Datenspeicher zu schützen. Die Verwaltung der Devices erfolgt über Policies. Vor der erstmaligen Verwendung ist ein Enrollment notwendig.

- *Schutz der gespeicherten Daten und Passwörter*
  Passwörter sollten immer geschützt sein. Das kann durch Ablage in die vom mobilen Betriebssystem vorgesehenen Bereiche (wie Keychain) erfolgen, durch die App selbst oder nur Server-seitig (mit der Einschränkung bezüglich Offline-Funktionen). Bei der App selbst stellt sich wie bei jedem Security-Code die Frage, wie sicher eine Eigenentwicklung gegenüber einem fertigen Paket/SDK sein kann. Gespeicherte und schützenswerte Daten sollten schon wegen potenziellen Verlustes verschlüsselt sein, etwa durch eine mit AES verschlüsselte Datenbank.

- *Integration der Device-Funktionen wie Entsperrung per Fingerprint, Voice oder Selfie*
  Zum Entsperren sollten die vom Hersteller vorgesehenen Mechanismen zum Einsatz kommen. Neben der PIN bieten biometrische oder Mehrfaktor-Authentifizierungen einen höheren Schutz. Dies sollte in der Anwendung zum Tragen kommen, damit der Vertrauensgrad der Umgebung mit berücksichtigt werden kann, etwa als aufsteigende Level: keine Sperre, PIN, Zertifikat, Bio, Einmal-Passwort. Auch kann eine Applikation weitere Authentifizierungen selbst verlangen, wenn entweder der Verdacht auf unsachgemäße Nutzung besteht (etwa Mehr-

fachaufruf von Personendaten) oder besonders sensible Funktionen (wie Buchung oder Überweisung) ausgeführt werden. Hier können auch zusätzliche, nicht vom Device bereitgestellte Verfahren integriert werden.

- *Single-Sign-on-Funktionen*
  Um dem Benutzer ein echtes Single-Sign-on zu ermöglichen, müssen die Zugriffstoken zwischen den Apps und dem App-Browser austauschbar sein. Dies lässt sich durch entsprechende SDKs erreichen. Basis ist immer ein Anmeldeverfahren, das beispielsweise OAuth-Token erstellt, die dann von der jeweiligen App dem Server präsentiert werden.

Der Service-Layer stellt die Kontaktpunkte der Front-end-App dar:

- *Anmeldeverfahren, Step-up-Möglichkeiten, Single-Sign-on*
  Wird für die App eine Identifikation des Benutzers serverseitig benötigt, würde sie diesen Service beziehen. Das Service-Prinzip ermöglicht den Austausch und die Erweiterung der Funktionalität, ohne die Apps ändern zu müssen, beispielsweise um eine stärkere Authentifizierung einzufordern oder ein Google-/Facebook-Login aus- beziehungsweise einzuschalten. Der Service arbeitet in der Regel über Token.

- *Abstraktion Back-End-Schnittstellen über mobile, verträgliche Schnittstellen (wie REST, OAuth) und Nutzerbezug beziehungsweise Trust-Verhältnis mit den Schnittstellen*
  Mobile Apps setzen typischerweise auf REST-Service-Aufrufe auf. Meist müssen diese noch propagiert oder erstellt werden. Im Zuge nicht trivialer Services ist auch hier ein Zugriffsschutz notwendig, der dann dort umgesetzt wird und den Zugriffstoken für das Back-End an sich in ein passendes Format (wie SAML) umwandelt.

- *Aggregation von Einzel- zu Compound-Services*
  Bestehende Services haben in der Regel eine zu breite Schnittstelle und stellen oft Einzelfunktionen dar. Im Bereich „Mobile“ sind wegen der Datenrate schmale Services erforderlich; um Entwicklern eine schnelle und einfache Erstellung zu ermöglichen, werden Services zusammengefasst. Auch hier gilt es, die entsprechenden Zugriffsberechtigungen zu propagieren.

- *Kontrollierte Weitergabe von Benutzer-Informationen*
  Der Service-Layer steuert die kontrollierte Weitergabe von Benutzer-Informationen und führt den Nachweis darüber. Er berücksichtigt dabei die jeweilige Einverständniserklärung des Benutzers (etwa mittels User Consent). Der Nachweis ist durch das kürzlich verabschiedete EU-Datenschutzgesetz gefordert und wird künftig eine stärkere Rolle spielen.

- *Verifikation berechtigter Benutzer unter Berücksichtigung des Kontexts*
  Um Risiko und Betrug zu entgegnen, erfolgt eine technische Bewertung des Aufrufs (etwa offenes WLAN vs. Firmennetz, registriertes vs. nicht-registriertes Device) und des Kontexts (wie zuvor durchgeführte Aktionen) dieses Aufrufs. Die Service-Komponente führt diese Bewertung durch und stellt das Ergebnis der App (oder deren Serverseite) zur Verfügung oder schreitet direkt ein (etwa durch nur noch Lesezugriffe oder Re-Authentifizierung mit Einmal-Passwort).

- *Administrationsmöglichkeit und Auditierung*
  Zur Verwaltung der Benutzer-Accounts und Berechtigungen ist eine Komponente notwendig. Abhängig vom Service muss auch der Nachweis geführt werden, wer Berechtigungen zugeordnet oder genehmigt hat. Für kritische Berechtigungen ist eine Rezertifizierung erforderlich. Im Blickfeld sind nicht nur Anwendungs-Accounts, sondern auch die Accounts der Betreiber und Entwickler. Der Übergang von Consumer-Services zu Unternehmensanwendungen ist hier fließend; diese Anforderungen betreffen auch Consumer-Anwendungen, wenn es etwa zum Abfluss von Accounts/Daten durch einen der privilegierten Accounts kommt.

- *Service gemäß CIA-Prinzip*
  CIA steht für „Confidentiality“, „Integrity” und „Availability”. Dies muss über die Services, je nach gewünschtem Service-Level, gewährleistet sein und kann über die „state of the art“-Mechanismen wie Zugriffsbeschränkung (Anmeldung, Autorisierung, rollenabhängige, zweckgebundene Nutzung), Verschlüsselung oder Anonymisierung, Auditierung sowie optional Hochverfügbarkeit erreicht werden.

Die Funktionen des Back-End-Layers stimmen größtenteils mit denen des Service-Layers überein und sind dort bereits beschrieben:

- Verifikation des Aufrufers (Trust, Weitergabe Benutzer-Informationen)
- Verifikation berechtigter Benutzer beziehungsweise Service-Schnittstelle unter Berücksichtigung des Kontexts
- Administrationsmöglichkeit und Auditierung
- Service gemäß CIA-Prinzip

Die Akteure

Eine grobe Unterscheidung in die verschiedenen Akteure „Endbenutzer“, „Entwickler“, „Betreiber“ und „Unternehmen“ hilft, weitere Aspekte darzustellen. Die Funktionen überschneiden sich teilweise mit den den jeweiligen Layern zugeordneten Funktionen. Für den Endbenutzer gilt:

- Auswahl des Identity-Providers je Nutzung
- Auswahl, ob Credentials lokal gespeichert werden sollen
- Einmalige Anmeldung und damit SSO zwischen Apps und App-Browser
- Auskunft über gespeicherte Daten
- Erlaubnis für einen Service, persönliche Daten nutzen zu können; einmalig, befristet oder bis auf Widerruf
- Steuerung der Security: soviel Security, wie der Benutzer als notwendig empfindet (mit dem Risiko, manche Funktionen nicht nutzen zu können)

Für den Betreiber gilt:

- Unterstützung bei Administration und Auditierung
- Unterstützung bei CIA

Für das Unternehmen gilt:

- Koppelung an das Unternehmens-Identity- und Access-Management

- Abbildungsmöglichkeiten von Schutzklassen
- Definition und Pflegemöglichkeiten für Policies
- Steuerung zulässiger Apps und Konfigurations-Einstellungen, wie verschlüsselte Verbindungen, Daten nur verschlüsselt gespeichert und nur nach Freigabe von anderen Apps nutzbar

Für den Entwickler gilt:

- Nutzung bereitgestellter Sicherheits-Funktionen
- Konzept/Vorgaben Best Practices
- Keine Entwicklungsmöglichkeit in der Produktion
- Möglichkeit, anonymisierte Testdaten zu nutzen
- Zugriffsmöglichkeit auf Notfall-Accounts

Abbildung 2: Front-End-Beispiel mit MAF und MAX

Umsetzung mit Oracle

Für das Front-End bietet Oracle zur Umsetzung der skizzierten Funktionen eine Entwicklungsumgebung zur Entwicklung mobiler Apps für iOS, Android und Windows (Desktop-Tool: MAF − Mobile Application Framework, Server-basiert: MAX − Mobile Application Accelerator, siehe *Abbildung 2*). Hier sind die grundlegenden Funktionen für Authentifizierung und Integration in ein Server-seitiges Identity- und Access-System. Eine verschlüsselte Datenspeicherung und Synchronisation mit einer Server-seitigen Datenhaltung ist möglich. Die Verbindungen sind optional verschlüsselt. Da dies native Apps sind, ist auch eine Offline-Funktion möglich. Optional kann diese App in einem 3rd-Party-Container betrieben werden.

Browser-basierte Anwendungen können mit den Fusion-Middleware-Komponenten auf WebLogic-Basis und beispielsweise mit dem JDeveloper entwickelt werden. Dort stehen alle Identity- und Access-Funktionen zur Verfügung (siehe Service-Layer). Natürlich lassen sich diese Anwendungen auch aus der Datenbank heraus mit Apex programmieren. Dabei kann Apex für Authentifizierung („ootb" mit Oracle) und Autorisierung („build") in das Identity und Access Management eingehängt werden.

Im Oracle-Cloud-Angebot gibt es Werkzeuge, um einfache Anwendungen ohne Entwickler-Know-how erstellen zu können (MAX − Mobile Application Accelerator).

Funktion	Cloudbasierte Lizenz	On-Premise-Lizenz
Mit Securityfunktionen integrierte Entwicklungsumgebung	Mobile Application Accelerator (MAX)	Mobile Application Framework (MAF), Oracle JDeveloper, Oracle APEX
Gekapselte oder standardisierte Security-Funktionen zur Verwendung in beliebigen Umgebungen	Oracle Identity Cloud Service (IDCS)*	SDK der Oracle Access Management Suite bzw. die Standard-schnittstellen wie OAuth, SAML etc.,
Verschlüsselte Datenhaltung auf Smartphone, Tablet		SQLite von MAF (DB), Oracle Mobile Security Suite (Container)
Verschlüsselte Übertragung	https	https

*) noch nicht verfügbar

Tabelle 1

Oracle bietet SDKs, um die zentralen Identity- und Access-Funktionen unabhängig der verwendeten Entwicklungsumgebung in native Apps zu integrieren. Das geht über pure OAuth- und User/PW-Funktionen hinaus, etwa mit Device-Registrierung, „step-up"-Authentifizierung oder kontextbasierter Autorisierung. Darüber hinaus wird ein Token-basiertes SSO zwischen Apps sowie zwischen App und Browser bereitgestellt.

Oracle und Partner bieten eigene SDKs an, um die Funktionen des Oracle Mobile Cloud Service (der auch Identity- und Access-Funktionen enthält) nativ in Entwicklungsumgebungen zu nutzen. Dies sind zurzeit Xamarin (jetzt Microsoft) und Sencha. *Tabelle 1* zeigt die Security-relevanten Funktionen im Überblick.

Der Service-Layer stellt viele der Sicherheits-relevanten Funktionen bereit. Manche Funktionen sind in mehreren Schichten vorhanden, um eine mehrschichtige Security (bei Oracle sogenanntes „defense in depth") umzusetzen *(siehe Abbildung 3)*.

Identity- und Access-Funktionen für die Anmeldung über Browser oder REST/App mittels Standard-Protokollen wie OAuth, SAML, OpenID, „Step-up"-Authentifizierung, OTP, TOTP, SSO und kontextbasierter Autorisierung werden bereitgestellt. Eine Administration der Accounts und Berechtigungen inklusive Governance-Funktionen steht zur Verfügung. Diese Services laufen auf einer WebLogic-Plattform und können mit bestehenden Funktionen (An-

meldeverfahren -> SSO, Userstore -> Konnektoren, Genehmigungs- und Rezertifizierungs-Workflows) verknüpft sein.

Die Identity- und Access-Services werden zeitnah auch stand-alone als Cloud Services (Identity Cloud Service) zur Verfügung stehen. Integriert in andere Oracle Public Cloud Services sind sie heute schon verfügbar.

Das API-Gateway sichert die Service-Aufrufe ab, es ist so etwas wie der erste Security-Layer, den die App oder -Browser-basierte Applikation aufruft. Hier wird bei Bedarf nochmals geprüft, ob eine ausreichend starke Anmeldung und Autorisierung vorliegt. Zudem ist dies der Punkt, bei dem „on the fly" in den Datenstrom eingegriffen werden kann (etwa zum Filtern oder Anonymisieren von Daten). Weitere Funktionen hinsichtlich Virenabwehr oder DoS sind ebenfalls vorhanden.

Die Integrations-Plattform nutzt die gleichen Mechanismen wie die genannten Identity- und Access-Funktionen. Sie ist sowohl als On-Premise-Installation als auch als Cloud Service verfügbar. Der Mobile Cloud Service bietet eine vorgefertigte Plattform, um Apps und Integration in Back-Ends zu bauen. Dieser Service wurde schon in einer der früheren Ausgaben vorgestellt. *Tabelle 2* zeigt die Security-relevanten Funktionen im Überblick.

Der Back-End-Layer ist entweder schon vorhanden und muss nur die Aufrufbarkeit über mobile Schnittstellen ermöglichen oder wird „from the scratch" zusammen mit der Service-Schicht erstellt. Sind die Service- und Back-End-Funktionen getrennt, bietet sich auch hier eine vorgeschaltete, Gateway-artige API-Funktion zur Entkoppelung und als Security-Layer an (siehe Beschreibung Service-Layer und *Abbildung 4*).

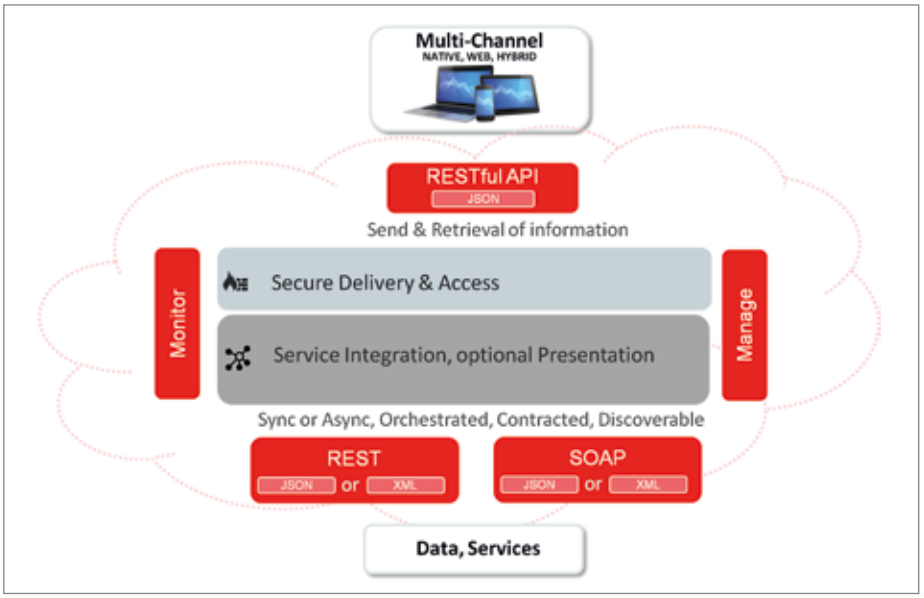

Abbildung 3: Service-Layer-Funktionen IAM und Service-Integration

Funktion	Cloudbasierte Lizenz	OnPremise-Lizenz
Authentifizierung, SSO	Oracle Mobile Cloud Service(MCS), Oracle Identity Cloud Service (IDCS)*	Oracle Access Management Suite
Starke Authentifizierung, Risk & Fraud-Detection	IDCS*	Oracle Access Management Suite
Autorisierung	MCS, IDCS*	Oracle Access Management Suite
API-Schnittstelle, REST/SOAP	MCS, Oracle API Cloud Service (API CS)*	Oracle API Gateway
Service-Composition & Integration	MCS, ICS, SOA CS	Oracle API Gateway, Oracle SOA Suite
Sichere Speicherung von Daten (inkl. sensibler Daten)	Oracle Database as a Service (DBaaS)	Oracle-Datenbank (ASO-Verschlüsselung, DBVault - Gewaltentrennung, AuditVault-Audit)

*) noch nicht verfügbar

Tabelle 2

Ein Beispiel

Das folgenden Beispiel zeigt die Umsetzung eines Kunden, der für seine Märkte zwei Typen von Apps zur Verfügung stellt: eine für seine Endkunden für Informationen und Bestellungen (zur Abholung) und eine Tablet-App für die Angestellten, damit diese eine persönliche Kunden-Ansprache durchführen können *(siehe Abbildung 5)*.

Als Front-End wurde eine native App für iOS und Android mit Oracle MAF umgesetzt. Oracle MAF authentifiziert den Benutzer gegen das bestehende Kunden-

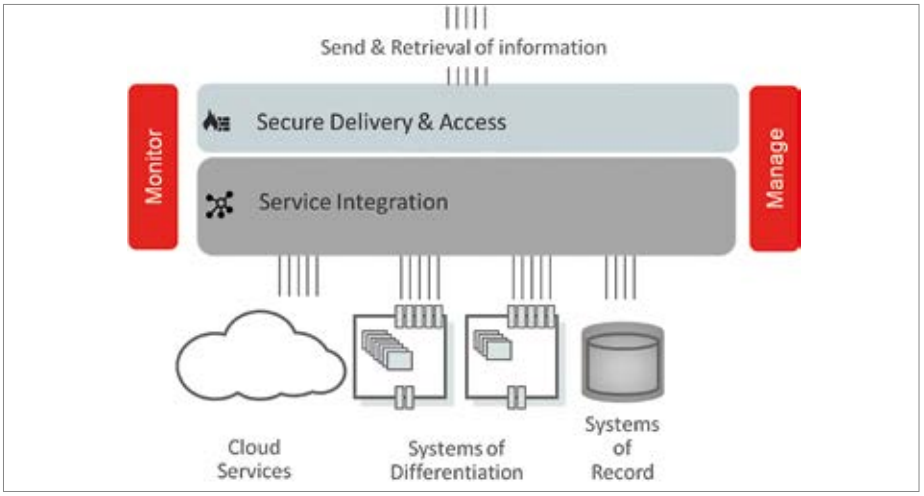

Abbildung 4: Back-End-Funktionen

Portal im Internet. Dabei sind verschiedene Verfahren möglich, etwa über Social Logins. Die App ist Offline-fähig, um Bestellungen auch ohne Netzzugriff in einer der 72 Filialen abholen zu können, ähnlich wie bei Flugtickets. Die Bestellmöglichkeit ist dabei nur einer von vielen bereitgestellten Services.

Die notwendigen Bestell- und Kundenservices bestanden bereits und werden aus dem Rechenzentrum des Kunden abgerufen oder manuell in den Filialen erbracht. Sie werden sowohl über die Kanäle „Web-Portal" und „Kiosk" als auch über andere Integrationen (Co-Shopping/Resell und InternetTV) genutzt.

Die bestehenden Services sind durch eine Service-Infrastruktur so zusammengesetzt, dass sie die passende Granularität für die Schnittstelle zur App bereitstellen. Im Back-End wird dabei auf SOAP als Schnittstelle verblieben. Damit konnten ohne Änderung die bestehenden „WS*"-Mechanismen zur Absicherung der Services beibehalten werden.

Das API-Gateway im Rechenzentrum nimmt den Request einer App entgegen. Es löst dabei drei Aufgabenstellungen: die Bereitstellung der Services als REST-Services, um diese aus einer App effizient nutzen zu können, die Absicherung der Aufrufe und die Transformation der Security-Informationen in/für nachfolgende Layer. Mit dieser Architektur hat der Kunde seine Multi-Channel Architektur umgesetzt, Silos vermieden und einen hohen Grad an Wiederverwendung erreicht.

Da sensible, personenbezogene Daten verarbeitet werden, ist die Security von Anfang an einbezogen und um möglichst flexibel zu bleiben, hat man auf die entsprechenden Standards gesetzt. Durch die konsequente Service-Orientierung, sowohl bei den Geschäftsprozessen als auch bei den Security-Funktionen lassen sich neue Kanäle und Funktionen effizient bereitstellen. Aktuell erfolgt die Erweiterung um Beacons in den Filialen und IoT-Devices, auch Wearables, um ein optionales persönliches Monitoring im Bereich „Gesundheit" zu unterstützen.

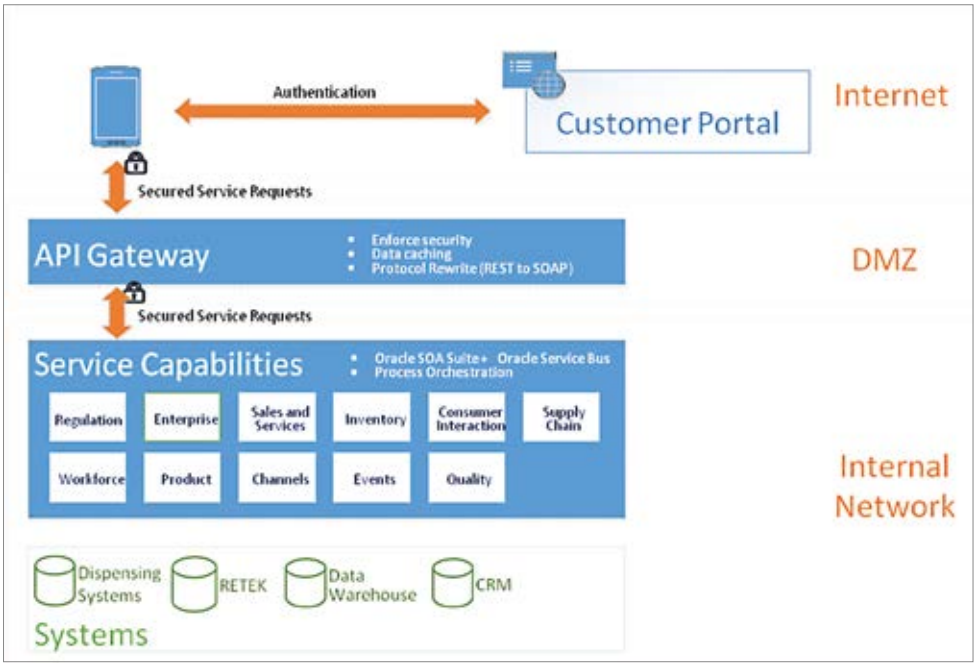

Abbildung 5: Umsetzungsbeispiel eines Oracle-Kunden

Fazit

Es ist sinnvoll, Mobile Security von Anfang an zu berücksichtigen, um nicht später Apps mit erhöhtem Aufwand umbauen oder erweitern zu müssen. Beispiele für eine Erweiterung sind aus technischer Sicht beispielsweise die Grenzen des Anmeldeverfahrens bezüglich Skalierung oder neue Technologien (wie OpenID Connect Integration). Auch von außen bestehen Anforderungen, etwa durch Auflagen von Auditoren oder allgemeine Regularien, ohne die Betrachtung von Breaches. Auch der Betriebsaspekt spielt eine Rolle, vielleicht nicht so sehr bei der ersten App, doch aber mit mehreren Apps, die vielleicht alle ein eigenes User-Management mitbringen. Daher könnten folgende Empfehlungen zum Tragen kommen:

- Ganzheitliches Konzept, keine Silos
- Sicherheit von Anfang an, etwa verschlüsselte Verbindungen bei sensitiven oder personenbezogenen Daten
- Sicherheit nach Bedarf: zum Beispiel „unprotected", „protected", „encrypted" sowie kontextabhängig
- Mindeststandards für Entwicklung
- Zentrale Kontrolle
- Audit und Regularien im Auge behalten
- Integration in Unternehmens-IT, sofern vorhanden
- Nicht alles selber aufsetzen/bauen
- Lifecycle im Auge behalten (Update, Multi-Plattform-Support)
- Gängige Protokolle und Plattformen unterstützen
- Testballons zum Start mit minimalen Kosten auf einer fertigen Plattform

Oracle unterstützt mit seinen Komponenten eine Umsetzung aller dieser Punkte On Premise, in der Cloud oder hybrid. Durch die Verwendung von standardisierten Schnittstellen und Protokollen ist dabei auch einen Mix mit Komponenten möglich, die nicht von Oracle kommen. Viele Kunden in allen Größenordnungen und durch alle Branchen nutzen diese Komponenten. Weitere Informationen unter *„http://www.oracle.com/us/technologies/security/overview/index.html".*

Michael Fischer
michael.fischer@oracle.com