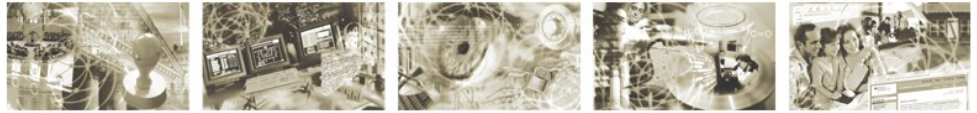




Bundesamt  
für Sicherheit in der  
Informationstechnik



# SOA-Security-Kompendium

Sicherheit in Service-orientierten Architekturen

Bundesamt für Sicherheit in der Informationstechnik  
Postfach 20 03 63  
53133 Bonn  
Tel.: +49 228 99 9582-5599  
E-Mail: [referat111@bsi.bund.de](mailto:referat111@bsi.bund.de)  
Internet: <http://www.bsi.bund.de>  
© Bundesamt für Sicherheit in der Informationstechnik 2008

## Inhaltsverzeichnis

1.	Motivation und Zielbestimmung.....	5
1.1	Motivation.....	5
1.1	Zielbestimmung.....	5
1.2	Gliederung.....	5
2.	Einführung in Service-orientierte Architekturen und Web- Services.....	6
2.1	Abstrakte Charakteristika Service-orientierter Architekturen.....	6
2.2	Ziele und Vorteile von SOA.....	9
2.3	Konkrete Umsetzungen.....	10
2.4	Beispielszenario.....	14
3.	Sicherheitsaspekte Service-orientierter Architekturen.....	15
3.1	Sicherheitsanforderungen SOA.....	15
3.2	Bedrohungspotentiale.....	20
4.	Schutzmaßnahmen.....	28
4.1	Schwächen bisheriger Sicherheitskonzepte.....	28
4.2	Non Best Practices.....	30
4.3	Standardisierte Schutzmaßnahmen.....	31
4.4	Nicht standardisierte Schutzmaßnahmen.....	35
5.	SOA-Security-Framework.....	37
5.1	Komponenten eines SOA-Security-Frameworks.....	37
5.2	Architekturansätze (Lokale Ebene).....	38
5.3	Zentrale Administration (Zentrale Ebene).....	40
5.4	Sicherheitsregeln.....	42
6.5	Ausgewählte Lösungen für das Beispielszenario.....	43
6.	Glossar.....	45
7.	Referenzen.....	46

## Abbildungsverzeichnis

Abbildung 1: Schematische Darstellung der Consumer-Provider-Beziehung.....	6
Abbildung 2: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung und unter Verwendung externer Dienste.....	7
Abbildung 3: Aufbau einer XML-basierten SOAP-Nachricht.....	11
Abbildung 4: Aufteilung einer SOAP-Nachricht in Header und Body.....	11
Abbildung 5: Ablauf eines Dienstaufwurfes unter Verwendung einer zentralen Registry.....	13
Abbildung 6: WebService-Protokollstack für die unterschiedlichen Phasen der Servicesuche bzw. -nutzung.....	13
Abbildung 7: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung und unter Verwendung externer Dienste.....	14
Abbildung 8: Sukzessives Bearbeiten einer SOAP-Nachricht durch mehrere Dienste.....	17
Abbildung 9: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung und unter Verwendung externer Dienste.....	24
Abbildung 10: Vergleich der Sicherheitskontexte bei TLS/SSL und SOA.....	28
Abbildung 11: Übersicht der wichtigsten Standards im Web-Service-Kontext.....	31
Abbildung 12: Einbettung von SAML-Informationen in eine SOAP-Nachricht.....	33
Abbildung 13: Ebenen eines SOA-spezifischen Security Framework.....	38
Abbildung 14: Sicherheit als eigenständiger Service.....	39
Abbildung 15: Sicherheit als integraler Bestandteil der Infrastruktur einer SOA.....	39
Abbildung 16: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung unter Verwendung externer Dienste.....	43

# 1. Motivation und Zielbestimmung

## 1.1 Motivation

Service-orientierte Architekturen (SOA) beschreiben einen allgemeinen Ansatz zur Realisierung komplexer IT-Systeme und der Abbildung von Geschäftsprozessen in solchen Systemen. Die Konzepte von SOA haben den Anspruch, viele bestehende Probleme bei der Integration und Interaktion unterschiedlicher Teilsysteme zu lösen. Zumindest in Teilen ist dies auch in der Praxis bereits umgesetzt.

SOA ist einer der modernsten Trends der IT und aufgrund seiner Geschäftsprozess-nahen Ausrichtung insbesondere im Bereich des Management sehr beliebt. Sehr wenig Beachtung wird jedoch den Sicherheitsaspekten von SOA geschenkt – insbesondere in den Teilen, wo solche sicherheitsrelevanten Anforderungen auftreten, die in konventionellen IT-Systemen bislang nicht beachtet wurden oder in dieser Form nicht relevant waren. Schließlich befinden sich im Hintergrund von SOAs auch immer entsprechende Geschäftsprozesse, die durchaus kritisch und daher schutzbedürftig sein können. Neben der Sicherheit von einzelnen Service-Anfragen (Vertraulichkeit, Authentizität, usw.) sind auch Aspekte wie Transaktionssicherheit von Bedeutung. Insbesondere wenn eine Service-orientierte Architektur nicht nur innerhalb einer Institution verwendet, sondern auch nach außen hin geöffnet wird, kommen zusätzliche Sicherheitsanforderungen hinzu. Als Beispiel sei etwa der Schutz gegen Denial-of-Service-Angriffe genannt.

Derzeit entworfene IT-Systeme auf Basis einer Service-orientierten Architektur werden meist zunächst unter rein funktionalen Gesichtspunkten entworfen. Sicherheitsfunktionalität wird i.d.R. nachträglich und beschränkt für die einzelnen Komponenten entwickelt. Als Folge dessen wird zumeist ein ganzheitliches Sicherheitskonzept verfehlt und viele SOA-spezifische Sicherheitsanforderungen bleiben unerfüllt. Es fehlt sowohl an einem angemessenen Sicherheitsbewusstsein als auch an einem ganzheitlichen Sicherheitskonzept und BestPractice-Ansätzen für diese neue Technologie. Dieser Umstand ist oftmals sogar Ursache für das Scheitern großer und vielversprechender IT-Vorhaben.

Damit IT-Systeme gemäß dem SOA-Paradigma in Behörden und anderen Institutionen unter Einhaltung der jeweiligen Sicherheitsanforderungen realisiert und betrieben werden können, ist es erforderlich, ein angemessenes Sicherheitsbewusstsein zu schaffen und geeignete Lösungsmöglichkeiten aufzuzeigen. Ein konkreter Bedarf liegt insbesondere in Behörden des Bundes vor, da verschiedene Großprojekte auf der besagten Architektur und den betroffenen Technologien basieren.

## 1.1 Zielbestimmung

Mit dem “Kompodium SOA-Security“ stellt das BSI eine Studie zur Sicherheit in Service-orientierten Architekturen (SOA) zur Verfügung. Hierdurch werden Entwicklern, Architekten von IT-Systemen und Entscheidern fundierte Grundlagen und Handlungsrichtlinien bereitgestellt, um sichere SOAs zu realisieren. Da es sich bei SOAs um eine neue Technologie handelt, wird hierdurch der Sicherheit kritischer Geschäftsprozesse in Behörden und Unternehmen Vorschub geleistet.

## 1.2 Gliederung

Ziel dieses Kompodiums ist es, zunächst eine Einführung in die grundlegenden Aspekte von Service-orientierten Architekturen zu geben. Hierzu wird u.a ein Beispielszenario eingeführt.

Der zweite Abschnitt dieses Kompodiums geht vertiefend auf die Sicherheitsaspekte von SOA und Web-Services ein. Es erfolgt eine Einführung in die zur Realisierung von SOAs verwendeten Technologien, wobei insbesondere WebServices relevant sind. Zusätzlich werden Interoperabilitäts- bzw. Kompatibilitätsaspekte erläutert und entsprechende Standardisierungsprojekte vorgestellt.

Der dritte Abschnitt dieses Dokuments stellt verschiedene BestPractice-Ansätze zur Realisierung von Schutzmaßnahmen im SOA-Umfeld vor. Unter anderem wird dabei ein Konzept für ein SOA-Security Framework beschrieben, welches die Umsetzung eines ganzheitlichen Sicherheitskonzepts für SOAs ermöglicht.

Im letzten Abschnitt dieses Dokuments werden dem Leser eine Sammlung von Referenzen sowie ein Glossar und weitere hilfreiche Ergänzungen präsentiert.

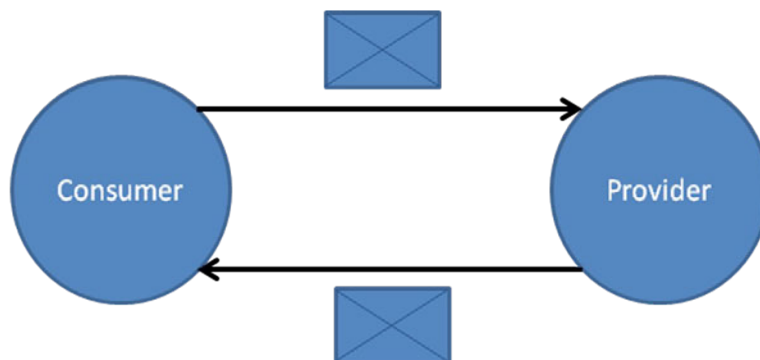
## 2. Einführung in Service-orientierte Architekturen und Web-Services

Geschäftsprozesse innerhalb einer SOA stellen sich als komplexe Gebilde dar, bei denen eine Vielzahl von Komponenten, Systemen und Applikationen involviert sind. In vielen Fällen erstrecken sich solche Prozesse sogar über Unternehmensgrenzen hinaus. Ziel dieses Moduls ist es, eine Einführung in SOA-Architekturen zu vermitteln und diese anhand typischer Szenarien im Enterprise-Bereich zu verdeutlichen. Neben grundlegenden Prinzipien wird insbesondere auf die Realisierung von SOA mittels Web-Services eingegangen.

### 2.1 Abstrakte Charakteristika Service-orientierter Architekturen

Service-orientierte Architekturen (SOA) sind in aller Munde. Zahlreiche Softwarehäuser bieten derzeit Infrastrukturlösungen für SOA an. Dabei ist das Konzept keineswegs neu und wurde bereits teilweise im Rahmen von Enterprise-Application-Integration (EAI) Projekten verfolgt.

Die Architektur moderner IT-Systeme orientiert sich immer stärker an den Geschäftsprozessen, die diese Systeme unterstützen bzw. abbilden. Aufgrund von Aspekten wie Modularität und Wiederverwendbarkeit ist es nicht sinnvoll, die dazu erforderlichen Applikations-Logiken monolithisch im jeweiligen Programm zu realisieren. Vielmehr bietet sich an, Geschäftsprozesse durch eine Aneinanderreihung von Aufrufen feingranularer und voneinander unabhängiger Services zu modellieren ("Komposition von Services").

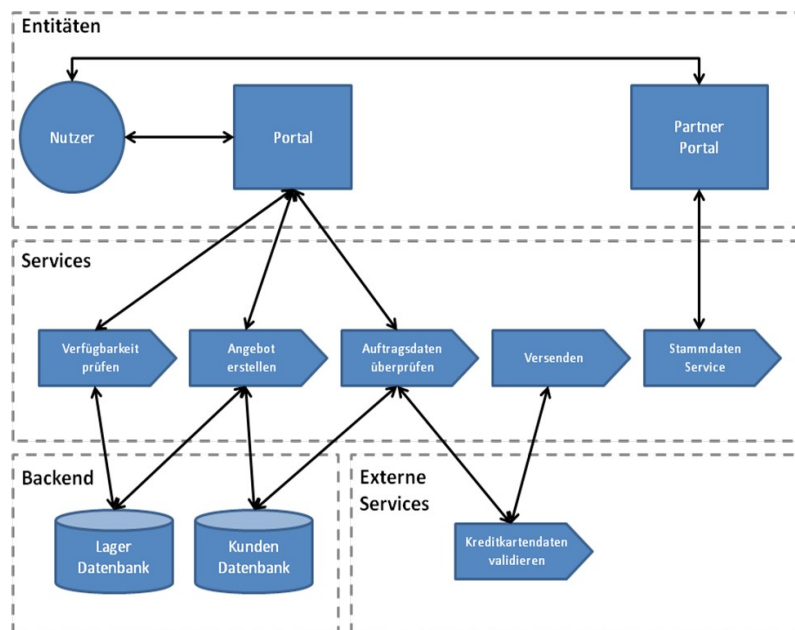


**Abbildung 1: Schematische Darstellung der Consumer-Provider-Beziehung**

Die Idee hinter Service-orientierten Architekturen ist folgende: Funktionalitäten und Aufgaben werden nicht wie bei herkömmlichen Systemen innerhalb einer einzelnen Anwendung, sondern als lose gekoppelte, unabhängige, austauschbare Dienste über standardisierte Schnittstellen von einem Service-Provider angeboten. Der Service-Consumer erhält als Antwort des Service-Requests einen Service-Response.

Der generische Lebenszyklus eines einfachen Dienstes stellt sich zusammenfassend wie folgt dar:

1. **Publish / Register:** Ein Serviceanbieter publiziert einen Dienst und registriert ihn in einem öffentlich zugänglichen Verzeichnis (Service-Repository).
2. **Find:** Ein Konsument greift mit einer Suchanfrage auf das Service-Repository zu, um einen geeigneten Dienst aufzufinden.
3. **Bind:** Der Konsument erhält vom Verzeichnisdienst die Adresse, unter welcher der Dienst aufgerufen werden kann, sowie die dazu erforderlichen Parameter, um den Dienst korrekt anzusprechen.
4. **Execute:** Der Konsument führt den Serviceaufruf unter der zuvor erhaltenen Adresse und unter entsprechender Wahl der Eingangsparameter aus. Als Ergebnis liefert der Dienst die Ergebnisparameter.



**Abbildung 2: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung und unter Verwendung externer Dienste**

Insbesondere die Angebote von Behörden auf Bundes-, Länder- oder kommunaler Ebene entsprechen genau dem Prinzip einer Dienstleistung, d.h. eines Services. Die Möglichkeit, diese Dienste miteinander zu verbinden und durch Kombination zu neuen Angeboten zu kombinieren, verspricht ein immenses Potential an neuen Möglichkeiten wie z.B. die einfache Umsetzung verschiedenster Mehrwertdienste oder Kosteneinsparung durch Synergieeffekte. Auch in anderen Institutionen und Unternehmen der Wirtschaft gilt dieser Grundsatz. Da die meisten IT-Systeme jedoch Individuallösungen mit proprietären Schnittstellen sind, ist die Realität von der Idealvorstellung weit entfernt. Als Folge dessen resultieren immense Aufwände bei der Integration verschiedener Systeme, häufige Neuentwicklungen und hohe Kosten. Die Lösung dieser Problematik besteht in einer Systemarchitektur, bei der Dienste im Fokus der Betrachtungen stehen.

Konkrete Anwendungsfälle von SOA finden sich bereits heute u.a. auch im E-Government, dem Finanzwesen oder in Service-Zentren. Insbesondere Global Player der freien Wirtschaft haben schon früh die Signifikanz und das Potential dieses neuen Architekturprinzips erkannt. Aber auch in anderen Anwendungsgebieten stellen Service-orientierte Architekturen einen vielversprechenden Ansatz dar. So wird der SOA-Ansatz u.a. im Bereich des Grid-Computing (vgl. OpenGrid) mit großem Interesse verfolgt und bekommt aus diesem Forschungsbereich auch eine Vielzahl von Anregungen und Weiterentwicklungen. Auch für die Verarbeitung großer Mengen von Geo- oder Infrastrukturdaten bringt SOA viele Vorteile mit sich.

Anschauliche praktische Beispiele für Services in einer dienstorientierten Umgebung sind z.B. in einem Bestellprozess zu finden. Ein Bestellvorgang eines Kunden bei einem Buchhändler kann durch das Zusammenspiel einzelner Services abgebildet werden. So sorgt der Service "Verfügbarkeit prüfen" für die Ermittlung der am Lager befindlichen Mengen der Ware. Angefragt über ein standardisiertes Format, greift der Service auf ein Warenwirtschaftssystem zu und antwortet dem Web-Portal mit dem Service-Response. Wie in diesem Beispiel können Services auch Schnittstellen zu weiteren Systemen anbieten. Für den Service-Consumer (bspw. ein Web-Portal) bleibt die Logik sowie die detaillierte Funktionalität der in Teilprozessen aufgerufenen Services verborgen. Auch unternehmensübergreifende Services sind z.B. bei der Kredit- und Bonitätsprüfung durch ein externes Unternehmen möglich. Soll ein solcher Service von einer Kreditanstalt genutzt werden, so wird der Service in den internen Bestellprozess "hineinmodelliert" und ist dadurch Teil des internen Vorgangs.

SOA ist somit grundsätzlich ein Managementkonzept, deren Infrastruktur an den Anforderungen des Geschäftsprozesses ausgerichtet wird und die strikte Trennung von Prozesslogik und Prozessfunktionen fordert. Einzelne Services werden wie Programmmodule bedarfsgerecht zusammengefügt. Der Geschäftsprozess ergibt sich somit durch die Orchestrierung von lose gekoppelten Services. Das eingesetzte Systemarchitekturkonzept muss daher in der Lage sein, Funktionen in Form von geeigneten Services bereitzustellen. Bei Systemarchitekturen nach dem SOA-Konzept stehen nicht Softwarekomponenten, Protokolle, Standards oder technische Implementierungsdetails im Mittelpunkt. Vielmehr orientiert sich dieser Ansatz an den abzubildenden Geschäftsprozessen und deren fachlicher Details, die in Form von Diensten modelliert werden.

In der Literatur gibt es keine eindeutige Definition von SOA. Dennoch lassen sich gemeinsame Schlüsselmerkmale benennen, die nach verbreiteter Meinung dienstorientierte Architekturen definieren.

#### *Standardisierte Service-Schnittstellen*

Eine der fundamentalen Forderungen ist die nach standardisierten Schnittstellen und Beschreibungen. Beschrieben werden muss, wie der Service genutzt werden kann, welche Daten-(Typen) benötigt werden und wie bestimmte Richtlinien verwendet werden können.

#### *Lose Kopplung*

Services sollen lose gekoppelt zu einem Prozess zusammengefügt werden können. Dies setzt ein Minimum an Abhängigkeiten zwischen einzelnen Services voraus. Es gilt der Grundsatz, gerade soviel Abhängigkeiten zu schaffen, dass eine Interoperabilität (auch Austauschbarkeit) noch gewährleistet ist.

#### *Funktionsabstraktion*

Um eine lose Kopplung zu ermöglichen, sollen Services nach außen von Details der konkreten Funktionalität und Implementierung abstrahieren. Lediglich die nach außen angebotenen und beschriebenen Funktionen werden gekapselt angeboten.

#### *Wiederverwendbarkeit*

Ein Grundgedanke ist die Wiederverwendbarkeit von Services in einem späteren Prozessschritt, von anderen Teilnehmern oder sogar für weitere Verwendungszwecke. Diese Forderung muss bereits bei der Entwicklung Berücksichtigung finden.

#### *Service-Autonomie*

Ein Service soll in der Lage sein, autark zu funktionieren. Wenn ein Service alle benötigte Logik, Ressourcen und Umgebung selbst verwaltet bzw. von externen Services unabhängig ist, spricht man von Service-Autonomie.

#### *Vorratsdatenfreiheit des Services (Statelessness)*

Services folgen dem Dienstleistungsgedanken, dass eine definierte Leistung erbracht wird. Diese kann auch die Vorratsdatenspeicherung beinhalten, allerdings nur, wenn explizit verlangt. Sonstige Services

halten keine persistenten Daten und müssen dadurch zwischen zwei Serviceaufrufen keine Statusverwaltung durchführen.

#### *Auffindbarkeit des Services*

Um einen Service zu nutzen, muss dieser auffindbar sein. In der Regel erfolgt dies über Service-Repositories. Ein Repository steht allen Konsumenten und Providern zur Verfügung und beinhaltet Beschreibungen von Serviceschnittstellen und -implementierungen. Es speichert alle Informationen, welche ein Consumer benötigt, um einen Service aufzurufen.

#### *Orchestrierbarkeit der Services*

Orchestrierung wird das Vorgehen genannt, einzelne Services aus fachlichen Gesichtspunkten heraus zu größeren Einheiten, den Geschäftsprozessen, zu verbinden. Da das Ziel der SOA die Abbildung des fachlichen Geschäftsprozesses ist, ist die Orchestrierbarkeit eine weitere Forderung an SOA-Services. Services sollen in der Lage sein, einzelne Aufgaben in einem Gesamtprozess effektiv zu erfüllen. Die Unabhängigkeit von der Komplexität und den Ausmaßen des jeweiligen Prozesses stellt dabei eine der zentralen Anforderungen dar.

In der Literatur lassen sich zudem zahlreiche weitere Forderungen und Definitionen finden. Die dargestellten Kriterien sollen an dieser Stelle jedoch ausreichen, um SOA und Services praktikabel zu definieren.

## **2.2 Ziele und Vorteile von SOA**

Primäre Ziele beim Einsatz von bzw. der Migration zu SOA Umgebungen sind die erhöhte Flexibilität von Geschäftsprozessen und die damit einhergehende Kostenersparnis. Weiterhin trägt die einfachere Einbindung bestehender (Alt-)Systeme maßgeblich zur Kostensenkung bei. Kernaufgaben des Unternehmens, wie etwa in der Umsetzung kundengerechter Prozesse oder die Anpassung an veränderte Marktbedingungen, können durch die strikte Trennung zwischen dynamischer Geschäftsprozesslogik und statischen Geschäftsfunktionen besser erfüllt werden. So können neue Zulieferer durch standardisierte Schnittstellen effizient eingebunden und Workflows für Mitarbeiter, Kunden und Partner schneller optimiert werden. Weiterhin ist der Vorteil der Wiederverwendbarkeit einzelner Services in diesem Kontext nicht zu vernachlässigen. Einsparungen von Systemen, Lizenzgebühren, Personal, etc. sind durch gemeinsame Nutzung oder gar Outsourcing bestimmter Funktionen möglich.

Enterprise Application Integration (EAI) ist ein Konzept zur unternehmensweiten Integration der Geschäftsfunktionen entlang der Wertschöpfungskette, die über verschiedene Applikationen auf unterschiedlichen Plattformen verteilt sind und die im Sinne der Daten- und Geschäftsprozessintegration verbunden werden können. Auf das Wesentliche reduziert ist EAI ein rein technischer Ansatz zur Integration von Anwendungssystemen. Die Hauptaufgabe besteht in der heterogenen Integration der verschiedenen Altsysteme. Hierzu haben die verschiedenen Hersteller hauptsächlich proprietäre Produktsuiten entwickelt, deren Einsatz ein sehr hohes Spezialistenwissen erfordert.

Gegenüber EAI ist SOA in hohem Maße standardisiert und bietet auch die technischen Möglichkeiten zur Abbildung eines fachlichen Geschäftsprozesses. Auch die Einbindung bestehender Systeme im Sinne von EAI lässt sich mit einem dienstorientierten Ansatz realisieren. Gerade bei der Integration von Anwendungen aus Altsystemen (sog. Legacy-Anwendungen) über Services lässt sich eine Flexibilisierung von monolithischen Systemen erreichen. Auf diese Weise kann auch eine schrittweise Migration großer Legacy Systeme erfolgen. Unternehmen benötigen nicht den gefürchteten Big-Bang, sondern migrieren Schritt für Schritt.

Auch die Integration zentraler Dienste spielt eine wichtige Rolle beim Einsatz von SOA. Mit der Service-Orientierung wird es möglich, Dienste wie Identity Management oder auch Public Key Infrastruktur (PKI) auf standardisierte Weise systemweit zur Verfügung zu stellen. Die Implementierung

von Authentisierungs- und Autorisierungsdiensten als Service ermöglichen Anwendungen oder anderen Services den Zugriff auf Usercredentials, um somit Single Sign-On-Methoden zu realisieren. Lokalisierung und Validierung von Zertifikaten kann in SOA-Umgebungen von zentralen Zertifikatsmanagement-Diensten übernommen werden. Hierbei tritt der Vorteil der Kapselung der Servicekomplexität in besonderem Maße hervor. Der Service-Consumer muss ausschließlich die Service-Schnittstelle kennen, er muss keine Details der dahinterliegenden Implementierung wissen.

Bei konsequenter Umsetzung bringen Service-orientierte Architekturen eine Reihe von Vorteilen mit sich, von denen zusammenfassend folgende genannt seien:

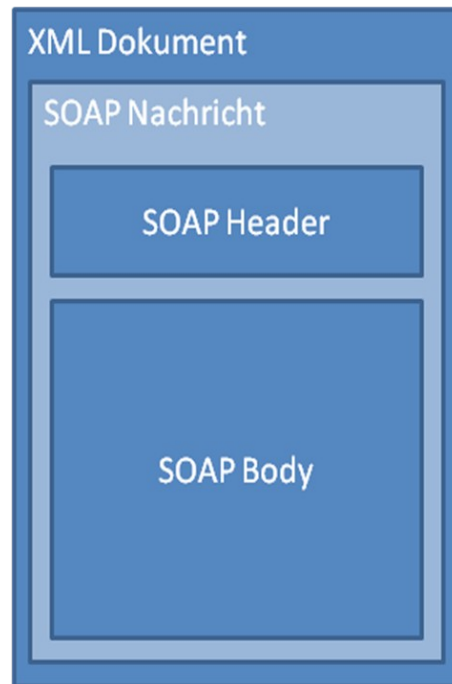
- Flexibilität (z.B. durch einfache Austauschbarkeit von Services),
- Wiederverwendbarkeit von Services (höhere Produktivität) und Vermeidung monolithischer Systeme (Reduktion der Komplexität),
- Hohe Dynamik: schnelle Anpassung an neue Gegebenheiten und verbesserte Möglichkeiten zur Restrukturierung, da die Softwarekomponente frei von der Geschäftslogik ist und diese statt dessen in der Orchestrierung der Services zu einem Geschäftsprozess enthalten ist,
- Kostengünstige Integration zentralisierter Dienste wie PKIs, Single-Sign-On oder Identity Management,
- Verbesserte Nachhaltigkeit der IT einer Institution oder eines Unternehmens,
- Verteilte Architektur (auch Redundanz) und
- Optimale Anpassung an den abzubildenden Geschäftsprozessen – somit auch leicht zugänglich für Management statt nur für Techniker.

Durch die neuen Möglichkeiten und den damit verbundenen Technologien ergeben sich aber auch eine Reihe möglicher Nachteile. In erster Linie ist diesbezüglich die Konfrontation mit neuen sicherheitsspezifischen Herausforderungen, z.B. mit Blick auf Federation oder Orchestrierung, zu nennen. Auch kann die Umsetzung SOA-basierter Systeme durchaus mit hohen Migrationskosten und einem weitreichenden Zeithorizont verbunden sein.

Insgesamt werden SOAs der Tatsache gerecht, dass Geschäftsprozesse eines Unternehmens und die zugehörigen IT-Landschaften immer stärker miteinander verschlungen sind. Im Gegensatz zu früheren Ansätzen stehen bei SOA die fachlichen Aspekte und insbesondere die Geschäftsprozesse im Mittelpunkt.

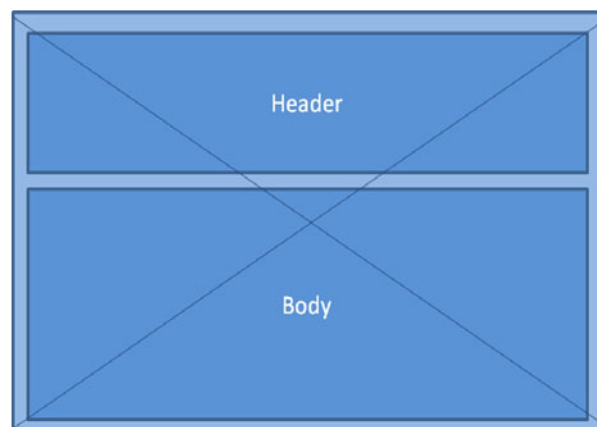
## 2.3 Konkrete Umsetzungen

Grundsätzlich ist SOA ein Architekturkonzept, welches keine konkrete technische Realisierung oder bestimmte Methoden vorschreibt. So kann eine SOA mit CORBA, Enterprise Java Beans, Web-Services oder anderen Technologien umgesetzt werden. Es haben sich allerdings Web-Services zur Realisierung durchgesetzt. Im Folgenden soll daher ausschließlich darauf fokussiert werden.



**Abbildung 3: Aufbau einer XML-basierten SOAP-Nachricht**

Ein Web-Service ist eine auf XML (eXtensible Markup Language) und SOAP (Simple Object Access Protocol) basierende Anwendung, die definierte Methoden über eine standardisierte Schnittstelle anbietet und über eine URI (Uniform Resource Identifier) eindeutig identifizierbar macht. Als Kommunikationsprotokoll dient SOAP (W3C Recommendation), mit dem Daten zwischen Systemen ausgetauscht werden können. SOAP verwendet XML zur Datenrepräsentation sowie in den meisten Fällen HTTP/TCP für den Transport.



**Abbildung 4: Aufteilung einer SOAP-Nachricht in Header und Body**

Im bildlichen Sinne stellt SOAP einen offenen Umschlag (*Envelope*) dar. Der Inhalt unterteilt sich in einen Nachrichtenkopf (Header) und einen Nachrichtenkörper (Body). Im Header werden Meta-informationen zum Datentransport sowie Informationen zu Absender und Empfänger der SOAP-Nachricht gespeichert. Im Body können als Inhaltsdaten der Nachricht Methodenaufrufe bzw. deren Antwort eingefügt werden.

Web-Services finden immer größere Verbreitung. Insbesondere im Kontext Service-orientierter Architekturen sind die entsprechenden Protokolle sehr attraktiv und bilden die am häufigsten verbreitete Realisierungstechnologie für SOAs. Web-Services können somit als State-of-the-Art zur Realisierung von SOA bezeichnet werden.

Beispiel:

Die Suchanfrage „SOA Einsteiger“ eines Kunden über das Web-Portal wird als Web-Service an das Warenwirtschaftssystem eines Buchhändlers geschickt. In diesem Fall wird die Methode “ItemInStock” mit den Parametern “SOA Einsteiger” des Web-Services aufgerufen. Dieser exemplarische SOAP-Request ist gleichzeitig ein Beispiel dafür, dass der SOAP-Header optional ist.

SOAP Request:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
  <soap:Body>
    <ItemInStock xmlns="http://www.mySearchService.de/soap">
      SOA Einsteiger
    </ItemInStock>
  </soap:Body>
</soap:Envelope>
```

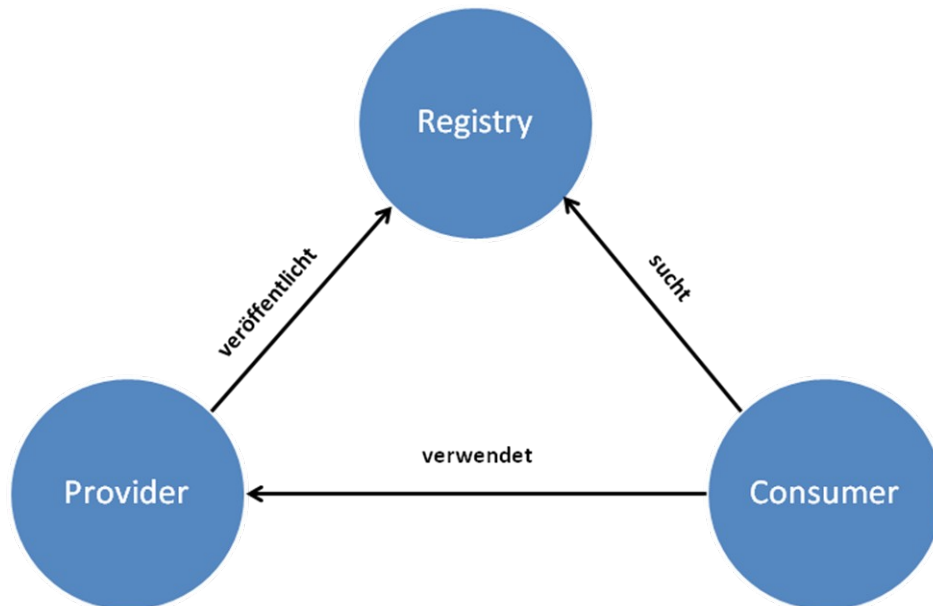
SOAP Response:

Der Search Web-Services antwortet mit zwei gefundenen Items im SOAP-Body.

```
<?xml version="1.0"?>
<soap:Envelope xmlns="http://www.w3.org/2001/12/soap-envelope">
  <soap:Header>
    <ReqId xmlns="http://www.mySearchService.de/soap">3462346123574</ReqId>
  </soap:Header>
  <soap:Body>
    <ItemResponse xmlns="http://www.mySearchService.de/soap">
      <title value="SOA Einsteiger">
        <Item value="1">SOA für Einsteiger</Item>
        <Item value="2">SOA Grundlagen – Serviceorientierung für Einsteiger</Item>
      </title>
    </ItemResponse>
  </soap:Body>
</soap:Envelope>
```

Populäre Beispiele für Web-Services sind z.B. die von Google oder Amazon, welche über diese Dienste die selben Funktionalitäten anbieten wie über ihr Webportal.

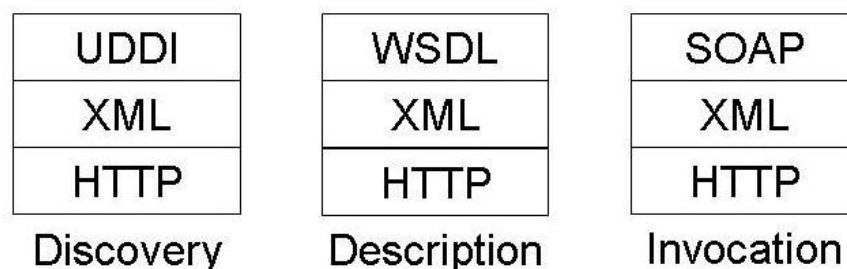
Neben dem (Web-) Service-Requests des Service-Consumers und dem (Web-) Service-Response des Service-Providers seien an dieser Stelle noch weitere Elemente der Web-Service-Architektur genannt.



**Abbildung 5: Ablauf eines Dienstauftrufes unter Verwendung einer zentralen Registry**

Wie oben erwähnt, müssen Web-Services beschrieben, veröffentlicht und auffindbar gemacht werden. Dafür kann der Service Provider den Web-Service bei einem Verzeichnisdienst (Service Repository) bekannt machen und den Service selbst sowie die verfügbaren Methoden beschreiben. Der Service Consumer kann den Service anschließend über den Verzeichnisdienst finden und mit Hilfe der Beschreibung aufrufen.

Als Standards haben sich dafür UDDI (Universal Description, Discovery and Integration) und WSDL (Web Services Description Language) etabliert. UDDI ist ein Standard für Verzeichnisdienste und bietet Möglichkeiten zum Einstellen, Suchen und Auffinden von Web-Services. Vereinfacht entspricht die Funktionsweise von UDDI der eines Telefonbuchs. Das Suchen eines Web-Services erfolgt über eine SOAP Schnittstelle. Die Antwort enthält einen Verweis auf ein Dokument im WSDL Format. WSDL ist der Standard zur Beschreibung von Web-Services. Neben Funktion, Syntax und Struktur werden darin auch zur Verfügung gestellte Methoden (bspw. „Verfügbarkeit prüfen“ aus dem Beispielszenario) beschrieben. Auf Basis dieser Informationen ist der Service Consumer in der Lage, den Service zu nutzen.



**Abbildung 6: WebService-Protokollstack für die unterschiedlichen Phasen der Servicesuche bzw. -nutzung**

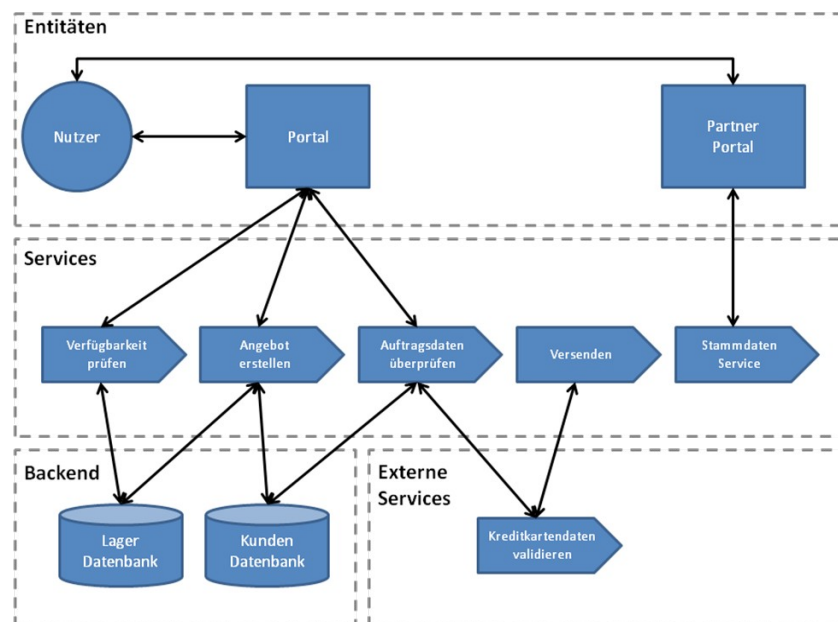
Ein weiteres Konzept, welches im Rahmen von SOA und Web-Services von zentraler Bedeutung ist, ist der Enterprise Service Bus (ESB). Der Begriff des ESB beschreibt im Grunde eine Kommunikationsinfrastruktur, welche den nachrichtenbasierten Informationsaustausch der Services steuert. Der Begriff ESB ist nicht eindeutig definiert. Oftmals bietet ein ESB ein zentrales Service Repository sowie Werkzeuge zur Datenkonvertierung.

Grundsätzlich ist die Integration und Migration von bestehenden Systemen eine Herausforderung. Konzepte wie der ESB fokussieren bei der Integration vornehmlich auf Applikationen wie CMS (Content Management Systems), ERP (Enterprise Resource Planning) oder Document Management Systems. Zudem gilt es Anwendungen wie Identity Management Systeme, Metaverzeichnisse und PKI in der SOA-Umgebung verfügbar zu machen. Benutzerdaten, Rechte, Rollen und Zertifikate können systemweit für Services zur Verfügung stehen. Für ein Single Sign-On (SSO) an unterschiedlichen Services wird ein Mapping der Credentials benötigt, welches Daten aus bestehenden Systemen wie LDAP oder Active Directory verwendet und Legacy Systeme mit den geforderten Formaten und Anmeldeinformationen bedient. Hintergrund ist, dass eine Entität meist unterschiedliche Credentials (Username/Passwort, Token, X.509, etc.) zur Anmeldung an unterschiedlichen Systemen, Ressourcen und Services benötigt. Über ein zentrales Mapping können diese Informationen auf den vorliegenden Daten abgebildet werden. Die lokale Anmeldung und die damit verbundene Autorisierung kann mit diesem "Wissen" automatisch erfolgen, und SSO Mechanismen können umgesetzt werden.

Eine vorhandene PKI kann durch die Nutzung eines Zertifikats Management Services abgebildet werden und somit Lokalisierungs- und Validitätsprüfungen jedem Service zur Verfügung stellen. So wird ein Zertifikatsmanagement in verteilten Umgebungen möglich, welches über standardisierte Schnittstellen einen Zertifikatsservice zur Verfügung stellt. Die Anforderung oder Prüfung eines benötigten Zertifikats erfolgt als Service Aufruf und ermöglicht so die sichere Kommunikation (Verschlüsselung, Signatur) oder Authentisierung (bspw. via SAML) zwischen Services und Systemen.

## 2.4 Beispielszenario

Nachdem der Begriff SOA sowie konkrete Umsetzungen in den vorangegangenen Abschnitten dargestellt wurde, wird ein Beispielszenario erläutert. Anhand dessen sollen im Folgenden Sicherheits- und Administrations-Aspekte illustriert werden.



**Abbildung 7: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung und unter Verwendung externer Dienste**

Als Beispielszenario soll der Bestellprozess bei einem Buchhändler dienen. Ein Kunde meldet sich am Web-Portal des Buchhändlers an und lässt sich die gewünschten Artikel auflisten. Der Service "Verfügbarkeit prüfen" wird angesprochen und liefert die am Lager befindlichen Mengeninformationen an das Web-Portal. Der Kunde gibt die Bestellung der gewünschten Artikel auf. Das System erstellt da-

raufhin ein Angebot. Der Service "Angebot erstellen" liefert unter Berücksichtigung des Kundenstatus (bspw. Rabatte, Konditionen, etc.) den kundenindividuellen Preis der Bestellpositionen. Bei Zahlung – hier via Kreditkarte – wird ein externer Service vom Kartenprovider aufgerufen und die Bonität des Kunden anhand der Kreditkartennummer geprüft. Im Positivfall wird die Zahlung akzeptiert und die Ware zum Versand gegeben. Der Service "Versenden" übermittelt dabei alle benötigten Informationen inkl. der Liefer- und Rechnungsadresse des Kunden an das Versandsystem.

Zusätzlich lässt sich das Szenario unternehmensübergreifend erweitern. So werden dem Kunden nach erfolgter Bestellung weitere produktnahe Artikel angeboten. Wurde das Buch "Umstieg auf SOA" erworben, so könnte ein Partnerunternehmen entsprechende Hardwareartikel anbieten. Ist der Kunde interessiert, so erfolgt die Weiterleitung auf den Internetshop eines Elektronikhändlers. Hierbei erfolgt ein automatisiertes Login (Single Sign-On) auf die Plattform des Elektronikhändlers, ohne dass dieser den Kunden und die damit verbundenen Stammdaten selbst vorhalten muss bzw. im Vorfeld kennen muss.

### **3. Sicherheitsaspekte Service-orientierter Architekturen**

#### **3.1 Sicherheitsanforderungen SOA**

Dieser Abschnitt befasst sich mit den Sicherheitsanforderungen von Service-orientierten Architekturen. Insbesondere wenn Prozesse einer Geschäftslogik nach außen hin geöffnet werden, werden besondere Sicherheitsanforderungen an die SOA-Implementierung gestellt. Doch auch bei Geschäftsprozessen innerhalb einer Organisation gibt es sicherheitsrelevante Anforderungen, welche in dieser Form in konventionellen IT-Systemen nicht auftreten. Neben technischen oder kaufmännischen Daten sind hinsichtlich der zu schützenden Werte insbesondere Aspekte wie Reputation des Betreibers einer SOA oder Kundenvertrauen zu nennen.

Die Sicherheitsanforderungen an SOA Infrastrukturen und den darin stattfindenden sicheren und vertrauenswürdigen Geschäftstransaktionen über multiple Parteien sind häufig hoch und ergeben sich aus der Summe herkömmlicher Herausforderungen unter Berücksichtigung SOA-spezifischer Besonderheiten. Dies betrifft neben der Problematik verteilter Strukturen, realisiert durch offene Formate, auch Administrations- und Beherrschbarkeitsaspekte. Die Realisierung eines Geschäftsprozesses über eine Vielzahl lose gekoppelter Services, erfordert beispielsweise mehr Authentisierungsvorgänge, eine jederzeit gewährleistete Vertraulichkeit sowie eine höhere Integrität als in einem monolithischen System. Ein Prozess-Design über Unternehmensgrenzen hinweg und zwischen quasi-anonymen Teilnehmern verstärkt diese Sicherheitsanforderungen an Services und Lösungen.

Im folgenden Abschnitt sollen daher grundsätzliche Sicherheitsanforderungen vorgestellt und deren SOA-spezifischen Erweiterungen diskutiert werden.

#### **Authentifizierung**

An dieser Stelle seien die häufig verwendeten Begriffe Credential und Security Token erläutert. Ein Credential ist ein Identifikations- und Berechtigungsnachweis, mit dem ein Subjekt – d.h. ein Anwender oder auch ein Service – nachweisen kann, dass er auf bestimmte Informationen oder Ressourcen zugreifen oder bestimmte Aktionen ausführen darf. Ein Security Token stellt ein Gerät oder eine Hardwarekomponente dar, auf dem Identifikations- und Berechtigungsnachweise gespeichert sind.

Durch die Authentifizierung sollen Entitäten in einem System identifiziert werden. Es muss gewährleistet werden, dass die vorgelegten Credentials korrekt sind und es muss verhindert werden, dass ein

Angreifer eine Identität vortäuschen kann. Häufig erfolgt die Identifikation mit Hilfe von Benutzernamen und einem dazugehörigen Passwort.

Da die Unsicherheit von Passwörtern hinlänglich bekannt ist, existieren alternative Credentials, die eine höhere Sicherheit aufweisen (z.B. biometrische Merkmale) und zusätzliche Sicherheitsanforderungen unterstützen können (z.B. Zertifikate). Insbesondere Zertifikate innerhalb einer Public Key Infrastructure (PKI) sind in der Lage, neben der sicheren Authentisierung weitere Anforderungen, wie Integrität und Vertraulichkeit zu erfüllen.

Während die Authentizität in einem herkömmlichen System relativ einfach beherrscht werden kann, ändert sich dies in einer verteilten Infrastruktur. Im Gegensatz zu einem monolithischen System, in dem die Authentizität meist über ein zentrales Login an einem definierten Punkt sichergestellt wird, ist man in SOA mit einer Vielzahl von Authentisierungen verschiedener Services konfrontiert. Ein kritischer Geschäftsprozess erfordert die Authentizität sämtlicher Service-Aufrufe, bspw. innerhalb einer Supply-Chain. In einer losen Kopplung sind zudem Fälle denkbar, in dem sich Service Consumer gegenüber bislang unbekanntem Service Providern authentisieren müssen. Authentisierungslösungen müssen in der Lage sein, benötigte Federation Konzepte technisch abzubilden und zudem eine Interoperabilität von Authentisierungsangaben und -formaten zu gewährleisten. Dies betrifft nicht nur die Inter-Service Kommunikation, sondern auch die Anbindung von Legacy-Systemen. Findet eine Migration schrittweise statt, werden Services als Middleware für den Zugriff auf Altsysteme eingesetzt. Dies bedeutet auch, dass solchen Services ein Zugriff auf die Altsysteme mit Hilfe von entsprechenden Credentials (in den benötigten Formaten) gestattet sein muss.

Eine zentrale Anforderung im Rahmen der Authentisierung im SOA-Kontext ist die Möglichkeit eines Single-Sign-On. Nicht zuletzt, da es sich in weiten Teilen um eine reine Maschine-zu-Maschine Kommunikation handelt, sind entsprechende Mechanismen zu implementieren.

### **Autorisierung**

Durch die Autorisierung soll sichergestellt werden, dass eine Entität befugt ist, auf die angeforderten Informationen oder Ressourcen zuzugreifen. Typischerweise wird die Autorisierung durch die Verwendung von Rollen und Gruppen umgesetzt. Bei jedem Zugriff auf geschützte Informationen oder Ressourcen sollte überprüft werden, ob die Berechtigungen diesen Zugriff erlauben.

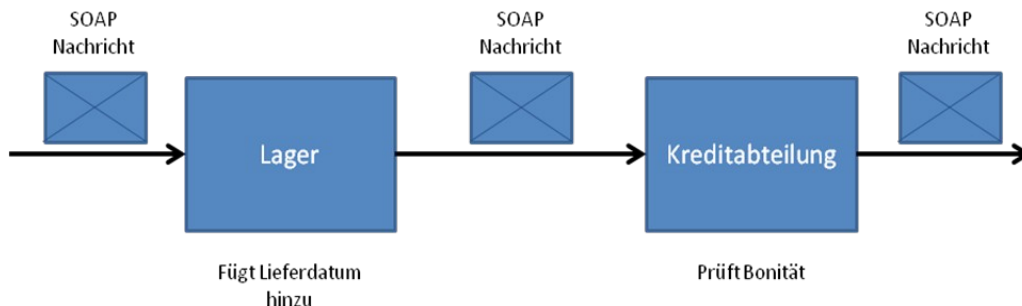
Wie bereits bei der Authentisierung dargestellt, vervielfachen sich in dienstorientierten Strukturen die sog. Policy Enforcement Points, an denen die Mechanismen zur Einhaltung der Berechtigungen implementiert wird. Nicht ein einzelnes Rechtssystem muss entsprechend konfiguriert werden, sondern es müssen Regeln und Rollen für sämtliche Services definiert und durchgesetzt werden. Entsprechend leistungsfähig muss die Administrationskomponente sein, die in der Lage sein muss, ein zentrales Management von Rechten und Rollen zur Verfügung zu stellen, das jederzeit flexibel beherrschbar ist.

Will man eine Autorisierungsprüfung auf Web-Service Ebene nutzen, so muss hier sichergestellt werden, dass die Nachrichten als Web-Services von dem Sicherheitssystem lesbar und verarbeitbar sind. Hierdurch entstehen neue Anforderungen an die Definition der Benutzerrechte. Zudem sind typische Anforderungen wie Mandantenfähigkeit und dedizierte Administrationskonzepte umzusetzen, da in der Regel verschiedene Organisationsstrukturen technisch abzubilden sind.

### **Integrität**

Integrität bezeichnet die Vertrauenswürdigkeit von Informationen und Ressourcen. Eine Verletzung der Integrität bedeutet daher die unberechtigte oder unangemessene Modifikationen von Daten. Integrität kann in die Bestandteile Datenintegrität und Herkunftsintegrität unterteilt werden. Die Datenintegrität soll gewährleisten, dass Daten nicht kompromittiert werden können, sondern über einen langen Zeitraum vertrauenswürdig sind. Die Herkunftsintegrität soll garantieren, dass der Absender der Informationen korrekt ist und die Absenderinformationen nicht gefälscht wurden.

SOA Konzepte sind Nachrichten-basierte Systeme, deren Kommunikation angemessen abgesichert werden muss. Durch die Verwendung offener Standards und einfache Lesbarkeit (XML) der Kommunikation ergeben sich dementsprechend einfache Möglichkeiten zur Manipulation durch Angreifer. Dies bedeutet, dass die Integrität auf Nachrichtenebene gewährleistet werden muss. Adäquate Methoden setzen daher nicht auf der Sicherung des Übertragungswegs an, sondern führen Transformationen der Nachricht selbst durch. Bei der Verwendung einer SOA muss bei der Integrität zudem beachtet werden, dass es in vielen Fällen nicht sinnvoll ist, die Integrität der gesamten Nachricht, sondern für definierte Teile zu gewährleisten. Dies kann z.B. aus Performance-Gründen erforderlich sein oder weil ein Empfänger der Nachricht einen Teil der Nachricht weiterbearbeitet und die Gesamtnachricht weiterleitet, ohne die restlichen Inhalte lesen zu dürfen.



**Abbildung 8: Sukzessives Bearbeiten einer SOAP-Nachricht durch mehrere Dienste**

Bei einem komplexen Workflow z.B. werden mehrere Knoten von einer Nachricht durchlaufen, die definierte Transformationen auf den Inhalt der Nachricht durchführen. Bereits nach der ersten Transformation würde eine Integritätsprüfung für das gesamte Dokument fehlschlagen. Typische Supply-Chain-Szenarien sehen oftmals die Änderung, Anpassung und Erweiterung der Nachricht durch unterschiedliche Teilnehmer (z.B. Zulieferer) vor. So muss die Forderung nach Daten- und Herkunftsintegrität entsprechend erweitert werden, so dass die Integrität einzelner, definierter Teile für unterschiedliche Teilnehmern gewährt sein muss. Integrität muss auf Elementebene bzw. XML-Teilbäume umzusetzen sein.

### Vertraulichkeit

Vertraulichkeit fordert eine Geheimhaltung von sensiblen Daten, um einen unberechtigten Zugriff auf Informationen zu unterbinden. Es muss gewährleistet werden, dass nur authentifizierte und autorisierte Entitäten auf geschützte Informationen zugreifen können. Diese Forderung umfasst das Lesen, Modifizieren und Löschen von Daten. Vertraulichkeit kann durch die Verwendung von Kryptographie erreicht werden.

Wie bereits zum Thema Integrität beschrieben, ist ein Nachrichtensystem mit klar und einfach lesbaren Nachrichten besonders gefährdet. Auch in Bezug auf die Vertraulichkeit gilt, eine Sicherung auf Nachrichtenebene – also die Transformation der Nachricht selbst – durchzuführen. So kann sichergestellt werden, dass auch in komplexen Abläufen über eine Vielzahl von Stationen (und damit verbundenen Zwischenspeicherungen) die Vertraulichkeit gegenüber Unberechtigten gewahrt bleibt.

Eine Sicherung auf Nachrichtenebene erfüllt diese und weitere Forderungen in Bezug auf die Vertraulichkeit. Stichwort ist hier der bereits angesprochene Supply-Chain-Gedanke mit mehreren Teilnehmern, der auch auf die Umsetzung von Vertraulichkeit Einfluss hat. So muss es im Kontext komplexer Geschäftsprozesse möglich sein, Teile von Nachrichten lediglich bestimmten Empfängern zugänglich zu machen. Eine Sicherung einzelner Nachrichtenelemente ist gefordert. Als Beispiel sei eine Bestellung genannt, die neben Artikelstammdaten weitere Angaben zum Kreditrahmen des Kunden enthält. Unterschiedliche Empfänger sollen dabei Zugang zu den für sie bestimmte Informationen erhalten.

### **Verbindlichkeit (Nichtabstreitbarkeit)**

Verbindlichkeit fordert, dass Aktionen, die von einer bestimmten Entität ausgeführt werden, im Nachhinein nicht abgestritten werden können. Zudem sollte gewährleistet werden, dass die Entitäten eine Bestätigung für die jeweils ausgeführten Aktionen erhalten.

Bei einem SOA-Szenario ist die Verbindlichkeit eine der wichtigsten Forderungen, die erfüllt werden muss, wenn geschäftskritische Prozesse abgewickelt werden. Gerade bei Involvierung verschiedener und teilweise anonymer Teilnehmer ist die Nicht-Abstreitbarkeit von getätigten Aktionen (Bestellungen, Aufträge, etc.) eine wesentliche Bedingung zur elektronischen Abwicklung von Transaktionen. In diesem Zusammenhang muss auch einer Vielzahl rechtlicher Aspekte Rechnung getragen werden. So stellt das Signaturgesetz die elektronische Unterschrift der handschriftlichen gleich und definiert gleichzeitig entsprechende Bedingungen an elektronische Signaturen, deren Erfüllung auch in verteilten Architekturen gewährleistet sein muss. Nachrichten sind ggf. mit einer qualifizierten digitalen Signatur zu versehen, um eine Verbindlichkeit sicherzustellen und juristisch verwertbar zu sein. Kosten und Zeitersparnisse sind daher mit dem Einsatz sicherer Methoden zu erreichen. Dies betrifft nicht nur Verträge i.e.S., sondern auch Transaktionsinformationen, wie bspw. Bestätigungen und Buchungsmeldungen. Nach erfolgreicher Prüfung der digitalen Signatur kann sichergestellt werden, dass die Nachricht willentlich vom angegebenen Absender stammt und dessen Inhalt unverfälscht ist. Die Verwendung von Zertifikaten innerhalb einer PKI kann neben den technischen auch die organisatorischen Aspekte – der Zuordnung von Person zu Zertifikat – abbilden. Zertifikatsklassen und Anforderungen an qualifizierte Signaturen definiert das Signaturgesetz.

### **Verfügbarkeit und Ausfallsicherheit**

Verfügbarkeit fordert, dass berechnete Entitäten auf die benötigten Informationen und Ressourcen ohne Einschränkungen zugreifen können. Um verlässliche Systeme zu bieten, muss eine Organisation garantieren, dass die Verfügbarkeit sichergestellt wird. Es ist häufig der Fall, dass die Vertraulichkeit und Integrität gewährleistet wird, aber durch einen Angriff auf die Verfügbarkeit eines Systems nicht mehr auf die benötigten Informationen zugegriffen werden kann.

In einem verteilten System ist das Ausfallrisiko als relativ hoch einzuschätzen, vor allem, da eventuell einige Teile nicht im eigenen Zugriffsbereich liegen. Daher ist ein Single-Point-of-Failure zu vermeiden und die Forderung zu erfüllen, dass auch bei Ausfall einiger Komponenten andere Teile weitestgehend unberührt bleiben. Bei Implementierung von zentralen Komponenten in einer SOA können kritische Geschäftsprozesse stark beeinträchtigt werden, was dazu führen kann, dass wichtige Prozesse innerhalb einer Organisation nicht korrekt ausgeführt werden können. Dies betrifft insbesondere das Sicherheitssystem, dessen Funktionsfähigkeit gewährleistet werden muss. Auch bei Ausfall von zentralen Teilen muss die Funktionsfähigkeit – bspw. durch lokal verteilte Logik und Policies – sichergestellt werden. Agenten-basierte Systeme ohne lokale Intelligenz/Logik sind zu vermeiden, da diese bei jeder Aktion Kontakt zu einem zentralen System aufbauen müssen, um Informationen zu erhalten. Auch bei verteilten Agenten ist die Verfügbarkeit eine der wichtigsten Anforderungen, da im Falle einer Abschottung eines Backend-Systems durch einen Agenten gewährleistet werden muss, dass der Agent und dadurch das Backend-System stets verfügbar ist. Falls dieser Agent nicht verfügbar ist, könnte man nicht mehr auf das Backend-System zugreifen. Somit könnte der Geschäftsprozess, der durch eine SOA abgebildet wurde, nicht mehr ausgeführt werden.

Daneben kann die Ausfallsicherheit sowie Skalierbarkeit durch bewährte Konzepte (Load-Balancing, Application-Server-Management, etc.) umgesetzt werden. Insbesondere die Skalierbarkeit der (Security-) Architektur muss sichergestellt sein, da in der Planungsphase nur bedingt festgelegt werden kann, wie viele Consumer auf einen Service zugreifen werden. Über die Zeit sind veränderte Nutzungsbedingungen ebenfalls zu berücksichtigen.

## **Administrierbarkeit**

Trotz der dezentralen und komplexen Architektur einer SOA muss gewährleistet werden, dass das Gesamtsystem beherrschbar bleibt und mit angemessenem Aufwand administriert werden kann. Bei der Umsetzung eines Sicherheitskonzeptes für eine Service-orientierte Architektur ist die Administration daher ein essentieller Bestandteil.

Schließlich ist es unabdingbar, dass ein Sicherheitskonzept flexibel auf sich ändernde Geschäftsprozesse angepasst werden kann. Nur dadurch wird es einer Organisation ermöglicht, sich schnell an geänderte Rahmenbedingungen anpassen zu können und damit den vollen Nutzen aus einer Service-orientierten Architektur zu ziehen.

Analog zur Service-Orchestrierung ist ein Administrationskonzept gefordert, welche die Beherrschbarkeit des Gesamtsystems von einem zentralen Punkt erlaubt. Weitere Anforderungen herkömmlicher Security-Systeme sind zu berücksichtigen, wie bspw. dedizierte Administrationskonzepte, Mandantenfähigkeit sowie Möglichkeiten zur Integration bestehender Systeme (bspw. PKI, Meta Directories, Identity Management, etc.). Gerade die Integration bereits vorhandener Systeme schafft neben Kostenersparnissen die Grundlage für einen reibungslosen Prozessablauf ohne Medienbrüche sowie die Sicherstellung konsistenter Informationen in allen beteiligten (Sicherheits-)Systemen.

Der Standardanforderung nach aussagekräftigen und aktuellen Protokoll- und Log-Dateien muss in einer SOA besondere Aufmerksamkeit geschenkt werden. Ein Logging- und Reportingsystem für verteilte Infrastrukturen muss in der Lage sein, Informationen von allen Laufzeitumgebungen zentral zur Verfügung zu stellen. Eine Analyse des Systems ist ansonsten schwer möglich. Zudem sollten führende Standards, wie bspw. ITO-Log, unterstützt werden, um vorhandene System Management Systeme einzubinden.

## **Datenschutz**

Durch den Datenschutz soll im Wesentlichen gewährleistet werden, dass personenbezogene Daten vor Missbrauch geschützt werden. Um diese Anforderung zu erfüllen, müssen diese Daten vertraulich behandelt werden und dürfen nur für den Zweck eingesetzt werden, für den sie erhoben wurden. Neben der organisatorischen Forderung, dass Unternehmen versuchen sollten, so wenig personenbezogene Daten wie möglich zu erheben und nur so viele wie unbedingt nötig, gibt es technische Rahmenbedingungen.

Bei Service-orientierten Architekturen müssen personenbezogene Daten ebenfalls angemessen geschützt werden, besonders wenn diese durch die abgebildeten Geschäftsprozesse die Organisationsgrenzen überschreiten. Umgesetzt werden kann dies durch Methoden der Nachrichtenverschlüsselung zur Gewährleistung der Vertraulichkeit. Die im Abschnitt Vertraulichkeit beschriebenen Techniken zur Sicherung einzelner Nachrichtenteile ermöglichen technischen Datenschutz auf Nachrichtenebene. Zusätzlich sollte der Prozess der Datenerhebung für den Betroffenen transparent gestaltet werden.

## **Audit**

Durch ein Audit wird überprüft, ob bei einem Prozess die jeweiligen Richtlinien und Rahmenbedingungen erfüllt werden. Im Rahmen von Service-orientierten Architekturen gibt es eine große Menge von Protokollen, die es zu überprüfen gilt. Es sollte hierbei die Möglichkeit geboten werden, dass die Informationen an einer zentralen Stelle einsehbar sind und die Informationen in einem einheitlichen Format vorliegen. Zudem ist es notwendig, dass ein System mit unterschiedlichen Detailstufen geboten wird und die Inhalte je nach Fragestellung gefiltert werden können. Bei einigen Protokollen kann es zudem nötig werden, dass nur bestimmte Personengruppen Zugriff erhalten. Dies ist mit entsprechenden kryptographischen und organisatorischen Methoden sicherzustellen. Die Granularität der gespeicherten Informationen muss angepasst werden können, so kann es bei einigen Services notwendig

sein, im Rahmen eines Audits den gesamten Web-Service zu loggen, doch bei anderen Services könnte diese Datenmenge zu groß sein.

## 3.2 Bedrohungspotentiale

Die Bedrohungspotentiale von SOA decken sich teilweise mit im allgemein im Betrieb vorhandenen Informationssystemen. Jedoch ist es aufgrund der Verteilung über mehrere Systeme und Geschäftspartner nicht mehr ausreichend, nur einzelne Computer und Netzwerke als solches zu schützen. Der klassische Ansatz von Sicherheit, die Ränder des Informationssystems zu schützen, greift zu kurz, da es keine klar definierten Ränder mehr gibt. Somit sind weitere Maßnahmen erforderlich, um die Sicherheitsziele Vertraulichkeit, Integrität, Authentizität, Nicht-Abstreitbarkeit und Verfügbarkeit sicherzustellen. Als Zielobjekte für mögliche Bedrohungen werden neben den eigentlichen transportierten Nutzdaten eines Serviceaufrufs u.a. auch kommunizierte Sicherheitstoken betrachtet.

Dieser Abschnitt unterteilt sich in vier Bereiche. Zunächst werden allgemeine Bedrohungspotentiale vorgestellt, bevor auf SOA-spezifische Bedrohungspotentiale eingegangen wird. Nach einer Risiko-/Bedrohungsanalyse des Beispielsprozesses werden die Bedrohungspotentiale für diesen konkretisiert.

### 3.2.1 Allgemeine Bedrohungspotentiale

Im Folgenden werden typische Quellen von Bedrohungen vorgestellt, die bei der Verwendung von Computersystemen existieren.

#### Schadsoftware

Als Schadsoftware werden Computerprogramme bezeichnet, die die Informationssicherheit gefährden. Die potentiellen Risiken reichen von Löschen bzw. Modifizieren einzelner Dokumente bis hin zum gezielten Ausspähen. Die Schadsoftware wird in der Regel ohne Wissen des Anwenders ausgeführt. Grundsätzlich können solche Schadsoftwareattacken auch in Services "verpackt" werden. Falsche Werte liefern, Daten ausspähen, etc. sind Angriffsszenarien, die in Betracht gezogen werden müssen.

#### Buffer Overflows

Als Buffer Overflows werden Fehler von Computerprogrammen bezeichnet, die aufgrund der ungültigen Eingabe von Daten zu unvorhergesehenem Verhalten führen. Die häufigste Form ist der Absturz des Computerprogramms. Eine weitere Möglichkeit besteht darin, dass das Computerprogramm einen Programmcode ausführt, der in den eingegebenen Daten enthalten ist. Dadurch kann der Angreifer unter Umständen die komplette Kontrolle über ein Computersystem übernehmen. Auch SOA-Services unterliegen als Netz-Anwendung solchen Gefahren. Services müssen ebenso vor falscher Parameter-eingabe geschützt werden wie herkömmliche Anwendungen.

#### Hintertüren

Hintertüren sind Zugangswege zu Computersystemen, die von Herstellern von Computerprogrammen in ihre Produkte integriert werden. Diese Zugangswege entziehen sich der Zugriffskontrolle des Betreibers des Computersystems und sind ihm im Allgemeinen auch nicht bekannt.

Im Rahmen einer SOA stellen Backdoors wie bei anderen Architekturen eine ernstzunehmende Bedrohung dar. Da wir bei einer SOA nicht über ein einzelnes System, sondern über eine Vielzahl von Systemen reden, ist das Bedrohungspotential dementsprechend höher. Zudem ergibt sich bei einer SOA eine weitere Problematik, die auf der allgemeinen Architektur basiert. Der Nutzer der angebotenen Services fragt bei einem Service-Repository nach einem passenden Service an und bindet sich an die-

sen. Es kann vorkommen, dass die genaue Herkunft der angebotenen Services nicht vollständig geklärt werden kann und somit potentiell gefährliche Services in den Geschäftsprozess integriert werden.

### Netzangriffe

Die Kommunikation in Computernetzen erfolgt durch so genannte Protokolle, welche die Form der Kommunikation festlegen. Durch Schwächen in diesen Protokollen können Angreifer Informationen erhalten oder Zugangskontrollsysteme umgehen.

Folgende Angriffsarten treten in Verbindung mit Computernetzen auf:

- **Scanning:** Es gibt Computerprogramme, mit denen überprüft werden kann, welche Dienste und Schwachstellen auf einem Computersystem vorhanden sind. Diese Computerprogramme reichen von einem einfachen Portscanner hin zu komplexen Anwendungen, die automatisiert Schwachstellen wie Buffer Overflows analysieren und ausnutzen.
- **Sniffing:** Mit Sniffing wird das Belauschen des Datenverkehrs in einem Computernetz bezeichnet. Dadurch kann dem Angreifer nicht nur der Zugang zu vertraulichen Daten ermöglicht werden, sondern auch bei einigen Programmen die Berechtigungsinformationen für Zugangskontrollsysteme.
- **Spoofing:** Mit dieser Technik erstellt der Angreifer Nachrichten, die ein gefälschtes Subjekt enthalten. Ein Angreifer kann mittels Spoofing Zugang zu Systemen oder Daten erhalten, für die er keine Zugangsberechtigung besitzt. Häufig wird das Internet Protocol (IP-Spoofing) oder der Domain Name Service (DNS-Spoofing) ausgenutzt.
- **Man in the middle:** Bei diesem Angriff schaltet sich ein Angreifer zwischen zwei Kommunikationsendpunkte ein und täuscht jedem Endpunkt die Identität der ursprünglichen Gegenseite der Kommunikation vor. So kann er Daten mitlesen oder auch modifizieren. Im SOA-Umfeld werden dieser und weitere Angriffe, bei denen XML-Nachrichten modifiziert weiterverwendet werden, als XML-Rewriting-Angriffe bezeichnet.

SOA Systeme basierend auf Nachrichtenformaten wie XML nutzen herkömmliche Protokolle wie http und sind daher diesen Gefahren ausgesetzt. Dabei muss allerdings berücksichtigt werden, dass herkömmliche Schutzmaßnahmen meist solche Angriffe auf Services nicht erkennen. Es müssen SOA- und Service-gerechte Methoden und Techniken eingesetzt werden.

### Denial of Service

Mit Denial of Service werden Angriffe gegen die Verfügbarkeit bezeichnet. Dieses Ziel wird durch eine Überlastung des Computersystems erreicht, so dass berechnete Subjekte ihre Aktivitäten nicht mehr oder stark eingeschränkt durchführen können. Die möglichen Angriffe reichen von primitiven verteilten Serviceaufrufen bis hin zum gezielten Überlasten des Services. Es stehen viele Angriffsziele zur Verfügung, bei denen an jedem Punkt das gleiche Schutzniveau gelten muss. Schließlich bewirkt ein Ausfall eines Services auf dem kritischen Pfad den Ausfall des gesamten Geschäftsprozess. Durch die Öffnung der Services ins Intra- bzw. Internet sind die dahinterliegenden Backendsysteme neuen Bedrohungen ausgesetzt. Reichten vormals klassische Applikations-Schutzmaßnahmen wie IP-Firewalls, so muss nun mit anderen Bedrohungen, wie Denial of Service, gerechnet werden. Dabei sollte zur Abwehr von Denial of Service Attacks die Forderung nach Skalierbarkeit des Sicherheitssystems ebenfalls bedacht werden.

SOAP-basierte Web-Services verwenden meist das http(s)-Protokoll zum Transport der Daten. Hierdurch werden mit Blick auf DoS-Angriffe viele Bedrohungen (Dumb Flooding, TCP/SYN-Flooding, ...) von http auch für Web-Services relevant. Auch ist es u. U. möglich, Schadcode innerhalb von SOAP-Nachrichten in ein Netzwerk einzubringen, ohne dass dies von Firewalls bemerkt und blockiert wird.

## **Passwortcracking**

Die Methoden des Passwortcrackings werden genutzt, um Passwörter für Computerprogramme oder Computersysteme wiederherzustellen. Mit diesen kann ein Angreifer dann Zugang erlangen. Im Allgemeinen werden Passwörter nicht im Klartext gespeichert, so dass die Passwortcracker versuchen, ein Passwort aus häufigen Wörtern zu raten oder einfach alle Zeichenkombinationen systematisch durchzuprobieren. Auch effizientere mathematische Kryptoanalysen sind u. U. möglich.

In SOA-Umgebungen sind die Angriffsmöglichkeiten relativ vielfältig, muss sich doch jeder Service Aufruf in irgendeiner Weise authentisieren. Werden lediglich Passwörter verwendet, ist ein nachrichtenbasiertes System an jedem Punkt angreifbar.

## **Kryptoanalyse**

Mit Kryptoanalyse werden Angriffe gegen kryptographische Verfahren bezeichnet. Ziel ist es hierbei, durch Schwächen des Kryptosystems sowohl Verschlüsselungsverfahren zu brechen, als auch digitale Signaturen fälschen zu können. In einer SOA, die über die Organisationsgrenzen hinweg betrieben wird, kann es zu der Verwendung unterschiedlicher Algorithmen kommen. Diese Situation resultiert aus den unterschiedlichen rechtlichen Rahmenbedingungen, die in einem internationalen Umfeld erfüllt werden müssen. Durch diese Bedingungen kann es dazu kommen, dass nicht immer die optimalen Algorithmen verwendet werden können, was sich selbstverständlich negativ auf die Sicherheit der SOA auswirkt. Im Falle eines Algorithmus, der gebrochen wurde, muss schnellstmöglich das gesamte System umgestellt werden, um die Sicherheit des gesamten Prozesses zu gewährleisten. Es sollten nur anerkannte Standardalgorithmen verwendet werden, da die einzelnen Web-Service Sicherheitsstandards von ihnen abgeleitet werden.

Neben den bereits vorgestellten Angriffen existieren noch weitere Angriffsmethoden. Hierzu zählen z.B. SQL-Injections oder Cross-Site-Scripting (XSS) in SOAP-Nachrichten. Auch durch nicht-technische Angriffe (z.B. Social Hacking) können Informationen oder Computersysteme kompromittiert werden. Da es sich bei SOA im Wesentlichen um Maschine zu Maschine Kommunikation handelt, soll dies an dieser Stelle vernachlässigt werden.

### **3.2.2 SOA-spezifische Bedrohungspotentiale**

Zusätzlich zu den allgemeinen Bedrohungspotentialen gibt es SOA-spezifische, die aus den grundlegenden Eigenschaften von Service orientierten Architekturen resultierten.

#### **Replay-Attacks**

Eine zentrale Eigenschaft von SOA ist die Zustandslosigkeit von Services. Durch diese Eigenschaft kann ein Angreifer eine durch ein berechtigtes Subjekt durchgeführte Handlung auch dann wiederholen, wenn die Nachricht durch Verschlüsselung und Signatur geschützt ist. Aus diesem Angriff kann beispielsweise ein wirtschaftlicher Schaden entstehen, wenn die Verwendung des Services für das Opfer pro Anfrage tarifiert ist. Am Beispiel des „Versand“-Services im Beispielszenario könnte ein Angreifer die Nachricht auf dem Weg zur Versandabteilung des Buchhandels abfangen und diese zu einem späteren Zeitpunkt erneut versenden oder kostenpflichtige externe Services („Kreditkarten Validierung“) mehrfach aufrufen. In SOA Umgebungen bieten sich für diese Attacken viele Angriffspunkte, da viele Systeme am Gesamtprozess beteiligt und Services häufig zustandslos sind. Ein System merkt sich nicht, welche Nachrichten bereits bearbeitet wurden und würde somit auch eine wiederspielte Nachricht bearbeiten.

#### **XML-spezifische Angriffe**

Aufgrund der mitunter sehr hohen Komplexität der Baumstruktur eines XML-Dokuments ist das erforderliche Parsen evtl. sehr ressourcenlastig. Unterschiedliche Parser eignen sich daher auch für

unterschiedliche Zugriffsverfahren unterschiedlich gut. Während einige Parser den wahlfreien Zugriff auf XML-Elemente effizient implementieren, ist das bei anderen Parsern für den seriellen Zugriff der Fall.

Durch die Verwendung von SOAP- bzw. XML-basierten Nachrichten ergeben sich eine Reihe möglicher Angriffe, die mit den Eigenheiten von XML eng verbunden sind. Hierzu gehören u.a. sog. XML-Bombs, wobei es sich um endlose Rekursionen in XML-Dokumenten handelt. Hierdurch kann z.B. ein XML-Parser möglicherweise zum Absturz gebracht werden. Abhilfe schafft hier das umfassende und sorgfältige Validieren bzw. Scannen von XML-Strukturen. Weitere XML-spezifische Angriff sind z.B. XPath-Injection oder XML eXternal Entity Attacks (XXE). Außerdem existieren weitere zusätzliche Angriffe gegen XML-Parser wie z.B. dem sog. Schema Poisoning, bei dem eine Modifikation der Grammatik von XML-Schemata zu Inkonsistenzen und Abstürzen bei der Verifikation durch XML-Parser führen kann.

### **WSDL- und Service-Scanning**

Details und Parameter zum Aufruf eines Web-Services sind in dessen WSDL-Beschreibung öffentlich verfügbar. Dies kann ein Angreifer ausnutzen, um etwa mittels Methoden der Fuzzing-Softwareanalyse gezielt diese Dienste möglichst effizient auf Schwachstellen hin zu untersuchen.

### **Kompromittierung eines Services**

Gemäß des Service-Gedankens bietet ein Dienst Funktionalität für andere Services oder Geschäftsprozesse. Die Services sind im Allgemeinen verteilt, auch über Organisationsgrenzen hinaus. Durch Manipulation eines Services oder Platzieren eines manipulierten Services im Service Repository können andere Services oder ganze Geschäftsprozesse manipuliert oder unterbunden werden. Um das Risiko, dass durch diese Bedrohungen verursacht wird zu minimieren, sollte garantiert werden, dass die Authentizität des Serviceanbieters überprüft wird und zusätzlich ob es sich bei diesem um einen vertrauenswürdigen Anbieter handelt.

### **Unberechtigte Servicenutzung**

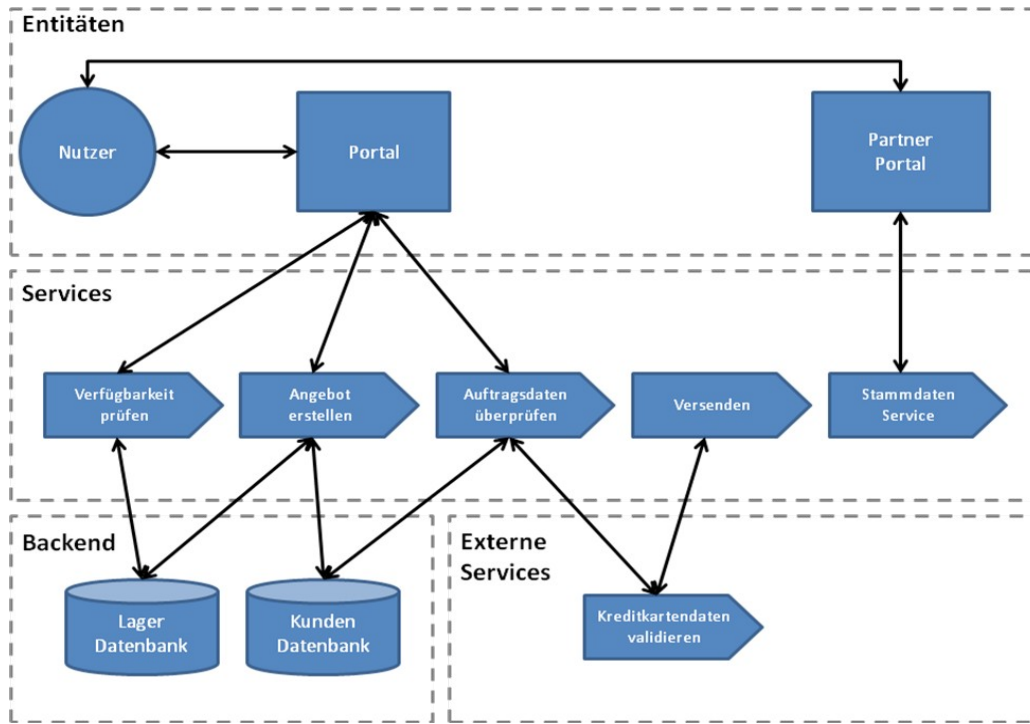
Da ein Service einen Bestandteil eines Geschäftsprozesses abbildet, müssen Subjekte, also Benutzer oder Computerprogramme, die diesen Service benutzen, authentifiziert werden. In einem verteilten Umfeld ggf. über Organisationsgrenzen hinaus, müssen die Mechanismen zum Austausch von Subjekten und Authentifizierungsinformationen geschützt werden, so dass ein Angreifer diese nicht manipulieren bzw. unterbinden kann. Da ein Service einen Bestandteil eines Geschäftsprozesses abbildet, muss die Benutzung durch Subjekte, also Benutzer oder Computerprogramme, die diesen Service verwenden, geregelt werden. Diese Services können je nach Szenario unternehmensintern oder unternehmensübergreifend abgerechnet werden. Die sichere Abrechnung von diesen Services muss gewährleistet werden, um zu verhindern, dass unberechtigte Subjekte Zugriff zu vertraulichen Daten erhalten oder dass unberechtigte Buchungen vorgenommen werden. Eine unberechtigte Buchung könnte erfolgen, falls die Benutzung von Services, die nicht verwendet wurden, gebucht wird oder eine inkorrekte Anzahl von Nutzungen gebucht wird.

### **Ausnutzung von Organisationsschwächen**

In Service-orientierten Architekturen können Angreifer durch Kompromittierung von Services Sicherheitsanforderungen umgehen. Aufgrund der Komplexität und ggf. Organisationsgrenzen kann die Kompromittierung unter Umständen schwer erkannt werden, was einem Angreifer weitere Attacken ermöglicht. Um einen ausreichenden Schutz zu bieten, sollte eine zentrale Administration für die Sicherheit in einer SOA gewährleistet werden. Dadurch wird ein durchgängiges und gleichmäßiges Schutzniveau für alle angebotenen Services geboten. Bei der Verwendung von proprietären Schutzmaßnahmen sollte man bedenken, dass diese kaum von zentraler Stelle überprüfbar sind. Es sollte daher darauf geachtet werden, nur standardisierte Verfahren einzusetzen.

### 3.2.3 Sicherheitsanforderungen des Beispielsprozesses

In diesem Abschnitt wird exemplarisch anhand des Beispielprozesses dargestellt, wie die Sicherheitsanforderungen eines Geschäftsprozesses ermittelt werden können.



**Abbildung 9: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung und unter Verwendung externer Dienste**

#### Verfügbarkeit prüfen

Der Service "Verfügbarkeit prüfen" liefert dem Consumer – d.h. dem Web-Portal – Informationen darüber, welche Gegenstände auf Lager sind. Dieser Service wird genutzt, um den Benutzer vor der Bestellung über Lieferbedingungen zu informieren.

Der Schutzbedarf dieses Services ist als niedrig bis mittel einzuschätzen. Das primäre Schutzziel ist die Verfügbarkeit, da sonst keine Bestellungen erfolgen können. Die Integrität und Authentizität müssen sicher gestellt sein, da durch falsche Angaben Bestellungen nicht durchgeführt werden können und dem Unternehmen hierdurch ggf. Schaden entsteht.

Zusätzlich sollte die Vertraulichkeit sichergestellt werden, da sich andernfalls ein Mitbewerber einen Überblick über den aktuelle Lagerbestand und die dahinter liegende Lagerhaltungsstrategie verschaffen könnte.

#### Angebot erstellen

Der Service "Angebot erstellen" erstellt basierend auf Artikelnummern und der Kundennummer ein für den Kunden spezielles Angebot unter Berücksichtigung unternehmensinterner Vertriebsstrategien.

Der Schutzbedarf dieses Services ist hoch. Er liefert Vertragsbestandteile, die das Schutzziel Verbindlichkeit erforderlich machen. Aus Kundensicht sollte die Nicht-Abstreitbarkeit nicht vernachlässigt werden. Die Integrität des Services, wie auch die Authentizität gegenüber den Subjekten muss sicher gestellt sein. Für den gesamten Bestellvorgang ist die Verfügbarkeit dieses Services wichtig.

Da das Angebot unternehmensinterne Vertriebsstrategien beinhaltet, sollte die Vertraulichkeit gewährleistet werden, so dass jeder Kunde nur Zugriff auf für ihn bestimmten Einkaufskonditionen erhält.

### **Auftragsdaten überprüfen**

Nachdem der Kunde das Angebot angenommen hat, gibt er ggf. seine Stammdaten ein und wählt die Zahlungsweise. Die Überprüfung der (syntaktischen) Korrektheit und die Validierung, ob die gewählten Optionen mit den Geschäftsbedingungen übereinstimmen, wird durch diesen Service durchgeführt.

Der Schutzbedarf dieses Services ist hoch. Da dieser Kundendaten verarbeitet, muss aufgrund von Datenschutzbestimmungen die Vertraulichkeit hergestellt werden. Das Schutzziel Verbindlichkeit ist durch die Verarbeitung von Vertragsdaten erforderlich. Die Integrität des Services, wie auch die Authentizität gegenüber Subjekten muss sicher gestellt sein. Für den gesamten Bestellvorgang ist eine Verfügbarkeit dieses Services essentiell.

### **Kreditkartendaten validieren**

Falls ein Kunde mit Kreditkarte zahlen möchte, wird von dem "Auftragsdaten überprüfen"-Service ein von einem externen Anbieter betriebener "Kreditkartendaten validieren"-Service angefragt. Da dieser Service zum einen mit personenbezogenen Daten arbeitet, zum anderen die Unternehmensgrenzen (zum Kreditinstitut hin) überschritten werden, ist der Schutzbedarf hoch. Für diese Anfrage ist es wichtig, dass die Antwort verbindliche und valide Aussagen über die Bonität beinhaltet. Da es sich um Daten handelt, die von Dritten missbraucht werden können, sind hohe Anforderungen an die Authentizität und Vertraulichkeit gegeben. Die Integrität und Verfügbarkeit dieses Dienstes sind wichtig.

### **Versenden**

Der "Versenden"-Service ist für die Bestellbearbeitung zuständig. Der Schutzbedarf ist eher niedrig bis mittel, jedoch an die Vertraulichkeit der Kundendaten herrschen höhere Anforderungen. Die Verfügbarkeit ist nicht hoch zu bewerten, sofern die Portalanwendung nicht zugestellte Bestellungen zwischenspeichern kann.

### **Stammdatenservice**

Durch das Anklicken des Partnerlinks auf der Website werden diesem Identifikationsdaten des Benutzers übermittelt. Greift der Partner im Rahmen des SSO auf den Stammdatenservice zu, so erhält er persönliche Daten des Kunden. An dieser Stelle sind Vertraulichkeit und Authentizität des Anfragenden besonders hoch zu bewerten, da es sich um personenbezogene Daten handelt. Integrität und Verfügbarkeit sind nachrangige Schutzziele.

	<b>Authentifizierung</b>	<b>Autorisierung</b>	<b>Integrität</b>	<b>Vertraulichkeit</b>	<b>Verbindlichkeit</b>	<b>Verfügbarkeit</b>
<b>Verfügbarkeit prüfen</b>			niedrig bis mittel	niedrig bis mittel		niedrig bis mittel
<b>Angebot erstellen</b>	niedrig bis mittel	niedrig bis mittel	sehr hoch	niedrig bis mittel	sehr hoch	sehr hoch
<b>Auftragsdaten prüfen</b>			sehr hoch	hoch		sehr hoch
<b>Kreditkartendaten validieren</b>	sehr hoch		sehr hoch	sehr hoch		hoch
<b>Versenden</b>		niedrig bis mittel	hoch	niedrig bis mittel		niedrig bis mittel
<b>Stammdatenservice</b>	sehr hoch	hoch	niedrig bis mittel	sehr hoch		niedrig bis mittel

**Tabelle 1: Ermittelter Schutzbedarf für das gewählte Beispielszenario in den Stufen „niedrig bis mittel“, „hoch“ und „sehr hoch“**

### 3.2.4 Bedrohungspotentiale für den Beispielsprozess

Die allgemeinen Bedrohungspotentiale gelten für jeden Service im Beispielsprozess.

Im Folgenden werden die SOA-spezifischen Bedrohungen für die einzelnen Services darstellt.

#### **Verfügbarkeit überprüfen**

Bei diesem Service könnte es verschiedene Angriffsszenarien geben, die eine erfolgreiche Ausführung verhindern oder Daten manipulieren. Ein Angreifer könnte sich zwischen das Portal und den Service einklinken und somit anstelle des Services Informationen an das Portal weiterleiten. Somit könnten falsche Informationen an den Kunden weitergeleitet werden, die eine Bestellung verhindern könnten, falls angegeben wird, dass ein vorhandener Artikel nicht verfügbar ist oder eine falsche Anzahl der im Lager befindlichen Artikel weitergegeben wird. Um solche Angriffe zu verhindern sollte darauf geachtet werden, dass die Integrität der Daten nicht verletzt, und die Authentizität der Services überprüft wird.

Zudem wäre es problematisch, wenn ein Angreifer sehen könnte, welcher Kunden bestimmte Produkte bestellt hat. Möglicherweise kann ein Angreifer auch umfangreiche Informationen über die Lagerbestände des Unternehmens gewinnen. Um diese Probleme zu lösen, sollte ein hinreichendes Maß an Vertraulichkeit gewährleistet sein. Selbstverständlich muss auch die grundlegende Sicherheitsanforderung „Verfügbarkeit“ umgesetzt werden, da ohne diese das Portal keine Informationen über aktuelle Lagerbestände an den Kunden weiterreichen könnte.

### **Angebot erstellen**

Bei der Erstellung von Angeboten für bestimmte Kunden werden Daten aus der Kundenverwaltung verwendet, um spezielle Konditionen einzubinden. Es könnte der Fall sein, dass spezielle Kunden besondere Rabatte auf Produkte des Buchhändlers angeboten bekommen. Diese Informationen stellen natürlich für Konkurrenten eine interessante Information dar, und ein potentieller Angreifer könnte versuchen, die Kommunikation abzuhören oder unautorisierte Anfragen an den Service zu senden.

Mit dem Abhören der Kommunikation könnte ein Angreifer Informationen über den Status eines Kunden gewinnen und generelle Informationen über die Vertriebsstrategie des Unternehmens erhalten. Mit Hilfe von Verschlüsselung kann bei diesen Angriffen die Sicherheitsanforderung „Vertraulichkeit“ umgesetzt werden und somit ein Auslesen von vertraulichen Daten verhindert werden.

Im Falle von unberechtigten Aufrufen eines Services sollte garantiert werden, dass diese verworfen und keinesfalls beantwortet werden. Um eine Schutzmaßnahme gegen solche Angriffe umzusetzen, muss die Integrität der Daten und die Authentizität des Anfragenden gewährleistet werden.

Bei diesem Szenario ist aus Kundensicht die Verbindlichkeit des erstellten Angebots interessant, da er annimmt, die gewünschten Daten zum genannten Preis zu erhalten. Sollte im Nachhinein ein anderer Preis für die Produkte verlangt werden, würde dies zu Verstimmungen beim Kunden führen.

Da auch dieser Prozessschritt eine Voraussetzung für die folgenden Schritte ist, ist er damit für den Gesamtprozess unabdingbar. Ist die Verfügbarkeit des Services gefährdet, so ist keine Beauftragung möglich und es kommt zu wirtschaftlichen Schäden.

### **Auftragsdaten überprüfen**

Bei diesem Service werden relevante Informationen des Auftrags überprüft, um sicherzustellen, dass alle Daten korrekt erfasst wurden. Hierfür muss garantiert werden, dass die Integrität der Daten nicht verletzt wurde. Korruptierte Daten hätten falsche, unerwünschte und fehlerhafte Auftragsbefüllungen zur Folge, die nicht zuletzt in rechtlichen Auseinandersetzungen enden können.

Im Zuge der Auftragsabwicklung werden auch personenbezogene Daten verarbeitet, und somit muss sichergestellt werden, dass diese Daten gemäß den gültigen Datenschutzbestimmungen des jeweiligen Landes bearbeitet werden. Rechtliche Probleme sowie Missbrauch dieser Daten durch Dritte wären mögliche Folgen einer Verletzung des Datenschutzes.

### **Kreditkartendaten validieren**

Im Rahmen der Validierung von Kreditkarten haben wir es mit sehr sensiblen Daten zu tun. Diese müssen auf angemessene Art und Weise vor Angreifern geschützt werden, die durch Gewinnung dieser Informationen in der Lage wären, diese für ihre eigenen Zwecke zu verwenden. Generell wird bei der Validierung überprüft, ob die eingegebenen Informationen korrekt sind und ob die benötigte Bonität vorhanden ist. Beim Versand der Daten an den Service muss sichergestellt werden, dass die Authentizität gewährleistet ist und die Informationen somit an eine vertrauenswürdige Stelle weitergeleitet werden. Falsche Aussagen zur Bonität des Kunden sind ebenso schädlich wie Kreditkartendaten des Kunden in den falschen Händen. Es muss auch darauf geachtet werden, dass die Informationen nicht auf dem Transportweg von einem Angreifer manipuliert werden können, was selbstverständlich einen wirtschaftlichen Schaden für das Unternehmen zur Folge hätte.

### **Versenden**

Auch bei diesem Service werden personenbezogene Daten weitergeleitet, in diesem speziellen Fall an die Versandabteilung des Unternehmens. Ist die Vertraulichkeit nicht garantiert, ist der Datenschutz verletzt. Eine nicht vorhandene Integrität würde falsche Liefer- und Rechnungsdaten zur Folge haben.

Im Falle einer erfolgreichen nicht autorisierten Bestellung könnte erheblicher wirtschaftlicher Schaden für das Unternehmen entstehen. Zusätzlich sollte bei diesem Service auch die Problematik des Wiedereinspiels von abgefangenen Nachrichten (sog. Replay-Angriffe) berücksichtigt werden, sodass verhindert wird, dass bereits versendete Bestellungen erneut bearbeitet werden.

### Stammdatenservice

Bei der Weiterleitung von Stammdaten ist es aus datenschutzrechtlicher Sicht zwingend erforderlich, die Authentizität des anfragenden Subjekts zu verifizieren und die Daten auf angemessene Art und Weise zu schützen, um somit die Sicherheitsanforderungen „Datenschutz“ und „Vertraulichkeit“ zu erfüllen. Bei Nichterfüllung könnten Unberechtigte Zugang zu personenbezogenen Daten erhalten, wodurch dem Kunden entsprechende Unannehmlichkeiten und dem Unternehmen rechtliche Probleme entstehen.

## 4. Schutzmaßnahmen

Im folgenden Abschnitt werden konventionelle Sicherheitsmechanismen betrachtet und es wird analysiert, warum diese bei einer SOA meist unzureichend sind. Zusätzlich wird hervorgehoben, welche klassischen Fehler bei der Umsetzung eines Sicherheitskonzeptes für eine SOA begangen werden können. Den Abschluss dieses Abschnittes bildet die Einführung anerkannter Sicherheitsstandards im SOA Umfeld und eine Vorstellung von Schutzmaßnahmen, die bisher nicht standardisiert sind.

### 4.1 Schwächen bisheriger Sicherheitskonzepte

Die Service-orientierte Architektur stellt neue Anforderungen an die Informationssicherheit. Somit kommt es häufig zu Problemen, wenn versucht wird, eine SOA mit klassischen Schutzmaßnahmen abzusichern. Speziell beim Einsatz herkömmlicher Schutzmaßnahmen auf Transport- und Applikationsebene stoßen diese Konzepte an ihre Grenzen und bieten keinen ausreichenden Schutz.

#### Transportebene

Bisherige Sicherheitsmaßnahmen waren häufig auf die Transportebene konzentriert, wie z.B. der sog. Transport Layer Security (TLS) und Firewalls. Der im Internet weit verbreitete Sicherheitsstandard TLS und sein Vorgänger SSL können selbstverständlich auch für die Absicherung von Web-Services verwendet werden. Allerdings verbleiben bei der Nutzung dieses Ansatzes viele Probleme.

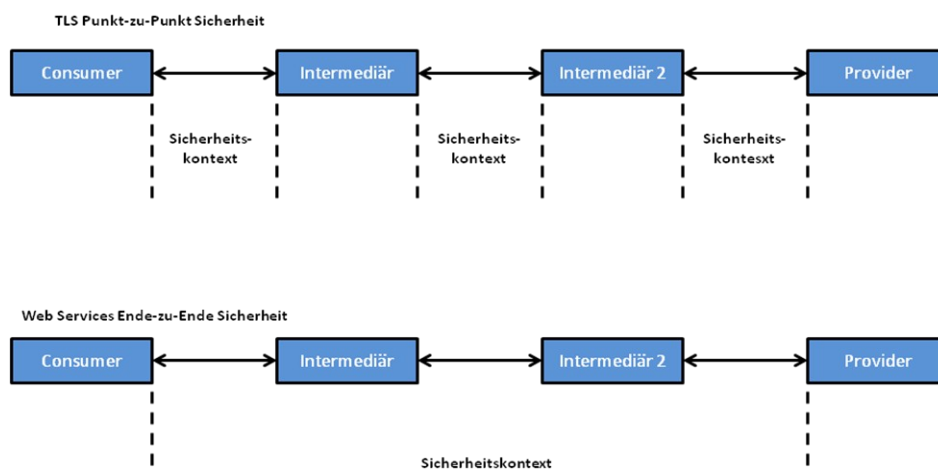


Abbildung 10: Vergleich der Sicherheitskontexte bei TLS/SSL und SOA

TLS stellt einen Sicherheitskontext zwischen einzelnen Punkten her und kann somit keine Ende-zu-Ende Sicherheit bieten, wie sie in SOA-Szenarien gefordert wird. Die Nachrichten, die durch die Verwendung von TLS ausgetauscht werden, sind nur auf dem Transportweg geschützt. Die Sicherheit kann nach dem Eingang im Zielsystem nicht mehr gewährleistet werden.

In klassischen SOA-Szenarien wird eine SOAP Nachricht über mehrere Knoten geleitet, die jeweils Operationen auf einzelne Elemente der Nachricht anwenden können. In diesem Fall ist es häufig notwendig, den Inhalt einzelner Elemente der Nachricht für einen bestimmten Knoten zu verschlüsseln und zu signieren. Außerdem ermöglicht eine teilweise Absicherung der Nachricht hohe Performancezugewinne. Dieses granulare Verschlüsseln und Signieren kann durch die Verwendung von TLS nicht erreicht werden, da die Nachricht nur komplett verschlüsselt werden kann. Für eine feingranulare Sicherheit muss auf andere Sicherheitsstandards zurückgegriffen werden. Es bleibt festzuhalten, dass mit dem Einsatz von TLS nur eine Sicherheit auf Transportebene realisiert werden kann, im Gegensatz zu Sicherheitsstandards, die Sicherheit auf Nachrichtenebene umsetzen. Neue Sicherheitsaspekte wie elementweise Signatur oder Verschlüsselung, sichere Kollaboration oder domänenübergreifende Sicherheitspolicies, werden hierdurch jedoch nicht abgedeckt.

Bei den häufig verwendeten Transport-Layer Firewalls bietet sich ein ähnliches Problem wie bei der Verschlüsselung auf dem Transportweg. Diese Firewalls bieten eine reine Sicherheit auf Transportebene und erlauben oder verweigern Verbindungen in das interne Netz. Im Normalfall findet der Nachrichtenaustausch über einen definierten Port statt. Oft wird im Rahmen einer SOA auf das HTTP-Protokoll zurückgegriffen, welches über Port 80 betrieben wird. Eine Firewall auf Transportebene kann nur definieren, ob ein Port geschlossen oder geöffnet ist. Es findet keine Inspektion von Nachrichten statt, die den offenen Port passieren. Da in der Regel alle Nachrichten über einen Port gesendet werden, kommen entweder alle Nachrichten oder keine Nachrichten an.

Für einen Einsatz von Firewalls im Rahmen einer Serviceorientierten Architektur muss auf deren Einsetzbarkeit in einem SOA-Umfeld geachtet werden. Um die Funktionalität einer gängigen Firewall auf XML-Nachrichtenaustausch abzubilden, wurden XML-Gateways und XML-Firewalls entwickelt. In die gleiche Gattung fallen "Application Level Gateways". Sie setzen im Gegensatz zu Firewalls ausschließlich auf der Ebene der High-Level Protokolle (HTTP, FTP, ...) an, dennoch gelten die folgenden Aussagen auch für Application Level Gateways. Diese Systeme befinden sich im Normalfall zwischen der Unternehmens-Firewall und den internen Systemen. XML-Firewalls konzentrieren sich auf das Abblocken von invaliden oder schädlichen XML-Dokumenten, während XML-Gateways weitere Mehrwerte bieten, wie z.B. Transformationen in den Nachrichten. XML-Firewalls und -Gateways bieten auf den ersten Blick eine recht einfache Möglichkeit, eine SOA auf der Basis von Web-Services abzusichern. Allerdings decken sie nur einen kleinen Teil der Sicherheitsanforderungen ab und können in einigen Szenarien gar nicht eingesetzt werden, da deren Filterlogik nicht hinreichend für die Eigenheiten von Web-Service- bzw. SOA-spezifischen Angriffen ausgelegt ist.

XML-Firewalls sind eine logische Weiterentwicklung von normalen Firewalls und daher an herkömmlichen Architekturen im Bereich der Informationstechnologie orientiert. Ein Unternehmen hat zwei Möglichkeiten, XML-Firewalls zu verwenden. Die erste Möglichkeit ist, ein zentrales System bereitzustellen, welches alle eingehenden Nachrichten überprüft; eine zweite ist es, vor jedem Web-Service des Unternehmens eine XML-Firewall zu installieren. Bei dem ersten Ansatz könnte es selbstverständlich zu Problemen bei der Skalierbarkeit kommen, und das Konzept würde keine Ende-zu-Ende Sicherheit bieten, sondern nur eine Punkt-zu-Punkt Absicherung wie bei SSL. Der zweite Ansatz wäre sehr kostenintensiv und würde eine Verwaltung erschweren, da nur selten eine zentrale Administration für (mehrere) Systeme dieser Art geboten wird.

Besonders bei verschlüsselten Inhalten in XML-Dokumenten stoßen XML-Firewalls und -Gateways an ihre Grenzen, wenn die Inhalte nicht für sie verschlüsselt wurden, sondern für einen Web-Service innerhalb des Unternehmensnetzes. Da XML-Firewalls oder -Gateways gesicherte Nachrichten nicht inspizieren können, könnte unter Umständen die XML-Schema-Validierung scheitern oder Schadcode die zentrale Absicherung passieren. Ein weiteres Defizit bei diesen Systemen ist die benötigte Flexibi-

lität im Rahmen einer SOA, da sich die Geschäftsprozesse, Sicherheitsmethoden und Standards häufig ändern. In der Regel werden Updates für XML-Firewalls und -Gateways im Rahmen eines Produktzyklus herausgegeben und der Anwender muss auf die nächste Version warten, um neue Anforderungen umsetzen zu können. Um eine umfassende Absicherung einer SOA zu bieten, muss jedoch gewährleistet werden, dass bei Änderungen das System schnell angepasst werden kann, ohne auf den nächsten Produktzyklus des Sicherheitssystems warten zu müssen. Auch seien die starren Sicherheitskonzepte innerhalb der monolithisch entwickelten Systeme erwähnt. XML-Firewalls und -Gateways gehen in der Regel davon aus, dass sich der Kommunikationspartner flexibel in der Wahl der Algorithmen zeigt, da XML-Firewalls und -Gateways selten eine Auswahl an gängigen Algorithmen bieten. Das Einbinden eigener (hochsicherer) Algorithmen ist daher ebenfalls nicht möglich, genau so wenig wie das Modellieren eigener gewünschter Sicherheitsabläufe. Es kann nur auf vorgefertigte und zumeist starre Abläufe zurückgegriffen werden.

### **Applikationssicherheit**

Klassische Schutzziele wie Authentifizierung und Autorisierung wurden in bisherigen Architekturen auf der Ebene von Applikationen durchgeführt. Durch den Einsatz einer SOA müssen diese Sicherheitsanforderungen an andere Stellen verlagert werden. Im Idealfall sollte in einer SOA die Anwendungslogik von der Sicherheitslogik getrennt werden. Dies bietet den entscheidenden Vorteil, dass Anwendungsentwickler weniger sicherheitsrelevante Aspekte beachten müssen, da viele Sicherheitsaspekte bereits durch vorgeschaltete Komponenten abgedeckt werden. Zusätzlich kann die Fragestellung auftauchen, auf welche Art und Weise Anwendungen integriert werden können, die neue Sicherheitsstandards nicht unterstützen. Hierfür muss eine getrennte Sicherheitslogik erstellt werden, um eine problemlose Integration in eine Serviceorientierte Architektur zu ermöglichen. Die Vorteile einer flexiblen SOA müssen von einer flexiblen Sicherheitsinfrastruktur untermauert werden, welche in hohem Maße auf die sich ändernden Anforderungen von innen und außen reagieren kann.

Zusätzlich gilt zu bedenken, dass nur ein ganzheitlicher Schutz zu einem gewünschten Sicherheitsniveau führt. Das gesamte Sicherheitsniveau ist nur so hoch wie die schwächste Absicherung eines Services. Denn wenn ein Angreifer diese Lücke ausnutzt, so ist der Gesamtprozess gefährdet. Verbleibt die Sicherheitslogik in der Applikation, so ist der Änderungsumfang enorm, da Änderungen aufwändig in jeder Anwendung (jeder Programmiersprache, jede Programmablauflogik,...) einzeln umgesetzt werden müssen. Ein zentrales Management und die damit verbundenen Vorteile entfallen ebenfalls.

## **4.2 Non Best Practices**

Der ausschließliche Einsatz von klassischen Schutzmaßnahmen ist nicht ausreichend, um eine angemessene Sicherheit in einer SOA zu gewährleisten. Eine Organisation sollte für die Umsetzung eines Sicherheitskonzeptes an SOA angepasste Sicherheitsstandards und Lösungen wählen. Diese Lösungen werden im Laufe dieses Kapitels vorgestellt, doch vorher werden typische Fehler analysiert, die während der Implementierung eines Sicherheitskonzeptes begangen werden können.

Ein häufiger Fehler bei der Implementierung von SOA Security Mechanismen beginnt bereits beim Projektstart. Häufig wird Security aus den ersten Pilot-Anwendungen ausgeblendet; dies bereitet im späteren Verlauf viele Probleme, da ad hoc (meist bei Start des Live-Betriebes) sicherheitsrelevante Entscheidungen getroffen werden müssen. Häufig sind diese dann unausgereift und unzureichend. Sicherheit sollte als ein zentraler Baustein von Anfang an mitgeplant und konzipiert werden. Dies betrifft neben der Formulierung von Sicherheitsanforderungen für einzelne Services auch die Planung geeigneter Maßnahmen sowie konkreter Lösungen. Welche Standards sollen eingesetzt werden, welche Schnittstellen zu vorhandenen Systemen müssen vorhanden sein, wie und wer administriert welche Teile der Sicherheitslösungen und vieles mehr sind Fragen, mit denen sich die Verantwortlichen auseinandersetzen müssen. Anhand dieser wenigen Beispiele zeigt sich bereits, dass die Implementierung von Sicherheit einem ganzheitlichen Ansatz folgen muss. Einzelne Komponenten abzu-

sichern, führt schnell zu unflexiblen und schwer zu beherrschenden Einzellösungen, mit der Folge, die durch die SOA-Einführung angestrebten Vorteile zu Nichte zu machen. Ebenso gilt gerade bei SOA-Implementierung auf Basis von Web-Services und anderen offenen Standards die gleiche Forderung nach Standardisierung und Interoperabilität wie für die Implementierung von Sicherheitsmechanismen. Proprietäre Formate und Methoden schränken diese ein und erschweren vor allem die unternehmensübergreifende Service-Kommunikation erheblich. Neben organisatorischen sind auch infrastrukturelle Entscheidungen zu treffen, insbesondere nach der Art der Implementierung von Sicherheit.

Dabei bietet es sich an, grundsätzlich nach Sicherheit als Service einerseits und Sicherheit als Infrastrukturkomponente andererseits zu unterscheiden. Es ist die Entscheidung, wie viel Sicherheitsmethodik in der Applikation verbleibt und welche Mechanismen gezielt aus Applikationen herausgehalten werden sollen. Sicherheit als Infrastrukturkomponente lagert die Sicherheit vor die Applikation. Somit muss die Anwendung selbst nicht angepasst werden.

Andererseits liegt es nahe, Sicherheitsmechanismen als Service zu implementieren, die von jedem berechtigten Service im im Sinne von SOA genutzt werden können. Dabei würde lediglich der Service-Aufruf innerhalb der Anwendungen (bzw. Service) erfolgen oder über eine Orchestrierungskomponente zentral gesteuert werden. Die gesamte Logik des Security-Services wäre in diesem Falle gekapselt und in seiner Funktionalität autonom.

Sich dieses Konzeptes zu vergegenwärtigen bietet die Möglichkeit, bereits während der Planung des SOA-Projekts notwendige Voraussetzungen zu schaffen.

### 4.3 Standardisierte Schutzmaßnahmen

Wie bereits dargestellt gelten im Kern auch für einen Web-Service die bekannten Anforderungen nach Vertraulichkeit, Authentizität von Sender und Empfänger, Datenintegrität und Verbindlichkeit bzw. Nicht-Abstreitbarkeit. Die dargelegten Schwächen herkömmlicher Sicherheitsmechanismen können mit dem Einsatz nachrichtenbasierter Web-Service-Security-Standards behoben werden. Diese Standards ergänzen die Web-Service-Kerntechnologien (SOAP, WSDL, UDDI, ...) um Sicherheitsfunktionalitäten.

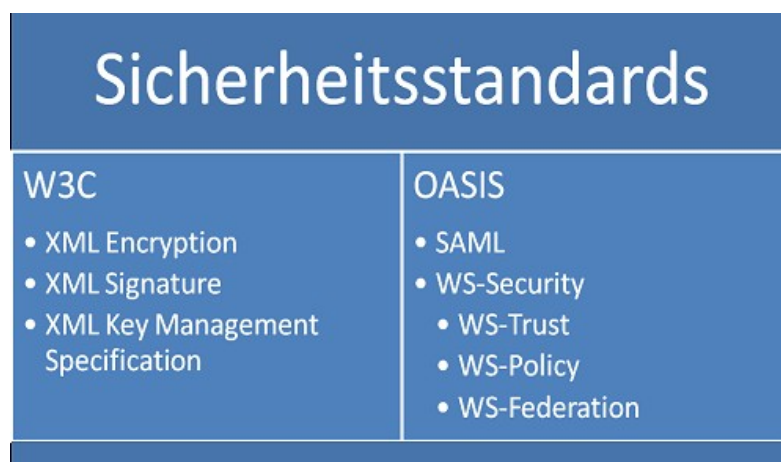


Abbildung 11: Übersicht der wichtigsten Standards im Web-Service-Kontext

Web-Service-Security-Standards bieten zahlreiche vom W3C und OASIS verabschiedete Methoden, um dienstorientierte Infrastrukturen mit der nötigen Sicherheit zu versehen. Hinzu kommt eine große Menge an Arbeitsgruppen, Initiativen und Gremien in diesem Bereich, die die Entwicklung neuer Techniken vorantreiben. Es überrascht nicht, dass dementsprechend auch die Menge an Technologien,

Methoden und Standards stetig wächst. In diesem Kapitel sollen die verbreiteten und etablierten Security-Standards kurz vorgestellt werden.

### **XML-Security (Encryption & Signature)**

Die Implementierung von Security-Funktionalitäten auf Nachrichtenebene bietet verschiedene Vorteile. So bleibt die Struktur des Dokuments erhalten und das XML-Dokument weiterhin wohlgeformt und valide. Außerdem können bspw. einzelne Teile des XML-Dokuments verschlüsselt werden, während andere lesbar bleiben. Neben Performance-Vorteilen kann somit der Header der Nachricht (vergleichbar mit einem Briefumschlag) erhalten bleiben, so dass die Nachricht weitergeleitet werden kann, ohne dass der Body (vergleichbar mit dem Briefinhalt) gelesen werden muss. Weiterhin sind Szenarien denkbar, in denen verschiedene Teile des Dokuments von verschiedenen Personen oder Institutionen signiert werden. Die Standardisierung ist dabei von zentraler Bedeutung. So beinhaltet jedes verschlüsselte oder signierte XML-Dokument alle Informationen in standardisierter Form, die zur Entschlüsselung bzw. Verifizierung nötig sind. Dadurch wird die durch den hohen Grad der Standardisierung im XML-Bereich gewonnene Interoperabilität der Systeme durch Sicherheitsbestrebungen beibehalten. Die Standardisierung im XML-Sicherheitsbereich ermöglicht die sichere Kommunikation verschiedener Kommunikationspartner, ohne dass man im Vorfeld die Protokolle abstimmen muss. Die Ergebnisse der Standardisierung wurden in folgenden W3C-Recommendations veröffentlicht:

#### **XML-Encryption**

Diese Spezifikation legt fest, wie und in welcher Form XML-Dokumente innerhalb der XML-Syntax verschlüsselt werden. Dabei ist es möglich:

- das gesamte XML-Dokument zu verschlüsseln,
- einzelne Elemente (und deren Unterelemente) zu verschlüsseln,
- den Inhalt eines XML-Elements zu verschlüsseln,
- die Verschlüsselung für mehrere Empfänger durchzuführen.

#### **XML-Signature**

Die XML-Digital-Signature-Spezifikation legt die Regeln und die Syntax fest, mit denen XML-Signaturen erzeugt werden. Wie eingangs erwähnt, können Signaturen auf Elementebene erzeugt werden, so dass einzelne Teile signiert werden können, während andere variabel bleiben. Werden die nicht signierten Teile verändert, behält die Signatur weiterhin ihre Gültigkeit. Dabei kann, ebenso wie bei der XML-Encryption, auf bekannte Hashverfahren (z.B. SHA-256) bzw. Verschlüsselungsverfahren (RSA) zurückgegriffen werden. Die Integration der Signatur in das XML-Dokument erfolgt gemäß der vom W3C veröffentlichten Spezifikation. Die benötigten zusätzlichen Einträge im Dokument (wie z.B. Hash-Code) werden in festgelegter Form am Anfang der Nachricht integriert.

#### **WS-Security**

Grundidee des von Microsoft, IBM und Verisign entwickelten Standards ist es, eine Spezifikation zu erstellen, die zeigt, wie die vorhandenen Basistechnologien XML-Encryption, XML-Signature und XML-Key Management in SOAP integriert werden können. Darin sind drei Mechanismen vereinigt:

- Security Token Propagation (Verteilung der sicherheitsrelevanten Information),
- Message Integrity (Integrität),
- Message Confidentiality (Vertraulichkeit).

Dabei werden Regeln und Standards definiert, wie und in welcher Form entsprechende Angaben in den Header von SOAP-Nachrichten eingebettet werden. Zum Verschlüsseln und Signieren von Nach-

richten wird auf die entsprechenden Basis-Technologien XML-Encryption bzw. XML-Signature zurückgegriffen. Dieser Standard bildet somit die Schnittstelle zwischen vorhandenen Security-Technologien und dem Web-Service Standard Format SOAP.

### SAML (Security Assertion Markup Language)

SAML ist ein Standard zur Authentisierung von Web-Services. Er ermöglicht die Einbringung erweiterter Sicherheitsinformationen in SOAP-Dokumente. Diese Informationen beziehen sich auf ein beliebiges "Subject", z.B. einen Service-Consumer, der einen Web-Service aufruft. Diese Informationen werden in Form von sog. Assertions eingebunden. Assertions sind festgelegte, zusätzliche Einträge in SOAP-Nachrichten.



**Abbildung 12: Einbettung von SAML-Informationen in eine SOAP-Nachricht**

Es sind bereits drei unterschiedliche Arten (Statements) standardisiert:

- Attribute Statement,
- Authentication Decision Statement,
- Authentication Statement.

Das Attribute Statement erlaubt vertrauenswürdige Aussagen über beliebige Attribute eines Subjects. Ein Beispiel für ein solches Attribut ist eine E-Mail-Adresse.

Das Authorization Decision Statement dient dazu, die Entscheidung, ob ein Subject autorisiert ist, auf eine Ressource (z.B. eine Datenbank) zuzugreifen, in standardisierter Form in die Nachricht zu integrieren.

Das Authentication Statement dient schließlich dazu, gegenüber dem WS-Security Standard erweiterte Authentifizierungsinformationen in eine SOAP-Nachricht einzubinden. Ein Beispiel für solche Informationen ist ein Benutzer-Zertifikat.

Mit SAML lassen sich auch auf standardisierte Weise speziellere Szenarien umsetzen. So lassen sich Assertions erzeugen, mit denen ein SSO (Single Sign On)-Szenario realisierbar ist. Der Vorteil hierbei ist, dass auch Web-Services, die keine direkte Verbindung zum Benutzer aufweisen, Zugriff auf die gesamten Authentifizierungsdaten des Benutzers haben.

### **XKMS – XML Key Management Specification**

Eine PKI wird verwendet, um digitale Zertifikaten erstellen, verteilen und prüfen zu können. Innerhalb einer PKI wird ein asymmetrisches Verschlüsselungssystem verwendet, das durch einen privaten und einen öffentlichen Schlüssel realisiert wird. Im Falle einer Signatur unterschreibt der Absender mit seinem privaten Schlüssel die jeweiligen Daten, und der Empfänger kann mit dem öffentlichen Schlüssel des Absenders die Echtheit überprüfen. Bei der Verwendung von Verschlüsselung werden die Daten vom Absender mit Hilfe des öffentlichen Schlüssels des Empfängers verschlüsselt und der Empfänger kann sie mit seinem privaten Schlüssel entschlüsseln.

Da im SOA-Umfeld die Kommunikation zwischen den einzelnen Services geschützt werden muss, stellt eine PKI einen wichtigen Aspekt für die Absicherung der Services dar. Mit Hilfe der PKI können die nötigen Schlüssel verteilt werden, um innerhalb des Systems die Sicherheitsanforderungen wie Vertraulichkeit, Authentizität, Integrität und Verbindlichkeit umsetzen zu können.

Eine PKI sollte aufgrund dieser Tatsachen als zentrale Komponente innerhalb der SOA-Sicherheitsinfrastruktur vorhanden sein. Allerdings besitzt eine PKI eine hohe Komplexität, die vor den Nutzern des Systems verborgen werden muss. Dies wird durch die Verwendung vom XKMS-Standard ermöglicht.

Die XKMS-Spezifikation beschreibt Möglichkeiten für den Austausch und die Überprüfung von Zertifikaten innerhalb einer PKI. Durch die Standardisierung wird zudem die Kompatibilität verschiedener PKI sichergestellt. Abstrahiert dargestellt bietet XKMS die Service-Fassade für eine PKI und kapselt deren komplexe Funktionalität.

XKMS gliedert sich in zwei Komponenten:

- XML Key Information Service Specification (X-KISS): Ermöglicht die Authentifizierung der Zertifikate mittels Abfrage beim entsprechenden PKI-Dienstleister. Im Einzelnen:
  - Bereitstellung eines Zertifikats,
  - Lokalisierung eines Zertifikats,
  - Validierung eines Zertifikats.
- XML Key Registration Service Specification (X-KRSS): Sorgt für das Management der Zertifikate, d.h. die Ausstellung, Widerrufung und die Wiederherstellung von Zertifikatsschlüsseln. Im Einzelnen:
  - Registrierung eines Zertifikats,
  - Annullierung eines registrierten Zertifikats,
  - Wiederherstellung eines Zertifikats,
  - Authentisierung der Anfrage.

### **WS-Trust**

WS-Trust ist eine Erweiterung des WS-Security Standards und ermöglicht SOAP-basierte Mechanismen für das Ausstellen, Erneuern und Bestätigen von Sicherheitstoken. Mit WS-Trust kann die ange-

fragte Stelle zudem Anforderungen für die Ausgabe von Sicherheitstoken (bspw. X.509 Zertifikate, Kerberos-Tickets oder BenutzerID) stellen. Voraussetzung ist daher, dass das Subject nicht gänzlich unbekannt ist. WS-Trust dient in erster Linie dazu, sichere Beziehungen zu etablieren, zu vermitteln und zu beurteilen. Anwendung findet WS-Trust auch zum Mapping verschiedener Credentials für unterschiedliche Applikationen und Zonen. So liefert WS-Trust beispielsweise ein X.509 Zertifikat nach "Vorlage" bzw. Erhalt eines Kerberos-Tokens desselben Benutzers.

### **WS-Policy**

Der W3C Standard WS-Policy ermöglicht es, Service-Providern und Service-Consumern, Sicherheitsrichtlinien an den Service-Request (resp. Reponse) zu stellen. Der Consumer (oder Provider) bestimmt dabei Anforderungen und Kriterien an die entsprechende Gegenstelle. So wird es z.B. möglich, dass der Anbieter eines Bestellservices eine verschlüsselte und signierte Nachricht fordert, dies den Konsumenten mitteilt und den Service Request nur bei Erfüllung der Richtlinien (und erfolgreicher Prüfung) akzeptiert. WS-Policy ermöglicht somit die Bekanntmachung und Durchsetzung von rudimentären Security-Policies auf Service-Ebene.

### **WS-Federation**

WS-Federation ermöglicht die Definition und den Zusammenschluss von Vertrauenszonen. Es setzt u.a. auf die Standards WS-Security, WS-Trust, WS-Policy und WS-Secure Conversation auf. WS-Federation benötigt daher Identity Provider, die ein Mapping unterschiedlicher User Credentials aus unterschiedlichen Quellen und über Zonen- und Domänengrenzen hinweg erlauben.

## **4.4 Nicht standardisierte Schutzmaßnahmen**

Neben den standardisierten Schutzmaßnahmen werden innerhalb einer Service-orientierten Architektur zusätzliche Maßnahmen benötigt, um einen ausreichenden Schutz zu gewährleisten. Ein kleiner Überblick über die zusätzlichen Sicherheitsmaßnahmen wird im folgenden Abschnitt geboten.

### **Single Sign-On**

Die Umsetzung eines Single Sign-On bietet neben dem offensichtlichen Vorteil, also der einmaligen Authentifizierung innerhalb eines Systemverbundes, weitere sicherheitsrelevante Vorteile. Aufgrund der Tatsache, dass die Authentifizierung nur einmal erfolgt, werden auch das Passwort oder sonstige Credentials nur einmal übertragen. Auf der administrativen Ebene ergibt sich der Vorteil, dass die Benutzerverwaltung an zentraler Stelle erfolgen kann. Beim Entfernen oder Hinzufügen von Benutzern muss dies somit nur an einer Stelle erfolgen, statt in vielen einzelnen Teilsystemen.

In einem SOA-Szenario authentifiziert sich ein Nutzer nicht direkt an den jeweiligen Backend-Systemen, sondern an zentraler Stelle. Durch diese Abschirmung der Hintergrundsysteme wird ein zentraler Aspekt einer SOA umgesetzt. Dies erfordert selbstverständlich, dass ein Single-Sign-On-System realisiert und zentral verwaltet wird.

SSO setzt ein Benutzer- und Credentialmapping zur Schaffung einer Entität voraus, die an zentraler Stelle verwaltet wird. Aufgaben der Administration sind daher die Anbindung, das Auslesen, das Importieren und Synchronisieren der Entitäten aus verschiedenen Quellen. Dabei ist weiterhin zu definieren, welches das führende System ist und wie Konflikte behandelt werden. Zudem müssen Werkzeuge zur Verfügung gestellt werden, die manuell oder über automatisierte Läufe ein Mapping (Beziehung schaffen) zwischen gleichen Entitäten erlauben.

Zudem sind dadurch erweiterbare Autorisierungsoptionen möglich, da die Benutzer-Entität im Sicherheitskontext grundsätzlich zur Verfügung steht. Dies bedeutet, selbst wenn die Anmeldung am

Backend-System, bspw. aus lizenzrechtlichen Aspekten auf Basis eines technischen Benutzers vorgenommen werden soll, können benutzerindividuelle Aktionen und Berechtigungen bereits vor dem Backend-System umgesetzt werden.

Durch die Abbildung der Anmeldung auf Nachrichtenebene (im Gegensatz zu sitzungsbasierten Systemen) ist zudem eine flexible Anmeldung in Formaten von Legacy-Anwendungen möglich. Durch Nachrichtentransformation vor den Systemen können systemspezifische Formate und Anmeldeinformationen geschaffen werden, die eine Autorisierung in der Sprache des Alt-Systems ermöglichen.

### **Web-Application-Firewall**

Anwendungen, die in öffentlichen Netzwerken zur Verwendung angeboten werden, sind häufig Gefährdungen wie SQL-Injections, Brute-Force- und Denial-of-Service-Attacken ausgesetzt. Gegen diese Angriffe werden Web-Application-Firewalls (WAF) eingesetzt. Innerhalb einer SOA kann ein Geschäftsprozess über eine Vielzahl von Systemen und Services ablaufen. Jeder einzelne dieser Services muss geschützt werden, um eine umfassende Absicherung zu gewährleisten. Hierfür muss auch in diesem Bereich ein Schutz vor den oben erwähnten Angriffen gewährleistet werden.

Problematisch in einer SOA-Umgebung ist die Verteilung von Services, was im Umkehrschluss bedeutet, vor jeden Service müsste eine WAF geschaltet werden, welche Angriffe erkennt und abwehrt. Eine solche Lösung wäre kostenintensiv, schwer administrierbar und deutlich überdimensioniert.

Dem SOA-Gedanken folgend ist es naheliegend, WAF-Funktionalität in Form von Services zu implementieren. Services innerhalb einer Zone oder Domäne wären in der Lage, einen WAF-Security, Content-Inspection und mehr als Service zu nutzen. Ein in die Sicherheitsanforderungen hinein modellierter Service-Aufruf bietet die Funktionalität, die sonst nur eine lokale WAF bieten würde. Die Service-Lösung wäre allerdings wesentlich kosteneffektiver, flexibler und besser skalierbar.

Zudem stoßen herkömmliche Web-Application-Firewalls bei der Behandlung von Web-Services schnell an ihre Grenzen. Gerade bei verschlüsselten SOAP-Nachrichten sind viele Lösungen nicht in der Lage, Angriffe zu identifizieren bzw. abzuwehren. Durch Nachrichtentransformation und dem Hinzufügen weiterer Sicherheitsmerkmale auf Nachrichtenebene gestaltet sich auch die Schema-Validierung als schwierig.

In der Modellierung muss es möglich sein, solche Aspekte zu berücksichtigen, bspw. durch eine Sicherheitslogik, die SOAP-Elemente erst entschlüsselt und den Klartext – oder Teile davon – zur Prüfung an eine Web-Application-Firewall übergibt. Dies setzt die gemeinsame Modellierbarkeit standardisierter (z.B. WS-Security) und nicht-standardisierter Sicherheitsmechanismen (z.B. WAF, Antivirus) in einem Werkzeug voraus.

### **Antivirus**

Auch in SOA Umgebungen müssen herkömmliche Bedrohungen wie Viren, Trojaner und Malware beachtet werden, da auch hier Dateien übertragen werden. Hierfür müssen Dokumente, Dateien und andere Inhalte vor der eigentlichen Bearbeitung durch einen Service auf potentiellen Schadcode kontrolliert werden. In den bereits vorgestellten Sicherheitsmaßnahmen findet dieser Aspekt keine Beachtung, trotzdem muss garantiert werden, dass ein Schutz gegen diese Bedrohungen auch innerhalb einer Service-orientierten Architektur gewährleistet wird.

Im Grunde handelt es sich bei Antivirus um nicht-standardisierte Sicherheitsmaßnahmen im SOA-Umfeld, und es gelten ähnliche Rahmenbedingungen, wie im Abschnitt Web-Application-Firewall beschrieben. Insbesondere Lizenz- und Betriebskosten lassen sich durch Antivirus-Services deutlich senken. Auch auf Agenten ergeben sich Vorteile durch die Möglichkeit der Modellierung von Virus-Prüfungen innerhalb einer gesamten Sicherheitslogik.

## PKI

Das Konzept einer PKI ist schon seit vielen Jahren bekannt und wird häufig verwendet, um sicherheitsrelevante Ziele umzusetzen. Eine PKI-Lösung muss seine Komplexität vor dem Anwender verbergen und eine einfache und nutzbare Lösung für die Umsetzung und Nutzung eines solchen Systems bieten. Innerhalb einer SOA werden geschäftskritische Informationen über Netzwerke ausgetauscht. Für diese Daten muss gewährleistet werden, dass sie auf angemessene Art und Weise geschützt werden. Zentrale Anforderungen sind die Integrität und die Vertraulichkeit dieser Daten, und diese Anforderungen können durch den Einsatz einer PKI umgesetzt werden. Zudem bildet eine PKI die Grundlage für den Einsatz von Web-Service-Security-Methoden, wie XML-Encryption und XML-Signature.

## 5. SOA-Security-Framework

Wie in den vorherigen Kapiteln dargestellt, ist die Eignung von SOA-Security-Ansätzen stark von unterstützten Architekturansätzen abhängig. Insbesondere die Frage nach der Implementierung der eigentlichen Security-Funktionalität ist dabei ausschlaggebend. Grundsätzlich lassen sich vier Ansätze unterscheiden, die unterschiedliche Charakteristika und damit verbundene Vor- und Nachteile aufweisen.

Eine Implementierung von Sicherheitsmechanismen innerhalb der Applikationen scheidet zumeist schon aufgrund der Inflexibilität und fehlenden Administrationsmöglichkeiten aus. Zudem ist die Forderung nach einer strikten Trennung von Applikationslogik und Sicherheitslogik nicht zu erfüllen. Auch der Aufwand bei Implementierung in jeder Applikation und jedem Service verhindert einen Einsatz in komplexen Prozessen.

Die Sicherheitsfunktionalität im Sinne eines zentralen Ansatzes aus der Applikation zu nehmen, ist ebenso nur bedingt geeignet. Zwar ist die Beherrschbarkeit und Administration hier gewährleistet, jedoch wirkt sich die monolithische Struktur auf Aspekte der Ausfallsicherheit, Lastverteilung und Skalierbarkeit aus.

Die Definition von Sicherheit als Infrastrukturkomponente hingegen ist für viele Anwendungsbereiche geeignet. Eine zentrale Administrierbarkeit vorausgesetzt, können lokale Agenten Sicherheitsfunktionen vor den Applikationen und Services umsetzen. Allerdings ist ein vollständig infrastrukturell basierter Ansatz in manchen Fällen schwer zu realisieren. Häufig müssen in der Praxis nicht alle Sicherheitsfunktionen an allen Punkten dezidiert zur Verfügung stehen. Eine Absicherung an den Übergängen des internen zum externen Netz alleine (wie bei Firewalls üblich) widerspricht dem Ansatz der "End-to-End-Security". Aus Lizenz-, Wartungs- und Performanceaspekten macht es Sinn, definierte Funktionen als Services systemweit zu implementieren.

Ein SOA-Security-Ansatz sollte in der Lage sein, Sicherheitsfunktionalität flexibel und anforderungsgerecht, wahlweise als Infrastrukturkomponente oder als zentralen Security-Service zur Verfügung zu stellen.

In diesem Kapitel wird ein Framework für SOA-Security vorgestellt, mit dessen Hilfe die vorher geforderten Sicherheitsanforderungen umsetzbar sind. Zunächst werden Konzepte für Security-Mechanismen im SOA-Umfeld eingeführt. Anschließend werden die Komponenten beschrieben, aus denen ein solches SOA-Security-Framework besteht. Auf zentrale Sicherheits-Services und Sicherheitsregeln wird in den folgenden Teilen eingegangen.

### 5.1 Komponenten eines SOA-Security-Frameworks

Grundsätzlich gelten für die Absicherung von SOA-Infrastrukturen zunächst ähnliche Anforderungen wie bei der Absicherung herkömmlicher Systeme – teilweise in SOA-typischen Ausprägungen. Wie

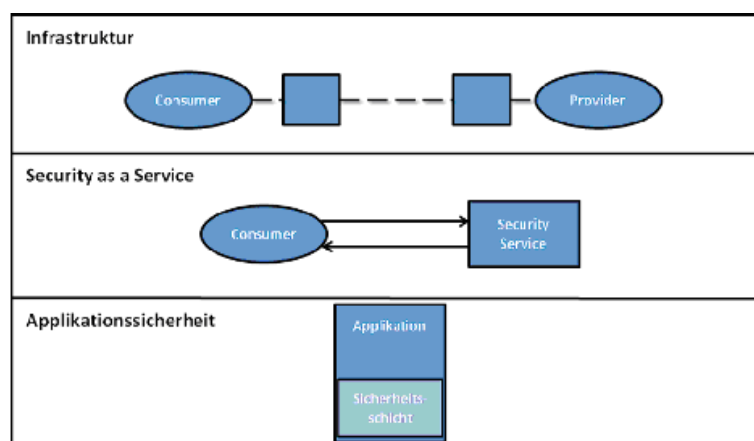
bei herkömmlichen Systemen üblich, müssen Identitäten und Berechtigungen verwaltet, Benutzer authentisiert, Zugriffsberechtigungen überprüft, Zugriffe und Zugriffsversuche aufgezeichnet und Daten mittels kartographischer Methoden signiert (Sicherstellung von Integrität) oder verschlüsselt (Erreichen von Vertraulichkeit) werden. Gerade in verteilten Umgebungen werden zudem systemweite Sicherheitsrichtlinien (Policies) benötigt, deren Einhaltung (Compliance) es permanent zu überwachen gilt. Der Grundgedanke von SOA wirft dabei weitere Anforderungen auf, die durch die Integration verteilter und teilweise heterogener Systeme insbesondere zu beachten sind. Zu nennen sind an dieser Stelle bspw. die Umwandlung von Identitäten und deren Formate, der Einsatz von Industriestandards und die Gewährleistung der Wiederverwendbarkeit von Security Diensten.

Solche Anforderungen können durch einen Framework-Ansatz erfüllt werden, der mindestens folgende sicherheitsspezifische Funktionalitäten bietet.

1. Verwaltungen von Identitäten (Identity Management),
2. Authentisierung und Autorisierung,
3. Absicherung der Nachrichten (Message Protection) - Kryptographie (Verschlüsselung und Signaturen) und Datenschutz (Privacy),
4. Sicherheitsrichtlinien – Durchsetzung und Entscheidung (Enforcement und Decision),
5. Einhaltung der Sicherheitsrichtlinien (Compliance),
6. Überprüfung (Auditing).

Analog zum Business-Layer bietet es sich an, ein Security-Layer bereitzustellen, welcher die Administration, Verteilung und Orchestrierung der Security-Logik in einem Gesamtsystem abbildet. Kernkomponente der Umsetzung ist daher die Sicherheitsrichtlinie, die definiert, an die Umsetzer (Policy-Enforcement-Points) verteilt und von diesen durchgesetzt werden muss. Zur Realisierung werden daher verschiedene Basiskomponenten benötigt. Hierzu gehören insbesondere die zentrale Administration für die Erstellung und Verteilung systemweiter Security-Regeln (Policy-Administration und -Distribution) sowie die Durchsetzung der Sicherheitsrichtlinie (Policy-Enforcement) durch lokale Agenten. Die Umsetzung kann dabei Agenten-seitig durch verschiedene Architekturansätze erfolgen.

## 5.2 Architekturansätze (Lokale Ebene)



**Abbildung 13: Ebenen eines SOA-spezifischen Security Framework**

Die Umsetzung der Sicherheitsrichtlinie für Services und Applikationen kann prinzipiell auf drei Arten erfolgen:

- durch die Anwendungslogik,
- durch die Verwendung von Security-Services (Security as Service),
- durch eine Proxyschicht vor den Services (Security as Infrastruktur).

In einem Nicht-SOA-Umfeld wird die Sicherheit durch Schutzmaßnahmen des Computersystems und einer Sicherheitslogik in den Anwendungen abgebildet. Dies ist in einem SOA-Umfeld jedoch nicht ausreichend. Eine Anforderung an eine Service-orientierten Architektur ist die Wiederverwendbarkeit von Services und die Zentralisierung derselben. Aus diesem Grund müssen die Berechtigungsverwaltung und die Security-Administration außerhalb der Service-Logik erfolgen.

Die folgenden Ansätze sind i.d.R. durch ein Agentenkonzept zu realisieren. Agenten oder Connectoren sind lokale Laufzeitkomponenten, die (Security-) Funktionalität umsetzen; als Fachterminus wird häufig Policy-Enforcement-Point verwendet. Dies kann die Signatur einer SOAP-Nachricht oder Verschlüsselung ebenso sein, wie die Bereitstellung einer Primärauthentisierung mittels Smart Cards. Agenten können daher im Sinne eines Proxy Konzepts (vgl. Security as Infrastruktur) "vor" den Systemen implementiert sein, oder als (Security-) Service-Provider (vgl. Security as Service) innerhalb einer Domäne oder Zone stehen und Funktionalität bereitstellen. Grundsätzliche Unterschiede bestehen oftmals in der Anbindung an zentrale Komponenten, wie Administration, Datenbestand, etc. Im Falle von Mini-Agenten liegt lokal kein Datenbestand vor, und zum Treffen von Entscheidungen (bspw. für den Zugriff eines Subjekts auf eine Ressource) muss eine zentrale Komponente angefragt werden. Liegt die Entscheidungsgrundlage in Form lokaler Richtlinien vor, können Agenten im Betrieb auch autark von zentralen Komponenten agieren.

### Security as a Service

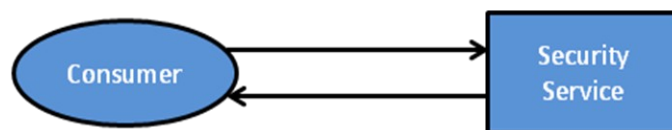


Abbildung 14: Sicherheit als eigenständiger Service

In einer SOA-Infrastruktur liegt es nahe, Sicherheit ebenso wie Anwendungslogik als wieder verwendbare Services bereitzustellen. Security-Services bedeutet dabei die Bereitstellung von Sicherheitsmechanismen als Dienst, die von allen Anwendungen/Komponenten über standardisierte Schnittstellen genutzt werden können. Über ein solches Konzept lassen sich viele Sicherheitsanforderungen realisieren. Ein solches Konzept trägt insbesondere der Wiederverwendbarkeit Rechnung, da viele Sicherheitsmechanismen von verschiedenen Anwendungen gleichartig genutzt werden können. Beispiele hierfür sind Web-Application-Firewall & Content-Filtering, Anti-Virus, Signatur oder Public-Key-Infrastruktur (PKI) via XKMS. Je nach Last- und Nutzverhalten sowie unter Berücksichtigung infrastruktureller (Domain-) Strukturen sind zentralisierte Services bereitzustellen.

### Security als Infrastruktur

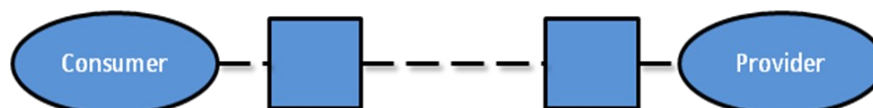


Abbildung 15: Sicherheit als integraler Bestandteil der Infrastruktur einer SOA

Security als Infrastruktur wird im Gegensatz zu den SOA-Security-Services genutzt, wenn Sicherheitsmechanismen von der Anwendungslogik getrennt werden sollen. Häufig ist es wünschenswerter, Systeme und Aufrufe durch eine Infrastrukturkomponente (Appliance oder Proxy Ansätze) zu schützen.

Ein Proxy ist als Dienstprogramm zu verstehen, welches als Mittler zwischen Netzen, Anwendungen und Services fungiert. Durch den Einsatz von Proxies für die Umsetzung der SOA-Sicherheit kann eine vollständige Plattform- und Systemunabhängigkeit erreicht werden. In diesem Szenario befinden sich Proxies zwischen den einzelnen Knoten der Architektur, so dass sie sicherheitsrelevante Informationen innerhalb der Nachrichten hinzufügen, bearbeiten oder kontrollieren. Durch die Verwendung dieses Konzeptes wird eine klare Trennung zwischen Anwendungs- und Sicherheitslogik geboten. Die Sicherheitsmechanismen können dann die benötigten Transformationen in der Nachricht durchführen, bevor diese anschließend für ihren eigentlichen Bestimmungszweck verwendet wird. Insbesondere Message-Transformation (bspw. für SSO an Legacy-Systemen) und Collaboration-Security (bspw. WS-Security, Signatur, Authentisierung) sind sinnvoll über einen Infrastrukturansatz zu realisieren. Folgende Services sind typischerweise durch einen solchen Ansatz umzusetzen:

**Authorization-Services** - entscheiden über den Zugriff eines Consumers auf eine Ressource. Grundlage hierfür sind Sicherheitsregeln (Security Rules).

**Message-Protection-Services** - schützen die eigentlichen Nachrichten ganz oder bis auf Statement-Ebene hinunter. Standards hierfür sind u.a. WS-Security und SAML.

**Interoperation-Service** – allgemeine standardisierte Web-Service-Schnittstellen zu Systemen. Führt benötigte Nachrichtentransformationen zur Anbindung von Systemen und Altanwendungen (Legacy Applications) durch.

Anforderungen wie Vertraulichkeit, Integrität, Authentizität und Verbindlichkeit können durch die Verwendung von WS-Security-Standards (WS-Security, XML-Encryption und Signature, etc.) auf Nachrichtenebene umgesetzt werden. Dafür bietet sich die Implementierung von lokalen Proxy-Agenten im Sinne einer Infrastruktur an. Für die Authentisierung bietet SAML 2.0 standardisierte Methoden, die eine Interoperabilität von Authentisierungsinformationen ermöglichen. Ein entsprechendes Entity- und Credential-Mapping vorausgesetzt, sind somit auch Federation-Ansätze umzusetzen. Ebenso für die Autorisierung für Service Aufrufe ermöglicht die zentrale Komponente die Verwaltung von Rollen und Rechte für den Zugriff. Am Policy-Enforcement-Point (vor oder auf den Service/Backendsystemen) kann auf deren Basis die Entscheidung getroffen werden. Liegt keine entsprechende Berechtigung vor, lehnt der Agent den Zugriff auf eine Ressource ab. Bei der Integration von Legacy-Systemen kann der Agent zusätzlich eine Nachrichtentransformation vor dem System vornehmen. Eine Änderung des Alt-Systems ist nicht notwendig, da der Agent das System im erwarteten Format anspricht.

### 5.3 Zentrale Administration (Zentrale Ebene)

Gerade in verteilten und heterogenen Strukturen mit einer Vielzahl beteiligter Systeme, Anwendungen, Services, Identitäten und Komponenten ist eine zentrale Komponente zur Administration unerlässlich. Security muss schließlich im gleichen Maße beherrschbar und flexibel sein, wie die Services und deren Orchestrierung selbst. Benötigt wird eine zentrale Administrationskomponente, die es erlaubt, eine systemweite Richtlinien-Administration und Richtlinien-Distribution zu gewährleisten. Die Leistungsfähigkeit der Administration ist dabei ein kritischer Faktor zur Zielerreichung des Gesamtprojekts. Änderungen in Services müssen durch die Security-Administration schnell und einfach abgebildet werden können.

Dem SOA-Ansatz folgend bietet es sich an, Administrationsfunktionen als Services zur Verfügung zu stellen, die sich innerhalb bestehender Infrastrukturen (bspw. Application-Server, Datenbanken, etc.) aufbringen lassen.

Zentrale Dienste umfassen mindestens:

## **Identity Services**

Benutzer, Rechte und Rollen verwalten ist eine Kernaufgabe der Administration. In einer SOA-Umgebung ist dabei jedoch die Hauptaufgabe Integrationschnittstellen und -Werkzeuge zur Verfügung zu stellen. Die im Unternehmen vorhandenen Benutzerverzeichnisse, Identity- Management-Systeme und Metadirectories müssen angebunden werden, um somit eine Nutzbarkeit vorhandener Entitäten in der SOA-Umgebung zu erreichen. Die Anbindung sollte durch automatisierte Mechanismen für den Import und die Synchronisation unterstützt werden.

## **Authentication-Services**

Es müssen Verfahren bereitgestellt werden, mit denen sich Benutzer, Server und Services ausweisen können. Neben der eigentlichen Unterstützung von Token, User/PW, Zertifikate, etc. liegt die besondere Anforderung im SOA-Umfeld auf dem Mapping von Entitäten und zugehöriger Credentials. Dies bildet die Grundlage für SSO-Anwendungen auf Agenten Seite und ermöglicht zudem die Anmeldung an Legacy-Systemen. Dies setzt in vielen Fällen auch die Formatkonvertierung von Credentials- und Anmeldeinformationen voraus.

## **Audit-Services**

Es muss eine zentrale Reporting- und Auditing-Infrastruktur bereitgestellt werden, in denen relevante Transaktionen festgehalten und archiviert werden. Ebenso umfasst ein Auditing die Speicherung und Bereitstellung der Historie aller Datentransfers. Dabei setzt ein rechtssicheres Auditing die Gewährleistung von Integrität, Authentizität und Nicht-Abstreitbarkeit voraus, die mit entsprechenden Methoden wie Verschlüsselung und Signatur erreicht werden kann.

## **Modellierungs- und Orchestrierungs-Funktionen**

Neben primären Services, die den sicheren Rahmen zur SOA-Security bilden, muss die zentrale Administration Modellierungs- und Orchestrierungswerkzeuge anbieten, mit denen Sicherheitsmechanismen und Services auf Basis der Business-Logik erschaffen und administriert werden können. Analog zur Business-Process-Modellierung muss die Administration in der Lage sein, im Sinne der Orchestrierung Sicherheitsmechanismen zu Services "zusammenzustecken". Es sind Werkzeuge gefordert, die es erlauben, WS-Security-Methoden, wie Verschlüsselung und Signatur, ebenso in einen Security-Workflow zu integrieren, wie bspw. Prüfungen nach Viren oder SQL-Injections. Eine Unterstützung von Beschreibungssprachen (bspw. BPEL) und entsprechende Schnittstellen ermöglichen dabei die Einbindung externer Modellierungswerkzeuge bzw. die Datenübernahme aus Business Prozessen.

## **Policy Roll-Out & Distribution**

Die Verteilung der erzeugten Security Regeln an lokale Agenten setzt einen Automatismus sowie Workflows voraus, die den Anforderungen und Systemgegebenheiten im Einsatzszenario angemessen sind. Hierbei sollten sowohl Push- als auch Pull-Mechanismen angeboten werden. Im Betrieb bietet sich die Verwendung von Push-Methoden an, die die Komponenten bei Änderungen mit (Delta-) Updates der Sicherheitsregeln versorgen. Insbesondere bei Push-Methoden ist auf eine effiziente Update-Logik zu achten, um den Datenverkehr auf ein Minimum zu beschränken. Dies gilt ebenso für Pull-Mechanismen, mit denen ein vom Agenten ausgelöstes Update forciert werden kann. Neben Delta-Updates ist es in manchen Fällen sinnvoll, ein Full-Update anzubieten, welches die Laufzeitkomponenten vollständig neu konfiguriert. Manuelle Updates einzelner Agenten sollten dabei ebenso unterstützt werden wie zeitgesteuerte automatische Updates. Ein entsprechender Service der Administrationsengine sollte zudem ein Log- und Monitoringsystem des Regel-Rollouts zur Verfügung stellen.

## Erweiterbarkeit, Patch- und Updatefähigkeit

Die flexible Unterstützung neuer Funktionalitäten ist eine fundamentale Forderung für eine ganzheitliche SOA-Security-Lösung. Gerade im Umfeld neuer Security-Standards, heterogener Systemumgebungen mit vielen Beteiligten, muss die Lösung in der Lage sein, flexibel auf neue Anforderungen angepasst zu werden. Ein Adapterkonzept mit definierten Schnittstellen ermöglicht die schnelle Einbindung neuer Funktionen. Ein Adapter ist in diesem Sinne als modulares Funktionsbündel definiert, welches alle von ihm benötigten Werkzeuge, Laufzeitcode und Schnittstellen gekapselt in das Gesamtsystem einbringt. Dabei ist insbesondere die Verteilung neuer Adapter ein kritischer Punkt. Eine erweiterte Definition der Richtlinien-Distribution ermöglicht die Verteilung einer neuen Funktionalität über bewährte Mechanismen. Dementsprechend werden neue Adapter in die Administration geladen. Die neue Funktionalität steht bei der Zentralen Administration (Modellierungsebene) direkt zur Verfügung und wird über Security-Regeln zur Agentenseite verteilt.

Die im vorherigen Kapitel diskutierten Sicherheitsanforderungen lassen sich durch die vorgestellten Ansätze erfüllen. Das Zusammenspiel zwischen zentraler Administration mit lokalen Agenten und (Security-) Service-Providern ermöglicht die Administration, Modellierung und Durchsetzung systemweiter Sicherheitsrichtlinien.

Auch weitere nicht-standardisierte Anforderungen lassen sich abbilden. Typische Beispiele sind Implementierung eines zentralen Anti-Virus oder Web-Application-Firewall-Service.

Auf die gleiche Weise kann das Zertifikatsmanagement realisiert durch einen XKMS-Service aufgesetzt werden. Zertifikate stehen dann sämtlichen Agenten zur Verfügung, kryptographische Operationen (z.B. für XML-Encryption, Signature) können durchgeführt und deren Validität überprüft werden. Grundsätzlich hängt die Wahl des Architekturmodells stark von den Einsatzbedingungen einzelner Funktionalitäten ab. So kann es in manchen Fällen auch sinnvoll sein, einen Signatur-Service zentral (Security as Service) oder Content-Inspection (WAF) vor einzelnen Systemen zu realisieren.

Verfügbarkeit und Ausfallsicherheit ist durch die Unterstützung bekannter Methoden wie Load-Balancing, Application-Server-Konzepte und lokalen Sicherheitsregeln (vgl. dazu auch Abschnitt "Sicherheitsregeln") zu erreichen.

Die Administrierbarkeit ist durch den Einsatz einer zentralen Komponente mit weitreichenden Integrationsfunktionen zu gewährleisten. Für die Modellierung und Definition der Agentenfunktionalität und deren Abbildung von Sicherheitsregeln, benötigt die zentrale Komponente Zugriff auf vorhandene Benutzer, Server, Rollen und Rechte sowie Zertifikate. Reporting, Logging und Audit lässt sich mittels der zentralen Komponente realisieren, da eine Datensammlung und -aufbereitung über entsprechende Funktionalitäten durchgeführt werden kann.

## 5.4 Sicherheitsregeln

Sicherheitsregeln sind die Steuerungseinheit der gesamten Security-Logik und beinhalten alle Informationen zur Durchsetzung der Security-Policy. Sicherheitsregeln sind somit das Ergebnis sämtlicher Administrations-, Konfigurations- und Modellierungsvorgänge und gleichzeitig die Grundlage des Verhaltens lokaler Komponenten. Sie sind damit die Schnittstelle zwischen Policy-Administration und Policy-Enforcement. Dafür müssen die zentral erzeugten Regeln an lokale Komponenten übertragen werden (vgl. Abschnitt – Roll-Out und Distribution). Zur Umsetzung von Sicherheitsregeln bieten sich grundsätzlich strukturierte und erweiterbare Formate wie bspw. XML an. So kann zum einen eine Trennung unterschiedlicher Teile an verschiedene Empfänger als auch die Integration in externe Process-Modelling-Tools gewährleistet werden. Für letzteren Punkt ist es sinnvoll, vorhandene Formate oder Standards (z.B. Business Process Execution Language (BPEL)) zu verwenden.

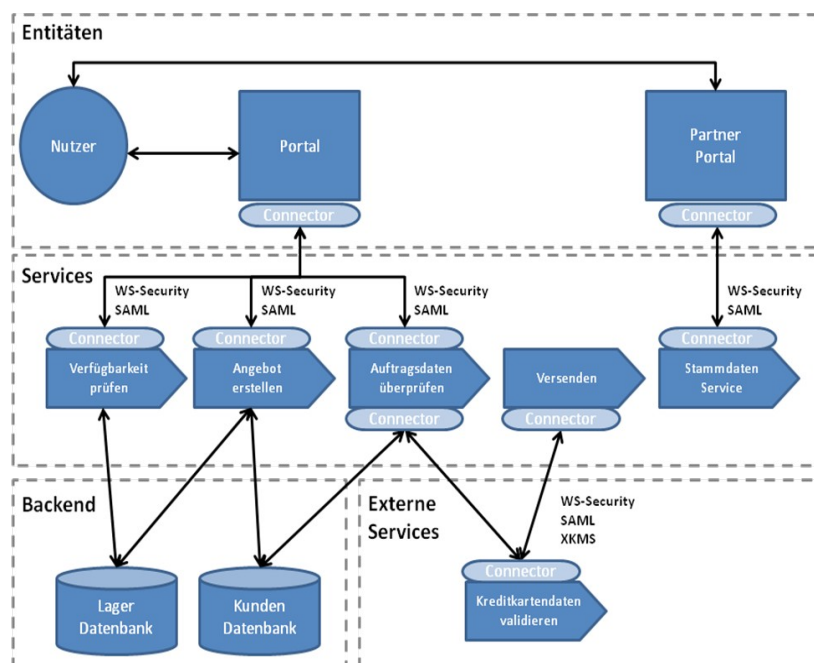
Um innerhalb einer SOA Sicherheitsrichtlinien umzusetzen, bietet sich die Verwendung von Standards wie WS-Policy und den darauf basierenden Standard WS-Security-Policy an. Mit Hilfe dieser Standards können die Sicherheitsrichtlinien, die eine Organisation für die Nutzung ihrer Web-Services fordert, definiert und durchgesetzt werden. Der Nutzer der Services muss diese Richtlinien erfüllen, um den Service nutzen zu können. Auch an dieser Stelle wird wiederum deutlich, wie essentiell die Verwendung von offenen Standards im SOA-Umfeld ist, um eine Interoperabilität mit externen Systemen zu gewährleisten. Die Verwendung von proprietären Standards in diesem Umfeld würde zu Problemen bei Geschäftsprozessen führen, die über die Organisationsgrenzen verlaufen und Grundideen, die hinter SOA stehen, ignorieren.

Die Umsetzung eines Security-Rule-Konzepts umgeht damit die in einer SOA unerwünschte Single-Point-of-Failure-Problematik, da den (produktiven) Laufzeitkomponenten neben der Funktionalität auch die (Entscheidungs-) Logik lokal vorliegt und damit nicht auf Antworten von zentralen Komponenten angewiesen sind. Durch diesen Architekturansatz wird gewährleistet, dass bei einem Ausfall eines sicherheitsrelevanten Systems das Gesamtsystem nicht gefährdet ist.

Das Konzept von Sicherheitsregeln lässt sich zudem weiter fassen und bietet auch die Möglichkeit, neben der reinen Logik, auch Funktionalität selbst zu kapseln. Aufbauend auf einem Adapterkonzept können so neue Funktionalitäten ebenso wie Patches und Updates auf lokale Komponenten ausgerollt werden.

## 6.5 Ausgewählte Lösungen für das Beispielszenario

Im Folgenden sollen für das skizzierte Beispielszenario und den einzelnen Service Requests bzw. Service Responses konkrete Umsetzungen dargestellt werden. Die Darstellung hat dabei keinen Anspruch auf Vollständigkeit, sondern soll die beispielhafte Anwendung vorgestellter Maßnahmen aufzeigen.



**Abbildung 16: Beispielszenario einer Service-orientierten Architektur mit Portalanbindung unter Verwendung externer Dienste**

Für den Service „Verfügbarkeit“ wurden in der Analyse der Bedrohungspotentiale Forderungen nach Authentizität, Integrität und Vertraulichkeit der Daten identifiziert. Der Web-Service muss ent-

sprechend abgesichert werden. Dies kann durch Anwendung von WS-Security-Methoden auf den Service Request und Service Response geschehen. Auf der Modellierungsebene wird die Security-Policy definiert, indem ein gültiges SAML-Statement zur Authentisierung des Service-Consumers gegenüber dem Service eingefügt wird. Eine Prüfung des SAML-Statements auf Service-Provider-Seite, stellt die Authentizität des berechtigten Zugriffs sicher. Neben der Sicherstellung der Authentizität, gelten für die Antwort des Services weitere Sicherheitsanforderungen. Mittels einer Signatur und der Verschlüsselung sensitiver Teile der Response können diese erfüllt werden. Der Service-Consumer kann nach erfolgreicher Prüfung sicher sein, dass die Antwort unverfälscht von dem angefragten Service kommt. Man-in-the-Middle-Attacken sind ausgeschlossen, da eine Verletzung der Integrität die Prüfung der Signatur fehlschlagen ließe und sensitive Teile für den Angreifer nicht lesbar wären.

Zur Implementierung bietet sich ein Proxy-Konzept (Security als Infrastruktur) an. Lokale Agenten hinter bzw. vor dem Service transformieren die Nachrichten und Prüfen nach definierten Sicherheitsmerkmalen. Nach dem vorgestellten Muster können auch die Verbindlichkeit und Vertraulichkeit für den "Angebot erstellen"- und "Auftragsdaten prüfen"-Service abgebildet werden.

Weitere Aspekte eines ganzheitlichen Sicherheitssystems werden an dem Beispiel "Kreditkartendaten validieren" deutlich. Im Gegensatz zu den bisher vorgestellten unternehmensinternen Prozessen handelt es sich hierbei um einen unternehmensübergreifenden Prozess, da der Service von einer externen Stelle angeboten wird. Zum einen kommen dabei höhere Sicherheitsanforderungen in Betracht, aber zum anderen müssen Federation Konzepte umgesetzt werden.

Der gesamte Kundenauftrag als SOAP-Dokument enthält auch die für den Kreditkarten Service interessanten Angaben bzgl. des Zahlungsmittels. Diese Daten sollten auf Elementebene für den Kreditkarten-Service verschlüsselt werden. Damit ist sichergestellt, dass andere Teilnehmer der Auftragsabwicklung nicht unbefugten Zugriff auf diese Daten bekommen. Zwischenspeicherungen auf internen sowie externen Systemen sind daher auch als unkritisch einzustufen, da die Sicherheit auf Nachrichtenebene erfolgt und der Sicherheitskontext jederzeit erhalten bleibt. Der für die Verschlüsselung notwendige öffentliche Schlüssel (Zertifikat) kann von einem zentralen XKMS-Service lokalisiert und angefordert werden. Für die benötigte Integration von Federation-Konzepten kann ebenfalls SAML verwendet werden. Hierbei wird im Vorfeld eine Vertrauensbeziehung zwischen dem Internetshop und dem Kreditkartenservice hergestellt. Somit muss nicht mehr genau definiert werden, welcher Service des Internetshops auf die Services der Kreditkartengesellschaft zugreift.

Die Vertrauenswürdigkeit externer muss zudem bei dem skizzierten Stammdatenservice gewährleistet sein. Wird der eingehenden Service Anfrage ein SAML-Statement eingefügt, indem die Vertrauenswürdigkeit von einer bekannten Stelle bestätigt wurde, gilt die Anfrage als authentisch.

Nicht nur der Schutz der Zahlungsdaten der Kunden ist ein Aspekt bei der unternehmensübergreifenden Service Kommunikation. Im Falle eines kostenpflichtigen Services muss sichergestellt sein, dass eine kostenpflichtige Prüfanfrage nicht von Dritten wiedereingespült wird und dadurch Kosten verursacht werden. Ein SAML-Token mit entsprechendem Zeitstempel und evtl. Message-ID ermöglicht die Festlegung einer Gültigkeitsdauer für eine Nachricht. Ist diese abgelaufen, akzeptiert der Service-Anbieter die Nachricht nicht mehr. Des Weiteren muss es für den Service Consumer und Provider möglich sein, die Inanspruchnahme des Services revisionssicher nachzuhalten. Daher kann bspw. auf Consumer-Seite nach der Prüfung der Authentizität und Integrität der erhaltenen Kreditkartenauskunft ein Auditservice die Inanspruchnahme protokollieren und das Protokoll wiederum digital signieren. So ist eine rechtssichere Protokollierung von Service-Aufrufen möglich.

Wie dem Beispiel zu entnehmen, lassen sich Sicherheitsanforderungen von SOA mit der konsequenten Anwendung standardisierter WS-Security-Mechanismen umsetzen. Authentizität, Vertraulichkeit, Integrität und Verbindlichkeit als Kernthemen können standardisiert auch über Unternehmensgrenzen hinweg durchgesetzt werden. Es zeigt zudem, dass Architekturansätze einen entscheidenden Einfluss auf die Realisierung haben, da Policy-Enforcement-Points an vielen Prozessschritten zu finden sind. Obwohl im obigen Szenario Änderungen in der Konfiguration ausgeklammert wurden, sei an dieser

Stelle nochmals auf die Notwendigkeit einer zentralen Administration hingewiesen, die die Flexibilität von Services mit trägt. Auch die Upgrade- und Erweiterungsfähigkeit im Hinblick auf die Unterstützung neuer Standards ist essentiell für eine SOA-Security-Lösung – nicht zuletzt um eine unternehmensübergreifende Interoperabilität zu gewährleisten.

## 6. Glossar

- Akteur/Actor: Teilnehmer in einer SOAP-basierten Kommunikation – dies kann sowohl ein Endpunkt als auch ein Intermediär sein, der über eine URI erreichbar ist und unter dieser er SOAP-Nachrichten entgegennimmt. Menschliche Benutzer oder Browser sind keine Akteure.
- BPEL4WS: Business Process Execution Language for Web-Services
- BPML: Business Process Modeling Language
- CDL4WS: Choreography Definition Language for Web-Services
- Claim: Verifizierbare Aussage über ein Subject, die für Sicherheitszwecke eingesetzt wird. Beispiele sind kryptographische Schlüssel oder digitale Zertifikate.
- Credential: Ein Credential ist ein Identifikations- und Berechtigungsnachweis, mit dem ein Subjekt nachweisen kann, dass er auf bestimmte Informationen oder Ressourcen zugreifen oder bestimmte Aktionen ausführen darf.
- Enterprise Application Integration (EAI): Enterprise Application Integration (EAI) ist ein Konzept zur unternehmensweiten Integration der Geschäftsfunktionen entlang der Wertschöpfungskette, die über verschiedene Applikationen auf unterschiedlichen Plattformen verteilt sind, und die im Sinne der Daten- und Geschäftsprozessintegration verbunden werden können.
- Enterprise Service Bus: Der Begriff des ESB beschreibt eine Kommunikationsinfrastruktur, welche den nachrichtenbasierten Informationsaustausch der Services steuert. Der Begriff ESB ist nicht eindeutig definiert. Oftmals bietet ein ESB z.B. ein zentrales Service Repository.
- Intermediär: SOAP-basierte Kommunikation zwischen zwei Endpunkten erfolgt über eine Kette von Zwischenstationen – den sogenannten Intermediären. Neben einer reinen Weiterleitungsfunktion können diese auch Mehrwerte auf den Nachrichten erbringen wie z.B. das Anbringen eines kryptographisch gesicherten Zeitstempels oder die Prüfung von Signaturen.
- MTOM: SOAP Message Transmission Optimization Mechanism
- OASIS: Organization for the Advancement of Structural Information Standards
- SAML: Security Assertion Markup Language
- Security Token: Ein Security Token stellt ein Gerät oder eine Hardwarekomponente dar, auf dem Identifikations- und Berechtigungsnachweise – also Credentials – gespeichert sind. In der Regel erfolgt diese Speicherung unter Einsatz kryptographischer Schutzmechanismen.
- Service: Funktionalitäten und Aufgaben werden nicht wie bei herkömmlichen Systemen innerhalb einer einzelnen Anwendung, sondern als lose gekoppelte, unabhängige, austauschbare Dienste über standardisierte Schnittstellen von einem Service-Provider angeboten. Der Service-Consumer erhält als Antwort des Service Requests einen Service Response. Aus Sicht der Geschäftsprozesse ist ein Service zugleich aber auch ein Dienst(leistung) mit gewissem (Mehr-)Wert.
- SOAP: Das Simple Object Access Protocol beschreibt den allgemeinen Aufbau von XML-Nachrichten im Web-Service-Kontext
- Subject: Natürliche oder juristische Person, auf die ein Security Token personalisiert ausgestellt wird. Das Token enthält somit Credentials zu dieser Person.
- SwA: Soap with Attachments
- UDDI: Universal Description, Discovery and Integration, <http://www.w3.org/TR/soap>

- W3C: World Wide Web Consortium, <http://www.w3.org/>
- Web-Service: Ein Web-Service ist eine auf XML (eXtensible Markup Language) und SOAP (Simple Object Access Protocol) basierende Anwendung, die definierte Methoden über eine standardisierte Schnittstelle anbietet und über eine URI (Uniform Resource Identifier) eindeutig identifizierbar macht. Als Kommunikationsprotokoll dient SOAP (W3C Recommendation), mit dem Daten zwischen Systemen ausgetauscht werden können. SOAP verwendet XML zur Datenrepräsentation sowie (in den meisten Fällen) HTTP/TCP für den Transport.
- WS: siehe Web-Service
- WS-A: Web-Service-Adressing, <http://www.w3.org/TR/soap>
- WS-Attach: WS-Attachments
- WS-CAF: Web-Service Composite Application Framework, <http://www.w3.org/TR/soap>
- WSCI: Web-Service Choreography Interface, <http://www.w3.org/TR/wsci>
- WSDL: Web-Service Description Language, <http://www.w3.org/TR/soap>
- XML: eXtensible Markup Language, <http://www.w3.org/TR/soap>
- XOP: XML-binary optimized packaging, <http://www.w3.org/TR/xop10>

## 7. Referenzen

1. <http://www.w3.org/2002/ws/>
2. <http://www-128.ibm.com/developerworks/webservices/standards/>
3. <http://msdn.microsoft.com/webservices/understanding/specs/default.aspx>
4. OASIS, <http://www.oasis-open.org/home/index.php>
5. OASIS SOA Reference Model TC, [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm)
6. Web Services Security Roadmap issued jointly by Microsoft, IBM and Verisign, <http://www.webservices.org/index.php/ws/content/view/full/1732>
7. Extensible Markup Language (XML) 1.0, W3C, <http://www.w3.org/TR/2006/REC-xml-20060816/>
8. Namespaces in XML 1.0, W3C, <http://www.w3.org/TR/REC-xml-names/>
9. XML Schema, W3C, <http://www.w3.org/TR/xmlschema-0/>, <http://www.w3.org/TR/xmlschema-1/>, <http://www.w3.org/TR/xmlschema-2/>, <http://www.w3.org/XML/Schema>
10. XPath, W3C, <http://www.w3.org/TR/xpath>
11. XML Encryption, <http://www.w3.org/TR/xmlenc-core/>
12. XML Digital Signature, <http://www.w3.org/TR/xmldsig-core/>
13. Web Services Security (WS-Security), <http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>
14. XKMS, <http://www.w3.org/TR/xkms/>
15. XKMS2, <http://www.w3.org/TR/xkms2/>
16. Web Services Trust Language (WS-Trust), <http://www-106.ibm.com/developerworks/webservices/library/ws-trust/>
17. SAML, <http://www.oasis-open.org/committees/security/>
18. XACML, <http://www.oasis-open.org/committees/xacml/>
19. Web Services Secure Conversation (WS-SecureConversation), <http://www-106.ibm.com/developerworks/webservices/library/ws-secon/>

20. Web Services Federation Language (WS-Federation), <http://www-106.ibm.com/developerworks/webservices/library/ws-fed/>
21. WS Kerberos Security Binding, <http://msdn.microsoft.com/ws/2003/12/wsskb/>
22. Web Services Eventing (WS-Eventing), <http://ftpn2.bea.com/pub/downloads/WS-Eventing.pdf>
23. WS Notification, <http://www.ibm.com/developerworks/library/specification/ws-notification/>
24. Web Services Addressing, <http://www-106.ibm.com/developerworks/library/specification/ws-add/>
25. WS-Routing Specification Index Page, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsroutspecindex.asp>
26. WS-ReliableMessaging Specification Index Page, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsrmspecindex.asp>
27. WS-Reliability and WS-ReliableMessaging, <http://xml.coverpages.org/ChappellReliability20030313.html>
28. Web Services Policy Framework (WSPolicy), <http://www-106.ibm.com/developerworks/library/ws-polfram/>
29. Web Services Metadata Exchange (WSMetadataExchange), <http://ifr.sap.com/ws-metadataexchange/WS-MetadataExchange.pdf>
30. Web Services Policy Attachment (WS-PolicyAttachment), <http://www-106.ibm.com/developerworks/library/ws-polatt/>
31. Web Service Policy Assertions, <http://www.ibm.com/developerworks/webservices/library/specification/ws-polas/>
32. Web Services Security Policy, <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>
33. Web Services Discovery, <http://msdn.microsoft.com/ws/2005/04/ws-discovery/>
34. Web Services Inspection, <http://www-128.ibm.com/developerworks/library/specification/ws-wsilspec/>
35. WSFL, <http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>
36. EbXML, <http://www.ebxml.org/>
37. Web Services Coordination (WS-Coordination), <http://www-106.ibm.com/developerworks/library/ws-coor/>
38. Web Services Transaction (WS-Transaction), <http://www-128.ibm.com/developerworks/library/specification/ws-tx/>
39. Web Services Atomic Transaction (WS-AtomicTransaction), <ftp://www6.software.ibm.com/software/developer/library/ws-atomictransaction.pdf>
40. Web Services Business Activity, <http://specs.xmlsoap.org/ws/2004/10/wsba/wsba.pdf>
41. Business Process Execution Language for Web Services Version 1.1, <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>
42. WS-CDL, <http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/>
43. WS-I, <http://www.ws-i.org>
44. Federation of Identities in a Web Services World, <http://www-106.ibm.com/developerworks/webservices/library/ws-fedworld/>
45. SOA Practitioner's Guide, <http://dev2dev.bea.com/pub/a/2006/09/soa-practitioners-guide.html>
46. WebServices Tutorial, SUN Microsystems, <http://java.sun.com/webservices/tutorial.html>
47. The Liberty Project, <http://www.projectliberty.org/>