Nils Gruschka

# Schutz von Web Services durch erweiterte und effiziente Nachrichtenvalidierung

Cuvillier Verlag Göttingen

# Schutz von Web Services durch erweiterte und effiziente Nachrichtenvalidierung

**Dissertation**

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

**Nils Gruschka**

Kiel, 2008

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.
1. Aufl. - Göttingen : Cuvillier, 2008
Zugl.: Kiel, Univ., Diss., 2008

978-3-86727-674-0


Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de


1. Auflage, 2008
Gedruckt auf säurefreiem Papier

978-3-86727-674-0

# Vorwort

Netzwerkdienste sind einer Vielzahl von Angriffen ausgesetzt. Insbesondere die sogenannten Denial-of-Service-Angriffe (kurz: DoS-Angriffe) stellen eine große Gefahr dar. Bei Web Services sind derartige Angriffe auch noch deutlich einfacher durchzuführen als gegen „klassische“ Dienste (wie dem WWW). Experimentelle Untersuchungen haben gezeigt, dass es problemlos möglich ist, die Verfügbarkeit mit wenigen oder sogar nur einer einzelnen Nachricht erheblich zu beeinträchtigen. Dabei wurde auch festgestellt, dass viele Angriffe auf der Abweichung vom korrekten Protokoll basieren. Einer der wichtigsten Gegenmaßnahme gegen DoS-Angriffs ist daher auch die Überprüfung der Konformität von Nachrichten bezüglich der Protokolldefinition. Allerdings sind die meisten heutigen Netzwerkschutzsysteme zur Analyse der Web-Service-Nachrichten (SOAP-Nachrichten) und damit zur Erkennung von Angriffen auf Web Services ungeeignet. Dies hat verschiedene Gründe. Zunächst erfordern die XML-basierten SOAP-Nachrichten eine grundlegend andere Verarbeitungsmethode als „klassische“ Protokollnachrichten. Weiterhin definiert SOAP nur eine generische Nachrichtenhülle. Die Definition für die konkreten Web-Service-Nachrichten ergibt sich erst aus einer Reihe von Metadaten, die für jeden Web Service unterschiedlich sind.

Zur Lösung dieser Probleme werden in dieser Arbeit Methoden zur Überprüfung von Web-Service-Nachrichten und zum Schutz von Web-Service-Systemen vor Angriffen entwickelt. Den Kern dieser Gegenmaßnahmen stellt dabei die sogenannte *erweiterte* und *effiziente* Nachrichtenvalidierung dar.

Unter erweiterter Validierung wird dabei die Untersuchung von Nachrichten auf Konformität zu sämtlichen beteiligten Definitionen verstanden. Dies umschließt sowohl die Web-Service-Standards, die Nachrichtenformate definieren, als auch die Metadaten, die jedem Web Service zu eigen sind und die an andere Web-Service-Systeme propagiert werden. Dabei hat sich gezeigt, dass diese Art der Validierung nicht nur gegen Denial-of-Service-Angriffe wirksam ist.

Das größte Problem der erweiterten Validierung ist die Realisierung der Nachrichtenverarbeitung. Zur Validierung ist eine Nachrichtenverarbeitung notwendig, die robust gegenüber Angriffsnachrichten ist. Die in heutigen Web-Service-Systemen überwiegend genutzte *baumbasierte* XML-Verarbeitung ist zur Analyse von potentiellen Angriffsnachrichten ungeeignet. Daher werden in dieser Arbeit Algorithmen zur Verarbeitung und Validierung von Web-Service-Nachrichten entwickelt, die vollständig auf der sogenannten *ereignisbasierten* Verarbeitungsmethodik aufbauen.

Die darauf basierenden Validierungsmechanismen und Schutzmaßnahmen wurden in Form einer Web-Service-Firewall realisiert, welche zum Schutz von herkömmlichenen Web-Service-Systemen genutzt werden kann. Die Evaluation der Implementierung hat die hergeleitete Laufzeit- und Speicherkomplexität und damit die theoretische Schutzwirkung bestätigt. Zusätzlich hat die Web-Service-Firewall ihre Schutzwirkung gegen konkrete Angriffe und geringen Ressourcenbedarf unter Beweis gestellt.

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Problemstellung

Die vorliegende Arbeit beschäftigt sich mit der Validierung von Web-Service-Nachrichten. Es wird gezeigt, wie diese Validierung wirkungsvoll zur Abwehr von Angriffen gegen Web-Service-Server eingesetzt werden kann.

Dienste, vor allem solche, die in öffentlichen Netzen angeboten werden, sind einer Vielzahl von Angriffen ausgesetzt. Insbesondere Angriffe, die die Verfügbarkeit von Netzwerk-Systemen reduzieren oder sogar vollständig eliminieren, die sogenannten Denial-of-Service-Angriffe (kurz: DoS-Angriffe, [115]), stellen eine große Gefahr dar. Denial-of-Service-Angriffe gegen weitverbreitete Dienste, wie das *World Wide Web* oder E-Mail, haben über die letzten Jahre stark an Häufigkeit und Auswirkungen zugenommen [138]. Eine der bekanntesten Angriffe dieser Art wurde im Mai 2007 mehrere Wochen lang gegen zahlreiche Webserver in Estland geführt [144]. Dabei wurden neben Webportalen von Zeitungen, Banken und anderer Unternehmen insbesondere auch viele Server der estnischen Regierung in Mitleidenschafft gezogen. Dadurch und durch Gerüchte, die Angriffe würden von Russland ausgehen, erhielt dieser Zwischenfall auch eine starke politische Brisanz. Dies wurde auch durch die Verwendung von Begriffe wie „Cyber-Krieg“ und „Cyber-Terrorismus“ verdeutlicht.

Technisch handelte es sich um einen verteilten DoS-Angriff, bei dem die Webserver von einer großen Anzahl von Rechnern aus mit extrem vielen regulären Anfragen überflutet wurden.

DoS-Angriffe auf derartige „klassische“ Dienste sind heutzutage in der Regel nur mit hohem technischen Aufwand durchführbar, beispielsweise durch Verwendung von *Bot-Netzen* [138]. Bei Web Services hingegen sind DoS-Angriffe deutlich einfacher durchzuführen. Wie in dieser Arbeit anhand von experimentellen Untersuchungen verschiedener Angriffstypen gezeigt wird, ist es problemlos möglich, die Verfügbarkeit mit wenigen oder sogar nur einer einzelnen Nachricht erheblich zu beeinträchtigen.

> *„A single XML Message can cause damage to a naive or misconfigured system.“*
>
> Mark O’Neill, Autor des Buches „Web Services Security“ [129]

Eine der wichtigsten Gegenmaßnahmen gegen derartige Angriffe ist die Untersuchung von eingehenden Nachrichten daraufhin, ob sie potentiell Teil eines Angriffs sind. Dabei gibt es in der Regel zwei Arten von Analysekriterien. Zum einen wird die Konformität der Nachricht bezüglich der Protokolldefinition überprüft. Da bei vielen Angriffsarten der Angriffseffekt durch die Abweichung vom korrekten Protokoll erreicht wird, können durch diese Überprüfung derartige

Angriffe erkannt werden. Zum anderen werden zur Erkennung von Angriffen, die innerhalb der Protokolldefinitionen bleiben, Nachrichten nach empirisch gewonnenen Angriffsmustern durchsucht.

Realisiert sind diese Gegenmaßnahmen typischerweise in *Firewalls* oder *Intrusion-Detection-Systemen*. Allerdings sind die meisten derartigen Systeme heutzutage zur Analyse der Web-Service-Nachrichten (SOAP-Nachrichten) und damit zur Erkennung von Angriffen auf Web Services ungeeignet.

> *SOAP goes through firewalls like a knife through butter.*
>
> Tim Bray, Verfasser der XML-Spezifikation [26]

Dies hat verschiedene Gründe. Zunächst erfordern die XML-basierten SOAP-Nachrichten eine grundlegend andere Verarbeitungsmethodik als „klassische“ Protokollnachrichten. Weiterhin definiert SOAP nur eine generische Nachrichtenhülle. Die Definition für die konkreten Web-Service-Nachrichten ergibt sich erst aus einer Reihe von Metadaten, die jeder Web Service verwendet und die auch für jeden Web Service unterschiedlich sind. Und schließlich lassen sich auch die Methoden zur Erkennung von Angriffsmustern in Nachrichtenströmen nicht auf XML-Strukturen anwenden.

Zur Lösung dieses Problems werden in dieser Arbeit Methoden zur Überprüfung von Web-Service-Nachrichten und damit zum Schutz von Web-Service-Systemen vor Angriffen entwickelt. Den Kern dieser Gegenmaßnahmen stellt dabei die sogenannte *erweiterte* und *effiziente* Nachrichtenvalidierung dar.

Unter erweiterter Validierung wird dabei die Untersuchung von Nachrichten auf Konformität zu sämtlichen relevanten Definitionen verstanden. Dies umschließt sowohl die Web-Service-Standards, die Nachrichtenformate definieren (SOAP, WS-Security, WS-Addressing usw.), als auch die Metadaten, die jedem Web Service zu eigen sind und die an andere Web-Service-Systeme propagiert werden (Web-Service-Beschreibung gemäß WSDL, Sicherheits-Policy gemäß WS-SecurityPolicy, Nachrichtenreihenfolge gemäß BPEL usw.). Weiterhin wird auch die Überprüfung gegen die Zugriffskontroll-Policy in diese Validierung mit einbezogen. Die Protokolldefinitionen, die sich aus all diesen Teilvorschriften ergeben, werden dabei zusätzlich derart modifiziert, dass sie Nachrichten mit bekannten Angriffsmustern ausschließen.

Dabei hat sich gezeigt, dass diese Art der Validierung nicht nur Denial-of-Service-Angriffe erkennen kann, sondern auch gegen andere Angriffe (beispielsweise die bekannten *XML-Rewriting*-Angriffe [110]) wirksam ist. Schließlich ist eine vollständige Protokollüberprüfung auch ein wichtiger Teil der Erfüllung von Dienstabsprachen (engl. *Compliance*)[1].

Das größte Problem der erweiterten Validierung ist die Realisierung der Nachrichtenverarbeitung. Die verbreitetste Art der XML-Verarbeitung (die sogenannten *baumbasierten* Methoden), die auch von den meisten Web-Service-Systemen verwendet wird, ist zur Analyse von potentiellen Angriffsnachrichten ungeeignet. Daher werden in dieser Arbeit Algorithmen zur Verarbeitung und Validierung von Web-Service-Nachrichten entwickelt, die vollständig auf der sogenannten *ereignisbasierten* Verarbeitungsmethodik aufbauen.

Darauf basierend wurden die Validierungsmechanismen implementiert und die Schutzmaßnahmen als Web-Service-Firewall realisiert. Die Evaluation der Implementierung hat die hergeleitete Laufzeit- und Speicherkomplexität und damit die theoretische Schutzwirkung bestätigt. Zusätzlich hat die Web-Service-Firewall ihre Schutzwirkung gegen konkrete Angriffe und ihren für praktische Einsätze verwendbaren Ressourcenbedarf bewiesen.

[1] Dies wird im Folgenden nicht weiter diskutiert werden.

## 1.2 Stand der Technik

In diesem Kapitel wird eine kurze Einführung in die Themengebiete gegeben, die dieser Arbeit zugrunde liegen. Diese sind Computersicherheit und Angriffe, Web Service und Web-Service-Sicherheit. Dabei werden auch bestehende Maßnahmen zur Sicherung von Web Services vorgestellt, die sich zumeist um die wichtigsten Web-Service-Sicherheits-Standards rangen: WS-Security und WS-SecurityPolicy.

### 1.2.1 Computersicherheit

Informationstechnische Systeme sind einer Vielzahl von Gefährdungen ausgesetzt. Dabei wird zwischen *Bedrohung* (engl. *Threat*) und *Angriff* (engl. *Attack*) unterschieden [142]. Eine Bedrohung ist eine Möglichkeit für eine Verletzung der Sicherheit, während ein Angriff ein konkreter Anschlag auf die Sicherheit eines Systems ist.

Nach klassischer Lehre sind die drei Hauptaspekte von Computersicherheit: *Vertraulichkeit* (engl. *Confidentiality*), *Integrität* (engl. *Integrity*) und *Verfügbarkeit* (engl. *Availability*); im Englischen auch oft (scherzhaft) als „CIA“ abgekürzt [18].

#### Vertraulichkeit

Vertrauliche Daten sind solche, deren Kenntnis nicht jeder erlangen darf. Zur Erreichung dieses Sicherheitsziels können beispielsweise kryptographische Verschlüsselung von Daten (bei der Speicherung und der Übertragung) und Zugriffskontrolle für die Dienste, die auf diesen Daten operieren, eingesetzt werden.

#### Integrität

Integrität beschreibt die Anforderung, dass Daten „unverfälscht“ sind. Dabei kann zwischen *Daten-Integrität* (Integrität des Inhalts) und *Ursprungs-Integrität* (Integrität der Herkunft, auch *Authentizität* genannt) unterschieden werden. Der Prozess der Feststellung der Authentizität wird *Authentifizierung* genannt. Zur Gewährleistung der Integrität kommen kryptographische Methoden wie *Hash-Bildung* (auch *Hashing* genannt) oder *digitale Signaturen* und wiederum Zugriffskontrollen zum Einsatz.

#### Verfügbarkeit

Verfügbarkeit bezeichnet die Möglichkeit, auf eine Ressource zuzugreifen. Obwohl diese Eigenschaft vielfach nicht als sicherheitsrelevant angesehen wird, ist sie für ein sicheres Systems doch zwingend erforderlich. Ein klassisches Beispiel ist ein Alarmdienst, der über einen Einbruch in einen Tresor informiert [126]. Bei diesem Dienst ist Vertraulichkeit und Integrität zweitrangig. Wenn aber die Verfügbarkeit nicht gegeben ist, ist die Sicherheit des Gesamtsystems nicht mehr gewährleistet.

Neben diesen „Axiomen“ der IT-Sicherheit werden teilweise noch weitere Sicherheitsanforderungen genannt, diese lassen sich aber meist auf die oben genannten zurückführen. Häufige Forderungen sind dabei

- Verlässlichkeit (engl. *Reliability*)
- Nicht-Abstreitbarkeit (engl. *Non-Repudiation*)
- Privatheit (engl. *Privacy*)

Die Aufgabe von Computer-Sicherheit ist die Analyse von Bedrohungen, mit dem Ziel Angriffe zu verhindern (engl. *Prevention*) oder zumindest zu erkennen (engl. *Detection*), sowie ein System nach einem Angriff wiederherzustellen (engl. *Recovery*). Ein Mechanismus, der dieses leistet, wird als *Sicherheits-Mechanismus* (engl. *Security Mechanism*) bezeichnet [145].

Angriffe lassen sich nach den durch sie gefährdeten Sicherheitsaspekten klassifizieren. Angriffe gegen die Verfügbarkeit werden als *Denial-of-Service-Angriffe*, im Deutschen teilweise auch als *Sabotageangriffe* bezeichnet. Dabei gibt es natürlich auch Angriffe, die gegen mehr als einen Sicherheitsaspekt wirken oder wirken können. So ist es beispielsweise mit einen *SQL-Injection*-Angriff [4] möglich, nicht intendierte SQL-Anfragen an die Datenbank eines Dienstes zu stellen. Diese Anfragen können dabei sowohl zur Reduktion der Verfügbarkeit als auch zum Erlangen von vertraulichen Informationen missbraucht werden.

Zur Sicherung der Verfügbarkeit sind in der Vergangenheit einige allgemeine Methoden entwickelt worden. Die wichtigsten sind die Folgenden:

- Überprüfung der Eingabe
- Beschränkung des Zugriffs
- Beschränkung des Ressourcenverbrauchs

### 1.2.2 Web Services

Das Konzept der *dienstorientierten Architektur* [52] (engl. *Service-Oriented Architecture*, SOA) stellt ein wichtigen Schritt in der Geschichte der Software-Entwicklung dar. Die Hauptidee von SOA besteht darin, monolithische Applikationen in kleinere Komponenten aufzubrechen und diese plattformunabhängig als Netzwerkdienst wiederverwendbar anzubieten. Die Erstellung komplexerer Applikationen erfolgt dann durch *Komposition* dieser Dienste [152]. Der bekannteste Standard zur Definition solcher Kompositionen ist die *Business Process Execution Language* (BPEL) [3].

Web Services [19, 2] sind die am weitesten verbreitete Realisierung der Kommunikation in einer dienstorientierten Architektur. Web Services werden oft als *Middleware* für verteilte Anwendungen betrachtet [98]. Im Gegensatz zu *Middleware*-Systemen wie DCOM oder RMI sind Web Services weder an bestimmte Betriebssysteme noch Programmiersprachen gebunden und sollen damit eine leichte Zusammenführung von verteilten Applikationen ermöglichen (beispielsweise im Rahmen der sogenannten *Enterprise Application Integration*).

Die Spezifikationen im Web-Service-Umfeld sind grundsätzlich sehr stark auf Flexibilität und Erweiterbarkeit ausgelegt. Zusätzlich sind Details teilweise nicht präzise oder eindeutig definiert. Da diese beiden Eigenschaften zu Problemen bei der Entwicklung entsprechender Systeme und deren Interoperabilität führen, hat die *Web Service Interoperability Organization* (WS-I) Empfehlungen herausgegeben, in denen die betroffenen Spezifikationen präzisiert oder eingeschränkt werden [9, 111]. Alle wichtigen Web-Service-Implementierungen berücksichtigen diese Empfehlungen inzwischen. In dieser Arbeit werden daher nur Web Services betrachtet, die dieser Empfehlung folgen.

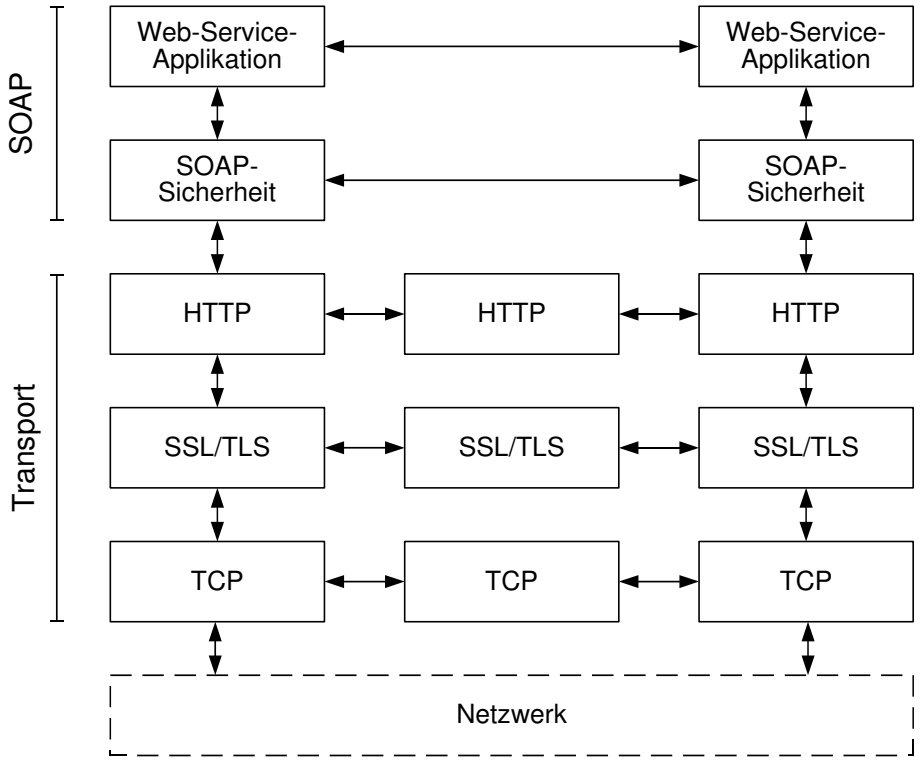


Abbildung 1.1: Protokollschichtung für Web Services (inklusive Sicherheitsprotokollen)

Der wichtigste designierte Anwendungsfall von Web Services ist die Kommunikation zwischen Unternehmen auch über offene Netze hinweg. Auch aus dieser Überlegung heraus ergibt sich die Notwendigkeit zur Sicherung von Web Services.

### 1.2.3 Web-Service-Sicherheit

Zur Sicherung von Vertraulichkeit, Integrität und Authentizität und zur Durchführung von Authentifizierung bei Netzwerkdiensten existieren einige weitverbreitete Standards wie z. B. *Secure Socket Layer* (SSL bzw. TLS) [59, 44] oder IPsec [46]. Diese Sicherheitsmechanismen sind an ein bestimmtes Transportprotokoll gebunden und realisieren damit *transportorientierte* Sicherheit. Solche Sicherheitsprotokolle sind für Web Services nur bedingt geeignet [40].

Eine der grundlegenden Ideen von Web Services ist die Unabhängigkeit vom verwendeten Transportprotokoll. Auch wenn für SOAP-Nachrichten de facto nur der Transport über HTTP verwendet wird, so sind grundsätzlich auch andere Transportprotokolle denkbar. Weiterhin wird selbst bei einem festen Transportmechanismus der Transport nicht immer Ende-zu-Ende bezüglich des Web Services (d. h. vom Nachrichtenersteller zum -empfänger) durchgeführt. Ein Beispiel sind SOAP-Zwischensysteme, die SOAP-Nachrichten weiterleiten und dabei auch Modifikationen durchführen, die typischerweise als Endpunkt des Transportprotokolls implementiert sind.

Zur Sicherung von SOAP-Nachrichten werden daher *nachrichtenorientierte* Mechanismen verwendet [80, 136]. Abbildung 1.1 zeigt die Einordnung dieser Mechanismen im Verhältnis zu anderen Internetprotokollen. Der wichtigste Standard für SOAP-Sicherheit ist WS-Security [124, 76].

WS-Security definiert folgende Sicherheitsmechanismen:

- Verschlüsselung von SOAP-Fragmenten
- Signierung von SOAP-Fragmenten
- Sicherheits-Token
- Zeitstempel

WS-Security ist bei Systemen für Web-Service-Sicherheit heutzutage weit verbreitet [27]. Viele Web-Service-Frameworks bieten WS-Security-Unterstützung, beispielsweise WSE [89] für Microsoft .NET oder Rampart [55] für Apache Axis2.

Aufgrund der Komplexität von WS-Security und insbesondere der komplexen Referenzierung zwischen den WS-Security-Elementen, wird die WS-Security-Verarbeitung typischerweise baumbasiert realisiert. Es existieren aber auch Ansätze für ereignis- bzw. strombasierte Lösungen. So wird in [108] ein Verfahren vorgestellt, welches SOAP-Nachrichten mittels vordefinierter Schablonen (engl. *Templates*) für WS-Security-Elemente verarbeitet. Dieses System funktioniert aber nur für ein festes WS-Security-Framework und lässt sich nicht verallgemeinern.

Eine ereignisbasierte Verifizierung von Signaturen in SOAP-Nachrichten wird in [107] präsentiert. Die dort vorgestellte Methode ist allerdings nicht geeignet, Signaturen mit Rückwärtsreferenzen zu behandeln, und berücksichtigt die Anforderung des WS-I [111] nicht. Außerdem wird die Problematik der Schlüsselreferenzierung und -erlangung nicht betrachtet.

Ein Ansatz zur ereignisbasierten Ver- und Entschlüsselung von XML-Dokumenten wird in [85] gezeigt. Die Arbeit beschäftigt sich aber nicht mit der Einbindung in SOAP-Nachrichten und insbesondere nicht mit der Referenzierung von Schlüsseln und verschlüsselten Schlüsseln.

WS-Security definiert generische Sicherheitselemente und erlaubt damit eine flexible Anwendung dieser Elemente auf SOAP-Nachrichten. Dies führt zu der Notwendigkeit, dass Web-Service-Partner sich gegenseitig über ihre Sicherheitsanforderungen und damit über die geforderten WS-Security-Elemente informieren müssen. WS-SecurityPolicy [95] erlaubt die Spezifikation solcher Anforderungen.

Policys werden typischerweise von einem Web-Service-Server angeboten und beschreiben seine Forderungen an eingehende Nachrichten sowie die Eigenschaften der ausgehenden Nachrichten. Ein Web-Service-Client ist damit in der Lage, seine ausgehenden Nachrichten entsprechend zu konstruieren bzw. seine eingehenden Nachrichten korrekt zu interpretieren.

Die Anbindung einer Policy an eine Web-Service-Beschreibung wird in WS-PolicyAttachment [48] beschrieben. Dies erlaubt die Referenzierung aus verschiedenen Teilen der Web-Service-Beschreibung auf mehrere Policys. So können dem Endpunkt, der Operation und der Nachricht jeweils eigene Teil-Policys zugewiesen werden. Diese heißen dann entsprechend *Endpunkt-Policy*, *Operations-Policy* und *Nachrichten-Policy*. Die *effektive* Policy für eine Nachricht ergibt sich aus der Vereinigung der Forderungen der (bis zu 4) zugehörigen Policys.

Das größte Problem bei der Benutzung von WS-SecurityPolicy ist die Erstellung von Policys für die gewünschten Sicherheitsanforderungen. Dabei ergibt sich zum einem das Problem der Umsetzung der abstrakten Sicherheitsanforderungen (z. B. „Kreditkartennummern dürfen nicht gelesen oder modifiziert werden.“) in einer konkreten Policy. In [146] und [87] werden Muster für die praktische Realisierung (engl. *Best Practice Pattern*) von Sicherheitsanforderungen vorgestellt.

Ein weiteres Problem ist, dass die einzelnen Teilanforderungen einer Sicherheits-Policy in komplexem Zusammenhang stehen. Dies führt dazu, dass Sicherheits-Policys oftmals Sicherheitsschwächen enthalten, die von Menschen schwer erkennbar sind. Ein einfaches Beispiel ist die Forderung von Integrität des SOAP-Bodys und Schutz gegen *Replay*-Angriffe. Die direkte Umsetzung in eine Sicherheits-Policy würde die Signierung des SOAP-Bodys, die Benutzung eines Zeitstempels und die Signierung des Zeitstempels erfordern. Eine Nachricht, die dieser Sicherheits-Policy entspricht, ist aber anfällig für einen einfachen *XML-Rewriting*-Angriff: der separat signierte Zeitstempel kann von einer anderen (aktuellen) Nachricht in die alte Nachricht kopiert werden. Zum Schutz vor solchen Angriffen gibt es eine große Anzahl von Arbeiten, die sich mit der Analyse von Sicherheits-Policys bzgl. derartiger Schwachstellen beschäftigen [14, 16, 130].

Für die praktische Realisierung hat WS-SecurityPolicy noch einige weitere Schwächen. So ist neben der komplexen Struktur mit Alternativenbildung und Verteilung der Anforderungen auf Teil-Policys das Fehlen von konkreten Identitäten ein Problem. Damit ist WS-SecurityPolicy allein nicht geeignet zur Definition von Zugriffskontroll-Policys. Einige Systeme lösen dies durch proprietäre Erweiterungen von WS-SecurityPolicy um konkrete Identitäten ([5], Apache Rampart 1.1). Damit sind derartige Policys aber nicht mehr zwischen Web-Service-Partnern austauschbar und widersprechen der Intention von WS-SecurityPolicy. In [68] wird eine Erweiterung um Identitäten vorgestellt, bei der die Austauschbarkeit der Policys erhalten bleibt. Sicherlich nicht zuletzt aufgrund dieser Eigenschaften verwenden Web-Service-Frameworks (Microsoft WSE [89], Apache Rampart 1.0) und Web-Service-Firewalls häufig nicht WS-SecurityPolicy, sondern eigene Policy-Sprachen für ihre Sicherheits-Policys.

Alle hier genannten Arbeiten und Spezifikationen zum Thema Web-Service-Sicherheit beschäftigen sich mit der Sicherung der Integrität und Vertraulichkeit bei Web Services. Die Sicherung der Verfügbarkeit bzw. der Schutz vor Denial-of-Service-Angriffen ist im Kontext von Web Services ein wenig beachtetes Thema. Dabei zeigen Untersuchungen [143, 104, 92], dass Web Services sogar noch verwundbarer für DoS-Angriffe sind als nicht-Web-Service-basierte Dienste.

Ein erhöhtes Bewusstsein für die Problematik der Verfügbarkeit lässt sich an der Entwicklung der Web-Service-Frameworks ablesen. So ließ sich unter Microsoft .NET Version 1.0 ein Web-Service-System noch problemlos durch eine übergroße SOAP-Nachricht zum völligen Stillstand bringen, während dies bei .NET Version 2.0 nicht mehr möglich ist.

Allerdings lösen die verwendeten Maßnahmen die Probleme oftmals nur teilweise oder führen zu neuen Schwachstellen. So wurde die oben genannte Verbesserung dadurch erreicht, dass der .NET-Service alle Nachrichten ablehnt, die eine bestimmte serialisierte Größe überschreiten, in der Standardeinstellung 4 MByte. Eine derartige Maßnahme ist aber nicht unproblematisch. Zunächst kann ein Web Service damit keine sehr großen legitimen Nachrichten verarbeiten. Weiterhin ist die Bewertung einer XML-basierten Nachricht an Hand der serialisierten Größe grundsätzlich ungenau, da ein und derselbe XML-Baum abhängig von der Serialisierung unterschiedliche Dokumentgrößen ergeben kann. So lehnt ein .NET-Service auch Nachrichten ab, die aus 4 MByte Leerzeichen bestehen. Schließlich zeigen die in dieser Arbeit präsentierten Angriffe, dass bei Web Services auch Nachrichten kleiner als 4 MByte zum Denial-of-Service führen können.

Eine weitere Maßnahme, die von vielen Systemen benutzt wird, um die Laufzeit und den Speicherverbrauch während der Verarbeitung zu reduzieren, ist die *tolerante* (engl. *lax*) Verarbeitung von Nachrichten. Dabei wird die Nachricht nicht auf genaue Konformität zum Protokoll überprüft. Bei der Web-Service-Verarbeitung wird beispielsweise keine Schema-Validierung

durchgeführt, was entsprechende Verarbeitungsressourcen einspart. Dies kann dann aber andere Angriffe ermöglichen, wie beispielsweise *XML Rewriting* oder *XML Injection*.

## 1.3 Beitrag der Arbeit

Der Beitrag der hier vorliegenden Arbeit ist eine Methodik zur Erkennung von Angriffen auf Web Services. Dies wird durch strenge Validierung von eingehenden Nachrichten gegen die Definitionen des konkreten Web Services erreicht. Diese Validierung besteht dabei aus folgenden Teilschritten:

#### Einschränkung der Nachrichtendefinitionen

Aus den Spezifikationen und Metadaten des Web Services werden die Definitionen für die erlaubten Nachrichten dieses Web Services generiert. Diese werden analysiert und dabei werden Vorschriften, die potentielle Angriffsnachrichten erlauben, entsprechend modifiziert. Dazu gehören beispielsweise unbeschränkte Listen in der Web-Service-Beschreibung oder zu schwache Sicherheitsforderungen in der Sicherheits-Policy.

#### Validierung von eingehenden Nachrichten

Eingehende Nachrichten werden gegen die generierten Definitionen überprüft. Dies umschließt die folgenden Teilvalidierungen:

- Validierung gegen das XML-Schema [54], das durch die Web-Service-Beschreibung definiert ist
- Entschlüsselung von verschlüsselten Dokumentteilen [86]
- Verifizierung der digitalen Signaturen [10]
- Validierung der Sicherheitseigenschaft gegen die Sicherheits-Policy [95]
- Überprüfung der Erfüllung der Zugriffs-Policy
- Validierung der korrekten Nachrichtenreihenfolge, die sich aus der BPEL-Beschreibung [3] ergibt

Durch diese Validierung werden die allgemeinen DoS-Gegenmaßnahmen „Überprüfung der Eingabe“ und „Beschränkung des Zugriffs“ realisiert. Gleichzeitig werden damit, insbesondere durch die Validierung gegen die (verschärfte) Sicherheits-Policy, Angriffe gegen die Sicherheitsziele Integrität und Vertraulichkeit verhindert oder zumindest erschwert.

#### Realisierung der Validierung

Die oben geschilderte Verarbeitung und Analyse von Nachrichten erfordert natürlich gewisse Ressourcen. Damit ist die Validierung selbst ein potentielles Ziel für Denial-of-Service-Angriffe, die auf Ressourcenerschöpfung abzielen. Ein solcher Angriff ist bei den üblichen baumbasierten XML-Verarbeitungsverfahren sogar sehr einfach, da diese einen extrem hohen Ressourcenbedarf haben. Insbesondere beim Speicherbedarf sind Faktoren von mindestens fünf im Vergleich zur Dokumentgröße üblich [84].

Das zweite Problem ist, dass hier die Verarbeitung erst beginnen kann, wenn das Dokument vollständig eingelesen wurde. Damit ist leicht einzusehen, dass eine reine Baumverarbeitung in jedem Fall durch ein „unendlich“ großes Dokument angegriffen werden kann. Die DoS-Gegenmaßnahmen „Überprüfung der Eingabe“ und „Beschränkung des Ressourcenverbrauchs“ widersprechen sich hier also offensichtlich. Damit ist eine baumbasierte Validierung von Nachrichten als Schutzmaßnahme für Web Services ungeeignet.

In dieser Arbeit wurde daher für die Durchführung der Validierung eine ereignisbasierte XML-Verarbeitungmethodik gewählt, welche die vorher erwähnten Nachteile nicht aufweist. Durch die direkte Verarbeitung des Eingabestroms wird das Dokument nur maximal bis zu dem Punkt verarbeitet, an dem eine der Validierungsoperationen eine Verletzung der Vorgaben feststellt. Damit lässt sich der Speicherverbrauch bei der Verarbeitung durch Einschränkung der Nachrichtendefinitionen beschränken. Weiterhin ist durch das Fehlen einer komplexen Dokumentrepräsentation auch der absolute Speicherverbrauch deutlich geringer als bei baumbasierten Methoden.

Der Effizienz der ereignisbasierten Verarbeitungsmodelle stehen aber auch einige Nachteile gegenüber. Die Verarbeitung von XML-Dokumenten erfordert typischerweise die Navigation innerhalb des XML-Baums und die Manipulation von Teilbäumen. Dies gilt auch für die Verarbeitung von SOAP-Nachrichten, beispielsweise bei der Referenzierung von signierten oder bei der Entschlüsselung von verschlüsselten Dokumententeilen. Solche Operationen sind bei der ereignisbasierten Verarbeitung nicht direkt durchführbar. Insbesondere das Navigieren zu vorherigen Dokumentteilen oder die Auswertung von XPath-Ausdrücken ist im Allgemeinen gar nicht möglich.

Einer der Hauptbeiträge dieser Arbeit stellt deshalb auch die vollständig ereignisbasierte Validierung von Web-Service-Nachrichten dar. Dafür wurden neue Algorithmen entwickelt, beispielsweise für ereignisbasierte Verarbeitung von WS-Security-Elementen oder für ereignisbasierte Validierung von Sicherheits-Policys.

## 1.4 Aufbau der Arbeit

Der erste Teil der Arbeit beschäftigt sich mit den Grundlagen für die erweiterte Validierung. Wie bereits motiviert, hat die XML-Verarbeitung großen Einfluss auf die Verletzbarkeit von Web Services und auf die Realisierung der Gegenmaßnahmen. Daher werden in Kapitel 2 die beiden wichtigsten Verarbeitungsmethodiken vorgestellt, abstrakt modelliert und in Bezug auf Speicherverbrauch und Laufzeit bewertet. Im nachfolgenden Kapitel wird eine Einführung in die wichtigsten Web-Service-Standards gegeben, die für das Verständnis diese Arbeit notwendig sind. Kapitel 4 präsentiert die für diese Arbeit wichtigste Klasse von Angriffen, die Denial-of-Service-Angriffe. Neben einem historischen Überblick und einer Vorstellung von Metriken und Gegenmaßnahmen wird ein Klassifizierungssystem für derartige Angriffe entwickelt. In Kapitel 5 werden Angriffe gegen Web Services vorgestellt und gemäß dem vorher genannten Klassifizierungssystem eingeordnet. Die Angriffe sind aus theoretischen Untersuchungen von Schwachstellen entstanden und zum großen Teil in praktischen Experimenten mit verbreiteten Web-Service-Systemen nachgewiesen worden.

Im zweiten Teil der Arbeit wird die Nachrichtenvalidierung behandelt. Zunächst wird in Kapitel 6 ein Überblick über alle Bestandteile der erweiterten Validierung und deren Schutzwirkung gegeben. Außerdem wird aus den theoretischen Überlegungen zur XML-Verarbeitung die ereignisbasierte Realisierung der Schutzmaßnahmen hergeleitet. In den folgenden Kapiteln wer-

den dann die einzelnen Bestandteile der Validierung beschrieben. Dabei werden jeweils abstrakt die Algorithmen zur Verarbeitung der Metadaten und zur ereignisbasierten Validierung der Nachricht sowie die Schutzwirkung gegen bestimmte Klassen von Angriffen erläutert. Kapitel 7 erläutert die Schema-Generierung aus Web-Service-Beschreibungen und die Validierung von SOAP-Nachrichten gegen solche Schemata. In Kapitel 8 wird die ereignisbasierte Verarbeitung von WS-Security-Elementen behandelt. Damit wird die Schema-Validierung von verschlüsselten Nachrichtenteilen sowie die Validierung gegen die Sicherheits-Policy ermöglicht. Kapitel 9 präsentiert dann die Validierung der Sicherheitselemente in der SOAP-Nachricht gegen die Forderung der Sicherheits-Policy. Anschließend wird in Kapitel 10 gezeigt, wie Zugriffskontrolle (also die Validierung gegen die Zugriffskontroll-Policy) effizient ereignisbasiert durchgesetzt werden kann. Kapitel 11 behandelt die Validierung der korrekten Nachrichtenreihenfolge. Dazu wird ein Automatenmodell vorgestellt, dass es erlaubt, aus BPEL-Prozessbeschreibungen Protokollablaufautomaten zu generieren. Diese Automaten werden verwendet, um zu überprüfen, ob eingehende Nachrichten im aktuellen Zustand des Ablaufprozesses erlaubt sind.

Der dritte Teil der Arbeit behandelt die Realisierung der Nachrichtenvalidierung zum Schutz von Web Services. Die vorgestellten Validierungsmaßnahmen wurden vollständig implementiert und als Web-Service-Firewall realisiert. Diese Implementierung wird in Kapitel 12 vorgestellt. Zum Nachweis der Funktionalität und der auch für praktische Zwecke ausreichenden Performance wurde das System einer Reihe von Evaluationstests unterzogen. Deren Ergebnisse finden sich in Kapitel 13. Die Arbeit schließt in Kapitel 14 mit einer Zusammenfassung und einem Ausblick.