Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung AW1 SS2006

Eike Falkenberg

Sicherheitsaspekte in Service Orientierten Architekturen

1 Einleitung

1.1 Motivation

Serviceorientierte Architekturen, kurz SOA, sind in aller Munde. Bisher wurden riesige, oftmals kaum noch beherschbare Anwendungs-Monolithen geschaffen. Die ursprünglich vielleicht gute Architektur wurde durch Patches, Bugfixes und permanente Weiterentwicklung ruiniert. Kommunikation zwischen Prozessen findet oftmals durch Schreiben in Dateien statt und viele Anwendundgsteile sind eng miteinander verzahnt. In diesem Szenario sind Änderungen an Geschäftsprozessen nur sehr schwer umzusetzen und die Anwendungen insgesamt nur schwer wartbar.

Dies alles soll SOA besser machen. SOA ist ein Architekturpattern, welches fachspezifische Anwendungsteile in wiederverwendbaren Services kapselt und so lose Kopplung erzielt. Ein einheitlicher Zugriff über einen *Enterprise Service Bus (ESB)* (siehe Grafik 1.1) ist empfehlenswert und ermöglicht die Anbindung von Legacy Systemen und den Zugriff auf diese. Technische Grundlage von SOAs sind meist Webservices. SOAs sind aber auch mit anderer *Message Oriented Middleware (MOM)*, wie zum Beispiel *CORBA*, realisierbar. Webservices scheinen jedoch am besten geeignet zu sein.

Große Unternehmen wie *Deutsche Post World Net* setzen bereits heute erfolgreich SOA ein. Somit ist SOA nicht mehr nur eine Idee, sondern ein konkretes Modell.
Da die technische Basis aktueller SOAs meist Webservices sind, beschränkt sich diese Arbeit auf Sicherheitsaspekte in SOAs im Zusammenhang mit Webservices. Das Thema Webservices und Sicherheit ist bereits im Vorjahr von Thies Rubarth (Rubarth, 2005) behandelt worden und kann nun durch aktuelle Entwicklungen ergänzt werden.

Diese Ausarbeitung zu meinem Vortrag am 2006-06-08 soll einen aktualisierten Überblick über die Sicherheitskonzepte für Webservices anbieten und darstellen, ob diese ausreichen um die Sicherheit einer SOA zu gewährleisten. Auf die Grundlagen der Themen SOA und Webservices wird, im Gegensatz zu dem Vortrag, nicht mehr eingegangen.
Diese Arbeit bezieht sich nicht auf eine konkrete Middleware und bietet daher auch keine Übersicht oder Bewertung der am Markt befindlichen Produkte.

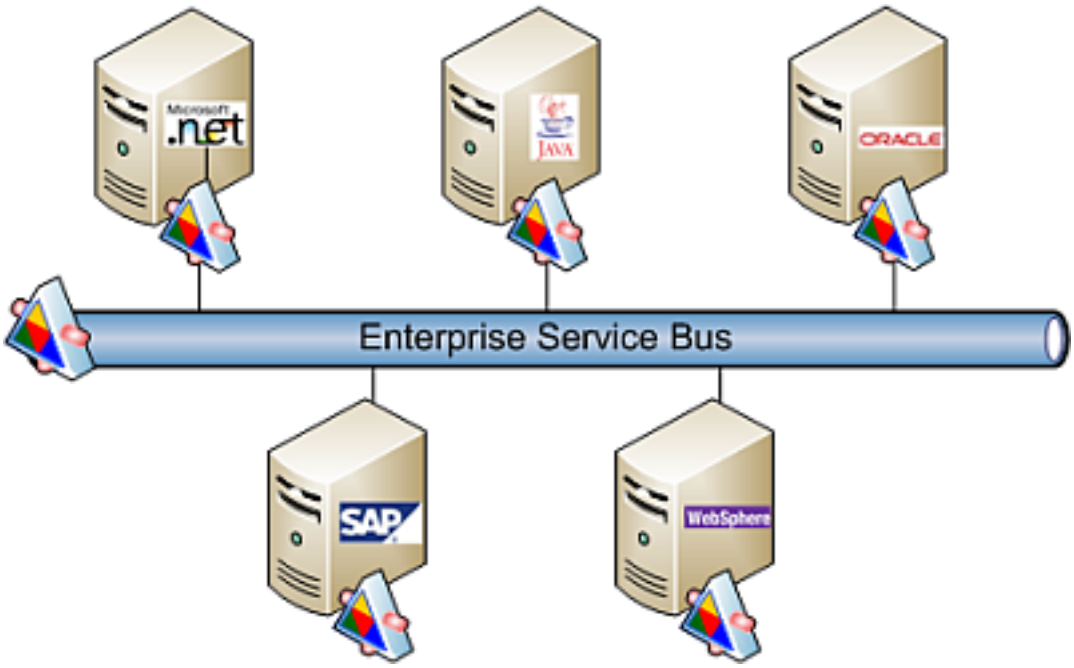

Abbildung 1.1: Enterprise Service Bus

Im folgenden Abschnitt soll die aktuelle Situation zum Thema Sicherheit und Webservices dargestellt werden. Der letzte Abschnitt ist die Bewertung der aufgelegten Lösungsversuche und ein daraus folgender Ausblick auf ein mögliches Thema im nächsten Semester.

2 Web Services und Sicherheit

2.1 Problem

Bei der Entwicklung der Webservices war man wie so oft zunächst fasziniert von dem Machbaren. Eine plattformunabhängige, relativ einfache und klare Kommunikationsform war etwas aufregendes und neues. Wie so oft wurde bei der Entwicklung zunächst der Aspekt der Sicherheit außen vor gelassen.
Webservices sollen lose koppelbar und von ihrer Struktur her offen sein. Sie nach außen hin abzuschotten ist daher nicht sinnvoll. Die klassiche Absicherung über Firewalls ist ebenfalls kein effektiver Ansatz, da Webservices normalerweise über Standardports angesprochen werden sollen und durch Firewalls auch nicht unnötig unflexibel werden dürfen. Solange Webservices singulär und nur im Intranet agierten, hatte das Thema Sicherheit keine hohe Priorität, man war der Ansicht „da passiert schon nichts“. SOA ist eine Öffnung der Systemarchitektur, oftmals sogar über das Intranet hinaus. Dies birgt neue Risiken und führte dazu, dass dem Thema Sicherheit bei Webservices die angemessene Bedeutung zukam.
Neben dem direkten Problem der Angreifbarkeit wurde bei der Konzeption von Webservices nicht die Integrität der Nachrichten bedacht. Ebenfalls Authentifizierungen und Autorisierungsmechanismen fehlen und führen somit zu fehleranfälligen und uneinheitlichen Eigenentwicklungen.
Im folgenden werden die wichtigsten Ansätze und Ideen zur Absicherung von Webservices aufgezeigt und eine erste Bewertung vorgenommen. Die abschließende Bewertung erfolgt im Schluss-Kapitel.

2.2 Secure Socket Layer - SSL

Die Absicherung von Webservices mittels SSL mag im ersten einleuchtend sein. SSL ist eine erprobte und bewährte Technik, ist sehr verbreitet und leicht einsetzbar.
SSL bietet jedoch keine Autorisierung und verschlüsselt immer die gesamte Nachricht, was einen Overhead bedeuten kann. SSL ist für Punkt-zu-Punkt Kommunikation konzipiert und somit wird an jedem Intermediary (Zwischenpunkt) sowohl ent- als auch wieder verschlüsselt. Die Intermediaries werden so zu Unsicherheitsfaktoren, da auf ihnen die Informationen

im Klartext ersichtlich sind.
Die Grafik 2.2 zeigt den Multihop-Zugriff auf einen entfernten Webservice. Der Intermediary entschlüsselt die Nachricht und Verschlüsselt sie wieder, um sie weiterzusenden.
SSL ist somit nur im einfachsten Fall geeignet, um Zugriffe auf Webservices abzusichern.

Abbildung 2.1: SSL Verschlüsselung mit Intermediary

2.3 Standardisierte Webservice Sicherheit

2.3.1 WS-Security (WSS)

WS-Security ist eine von der *Organization for the Advancement of Structured Information Standards (OASIS)* verabschiedete Spezifikation für einen einheitlichen Sicherheitsansatz im Webservice Umfeld.(OASIS)
Die erste Version der Spezifikation beschränkte sich auf drei Kernelemente:

- *Security Token* als Authentifizierung für den sicheren Zugriff
- *Signaturen* um die Integrität der Nachrichten gewährleisten zu können
- *Verschlüsselung* um zu verhinden, dass die Informationen der Kommunikation an dritte gelangen kann

deren Merkmale folgend beschrieben werden.

2.3.1.1 Security Token

Ein Security Token ist eine bestimmte Form, sich zu Authentifizieren. Technisch wird dies über spezielle Header in den SOAP-Nachrichten realisiert. Es stehen mehrere Möglichkeiten zur Verfügung um, den Anforderungen des Umfelds entsprechend, eine einheitliche Authentifizierung und Authorisierung zu realisieren.

2.3.1.1.1 UsernameToken ist die einfachste Form der Authentifizierung. Der Client sendet mit jeder SOAP Nachricht einen Benutzernamen und das Passwort als Feld im Header mit. Das Passwort kann und sollte als Hashwert übertragen werden (siehe Grafik 2.3.1.1.1). So wird verhindert, dass dritte an das Passwort gelangen können. Diese Technik ist nur im einfachsten Fall geeignet, für komplexere Strukturen eher ungeeignet.

```
<wsse:UsernameToken>
    <wsse:Username>Eike</wsse:Username>
    <wsse:Password Type="wsse:PasswordDigest">
        KE6QugOpkPyT3EoOSEgT3OW4Keg=</wsse:Password>
    <wsse:Nonce>5uW4ABku/m6/S5rnE+L7vg==</wsse:Nonce>
    <wsu:Created xmlns:wsu=
        "http://schemas.xmlsoap.org/ws/2002/07/utility">
            2006-06-04T08:44:02Z
    </wsu:Created>
</wsse:UsernameToken>
```

Abbildung 2.2: UsernameToken mit Passwort Hashwert

2.3.1.1.2 BinarySecurityToken bietet die Möglichkeit, eine Authentifizierung mittels Kerberos Ticket oder PKI mit X.509 Zertifikat durchzuführen. Auch hierfür werden SOAP-Header definiert, in denen die entsprechenden Informationen mit jedem Aufruf vom Client an den Server übertragen werden. Gerade für größere Infrastrukturen ist diese Technik wesentlich geeigneter als ein einfaches UsernameToken.

2.3.1.2 Signaturen

Mittels XML Signature (XMLDsig) wird sichergestellt, dass die Nachricht nicht verändert wurde und auch wirklich von der angegebenen Identität stammt. Vergleichbar ist dies mit der Signierung von Emails bei PGP. Der Nachrichten Text wird durch einen, beiden Teilnehmern bekannten, Algorithmus gehasht und der meist 64-Bit lange Hashwert an die Nachricht angehängt. Der Empfänger wiederholt den Algorithmus und prüft, ob der Hash Identisch ist. Bei WS-Security wird der Hashwert in spezielle Header-Felder eingetragen. Auch der Algorithmus und der gehashte Bereich, bei WS-Security meist der komplette SOAP-Body, wird im Header definiert.
Das Hashing kann im einfachsten Fall mit einem UsernameToken, normalerweise aber einem X.509 Zertifikat oder auch einem Kerberos Ticket erfolgen. Grafik 2.3.1.2 zeigt ein BinarySecurityToken, dessen Kerberos Ticket signiert wurde.

```
<wsse:Security
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
  <wsse:BinarySecurityToken id="Kerberosv5ST"
                            wsse:ValueType="wsse:Kerberosv5ST"
                           wsse:EncodingType="wsse:Base64Binary">
    MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
  </wsse:BinarySecurityToken>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/
      <ds:SignatureMethod Algorithm="...#DES-MD5"/>
      <ds:Reference>
        <ds:Transforms>
          <ds:Transform>Algorithm="http://...#MyTranform"/>
          <ds:Transform>Algorithm="http://www.w3.org/2001/10/xml-exc
        </ds:Transforms>
        <ds:DigestMethod Algorithm="...#DES-MD5"/>
        <ds:DigestValue>AhdsgHu...</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>BL8jdfToEb11vXcMZNNjPOV...</ds:SignatureValue>
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#Kerberosv5ST"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
</wsse:Security>
```

Abbildung 2.3: Signiertes Kerberos Ticket

2.3.1.3 Verschlüsselung

Signierung und Verschlüsselung gehen bei WS-Security, ähnlich wie bei PGP, Hand in Hand. Die Verschlüsselung mittels XML Encryption soll die Vertraulichkeit der in den SOAP Nachrichten enthaltenen Informationen gewährleisten. Sie verhindert, dass die in SSL beschrieben Probleme entstehen und baut so aus Nachrichtensicht eine Ende-zu-Ende Verbindung auf (siehe Grafik 2.3.1.3).

Abbildung 2.4: Ende-zu-Ende Verbindung mit WS-Security Verschlüsselung

2.3.2 WS-Security 1.1 (WS-*)

In der Praxis hat sich gezeigt, dass die erste Spezifikation von WSS nicht der Weißheit letzter Schluss war.
Es hat sich gezeigt, dass die Authentifizierung nicht flexibel genug ist. Stellt ein Webservice Provider beispielsweise seine Authentifizierungs Methoden um, so muss er den Webservice selber anpassen. Weiterhin müssen auch noch die Clients informiert werden, was in einem SOA Umfeld, in dem der Server nicht alle potentiellen Clients kennen kann, zu einer ziemlich schwierigen Aufgabe wird.
Ebenfalls als problematisch wurde das permanente mitsenden der Authentifizierungsdaten bei jedem Aufruf empfunden. Hier macht es mehr Sinn, einen gemeinsamen Sicherheitskotext aufzubauen und in diesem zu Kommunizieren.
WS-* ist quasi die Vortsetzung von WSS und bietet neue Spezifikationen für die zuvor beschriebenen Probleme.

2.3.2.1 WS-Trust

Um mehr Flexibilität bei den unterschiedlichen Authentifizierungstechniken zu erreichen, führt WS-Trust eine zentrale Instanz für das Handling der Security Tokens ein, den Security Token Service. Clienten authorisieren sich bei diesem einen Server und bekommen dann ein Token vom Token Service, mit dem sie den Dienst in Anspruch nehmen können. Dieses Prinzip ist stark an das erfolgreiche Kerberos-Konzept angelehnt. Die Einführung des Security Token Services bewirkt, dass die Authentifizierung aus den Webservices extrahiert wird und somit diese unabhängig von der verwendeten Authentifizierungstechnik sind. Der Security Token Service ist zuständig für die Ausgabe, Prüfung, Erneuerung und Entwertung von Tokens.
Grafik 2.3.2.1 zeigt die Einbindung des Security Token Services (STS). Der Client sendet vor dem Zugriff auf den eigentlichen Webservice einen *Request Security Token (RST)* an den STS. Das RST beinhaltet das Authentifizierungstoken mit zusätzlichen Informationen darüber, welcher Webservice wie in Anspruch genommen werden soll.
Der STS prüft die Daten und erstellt entsprechend der Berechtigungen des Clients ein *SAML-Token* und signiert es bei Bedarf. Das SAML-Token sendet er als *Request Security Token Response (RSTR)*, an den Client zurück. Ein SAML-Token ist eine XML-Strukur, welche Berechtigungen für Dienste beschreibt.
Der Client greift nun mit dem SAML Token auf den Webservice zu. Da der Webservice und der STS sich kennen und vertrauen, muss sich der Client nicht mehr gesondert bei dem Webservice authentifizieren.

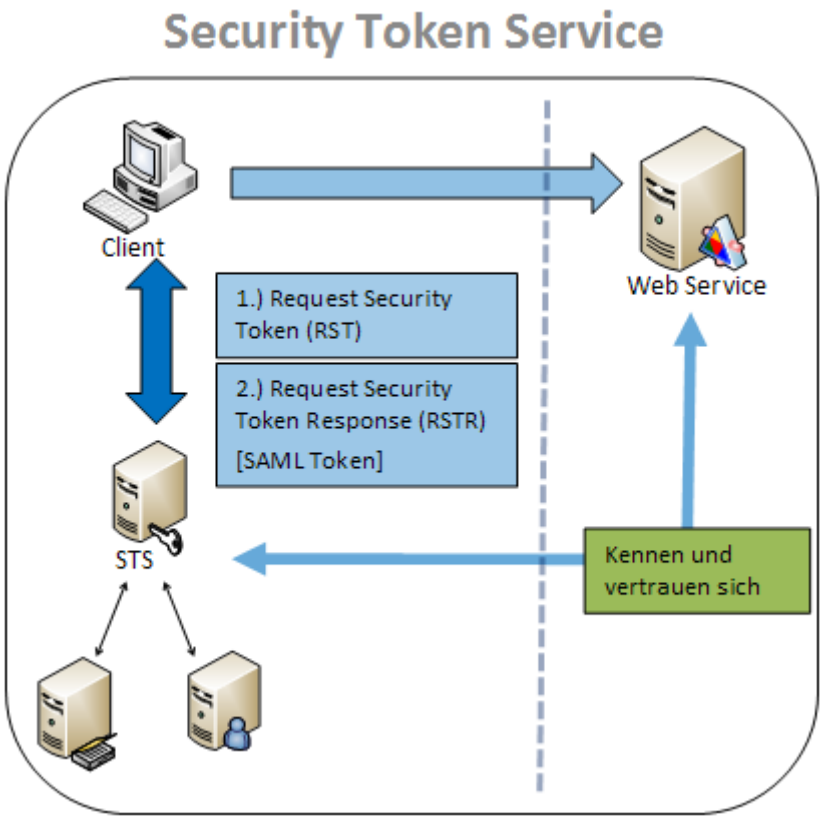

Abbildung 2.5: Security Token Service

2.3.2.2 WS-SecurityPolicy

WS-SecurityPolicy bietet eine Möglichkeit, die Sicherheitsanforderungen einen Webservices deklarativ und in maschinenlesbarer Form abzulegen. Dies ermöglicht, dass ein Service seine Sicherheitsanforderungen ändert, ohne jeden einzelnen Client informieren zu müssen. Die WS-SecurityPolicy beschreibt das erforderliche Token-Format und welchem Security Token Service der Webservice vertraut. Die WS-SecurityPolicy kann mehrere Alternativen anbieten, der Client wählt dann eine, die er erfüllen kann.
Die Einführung war nötig, da nur so lose gekoppelte Systeme mit unterschiedlichen Sicherheitsanforderungen Ad-Hoc eingebunden werden können. Der Client wertet die WS-SecurityPolicy des Dienstanbieters aus und liefert dann die erforderlichen Daten.

2.3.2.3 WS-Secure-Conversation

Das mitsenden der Authentifizierungsdaten mit jeder SOAP Nachricht erzeugt einen Protokoll-Overhead, dies soll durch WS-Secure-Conversation reduziert werden. WS-Secure-

Conversation dient dazu, mehrere aufeinanderfolgende Nachrichten in einem gemeinsamen Sicherheitskontext auszutauschen. Hierfür wird zu Beginn der Unterhalten (Conversation) ein Sicherheitskontext aufgebaut, in dem sich die beiden Teilnehmer bei folgenden Nachrichten weitere Schlüssel von dem Initialisierungsschlüssel ableiten.
Neben dem reduzierten Protokoll-Overhead wird der Security Token Service entlastet, der sich in der Praxis oftmals als potentielles Bottleneck erwiesen hat.

2.3.2.4 WS-Federation

Komplexere Infrastrukturen koppeln Zugriffe auf Webservices aus unterschiedlichen Sicherheitsdomänen. Es ist nicht optimal, wenn sich ein Client in jeder Sicherheitsdomäne vom neuen Authentifizierung muss. WS-Federation (WS-Fed:2004) ist eine Spezifikation, die die einheitliche Behandlung von Entitäten über Sicherheitsdomänengrenzen hinweg zu handhaben ist. Leider sind sich die unterschiedlichen Mitwirkenden an dieser Spezifikation (Microsoft, IBM, BEA u.A.) über einige Details aus Markttechnischen Gründen noch nicht einig, so dass sich die Verabschiedung dieser Spezifikation verzögert. WS-Federation ist für große SOAs von entscheidender Bedeutung, da sich so Systeme unterschiedlicher Hersteller koppeln ließen, genau das ist auch der Grund, warum die Vertreter der großen Firmen so zögerlich mit dieser Spezifikation umgehen. Nachdem IBM und Microsoft sich weigerten die Spezifikation von WS-Federation an das W3C oder OASIS abzugeben, ist es recht still um WS-Federation geworden und es ist zu befürchten, dass dies nie eine offzielle Spezifikation wird.

3 SOA, Webservices und Sicherheit

3.1 Bewertung

Die fast naiv anmutende Offenheit von Webservices musste leider nachträglich abgesichert werden. Sinnvoller wäre es gewesen, von vornherein Sicherheitaspekte in die Webservice Spezifikation mit einfließen zu lassen. Die erste Version von WS-Security war ein guter Ansatz, langte aber bei weiten nicht aus, um in großen, verteilten System für Sicherheit sorgen zu können.
Die neuen Standards lösen viele Probleme einheitlich und bilden eine solide Grundlage für den Aufbau künftiger SOAs. Die Spezifikationen basieren auf soliden, erprobten Technologien. Ein kluger Zug, da man dadurch verhindert hat, sich durch neue Techniken wieder neue Risiken in die Spezifikation zu holen. Die neuen Spezifikationen decken die derzeit offensichtlichsten Sicherheitsanforderungen ab und sind daher als positiv zu bewerten.
Leider ist es wie so oft - die meisten Anbieter unterstützen noch nicht die neuen Standards. Diese Standards einzusetzen macht aber erst Sinn, wenn sie auch von einem Großteil der teilnehmenden Entitäten in einer SOA angeboten werden. Wie soll sich beispielsweise ein Client verhalten, wenn der Webservice keine WS-Security-Policy anbietet? Die Einführung solcher Systeme an sich stellt schon eine gewaltige Aufgabe dar, wenn dann auch nur eine Teilmenge der Teilnehmer diese anbieten, wird es fast unmöglich.
Es besteht weiterhin die Gefahr, dass die üblichen proprietären Rangeleien zu Inkompatibilitäten führen und am Ende die Struktur einer Architektur doch nicht so offen und flexibel ist, wie man es ursprünglich erhofft hatte.
Weiterhin ist sehr ärgerlich, wenn auch wirtschaftlich nachvollziehbar, dass gewisse Spezifikationen nicht an unabhängige Gremien abgegeben werden. Dies führt dazu, dass Spezifikationen nie offziell werden und somit kaum mehr als eine fundierte Dokumentation über Produkteigenschaften bestimmter Hersteller sind.
Neben technischen unzulänglichkeiten bleibt natürlich die Frage, ob diese Absicherungen wirklich auslangen, um eine SOA als sicher zu bezeichnen. Selbstverständlich ist dies nicht der Fall. Auch mit WS-Security & Co. ist eine SOA nicht 100% sicher. Auch weiterhin besteht die Gefahr von unerlaubten Systemangriffen wie SQL-Injection oder Buffer-Overflows (Giblin, 2005).
Während des Vortrags kam die Frage auf, wie sich die WS-Security auf die Performance

auswirkt. Da WS-Security eine nachträglich aufgesetzte Technik ist, wirkt sie sich negativ auf die Performance aus. Wie negativ sich WS-Security auf die Perormance auswirkt ist von der Situation abhängig. Natürlich sollte man aus Performanzgründen nicht auf Sicherheitsaspkete verzichten. (Satoshi Makino und Nakamura, 2005) schlägt einen Template-basierten Ansatz als Umgang mit dem Problem vor.

3.2 Aussicht

Leider fehlte für diese Arbeit die abschließende Zeit für einige praktische Tests vor allem hinsichtlich der Kompatibilität. Dies, in Verbindung mit weiterführenden Tests wäre ein interessantes Thema für das nächste Semester.
Weiterhin interessant wären Konzepte, wie man mit den angesprochenen Problemen der Inkompatibilität und sukzessiven Einführung der Spezifikation in Service Orientierten Architekturen umgeht.
Das Projekt „Pervasive Gaming Framework“ im nächsten Semester wird sicherlich einige Möglichkeiten bieten, das gewonnene Wissen einfließen zu lassen.

Literaturverzeichnis

[WS-Fed:2004] : *WS-Federation*. – URL ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf

[et Al 2006] AL, Epstein et: Software security and SOA: danger Will Robinson! (2006). – URL http://ieeexplore.ieee.org/iel5/8013/33481/01588834.pdf?tp=&arnumber=1588834&isnumber=33481

[Eckert 2004] ECKERT, Claudia: *IT-Sicherheit*. Oldenbourg Wissenschaftsverlag, 2004. – ISBN 3-486-20000-3

[Giblin 2005] GIBLIN, Takeshi Imamura; Michiaki Tatsubori; Yuichi Nakamura; C.: Web services security configuration in a service-oriented architecture. (2005). – URL http://portal.acm.org/citation.cfm?id=1062745.1062898

[OASIS] OASIS: *Web Service Security (WSS)*. – URL http://www.oasis-open.org/committees/wss

[Peikari und Chuvakin 2004] PEIKARI, Cyrus ; CHUVAKIN, Anton: *Security Warrior*. O'Reilly, 2004. – ISBN 0-596-00545-8

[Rubarth 2005] RUBARTH, Thies: *Web Service Security*. 2005. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/rubarth/abstract.pdf

[Satoshi Makino und Nakamura 2005] SATOSHI MAKINO, Kent T. ; NAKAMURA, Yuichi: Improving WS-security performance with a template-based approach. (2005). – URL http://ieeexplore.ieee.org/iel5/10245/32665/01530849.pdf?tp=&arnumber=1530849&isnumber=32665

[Zimmermann 2005] ZIMMERMANN, Olaf: *Jumpstarting SOA Projects*. 2005. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2005/zimmermann/slides.pdf