"SAML - Security Assertion Markup Language"

Björn Waide

Seminararbeit

im Institut für

Betriebssysteme und Rechnerverbund

Abstract: Die folgende Arbeit beschreibt die Entwicklung vom Benutzer-zentrierten WWW hin zum Service orientierten Netz und die sich daraus ergebenen Probleme insbesondere bezüglich der Sicherheit in solchen Systemen. Als eine mögliche Lösung wird der OASIS Standard SAML diskutiert.

1. Einleitung

1.1 Motivation

Die vorliegende Arbeit im Rahmen des Seminars „Sicherheit in Verteilten Systemen“ beschäftigt sich mit dem Aspekt der Sicherheit von WebServices und dort im Besonderen mit SAML.
Die Notwendigkeit für die Entwicklung von SAML ergibt sich aus der Etablierung von WebServices als festem Bestandteil der verteilten IT-Struktur. Genügte es in der Vergangenheit, zur Authentifizierung von Nutzern verteilter System Login-Masken oder ähnliches bereitzustellen, gestaltet sich das unter dem Gedanken des „Service-Nets“ komplizierter. Im Gegensatz zu herkömmlichen Anwendungen im WWW, die Menschen als Endnutzer voraussetzen, richten sich Anwendungen im „Service-Net“ primär an andere Anwendungen bzw. Dienste, also Services. Das „Service-Net“ ist also nicht anders als ein Netzwerk unterschiedlichster Anwendungen, die als Dienst anderen Anwendungen zur Verfügung stehen. Durch den Gedanken der losen Kopplung und des fehlenden menschlichen Endanwenders scheiden die üblichen Mittel zur Authentifizierung z.B. mittels Login-Masken aus. An diese Stelle soll zukünftig SAML treten, das sichere Authentifizierung auch zwischen Diensten möglich machen soll.

1.2 Aufbau der Arbeit

Die Arbeit gliedert sich in fünf Kapitel. Nach der Einleitung folgt im Kapitel 2 ein technologischer Überblick. Dieser Teil richtet sich primär an Leser, die mit den Technologien, die SAML zugrunde liegen, noch nicht hinreichend vertraut sind. Es wird eine kleine Einführung in die Historie von Web-Anwendungen sowie in die Themen WebServices, XML und SOAP gegeben.
Im darauf folgenden Kapitel 3 geht es um die Probleme, die der Auslöser für die Entwicklung von SAML waren. Es werden einige Szenarien aufgezeigt, in denen Sicherheit im

Zusammenhang mit WebServices eine Rolle spielt. Darüber hinaus wird gezeigt, weshalb bereits existierende Sicherheitsmechanismen auf WebServices nicht angewandt werden können oder sollten.
Das vierte Kapitel behandelt die Technologie, um die es in dieser Arbeit gehen soll. Es wird gezeigt, wie die im vorangegangenen Kapitel erwähnten Probleme durch SAML gelöst werden.
Nach einer abschließenden Zusammenfassung der Ergebnisse dieser Arbeit folgt im fünften und letzten Kapitel ein Ausblick auf zukünftige Weiterentwicklungen von SAML und möglicher Konkurrenten.

2. Technologischer Überblick

2.1 Historie

Zu Beginn des World Wide Web in den Anfängen der 90er Jahre erstellte man statische Seiten mittels einer eigens dafür entwickelten Auszeichnungssprache namens HTML ([FISCHER]). Diese Seiten verknüpfte man untereinander mit Hilfe von Hyperlinks, die der Auszeichnungssprache ihren Namen gaben. Diese Seiten konnten nun von jedem, der Zugriff auf das jeweilige Netzwerksegment hatte, über zunächst text-, später grafischbasierte Browser angesehen werden.
Die Idee lag nahe, bestimmte sich häufig ändernde Inhalte nicht händisch aktualisieren zu müssen, sondern die entsprechenden Seiten dynamisch zu erzeugen. Mit denen sich hieraus entwickelnden Möglichkeiten veränderten sich rasch sowohl Inhalte als auch Publikum des anfangs als reines Wissenschaftsnetzwerk gedachten Systems. Die bereitgestellten Informationen sollten bald nicht mehr uneingeschränkt allen Interessierten zur Verfügung stehen, sondern nur einer gezielten Personengruppe zugänglich gemacht werden. Diese Anforderungen verschärften sich, als aus dynamischen Web-Seiten bald ganze Anwendungen erwuchsen. Es entstanden im Laufe der Zeit etliche Technologien, die diesem Umstande Rechnung trugen.
Aufgrund der sich verschlechternden wirtschaftlichen Lage sahen sich vor allem die Global Player immer mehr gezwungen, ihre Kosten in allen Bereichen zu senken. Insbesondere bei den Kosten für die Kommunikation zwischen den Firmen, die eng miteinander verzahnt sind, wie z.B. Automobilkonzerne mit ihren Zulieferern, erhoffte man sich durch den Einsatz von Internet-Technologie erhebliche Einsparmöglichkeiten. Dies gab der Entwicklung in diesem Bereich einen gehörigen Schub. Zur weiteren Kostensenkung sollte die B2B (Business to Business) Kommunikation nicht nur vereinfacht, sondern auch in weiten Teilen automatisiert werden. Hierzu war ein Umdenken bei der Programmierung derartiger System erforderlich. Anwendungen mit Bedienoberflächen in HTML sind zur automatisierten Bearbeitung nicht gut geeignet, da mit HTML das Layout der Daten beschrieben wird und sich die Semantik nur einem menschlichen Betrachter aus dem Inhalt und dem Kontext erschließt. Mit XML stand nun schon seit geraumer Zeit ein Standard zur Verfügung, der mit genau dieser Problematik aufräumen sollte, indem er die Semantik der Daten in den Vordergrund stellte. Ein Datenaustauschformat war also vorhanden, lediglich Art der Übertragung gestaltete sich nach wie vor schwierig. Zwar waren mit CORBA und Java RMI Technologien zur entfernten Kommunikation zwischen Systemen vorhanden, jedoch erwies sich die Umsetzung dieser zum Teil sehr komplexen Technologien in einem Internet-Szenario als sehr aufwendig und oft nicht praktikabel. Genau in diese Lücke traten dann WebServices. Sie sollten das Internet zu einem „Service-Net“ erweitern, in dem von Rechnern benutzbare Dienste angeboten werden.
Funktionsweise und Probleme von WebServices sowie die Rolle von SAML in diesem Szenario sollen in den folgenden Kapiteln erläutert werden.

2.2 WebServices

WebServices sind mehr als ein Buzz-Word, auch wenn die inflationäre Verwendung dieses Begriffes bei gleichzeitig ausbleibender sichtbarer Realisierung verwendbarer Lösungen dies vermuten lässt. Das Problem bei WebServices ist, dass sie nicht einfach nur eine neue Technologie sind, die man lernen muss, sondern in gewisser Weise auch eine neues Denken erfordern. Der Begriff Service, also Dienst, wird oft – da nahe liegend – mit bereits existierenden Diensten im Internet assoziiert, wie z.B. Google als Suchdienst begriffen wird. Alle diese „herkömmlichen" Dienste im Netz kennzeichnet jedoch die Voraussetzung eines menschlichen Nutzers. Das Ziel von WebServices ist aber genau eine Entkopplung des Dienstes vom Nutzer, einer allgemeineren Bereitstellung der angebotenen Dienstleistung auch für nicht-menschliche Gegenüber, also z.B. Rechner oder weitere WebServices.
An dieser einfachen Anforderung kann man bereits erkennen, warum der Umgang mit WebServices ein neues Denken erfordert. Zum einen lassen sich viele der etablierten Geschäftsmodelle auf WebServices nicht anwenden. Eine zumeist werbefinanzierte Suchmaschine wird ihren Dienst nicht ohne weiteres als WebService anbieten, da eine Platzierung von Werbung hier nicht sinnvoll möglich ist. Hier sind Marketing-Strategen und Manager gefragt, neue Geschäftsmodelle für WebServices zu erdenken.
Zum anderen kommen WebServices a priori ohne grafische Benutzerschnittstelle daher, sind also unsichtbar.
Zusammengenommen führt das zu dem Eindruck, WebServices wären ein Lösung auf der Suche nach einem Problem. Wie im Kapitel 2.1 Historie bereits ausgeführt, existieren diese Probleme jedoch bereits seit einigen Jahren und WebServices scheinen in der Lage zu sein, einen Teil davon lösen zu können. Einer der Bereiche, in dem sich WebServices bereits erfolgreich etabliert haben, ist EAI – Enterprise Application Integration. Große und gewachsene Unternehmen besitzen meist eine ebensolche IT-Landschaft: Verteilte und heterogene Systeme bestimmen das Bild. Diese unter einer einheitlichen Fassade zu integrieren ist das Ziel von EAI. Und WebServices sind der Weg dahin.
In einem anderen Bereich, für den man sich durch den Einsatz von WebServices viel versprochen hatte, hinken die Ergebnisse den Erwartungen ein gehöriges Stück zurück. Gemeint sind B2B-Anwendungen, wie sie ebenfalls im Kapitel 2.1 Historie bereits kurz angerissen worden sind. Die Gründe für die Erfolge in dem einen und insbesondere die Verzögerungen in dem anderen Einsatzfeld sind Thema dieser Arbeit und werden in den Kapiteln 3. Webservices und Sicherheit und 4. SAML detaillierter behandelt.
Der Erfolg im Bereich EAI lässt sich zum großen Teil darauf zurückführen, dass WebServices auf etablierte Standards aufsetzen und dabei selbst ein plattformübergreifender Standard sind (siehe [W3CWS]). WebServices sind kurz gesagt XML-basierte RPCs (Remote Procedure Calls). Diese entfernten Methodeaufrufe nutzen als Übertragungsprotokoll üblicherweise HTTP, sind also „Firewall-geeignet".
Für eine etwas detailliertere Übersicht über die Architektur von WebServices soll die schematische Abbildung 1 dienen.
Zu sehen ist die an das OSI-Schichten-Modell angelehnte Einteilung in verschiedene Layer, von denen zwei vertikal verlaufen. Mit linken dieser beiden Layer mit dem Namen „Security" werden wir uns im Laufe dieser Arbeit noch genauer beschäftigen. Sichtbar wird auch die lose Kopplung und die damit weitestgehende Unabhängigkeit vom Transportprotokoll, die in einem extra Layer angedeutet sind. HTTP ist dabei aber das verbreitetste für die Kommunikation mit und zwischen WebServices. Einige der in der Abbildung erwähnten Begriffe wie z.B. SOAP und XML werden in dieser Arbeit noch erläutert. Für tiefergehende Informationen insbesondere zu den hier unerwähnt bleibenden Aspekt wie WSDL sei an dieser Stelle auf [W3CWS] verwiesen.

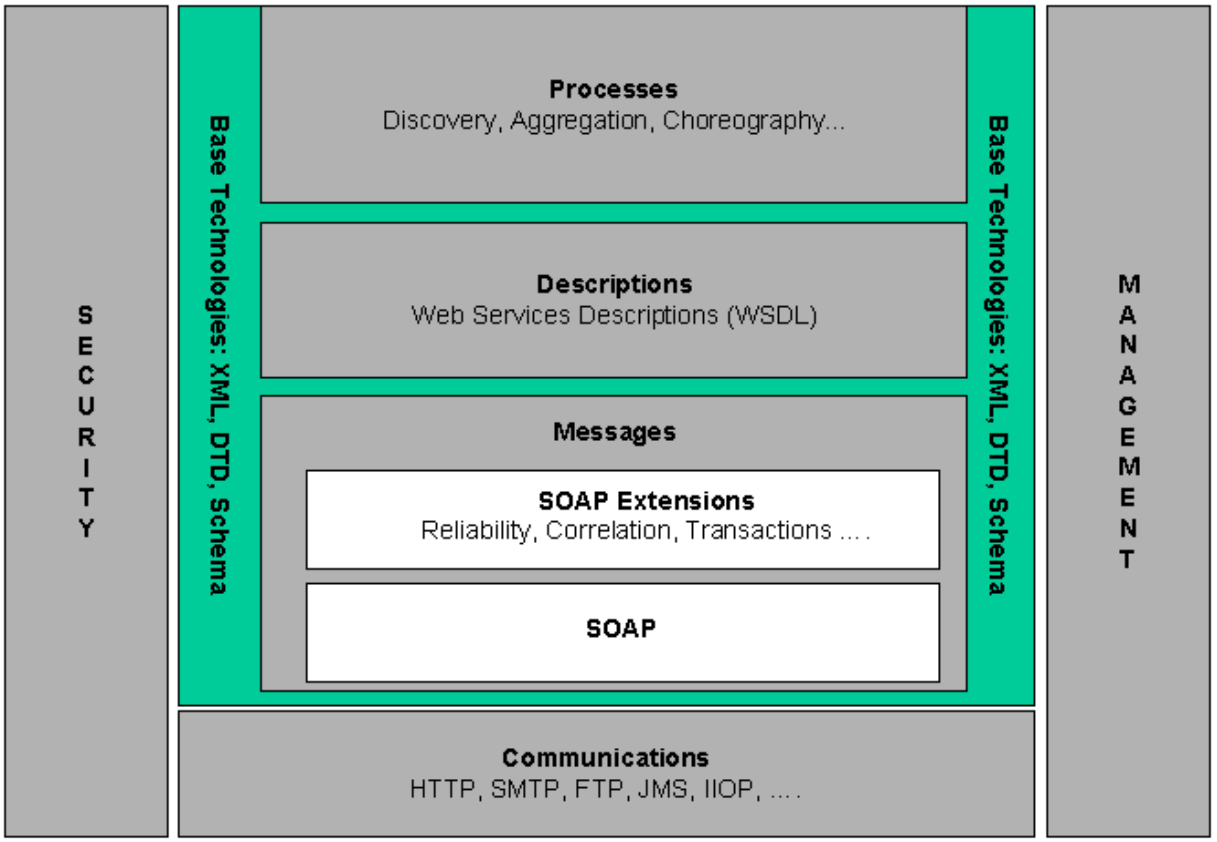

Abbildung 1: Architektur von WebServices (Quelle: [W3CWSA])

2.3 XML

Als einer der WebServices zugrunde liegenden Standards soll an dieser Stelle auch kurz auf XML eingegangen werden. Das Akronym steht für eXensible Markup Language und damit ist auch schon das Wichtigste gesagt: XML ist eine Meta-Sprache zur Auszeichnung von Daten durch so genannte Tags. Diese Tags lassen sich selbst definierten, wofür das „X“ in XML steht. Das Auszeichnen eines Textes ist dabei als semantische Auszeichnung zu verstehen, im Gegensatz zur HTML, bei dem das Layout der Daten im Vordergrund steht.
XML ist damit die Grundvoraussetzung für die Idee des „Service-Nets“, in dem Anwendungen automatisiert miteinander kommunizieren und daher die Sprache des Gegenübers verstehen müssen.
Auf XML basieren eine Reihe weiterer Sprachen, denen dann oft das „X“ im Namen fehlt, da die verfügbaren Tags und ihre Bedeutung dort festgeschrieben und eben nicht erweiterbar sind. SAML ist eine dieser Sprachen und wird im Kapitel 4. SAML behandelt.

2.4 SOAP

Der Begriff SOAP steht für Simple Object Access Protocol und beschreibt ein XML basierendes Protokoll zum Austausch von Informationen über Objekte oder Entitäten. Die beiden Begriffe WebServices und SOAP werden in der einschlägigen Literatur oft synonym verwendet. Das liegt sicher darin begründet, dass die WebService Working Group des W3C [W3CWS] SOAP als Binding zum Zugriff auf WebServices definiert. SOAP ist dabei aber keineswegs die einzige Möglichkeit, sondern nur die einzige als Standard definierte Variante. Es ließen sich jederzeit weitere Bindings definieren, z.B. ein Binding für die Bereitstellung von Enterprise Java Beans als WebService über RMI/IIOP. Da sich SOAP aber inzwischen als Quasi-Standard für den Zugriff auf WebServices etabliert hat, soll hier ein kurzer Überblick über das Protokoll gegeben werden.

Ein einfaches Beispiel soll den grundsätzlichen Aufbau einer SOAP Nachricht aufzeigen.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

Beispiel 1: Eine einfache SOAP Nachricht

Die drei wesentlichen Bestandteile einer SOAP Nachricht sind:

- **Envelope**: Der „Umschlag“ für eine Nachricht
- **Header**: Enthält einige Meta-Daten zu der Nachricht
- **Body**: Enthält die eigentliche Nachricht

Zwei Hauptziele bei der Entwicklung dieses W3C Standards [W3CSOAP] waren Einfachheit und Erweiterbarkeit. Insbesondere die zweite Eigenschaft macht sich SAML zunutze, wie in einem späteren Kapitel noch erläutert wird.

3. Webservices und Sicherheit

Wie bereits an früherer Stelle besprochen wurden WebServices zunächst im Bereich EAI erfolgreich eingesetzt. Im großen Feld der B2B Kommunikation verlief die Implementierung um einiges schleppender. Für diesen Umstand gibt es Gründe, die in diesem Kapitel aufgezeigt werden sollen.

Der erste Vorschlag für WebServices brachte bereits alles mit, was man für einen erfolgreichen Einsatz bei der Integration von Legacy Anwendungen brauchte. Viele Unternehmen und auch Open Source Entwickler brachten sehr schnell erste Frameworks auf den Markt, mit denen sich WebServices implementieren ließen, die eine Schnittstelle zu der darunter liegenden Anwendung lieferten. Nun war es recht einfach, einige weitere WebServices zu entwickeln, die diese abfragten und so nach außen hin den Anschein eines homogenen Systems abgaben. Alle abzufragenden WebServices befanden sich innerhalb des gleichen Unternehmens. Sicherheitsaspekte konnten so im einfachsten Fall vernachlässigt oder aber durch Implementierung proprietärer Sicherheitsmechanismen abgedeckt werden.
Ganz anders dagegen verhält es sich bei B2B Szenarien. Hier sind zum einen die Sicherheitsanforderungen schon um einiges höher und zum anderen muss eine standardisierte Lösung gefunden werden, da in diesem Fall ja Anwendungen verschiedener Unternehmen kommunizieren sollen.
Aus strategischen Gründen enthielt die erste Version der WebService Spezifikation noch keine Sicherheitsmechanismen für B2B Szenarien und ähnlich sicherheitskritische Anwendungsfälle. Hier seien beispielhaft noch Bezahldienste und Internet Banking genannt.
Stattdessen setzte man auf etablierte Verfahren weiter unten liegender Schichten im OSI-Modell. Mit SSL bzw. HTTPS gab es bereits Sicherheitsprotokolle auf der Transportschicht. Da WebServices primär HTTP als Übertragungsmedium nutzen, ist hier die Sicherheit auf

dem Übertragungsmedium bereits implizit gegeben. SSL-Verschlüsselungen bergen jedoch einige Nachteile, die die Nutzung in einigen WebService Szenarien ausschließen.

1. Eng verknüpft mit dem Übertragungsendpunkt; erlaubt es so mehreren WebServices nicht, den Transport für ein unabhängiges Sicherheitsschema zu teilen
2. Sicherheit gilt nur für das Übertragungsmedium, nicht dahinter
3. Nur Ende-zu-Ende Sicherheit, keine Schutz für Zwischenstationen

Insbesondere Punkt 3 verhindert, dass WebServices in sicherheitskritischen Szenarien ihr volles Potential ausspielen können. Die hierarchische Struktur, in der ein WebService weitere WebServices abfragt, die wiederum die Bearbeitung der Anfrage an andere WebServices delegieren, kann auf diesem Wege nicht ausgenutzt werden. Eine Weitergabe des Client-Zertifikats an weitere Server ist nicht möglich, da ein Zertifikat immer zwischen dem Server als Endpunkt und dem Client ausgehandelt wird.
Auch andere Sicherheitsmechanismen wie z.B. S/MIME waren für andere Anwendungszwecke entwickelt worden und genügten so nicht allen Anforderungen, die WebServices an sie stellten.
Zu den geforderten Features eines Sicherheitsmechanismus für WebServices gehören unter anderen die Standardisierung des Verfahrens, die Möglichkeit für Server-zu-Server Verbindungen sowie die Portabilität der Identitäten, Profile und Zertifikate.

Die Architektur sah genau für diese Fälle aber Erweiterungsmöglichkeiten vor. In Abbildung 1 wird dies durch eine parallel verlaufende Box mit dem Inhalt „Security“ angedeutet. Formal werden diese Mechanismen in der SOAP Spezifikation beschrieben. Hier existieren zwei Punkte mit den Namen Feature und Module, die für Erweiterungen aller Art genutzt werden können. Bei der Entwicklung des Protokolls hatte man den Sicherheitsaspekt aber bereits ins Auge gefasst. Als Nutzungsmöglichkeit dieser Schnittstellen sind eben solche Szenarien explizit erwähnt.
Inzwischen gibt es nicht nur eine Erweiterung, die sich diese Schnittstelle zunutze macht. Die beiden größten sind das in dieser Arbeit behandelte SAML sowie das von Microsoft als Quasi-Konkurrenz ins Spiel gebrachte WS-Security Framework, das die Aspekte von SAML in sich subsumiert.

4. SAML

Aus der Not geboren, für bestimmte Szenarien der WebService Nutzung keine geeigneten Sicherheitsmechanismen zur Verfügung zu haben, entwickelte eine Gruppe der OASIS in der zweiten Hälfte des Jahres 2001 den SAML Standard (gesprochen „Semmel“ oder „Samuel“) ([OASISSEC]). Das Akronym steht für *Security Assertion Markup Language* („Auszeichnungsspreache für Sicherheitsbestätigungen“) und beschreibt ein XML-basierendes Framework zum Austausch von Sicherheitsinformationen zwischen Geschäftspartnern über das Internet. OASIS ([OASIS]) ist eine Non-Profit Organisation, die sich der Entwicklung von eBusiness Standards verschrieben hat.

SAML basiert zu großen Teilen auf einer Arbeit der Firma Netegrity [NETEGRITY], die Ende 2000 bei der OASIS einen S2ML genannten Vorschlag für ein Sicherheitsframework eingereicht hatten. Die Gruppe der OASIS, die ausgehend von diesem Vorschlag SAML entwickelten, besteht aus namhaften Mitgliedern. Exemplarisch seien IBM, Nokia, SAP und SUN genannt. Diese kurze Auflistung der Who-is-who in der IT-Branche zeigt schon, dass SAML nicht fürchten muss, von der Wirtschaft nicht akzeptiert zu werden. Konkurrenz ist trotzdem vorhanden, denn mindestens ein großer Name fehlt: Microsoft. Anfangs noch in den Entwicklungsprozess mit einbezogen, ging Microsoft bald eigene Wege, als sich zeigte, dass sie einige ihrer Interessen nicht durchsetzen konnten. Microsoft verließ das Gremium und entwickelte eine eigene Lösung unter dem Namen WS-Security.

4.1 Anwendungsfälle

SAML wurde im Hinblick auf drei Anwendungsfälle oder Use Cases entwickelt:

1. Single Sign On (SSO)
2. Distributed Transactions
3. Authorization Services

Zu jedem dieser Anwendungsfälle gehören wiederum Szenarien, die den jeweiligen Use Case detaillieren. Zu den folgenden drei Diagrammen sei noch angemerkt, dass die dargestellten Figuren keineswegs notwendigerweise als menschliche Nutzer interpretiert werden müssen. In den meisten Fällen geht es bei WebServices ja gerade um die Inter-Maschinen Kommunikation.

1. Single Sign On (SSO)

In diesem Szenario geht es darum, dass ein einmal an der Domain X angemeldeter Nutzer sich bei dem Besuch der Domains Y und Z nicht erneut anmelden muss, da X, Y und Z in irgendeiner Weise einem Verbund angehören und sich gegenseitig trauen. Die Anmeldeinformationen von X werden beim Besuch von Y und Z weitergeleitet.

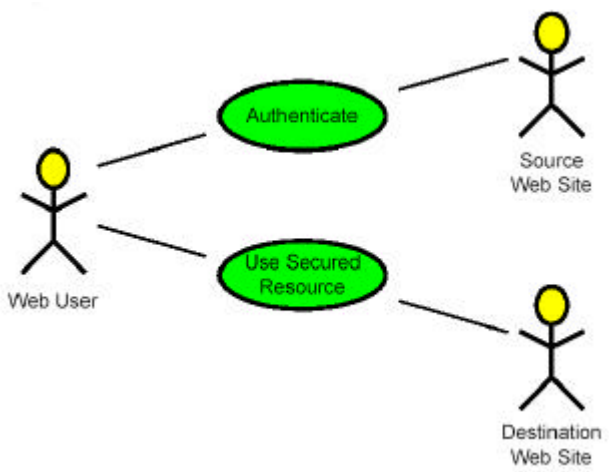

Abbildung 2: Single Sign On (Quelle: [MALER])

Microsoft brachte mit dem System Passport vor einigen Jahren bereits eine Lösung auf den Markt, die den gleichen Anwendungsfall im Blick hatte. Da Microsoft anfangs das Hosting der Nutzerinformationen vollständig selbst übernehmen wollte, scheiterte das Vorhaben zu Beginn an mangelnder Akzeptanz. Inzwischen kann man Passport als Produkt erwerben und in seinem Unternehmen einsetzen, ohne Nutzer spezifische Informationen in die Hände von Microsoft zu geben. Mit DirectorySmart von OpenNetwork [OPENNET] existiert auch ein Produkt, das Sicherheitsinformationen über einen Passport authentifizierten Nutzer an vertrauenswürdige Partner per SAML weiterleitet.

2. Distributed Transactions

Das Szenario für verteilte Transaktion ist dem von SSO sehr ähnlich. Nur geht es hierbei nicht um die Anmeldung eines Users an mehreren verschiedenen Web-Seiten, sondern um eine Transaktion, z.B. eine Bestellung, bei der mehrere Parteien beteiligt sind und die sich die Sicherheitsinformationen teilen.

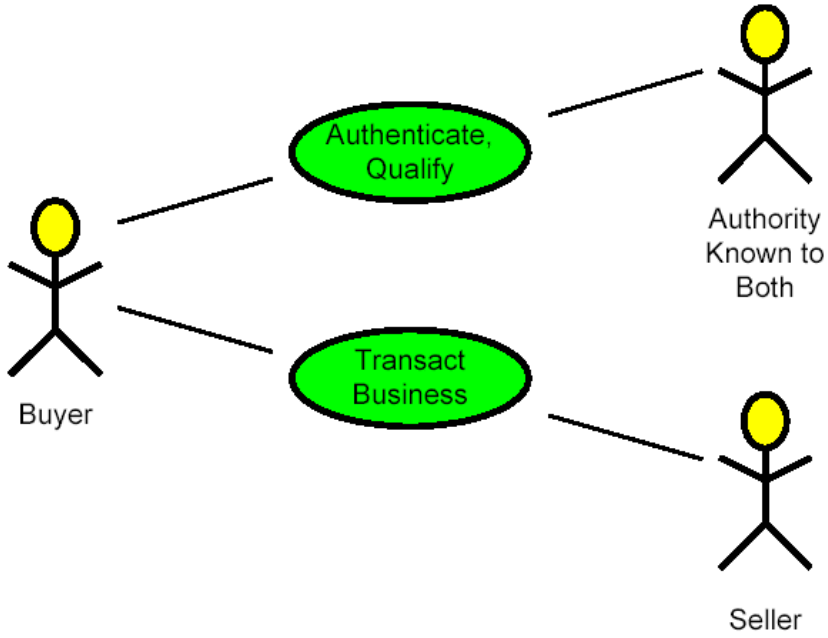

Abbildung 3: Distributed Transactions (Quelle: [MALER])

3. Authorization Services

Beim Szenario der Autorisierungsdienste geht es primär um hierarchisch verschachtelte WebService Aufrufe. Eine andere Bezeichnung für dieses Szenario lautet „Man-in-the-middle“. Hierbei läuft also die Kommunikation mit dem Endpunkt über eine Zwischenstation, die ebenfalls am Autorisierungsprozess beteiligt ist. Dieses Szenario lässt sich wie eingangs erwähnt mit SSL nicht abbilden, da hier eine Weitergabe der Security Informationen nicht möglich ist.

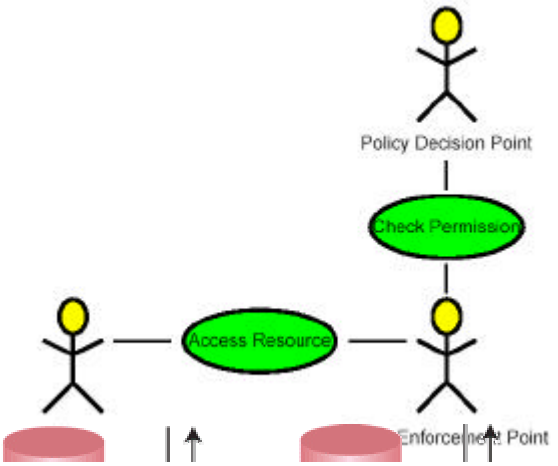

Abbildung 4: Authorization Services (Quelle: [MALER])

4.2 Technische Realisierung

Auch wenn bisher in dieser Arbeit von SAML nur im Zusammenhang mit WebServices die Rede war, ist die Spezifikation in dieser Hinsicht weiter gefasst. Allerdings schreibt die Spezifikation vor, dass jede Implementierung mindestens ein SOAP Binding mitbringen muss. Somit ist die Intention und Ausrichtung von SAML trotz prinzipieller Erweiterungsmöglichkeiten klar.
Weitaus allgemeiner sind dagegen die Begriffe gefasst, die sich auf die eigentlichen Sicherheitsaspekte beziehen. Einen Überblick über die in SAML vorkommenden Elemente und ihre Beziehungen liefert Abbildung 5.

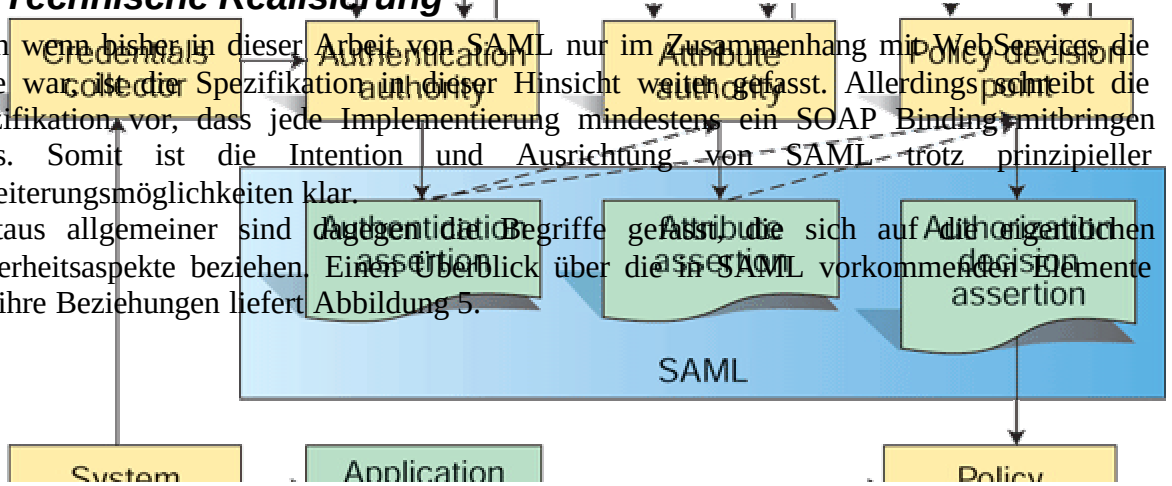

Abbildung 5: SAML Domain Model (Quelle: [XMLMAG])

Der zentrale Begriff ist dabei der der *Assertion*. Zwei weitere Begriffe finden sich in der Abbildung 4: Authorization Services (Quelle: [MALER]) wieder, nämlich *policy enforcement point* (PEP) und *policy decision point* (PDP). Ersterer beschreibt abstrakt eine Stelle – meist ein WebService – der eine Authentifizierung erfordert. Diese kann jedoch nicht immer vom befragen WebService – oder abstrakter PEP – selbst vorgenommen werden. Stattdessen wird die Anfrage an einen weiteren Punkt - den PDP – weitergeleitet, der diese Aufgabe erledigt und eine entsprechende Assertion zurückschickt.

SAML geht dabei den in .
Im darauf folgenden Kapitel 3 erwähnten Weg und implementiert die Sicherheitsmechanismen auf der Dienstschicht. Dabei definiert es lediglich eine XML-Syntax, in der sich sicherheitsbezogene Informationen ausdrücken lassen und stellt keine API dar.

Eine *Assertion* oder *Bestätigung* ist also nichts weiter, als ein einem bestimmten Syntax genügendes XML-Dokument.
SAML definiert oder implementiert auch keine neuen kryptografische Verfahren zur Verschlüsselung der Daten, sondern beschreibt nur, wie man mittels Standardverfahren verschlüsselte oder signierte Daten zwischen zwei Parteien austauscht. Eine solche Assertion zeigt Beispiel 2: SAML Assertion.

```
<?xml version="1.0" encoding="utf-8"?>
<Assertion
    xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    MajorVersion=”1”
    MinorVersion=”0”

AssertionID=”http://www.myEMarketPlace.com/AuthenticationService/SAMLAssert
ions/786”
    Issuer=”http://www.myEMarketPlace.com”
    IssueInstant=”2003-03-11T02:00:00.173Z”>
    <Conditions
        NotBefore=”2003-03-11T02:00:00.173Z”
        NotOnOrAfter=”2003-03-12T02:00:00.173Z”/>
    <AuthenticationStatement
        AuthenticationMethod=”urn:ietf:rfc:3075”
        AuthenticationInstant=”2003-03-11T02:00:00.173Z”>
        <Subject>
            <NameIdentifier
                NameQualifier=”http://www.myEMarketPlace.com”>
                MyTourOperator
            </NameIdentifier>
            <SubjectConfirmation>
                <ConfirmationMethod>
                    urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
                </ConfirmationMethod>
                <ds:KeyInfo>
                    <ds:KeyName>MyTourOperatorKey</ds:KeyName>
                    <ds:KeyValue> ... </ds:KeyValue>
                </ds:KeyInfo>
           </SubjectConfirmation>
        </Subject>
    </AuthenticationStatement>
    <ds:Signature>...</ds:Signature>
</Assertion>
```

Beispiel 2: SAML Assertion

Übersetzt in natürliche Sprache bedeutet diese Bestätigung:
Das Objekt oder die Entität (das kann eine Person oder eine Organisation sein) mit dem Namen „MyTourOperator“ ist Besitzer des öffentlichen Schlüssels „MyTourOperatorKey“. Die sogenannte Authentication Authority hat MyTourOperator mittels XML digital signature bestätigt. Diese Bestätigung ist gültig vom 11.03.2003 bis zum 12.03.2003.

4.3 Stand der Dinge

Die aktuelle Versionsnummer der SAML Spezifikation ist 1.0. Doch in wenigen Tagen steht bereits Version 1.1 zur Veröffentlichung an.
Trotzt dieser kleinen Versionsnummern existieren bereits einige Implementierungen für SAML.
Die den für SAML Ausschlag gebenden Vorschlag einreichende Firma Netegrity ([NETEGRITY]) stellt eine Java API namens *JSAML* ([JSAML]) zur Verfügung, mit der sich programmatisch SAML *Assertions* erstellen lassen.
SUN liefert ein komplettes Produkt unter dem Namen *SUN ONE Identity Server* ([SUNONE]), das als Out-of-the-box Lösung für Single Sign On Szenarien daherkommt mit voller SAML Unterstützung.
Ob bereits Anwendungen auf SAML Basis produktiv im Einsatz sind, ließ sich über die öffentlich zugänglichen Quellen nicht in Erfahrung bringen. Es ist jedoch davon auszugehen, dass es bisher lediglich vereinzelte Implementierungen gibt, da SAML erst Ende 2002 offiziell veröffentlicht wurde.

5. Zusammenfassung und Ausblick

Mit Aufkommen der WebServices ergaben sich unzählige neue Möglichkeiten zur Implementierung verteilter Systeme. An vielen Stellen konnten WebServices bereits erfolgreich eingesetzt werden, insbesondere bei EAI. Die großen Kosteneinsparungen bei der Automatisierung der B2B Kommunikation ließen sich mangels geeigneter Sicherheitsmechanismen für WebServices nicht so schnell umsetzen. Mit SAML existiert jetzt ein offener und von der Industrie breit unterstützter Standard, der dieses Manko zu beseitigen verspricht.
Trotz der Konkurrenz aus dem Hause Microsoft mit ihrem WS-Security Framework wird sich SAML in Zukunft als Sicherheitsmechanismus für WebServices etablieren können. Mit Microsoft Marktmacht und ihrem bereits seit einiger Zeit verfügbaren Produkt *Passport* wird SAML vorerst nicht der einzige Weg sein, um WebServices sicher zu gestalten. Wie bei den anderen „großen“ Protokollen wie HTTP oder TCP werden sich hier auf Dauer aber keine zwei konkurrierenden Standards halten können. Natürliche Auslese wird hier für eine Marktbereinigung sorgen. Durch das namhafte Konsortium, das hinter SAML steht, scheinen hier die Sterne ausnahmsweise mal nicht für das Microsoft Produkt, sondern für seinen Konkurrenten gut zu stehen.

Literaturverzeichnis

[FISCHER] Prof. Dr. Fischer – Folien zur Vorlesung “Enterprise Applications” – TU Braunschweig – Sommersemester 2003

[JSAML] Netegrity – JSAML Framework - http://www.netegrity.com/files/JSAMLwhitepaper.pdf

[NETEGRITY] Netegrity – Homepage - http://www.netegrity.com

[OASIS] OASIS – Homepage - http://www.oasis-open.org

[OASISSEC] OASIS – Security Gruppe - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[OPENNET] OpenNetwork – DirectorySmart - http://www.opennetwork.com/solutions/standards/saml/
[SUNONE] SUN – SUN ONE Identity Server - http://wwws.sun.com/software/products/identity_srvr/home_identity.html
[W3CWS] W3C – WebServices - http://www.w3.org/2002/ws
[W3CWSA] W3C - WebService Architecure - http://www.w3.org/TR/ws-arch/
[W3CSOAP] W3C – SOAP Version 1.2 Part 1 - http://www.w3.org/TR/soap12-part1/
[WSSOAP] J. Snell, D. Tidwell und P. Kulchenko - Webservice-Programmierung mit SOAP - O'Reilly - ISBN 3-89721-159-9, -1. Auflage Juli 2003
[XMLCOVER] Coverpages – XML Magazin - http://xml.coverpages.org/saml.html
[XMLMAG] Andy Patrizio- http://www.fawcette.com/xmlmag/2002_03/magazine/departments/marketscan/SAML/