

# MASTERARBEIT

## Konzeption und prototypische Implementierung eines Triple-A-Systems zum Schutz verteilter Geo-Webervices

durchgeführt am  
Studiengang Informationstechnik und System-Management  
an der  
Fachhochschule Salzburg

vorgelegt von:  
**DI (FH) Bernhard Schulz**



Studiengangsleiter:                    FH-Prof. DI Dr. Gerhard Jöchtl  
Betreuer:                                FH-Prof. DI Dr. Thomas Heistracher

Salzburg, August 2010

# **Eidesstattliche Erklärung**

Hiermit versichere ich, Bernhard Schulz, geboren am 07.09.1981, dass die vorliegende Masterarbeit von mir selbstständig verfasst wurde. Zur Erstellung wurden von mir keine anderen als die angegebenen Hilfsmittel verwendet.

---

Bernhard Schulz

---

0910581010  
Matrikelnummer

# Danksagung

Sage dir immer: ich kann wenn noch so einsam, an allen Orten glücklich sein; denn glücklich ist, wer sich selbst ein glückliches Los bereitet, dies ist: gute Gemütsstimmung, gute Neigungen, gute Handlungen.  
*Marcus Aurelius Antonius, röm. Kaiser u. Philosoph (121 - 180)*

Ich möchte mich hiermit herzlich bei Dr. Bernd Resch und Mag. Manfred Mittlböck vom Research Studio iSpace aus Salzburg<sup>1</sup> bedanken, die viele Stunden ihrer Zeit für Besprechungen, angeregte Diskussionen und Reviews geopfert haben.

Besonderer Dank gilt meinem Masterarbeitsbetreuer FH-Prof. DI Dr. Thomas Heistracher, der trotz seiner vielen Aufgaben und Verpflichtungen an der FH für Fragen und Besprechungen in großzügigem Maße zur Verfügung gestanden ist.

Ohne die technisch/wirtschaftliche Betreuung durch Dr. Resch und Mag. Mittlböck sowie die wissenschaftliche Betreuung durch FH-Prof. DI Dr. Heistracher wäre die Arbeit heute nicht in dem Zustand, in dem sie nun vorliegt.

Mein Dank gilt auch dem Studiengangsleiter FH-Prof. DI Dr. Jöchtl und seinem Team, ohne dessen besondere Anstrengungen es wahrscheinlich nie dieses Masterstudium gegeben hätte.

Schlussendlich möchte ich mich hiermit bei meiner Familie bedanken, welche mich in all meinen Bestrebungen stets unterstützt.

---

<sup>1</sup><http://ispace.researchstudio.at>

# Informationen

Vor- und Zuname:	DI (FH) Bernhard Schulz
Institution:	Fachhochschule Salzburg GmbH
Studiengang:	Informationstechnik & System-Management
Titel der Diplomarbeit:	Konzeption und prototypische Implementierung eines Triple-A-Systems zum Schutz verteilter Geo-Webservices
Betreuer an der FH:	FH-Prof. DI Dr. Thomas Heistracher
Betreuer in der Firma:	Dr. Bernd Resch/Mag. Manfred Mittelböck

## Schlagwörter

1. Schlagwort: Triple-A-System
2. Schlagwort: OGC-Webservice
3. Schlagwort: OAuth

## Abstract

This master thesis describes the conceptual design and prototypical implementation of a *Triple-A-System* to secure distributed geo-web services.

First, the three components of Triple-A-Systems are described: authorisation, authentication and accounting. This is followed by an introduction of common security concepts for secure message transmission. Furthermore, existing web-based authentication and authorisation technologies are presented with a special focus on *OAuth* and *OpenID*. The introduction concludes with a general description of the *Open Geospatial Consortium (OGC)*-web services like *Web Map Service (WMS)* or *Web Feature Service (WFS)* and existing security concepts for these services.

Second, the required components for a *service-oriented security architecture for geo-web services* are defined. This is done by describing the demands of geo-web service providers and customers and by developing a system based on existing infrastructure into an architecture which takes all demands into account. As a result, a simple and yet flexible and powerful architecture is presented.

Third, an *evaluation* of different authorisation protocols for an application in this derived architecture is presented as well as an evaluation of the different formats to store geo-access-rules. Therefore, all criteria are explained, weighted and evaluated.

In the practical part of this thesis a *prototypical implementation* is introduced, which was developed within the scope of this work. A distributed system was developed, which is capable of controlling the access to a WMS. This prototype is also able to filter parts of the resulting maps. Verification of the results showed that all aspects of the derived service-oriented security architecture for geo-web services and the prototypical implementation are correct and applicable.

A discussion of implementation details and an outlook on future development conclude this work.

# Inhaltsverzeichnis

<b>Eidesstattliche Erklärung</b>	ii
<b>Danksagung</b>	iii
<b>Informationen</b>	iv
<b>Schlagwörter</b>	iv
<b>Abstract</b>	iv
<b>Inhaltsverzeichnis</b>	v
<b>Abbildungsverzeichnis</b>	ix
<b>Tabellenverzeichnis</b>	xii
<b>1 Einleitung</b>	1
1.1 Anforderungen der Geodiensteanbieter und deren Kunden . . . . .	2
1.2 Aufbau der Arbeit . . . . .	3
<b>2 Einführung in Triple-A-Systeme</b>	5
2.1 Grundlagen der Authentifizierung . . . . .	5
2.1.1 Methoden der Authentifizierung . . . . .	7
2.1.2 Sicherung der Übertragung . . . . .	8
2.2 Die Autorisierung als Geschäftsentscheidung . . . . .	12
2.3 Abrechnung bei Benutzung von Diensten . . . . .	12
2.4 Zusammenfassung von Triple-A-Systemen . . . . .	13

<b>3 Sicherheitskonzepte in der Nachrichtenübermittlung</b>	<b>14</b>
3.1 Ende-zu-Ende Sicherung mittels Nachrichten-basierter Sicherheit . . . . .	14
3.2 Punkt-zu-Punkt Sicherung mittels Transport-basierter Sicherheit . . . . .	15
3.3 Sicherheitsgewinn mittels Token-basierter Sicherheit . . . . .	16
3.4 Scheinbare Sicherheit durch Verschleierung . . . . .	17
3.5 Zusammenfassung der Sicherheitskonzepte in der Nachrichtenübermittelung . . . . .	17
<b>4 Webbasierte Authentifizierungs- und Autorisierungstechnologien</b>	<b>19</b>
4.1 Authentifizierungsprotokolle . . . . .	19
4.1.1 Authentifizierungsmethoden im Hypertext Transfer Protokoll . . . . .	20
4.1.2 Kerberos . . . . .	21
4.1.3 OpenID . . . . .	21
4.2 Autorisierungsprotokolle für verteilte Dienste . . . . .	29
4.2.1 OAuth . . . . .	29
4.2.2 Security Assertion Markup Language . . . . .	35
4.2.3 Shibboleth . . . . .	36
4.2.4 WS-Federation . . . . .	38
4.3 Zusammenfassung webbasierter Authentifizierungs- und Autorisierungs- technologien . . . . .	40
<b>5 Verteilte Geoinformationssysteme</b>	<b>42</b>
5.1 Webbasierte Geoinformationsdienste . . . . .	42
5.1.1 Informationen als Grafik vom Web Map Services . . . . .	42
5.1.2 Geografische Vektordaten vom Web Feature Services . . . . .	44
5.1.3 Datenpflege per Transactional Web Feature Service . . . . .	45
5.1.4 Raum- und Zeitdimensionen durch den Web Coverage Service . . . . .	46
5.1.5 Online Datenverarbeitung mit dem Web Processing Service . . . . .	46
5.1.6 Sensordaten aus dem Sensor Observation Service und Sensor Alert Service . . . . .	49
5.2 Kartenausschnitt per Bounding Box . . . . .	52
5.3 Kartenaufbau aus einzelnen Ebenen . . . . .	54
5.4 Schützenswerte Aspekte von Geodaten . . . . .	55
5.5 Existierende Schutzmaßnahmen für Geoinformationsdienste . . . . .	58
5.6 Zusammenfassung verteilter Geoinformationssysteme . . . . .	60

<b>6 Sicherheitsarchitektur für Geoinformationsdienste</b>	<b>61</b>
6.1 Anforderungen der GIS-Anbieter und -Kunden . . . . .	61
6.2 Authentifizierung, Autorisierung und Abrechnung am Geodienst selbst	62
6.3 Einführung eines Proxydienstes . . . . .	63
6.4 Einführung einer zentralen Datenbank für Triple-A-Systeme . . . . .	65
6.5 Einführung eines Authentifizierungs-, Autorisierungs- und Abrechnungsdienstes . . . . .	66
6.6 Betrachtung komplexer Szenarien mit Web Processing Services . . . . .	68
6.7 Einführung von Token-basierter Sicherheit . . . . .	69
6.7.1 Bereitstellung des Tokens . . . . .	72
6.8 Zusammenfassung zur serviceorientierten Geoinformationsarchitektur .	73
<b>7 Evaluierung zum Schutz von Geoinfrastruktur</b>	<b>75</b>
7.1 Begründete Auswahl eines Autorisierungsprotokolls für verteilte Dienste	76
7.1.1 Vergleichstabelle der Technologien zum Schutz der GIS-Infrastruktur	80
7.2 Auswahl des Authentifizierungsprotokolls . . . . .	81
7.3 Format zum Speichern der Geoautorisierungsregeln . . . . .	81
7.3.1 Vergleich der Formate zum Speichern von geografischen Zugriffsregeln . . . . .	89
7.3.2 Gewichtete Vergleichstabelle der Formate zum Speichern von geografischen Zugriffsregeln . . . . .	92
7.4 Übertragung von Geoautorisierungsregeln . . . . .	93
7.5 Abrechnung der Zugriffe . . . . .	94
7.6 Zusammenfassung der Evaluierung bestehender Protokolle und Implementierungen zum Schutz von Geoinformationssystemen . . . . .	95
<b>8 Prototypische Implementierung zum Schutz von Geoinfrastruktur</b>	<b>97</b>
8.1 Implementierung von OAuth . . . . .	97
8.2 Softwarekomponenten des Endnutzers . . . . .	102
8.3 Softwarekomponenten des Konsumenten . . . . .	102
8.4 Softwarekomponenten des Service-Providers . . . . .	104
8.5 Anwendung der Servicearchitektur aus Benutzersicht . . . . .	104
8.6 Filtern der Anfragen . . . . .	110
8.7 Diskussion der Umsetzung . . . . .	113
8.8 Zusammenfassung der prototypischen Implementierung . . . . .	116

<b>9 Resümee, Weiterentwicklung und Ausblick</b>	<b>117</b>
9.1 Resümee . . . . .	117
9.2 Weiterentwicklung und Ausblick . . . . .	118
<b>Literaturverzeichnis</b>	<b>120</b>
<b>Abkürzungsverzeichnis</b>	<b>127</b>
<b>Anhang</b>	<b>129</b>
<b>A Aufgabenstellungen und Anforderungen durch Research Studio iSpace</b>	<b>130</b>
<b>B WFS-Anfrage mit Geometriedaten</b>	<b>133</b>
<b>C Bestätigungsemail bzgl. defekten Projektsetups der 52North Security Suite</b>	<b>141</b>

# Abbildungsverzeichnis

4.1	Login beim OpenID-Provider mit persönlichem Symbol zur Abwehr von Phishing-Attacken. . . . .	22
4.2	Autorisierung einer Authentifizierungsanfrage beim OpenID-Provider. .	23
4.3	Einstellungen zur starken Authentifizierung per Telefon beim Provider myOpenId. . . . .	25
4.4	Erstellen eines SSL-Client-Zertifikates für starke Authentifizierung. . .	25
4.5	OpenID Identität mit korrekten persönlichen Angaben. . . . .	26
4.6	Phantasie OpenID Identität für die anonyme Nutzung von Diensten. .	26
4.7	Authentifizierung über Benutzername und Passwort oder OpenID bei Slashdot.org. . . . .	27
4.8	Inhaber eines Benutzerkontos bei großen Anbietern wie Google oder Yahoo haben automatisch eine OpenID, oft ohne dies zu wissen. . . . .	28
4.9	Verknüpfung eines Facebook-Kontos mit einer OpenID für eine zentrale Authentifizierung. . . . .	28
4.10	Klassische (unsichere) Authentifizierung eines Twitter Clients beim Twiterdienst per Benutzername und Passwort. . . . .	30
4.11	Der Facebook-„Freundefinder“, welcher nach Angabe von Benutzername und Passwort die Adressbücher von Webmailbetreibern auslesen kann, um Kontakte auf Facebook zu finden. Facebook verspricht, das Passwort nicht zu speichern, der Benutzer kann dies jedoch nicht kontrollieren. .	31
4.12	Verwalten der Anwendungen, die auf den eigenen Twitter Account zugreifen dürfen, inklusive der Möglichkeit, dies jederzeit zu widerrufen. .	32
4.13	Nutzung von Twitter per OAuth. . . . .	33
4.14	Gleichzeitige Authentifizierung und Autorisierung in einem Anwendungsdialog. . . . .	35
4.15	Erster Zugriff auf eine per Shibboleth geschützte Ressource. . . . .	37
4.16	Auswahl der Heimatuniversität für die Shibboleth-Authentifizierung. .	37
4.17	Shibboleth-Authentifizierung bei der Heimatuniversität. . . . .	37
4.18	Zugriff auf die per Shibboleth geschützte Ressource. . . . .	37

4.19	Der gesamte Shibboleth-Authentifizierungsvorgang.	39
5.1	Einfacher Ablauf bei direkter Nutzung eines webbasierten Geodienstes: HTTP Anfrage an eine WMS-Schnittstelle und Antwort im Bildformat.	43
5.2	Gerasterte Grafik als Ergebnis einer einfachen WMS-Anfrage.	44
5.3	Teil des galizischen Straßennetzes (Spanien).	47
5.4	WMS-Darstellung zweier Kartenebenen mit Straßennetz und Waldbrand.	48
5.5	Ergebnis der WPS-Anfrage zur Berechnung der brennenden Straßenabschnitte	48
5.6	Ein WPS nutzt andere Geoinformationsdienste als Basis für seine Berechnungen.	48
5.7	„Sun SPOT“ als Beispiel eines Embedded Devices.	49
5.8	Abfrage der grünen, das Bundesland Salzburg umschließenden Bounding Box innerhalb des erlaubten, roten Bereiches.	53
5.9	Gleiche Abfrage des Bundeslandes Salzburg wie in Abbildung 5.8, jedoch innerhalb einer komplexeren, den Zugriff steuernden Bounding Box.	53
5.10	Ergebnis einer WMS-Anfrage, wobei die Bounding Box der Anfrage die Bounding Box der erlaubten Regionen überlappt.	54
5.11	Zusammensetzung von Karten aus verschiedenen Ebenen.	55
6.1	Anfrage an eine WMS-Schnittstelle mit Authentifizierung, Autorisierung und Abrechnung.	62
6.2	Anfrage an eine WMS-Schnittstelle über einen Proxydienst für Authentifizierung, Autorisierung und Abrechnung.	64
6.3	Anfrage an zwei WMS-Schnittstellen und doppelten Benutzerdatenbanken.	65
6.4	Zentrale Datenbank für Authentifizierung, Autorisierung und Abrechnung.	66
6.5	Serviceorientierte Systemarchitektur mit zentralem Authentifizierungs-, Autorisierungs- und Abrechnungsdienst für Webservices.	67
6.6	WPS-Anfrage mit einer zentralen Autorisierungsprüfung beim AAAD.	68
6.7	WPS-Anfrage geschützter Geoinformationsdienste und deren jeweilige Autorisierungsanfragen beim AAAD.	69
6.8	Unsichere Authentifizierung mittels Benutzername und Passwort.	70
6.9	Sichere Authentifizierung mittels Token.	71
6.10	Anfordern eines Tokens beim AAAD bei lokaler Authentifizierung.	72
6.11	Anfordern eines Tokens beim AAAD bei externer Authentifizierung.	73

7.1	Ergebnis einer ungefilterten WFS-Anfrage zum Auslesen der amerikanischen Bundesstaaten . . . . .	85
7.2	Ergebnis einer mit dem Attribut MALE >FEMALE gefilterten WFS-Anfrage, um darzustellen, wo der Männeranteil der Bevölkerung größer ist als der Frauenanteil. . . . .	86
7.3	Ergebnis einer WFS-Anfrage, gefiltert nach Bundesstaaten mit einer Arbeitslosenquote unter 7 %. . . . .	86
8.1	Darstellung des Ablaufes beim erstmaligen Zugriff auf eine geschützte WMS-Ressource. . . . .	98
8.2	Darstellung des Ablaufes bei weiteren Zugriffen auf eine geschützte WMS-Ressource mit vorhandener SessionID und gültigem Zugriffstoken. . . . .	101
8.3	Screenshot der Willkommensseite des WMS-Clients mit kurzen Erklärungen. . . . .	103
8.4	Abbildung 1 von 3 mit Screenshots beim Benutzerzugriff auf einen geschützten WMS-Dienst. . . . .	106
8.5	Abbildung 2 von 3 mit Screenshots beim Benutzerzugriff auf einen geschützten WMS-Dienst. . . . .	107
8.6	Abbildung 3 von 3 mit Screenshots beim Benutzerzugriff auf einen geschützten WMS-Dienst. . . . .	108
8.7	Screenshot der Zugriffsverweigerung wegen ungültigem Zugriffstoken. . . . .	109
8.8	Ergebnis der ungefilterten WMS-Anfrage. . . . .	112
8.9	Definition zweier Bounding Box-Filter. Der Benutzer erhält Zugriff auf die beiden grün markierten Bereiche. Der Zugriff auf den roten Bereich wird unterbunden. . . . .	112
8.10	Ergebnis der gefilterten WMS-Anfrage. . . . .	113
8.11	Vergleich der abgeleiteten Sicherheitsarchitektur mit GeoXACML. . . . .	115

# Tabellenverzeichnis

7.1	Gewichtete Bewertung der Autorisierungstechnologien zum Schutz der verteilten Geoinfrastruktur. . . . .	80
7.2	Gewichtete Bewertung der Formate zum Speichern von geografischen Zugriffsregeln. . . . .	92

# 1

---

## Einleitung

Diensteanbieter im Geoinformationsbereich stellen Daten statischer oder zeitdynamischer Natur mit geografischem Bezug zur Verfügung [47]. Diese Daten, gewonnen aus Satellitenbildern, durch Landschaftsvermessung und vielen verschiedenen Sensoren werden über einheitliche Webservice-Schnittstellen an Partner und Kunden weitergegeben [47]. Die Rohdaten selbst werden dabei in der Praxis in vielen unterschiedlichen Formaten erfasst und müssen vor der Weiterverarbeitung erst konvertiert und aufbereitet werden, was einen bedeutenden Kostenfaktor darstellt. Daher werden diese gewonnenen und bereinigten Daten üblicherweise nur gegen entsprechende Bezahlung bzw. Beteiligung an den Erhebungs- und Wartungskosten bereitgestellt [9]. Die Kosten richten sich dabei unter anderem nach

- Aktualität / Korrektheit
- Vertrauenswürdigkeit
- und Vollständigkeit

der Daten [81].

Der Schutz von Geodaten setzt heute noch den Einsatz erheblicher Ressourcen voraus (siehe Kapitel 5.5).

## 1.1 Anforderungen der Geodiensteanbieter und deren Kunden

Geodiensteanbieter, wie das Salzburger Research Studio iSpace, und deren Kunden wünschen sich eine einfache Autorisierungssteuerung. So sollte ein Kunde, wenn er Zugriff auf einen Server mit Österreichs gesamter Straßenkarte hat, beispielsweise nur das Straßennetz des Bundeslandes Salzburg abrufen können, sofern dies zuvor vertraglich so vereinbart wurde. Der Kunde hat dabei den Vorteil, nur für den Kartenausschnitt zu bezahlen, den er benötigt und abrufen kann. Der Kartenanbieter wiederum kann sich darauf verlassen, dass die restlichen Daten geschützt und für den Kunden nicht abrufbar sind.

Werden am Kartenserver auch potentiell sicherheitskritische Informationen hinterlegt, wie etwa die Positionen von Atomkraftwerken und Militärbasen oder der Verlauf von Gas- und Stromleitungen, so sind diese Informationen auch hinsichtlich des terroristischen Gefahrenpotentials zu sichern und nur für autorisierte Benutzer zugänglich zu machen [30].

Im Rahmen der Geoautorisierung sollte es möglich sein, Kunden nur bestimmte Informationen einer Region zukommen zu lassen, so der Wunsch der Geodiensteanbieter wie iSpace. Mitarbeiter der Landesregierung und Feuerwehr könnten zum Beispiel visualisierte Evakuierungspläne und Katastrophenschutzmaßnahmen vom Kartenserver abrufen, Mitarbeiter der Wetterwarte jedoch vom selben Server nur Standortinformationen der einzelnen Wetter-Sensoren.

Im Bereich der Authentifizierung gibt es ebenfalls Wünsche, so geäußert von den Mitarbeitern von iSpace (siehe Anhang A), um die Nutzung von Geoinformationsdiensten zu vereinfachen. Üblicherweise werden Geoinformationsdienste nicht nur von einzelnen Personen verwendet, sondern ganze Organisationen wie Landesregierungen, Feuerwehren, Energieversorger oder andere Infrastrukturbetreiber nutzen diese Dienste, beispielsweise für die Planung neuer Stromleitungen, Eisenbahn- und Straßenabschnitte sowie im Katastrophenfall durch Hochwasser, Brände, Chemieunfälle oder Pandemien zum Planen von Abwehrmaßnahmen. Geodiensteanbieter können hierbei nicht von

jedem Unternehmen die Benutzerdatenbanken mit Benutzername und Passwort kopieren, um den Zugriff auf die Geoinformationsdienste zu steuern. Der Pflegeaufwand zum Abgleich der Benutzerdatenbank mit den jeweiligen Datenbanken der Anbieter wäre viel zu aufwändig und ist oftmals technisch gar nicht realisierbar.

Ein weiterer Wunsch von Kunden und Geodiensteanbietern betrifft die Abrechnung der Zugriffe. Der Kunde möchte nur für die Daten bezahlen, die er tatsächlich abgerufen hat und der Geodiensteanbieter möchte jeden Zugriff zuverlässig protokollieren und abrechnen können, um seine Kosten zu decken [62].

## 1.2 Aufbau der Arbeit

Diese Arbeit, welche in Zusammenarbeit mit dem Research Studio iSpace aus Salzburg geschaffen und aus Mitteln des Bundesministeriums für Wissenschaft und Forschung gefördert wurde, beschäftigt sich mit den theoretischen Grundlagen von Authentifizierung, Autorisierung und Abrechnung im Geoinformationsbereich, wobei der Fokus auf Authentifizierung und Autorisierung liegt. Aus Gründen der Lesbarkeit wurde in dieser Masterarbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Der Verfasser weist jedoch ausdrücklich darauf hin, dass die bei Personen maskuline Form für beide Geschlechter zu verstehen ist.

Zur Einführung werden die zentralen Begriffe Authentifizierung, Autorisierung und Abrechnung in Kapitel 2 erläutert. Kapitel 3 führt in die verschiedenen Sicherheitskonzepte der Nachrichtenübermittlung ein, denn Authentifizierung und Autorisierung können nur zuverlässig funktionieren, wenn die Informationen sicher übertragen werden. Entsprechende Technologien und Implementierungen werden in Kapitel 4 vorgestellt. Kapitel 5 führt in die webbasierten Geoinformationsdienste ein, um dem Leser die Komplexität dieser Dienste erahnen zu lassen. Nachdem die grundlegenden Konzepte und Dienste erklärt wurden und im Kapitel 5.5 existierende Lösungen und ihre Einstellungen vorgestellt werden, finden in Kapitel 6 theoretische Überlegungen zu einer Sicherheitsarchitektur für Geoinfrastruktur statt. Ausgehend von diesen Überlegungen werden in Kapitel 7 verschiedene Möglichkeiten genannt, die Architektur durch konkrete Protokolle und Frameworks zu schützen. Die einzelnen Technologien werden

dabei miteinander verglichen, um schlussendlich eine Wahl zu treffen, welche dieser Technologien im Rahmen einer Implementierung verwendet werden sollten. Auf Basis der theoretischen Überlegungen aus Kapitel 6 sowie der Technologiewahl aus Kapitel 7 wird im Kapitel 8 das Konzept prototypisch implementiert und die Implementierung verifiziert. Im letzten Kapitel 9 wird ein Ausblick für zukünftige mögliche Erweiterungen gegeben und ein Resümee gezogen.

# 2

---

## Einführung in Triple-A-Systeme

Triple-A-Systeme<sup>1</sup> sind Softwareframeworks, die Authentifizierung, Autorisierung und Abrechnung abdecken [74]. Diese drei zentralen Aspekte werden im Folgenden erklärt.

### 2.1 Grundlagen der Authentifizierung

Bei der Authentifizierung handelt es sich um die Überprüfung der Identität<sup>2</sup> des Anwenders. Diese Überprüfung kann auf mehrere verschiedene Arten geschehen, welche unterschiedlich sicher sind. Man unterscheidet daher starke und schwache Authentifizierung [40].

Unter *schwacher Authentifizierung* versteht man die Authentifizierung durch Eigenschaften, welche leicht weitergegeben, gefälscht oder gestohlen werden können [40]. Eine Authentifizierung des Benutzers durch Benutzername und Passwort kann problemlos von einem Angreifer abgefangen und missbraucht werden, indem dieser den Login-Vorgang und die Tastenanschläge auf der Tastatur beobachtet und notiert. Der Angreifer kann mit diesem Wissen Dienste im Namen des ausgespähten Benutzers nutzen und dadurch Schaden anrichten [39].

---

<sup>1</sup>Auch AAA-Systeme genannt.

<sup>2</sup>Als Identität bezeichnet man die Summe von Eigenschaften, die eine Instanz eindeutig identifizieren [13]

Der Benutzername und das Passwort kann in vielen Fällen auch per Brute-Force<sup>3</sup> Methode oder Social-Engineering<sup>4</sup> erschlichen werden [48].

Unter *starker Authentifizierung* versteht man eine Authentifizierung, welche nicht leicht erraten oder manipuliert werden kann [40]. Die Authentifizierung über biometrische Merkmale ist ein Beispiel hierfür. Müssen sich Benutzer an einem System per Fingerabdruckerfassung oder Irisscan authentifizieren, sind die Möglichkeiten für einen Angreifer deutlich geringer, da der Aufwand an diese Daten zu gelangen und dem Scanner so vorzulegen, dass sich dieser täuschen lässt, erheblich steigt [40].

Beim Design von Sicherheitssystemen ist jedoch zu beachten, dass praktisch alle Authentifizierungssysteme überlistet werden können. So gibt es Techniken, Fingerabdrücke oder Abbilder der Iris zu fälschen und so zu präparieren, dass die Scanner den Unterschied zur originalen, lebenden Iris bzw. dem Fingerabdruck nicht erkennen [40].

In der Praxis werden daher, je nach Sensibilität des Systems, mehrere Methoden kombiniert, um die Sicherheit weiter zu erhöhen [18].

Beim Online Banking ist es heute üblich, neben dem Login auf der Hauptseite jede Finanztransaktion mit einem Transaktionsnummer (TAN)-Code zu bestätigen [36]. Dieser TAN-Code wurde dem Bankkunden dabei entweder in einem Umschlag per Post zugesandt oder wird live per SMS-Nachricht übermittelt. Manche Bankinstitute bieten ihren Kunden Geräte zum Erzeugen von Tokens an, welche kurze Zeit gültig sind und zum Bestätigen einer Transaktion benötigt werden<sup>5</sup> [23]. Doch kann auch dieses Gerät zur Tokenerzeugung gestohlen oder die Anzeige bei unsachgemäßer Handhabung mit einem Fernglas abgelesen werden.

Letztendlich geht es daher darum, den Aufwand für potentielle Angreifer so hoch zu setzen, dass ein Angriff nicht mehr vertretbar erscheint [40].

---

<sup>3</sup>Ausprobieren aller Buchstaben/Zahlen/Sonderzeichen-Kombinationen [48].

<sup>4</sup>Erschleichen von Informationen durch das Aufbauen einer Vertrauensbeziehung zum Opfer, welches dann Geheimnisse preisgibt [48].

<sup>5</sup>Die Firma RSA Security bietet beispielsweise das Produkt SecurID an, welches jede Minute einen neuen Token erzeugt, der nicht erraten werden kann. Somit hat das Ablesen des Tokens und die Eingabe in das System innerhalb von maximal 60 Sekunden zu erfolgen. Phishing-Angriffe sind somit in der Praxis kaum möglich, da diese innerhalb der Lebensdauer des Tokens erfolgen müssten.

### 2.1.1 Methoden der Authentifizierung

#### Wissensbasierte Authentifizierung

Die wissensbasierte Authentifizierung stellt heute die am weitesten verbreitete, aber auch unsicherste Form der Authentifizierung dar [40].

*Der Identitätsnachweis durch Wissen basiert prinzipiell auf der Abmachung eines gemeinsamen Geheimnisses zwischen einem Subjekt und der authentifizierenden Instanz. Die Identität eines Subjekts ist somit an dieses Geheimnis geknüpft [66, Seite 7].*

Das bedeutet, ein Benutzer meldet sich mit seinem Benutzernamen und seinem geheimen Passwort an einem System an, woraufhin dieses die Korrektheit der Logindaten mit der internen Datenbank abgleicht und den Zugriff gewährt oder diesen ablehnt.

#### Besitzbasierende Authentifizierung

Bei der besitzbasierten Authentifizierung besitzt ein Subjekt etwas, was seine Identität beim Authentifizierungssystem nachweist [71]. Dies kann etwa eine Chipkarte oder ein Token-Erzeuger, wie oben beschrieben, sein.

*Analog zu wissensbasierten Authentifizierungsverfahren sind hierbei die zum Identitätsnachweis herangezogenen Nutzerattribute nicht an die Person gebunden und können gewollt oder ungewollt an andere Personen weitergegeben werden [66, Seite 9].*

#### Biometrische Authentifizierung

Biometrische Authentifizierung basiert auf der Messung spezifischer biologischer Merkmale zur Bestimmung der Identität einer Person [34] [71].

Diese Merkmale können dabei physiologische Eigenschaften wie der Fingerabdruck, die Iris, die Stimme oder Körperproportionen, aber auch verhaltensbasierte Eigenschaften wie Sprache, darstellen [42].

## Public-Key-Authentifizierung

Bei der Public-Key-Authentifizierung wird zu Authentifizierung ein asymmetrisches Schlüsselpaar, bestehend aus je einem privaten und öffentlichen Schlüssel, verwendet. Damit der Benutzer Zugriff auf einen Server erhält, muss zuvor der öffentliche Schlüssel des Benutzers am Server hinterlegt worden sein, beispielsweise durch den lokalen Administrator. Der private Schlüssel des Benutzers verbleibt am Rechner des Benutzers und darf auf keinen Fall veröffentlicht werden. Will sich der Benutzer nun am Server anmelden, muss er dem Server beweisen, dass er in Besitz des privaten Schlüssels ist (siehe 2.1.1, Abschnitt „Besitzbasierende Authentifizierung“). Dazu stellt der Server dem Client mit Hilfe des hinterlegten öffentlichen Schlüssels eine mathematisch aufwändige Aufgabe, die er nur lösen kann, wenn er den passenden privaten Schlüssel hat (siehe 2.1.2, Abschnitt „Challenge-Response-Verfahren“) [8].

### 2.1.2 Sicherung der Übertragung

Während der Authentifizierung werden Daten zwischen dem zu authentifizierendem Objekt und dem Authentifizierungssystem übertragen. Wenn diese Daten abgehört werden, so könnten Angreifer die übertragenen Informationen nutzen, um eine falsche Identität beim Authentifizierungssystem zu erlangen [49].

Des weiteren ist es von großer Wichtigkeit, dass die Daten während der Übertragung nicht verändert werden [49]. Es wäre beispielsweise fatal, wenn ein bereits authentifizierter und autorisierter Benutzer einen einzelnen Datensatz einer Datenbank ändern möchte und ein Angreifer den über das Netzwerk gesendeten Befehl so abändert, dass alle Datensätze gelöscht werden.

Es ist daher wichtig die Datenübertragung abzusichern. Dafür gibt es eine Reihe von Konzepten, welche im Anschluss vorgestellt werden.

Um die Angriffsszenarien anschaulicher zu beschreiben, verwendet man in der Kryptographie und bei der Beschreibung von Netzwerkprotokollen bestimmte Namen für einzelne Kommunikationspartner. Gemäß dieser Konvention werden in den folgenden

Beispielen „Alice“ und „Bob“ miteinander kommunizieren. „Mallory“ ist die Angreiferin, welche Nachrichten fälscht oder austauscht, um eine falsche Identität zu erlangen [49].<sup>6</sup>

### **Einmalpasswörter**

Authentifiziert sich Bob gegenüber Alice immer mit dem gleichen Passwort, so ist Mallory in der Lage, das Passwort abzuhören und sich selbst gegenüber Alice als Bob zu authentifizieren, indem sie das zuvor abgehörte Passwort zum Einloggen verwendet. Um diese Sicherheitslücke zu schließen, können Einmalpasswörter verwendet werden, welche, wie der Name schon impliziert, nur für einen einzigen Loginversuch verwendet werden können [71]. Dabei teilen sich Alice und Bob eine Liste von Passwörtern, welche zuvor über einen sicheren Kanal übertragen wurde. Beide Parteien müssen diese Liste außerdem gut geschützt abspeichern, sodass die Liste nicht gestohlen oder ausspioniert werden kann [14]. Wann immer sich Bob bei Alice authentifizieren möchte, verwendet er eines der Einmalpasswörter, welches nach dem Loginversuch von der Liste der Passwörter gestrichen wird. Versucht Bob oder Mallory später sich mit einem bereits verwendeten Einmalpasswort anzumelden, so erkennt Alice, dass das Passwort bereits verwendet wurde und verweigert die Authentifizierung.

### **Challenge-Response-Verfahren**

Beim sogenannten Challenge-Response Verfahren prüfen die Kommunikationspartner Alice und Bob die Identität des jeweilig anderen, indem sie ihm eine Aufgabe („Challenge“) zusenden, die er nur lösen kann, wenn es sich tatsächlich um den erwarteten Kommunikationspartner handelt, denn nur er kennt das gemeinsame Geheimnis, um die Aufgabe korrekt lösen zu können. Der Kommunikationspartner sendet das Ergebnis („Response“) zurück zum Aufgabensteller, der die Lösung des Partners mit seiner eigenen Lösung vergleicht, um so festzustellen, ob der Partner der ist, für den er sich auszugeben scheint [71].

---

<sup>6</sup>Bei der kryptographischen Beschreibung des Datenaustausches kommen in der Literatur noch weitere Rollen und Namen ins Spiel, welche aber im Rahmen dieser Arbeit nicht benötigt und daher nicht erwähnt werden [49].

Im Idealfall wird eine bestimmte Challenge nur einmal von Alice an Bob gesendet, denn Mallory, die die Kommunikation abhört, könnte die Challenge und die Response aufzeichnen und sich später als Bob ausgeben, indem sie die schon bekannten Challenges von Alice mit den zuvor aufgezeichneten Responses von Bob beantwortet [45].

Ein Beispiel für das Challenge-Response Verfahren mit Einmalpasswörtern ist das bereits oben vorgestellte TAN-Verfahren beim Online Banking. Alice, der Server der Bank, stellt an Bob, den Benutzer des Online Bankings, zum Beispiel die Aufgabe, den TAN-Code Nummer 5 anzugeben, um die Finanztransaktion zu autorisieren. Bob, der mit Alice das Geheimnis der TAN-Codes teilt, gibt den TAN-Code an und Alice führt die Transaktion durch, wenn der TAN-Code von Bob mit dem hinterlegten Code übereinstimmt. Der verwendete TAN-Code Nummer 5 wird dann von der Liste gestrichen und Alice wird Bob nie wieder nach TAN-Code Nummer 5 fragen. Auch wenn Bob den Code Nummer 5 falsch angegeben hat und Alice die Finanztransaktion nicht durchführt, wird der Code Nummer 5 gestrichen und nie wieder verwendet, um maximale Sicherheit zu gewährleisten, denn ansonsten könnte Mallory mit einer Brute Force Attacke alle möglichen Kombinationen für den TAN-Code Nummer 5 an Alice senden, bis sie den richtigen Code erraten hat [36].

Bei der Authentifizierung in GSM/UMTS-Mobilfunknetzen oder WLAN-Netzen schickt Alice an Bob in der Regel eine Zufallszahl als Challenge. Bob nutzt eine zuvor definierte Funktion

$$\text{Response} = f(\text{Challenge})$$

die nur ihm und Alice bekannt ist. Bob berechnet die Response, sendet das Ergebnis an Alice, und Alice vergleicht das selbst berechnete Ergebnis mit dem Response von Bob, um so die Identität von Bob zu prüfen [31] [18].

### **Zero Knowledge Verfahren**

Das Zero Knowledge Verfahren stellt eine spezielle Variante des Challenge-Response Verfahrens dar [31]. Hierbei überzeugt Bob Alice davon, ein Geheimnis zu kennen, ohne Details darüber zu verraten.

Das historisch erste bekannte Beispiel für Zero Knowledge Verfahren geht auf das Jahr

1535 zurück, in dem der italienische Mathematiker Nicolo Tartaglia eine Lösungsformel für die Gleichung

$$x^3 + ax^2 + bx + c = 0$$

fand, diese aber aus politischen Gründen nicht veröffentlichte. Er bewies die Kenntnis der Lösungsformel, indem er 30 ihm gestellte Gleichungen korrekt lösen konnte. Seine Konkurrenten wussten nun, dass er tatsächlich eine korrekte Lösungsformel gefunden hatte, ohne selbst das Geringste über die Formel zu erfahren [12]. Diese historische Gegebenheit stellt das Verfahren anschaulich dar. In der Kryptographie verwendet man heute das Fiat-Shamir Protokoll. Die Details dieses Protokolls würden den Rahmen dieser Arbeit sprengen. Interessierte Leser finden eine genau Beschreibung bei [12].

## Digitale Signaturen

Durch den Einsatz digitaler Signaturen, die aufgrund des Nachrichteninhaltes berechnet werden, kann der Empfänger prüfen, ob die Nachricht während der Übertragung manipuliert wurde. Dazu werden die Nachricht sowie - je nach Verfahren - ein symmetrischer oder asymmetrischer Schlüssel als Parameter für eine Hashfunktion verwendet, deren Ergebnis die Signatur repräsentiert. Diese Signatur wird gemeinsam mit der Nachricht an den Empfänger geschickt. Der Empfänger, der den symmetrischen Schlüssel ebenfalls kennt bzw. das Gegenstück zum asymmetrischen Schlüssel vorliegen hat, berechnet erneut den Hashwert und vergleicht diesen mit dem Hashwert des Absenders. Stimmen beide Werte überein, wurde die Nachricht nicht verändert [11].

## Verschlüsselung

Um den Inhalt der Nachricht zu schützen, kann die Nachricht selbst verschlüsselt werden. Die Sicherheit ist gegeben, wenn nur der Empfänger den symmetrischen oder asymmetrischen Schlüssel zu Dekodierung kennt und ein aktuell gültiger Verschlüsselungsalgorithmus verwendet wird, bei dem keine Sicherheitslücken oder Angriffsszenarien bekannt sind [19].

Nachrichten können verschlüsselt und signiert sein, sodass der Inhalt für Angreifer geheim und zusätzlich gewährleistet ist, dass sich der Inhalt der Nachricht während der Übertragung (durch Übertragungsfehler oder Manipulation) nicht verändert hat [40].

## 2.2 Die Autorisierung als Geschäftsentscheidung

Die Autorisierung des Benutzers findet nach seiner Authentifizierung statt. Sie entscheidet über die Berechtigungen des Benutzers innerhalb des Systems.

Üblicherweise wird Autorisierung als eine Sicherheitsfunktion betrachtet, tatsächlich handelt es sich aber bei der Entscheidung, welcher Benutzer Zugriff auf welche Funktionen erhält, um eine Geschäftsentscheidung. Die Regeln für die Implementierung der Autorisierung sind daher Geschäftsregeln [61].

In der Praxis ist es zwar oft so, dass ein Benutzer mit Administratorberechtigung mehr Möglichkeiten innerhalb einer Software oder eines Dienstes hat, als ein Gastbenutzer, aber trotzdem obliegt diese Entscheidung der Geschäftslogik der Anwendung.

Geschäftsentscheidungen von Software, die den Zugang zu geografischen Daten regelt, basieren unter anderem auf den in Kapitel 5.4 vorgestellten Aspekten.

## 2.3 Abrechnung bei Benutzung von Diensten

Das dritte „A“ im Rahmen von Triple-A-Systemen steht für die Abrechnung. Dabei geht es darum, den Kunden die in Anspruch genommenen Ressourcen in Rechnung zu stellen [16].

Ein Beispiel: Der Endbenutzer authentifiziert sich per Benutzername und Passwort. Nachdem er als Kunde einen Vertrag mit dem Geodaten-Provider abgeschlossen hat und von diesem autorisiert wurde, kann er auf die Geodaten zugreifen. Jeder Zugriff kostet 5 Cent. Macht er im Monat 1.000 Zugriffe, so hat er am Ende des Monats € 50 zu bezahlen.

In der Praxis spielen für die Preisberechnung viele Faktoren eine Rolle, wie Aktualität der Daten, Genauigkeit, Exklusivität oder Dauer der Vertragsbindung (siehe auch 5.4).

## 2.4 Zusammenfassung von Triple-A-Systemen

Triple-A-Systeme decken die Aspekte der *Authentifizierung* von Benutzern, die *Autorisierung* der Benutzer sowie die *Abrechnung* der Zugriffe ab.

Während die Verfahren zur Authentifizierung von Benutzern relativ umfassend und universell sind und daher allgemeingültig und detailliert beschrieben werden können, sind die Aspekte der Autorisierung und Abrechnung als Geschäftsentscheidung von System zu System abhängig. Es lassen sich in der Praxis zwar wiederkehrende Muster in Anwendungen finden, sodass ein Administrator mehr Rechte in einer Anwendung hat als ein Gastbenutzer, jedoch gibt es hierfür keine verbindlichen Regeln oder Vorschriften. Dementsprechend nimmt das Kapitel Authentifizierung (Kapitel 2.1) den meisten Platz in dieser allgemeinen Einführung zu Triple-A-Systemen ein.

Die konkreten Aspekte der Autorisierung in Geoinformationssystemen (GIS) sind erst in Kapitel 5.4 beschrieben und auf die zu berücksichtigenden Faktoren bei der Abrechnung in GIS geht Kapitel 7.5 ein.

# 3

---

## Sicherheitskonzepte in der Nachrichtenübermittlung

Damit Systeme untereinander, sowie Benutzer und Systeme, Daten austauschen können, sollten diese geschützt sein. Dieser Schutz kann sich gegen das Ausspionieren der Daten während der Übertragung, das erneute Senden der Nachrichten sowie der Manipulation des Inhalts der Nachrichten richten. Hierbei gibt es unterschiedliche Konzepte, welche im Folgenden vorgestellt werden.

### 3.1 Ende-zu-Ende Sicherung mittels Nachrichtenbasierter Sicherheit

Bei der Nachrichten-basierten Sicherheit<sup>1</sup> werden die einzelnen Nachrichten(pakete) selbst geschützt, indem die Nachricht verschlüsselt und die Authentizität der Nachricht mittels digitaler Signaturen geprüft wird. Durch spezielle Schutzmechanismen werden Replay-Attacken verhindert [38]. Die einzelnen Mechanismen, um dies zu erreichen, wurden in Kapitel 2.1.2 vorgestellt. Die Nachrichten-basierte Sicherheit stellt eine sogenannten Ende-zu-Ende Sicherung dar. Das bedeutet, dass die Nachricht von niemand anderem als dem Empfänger gelesen werden kann, unabhängig davon, über welche Nachrichtenkanäle die Nachricht übertragen wurde [38].

---

<sup>1</sup>Englisch: Message based Security

## 3.2 Punkt-zu-Punkt Sicherung mittels Transport-basierter Sicherheit

Im Gegensatz zur Nachrichten-basierten Sicherheit, wo jede einzelne Nachricht geschützt wird und über einen ungesicherten Kanal gesendet werden kann, wird bei der Transport-basierten Sicherheit<sup>2</sup> der gesamte Übertragungskanal gesichert. Die Sicherung des Kanals und der darüber fließenden Nutzdaten findet dabei zwischen den zwei Endpunkten des Kanals statt [38]. Eine Ende-zu-Ende Sicherung, wie dies bei der Nachrichten-basierten Sicherheit der Fall ist, ist nicht gegeben [53].

Ein Beispiel soll dies verdeutlichen: Sendet ein Benutzer eine Email an seinen Mailserver und nutzt dabei einen sicheren Übertragungskanal, kann ein lokaler Angreifer den Inhalt der Email nicht mitlesen, da diese eben über einen gesicherten Kanal gesendet wurde. Ruft der Empfänger der Email diese aber über einen ungesicherten Kanal ab, so kann ein Angreifer im Netzwerk des Empfängers den Inhalt der Email kopieren. Für das Versenden vertraulicher Emails ist es daher ratsam, Nachrichten-basierte Sicherheit und Transport-basierte Sicherheit zu kombinieren. Dabei wird die Email verschlüsselt und in dieser Form an den Empfänger übertragen, welcher die Nachricht wieder entschlüsseln kann. Durch diese Nachrichten-basierte Sicherheit ist die Ende-zu-Ende Sicherung gegeben. Zum Übertragen der Email zum Mailserver sowie zum Abrufen der Email vom Mailserver ist es ratsam Transport-basierte Sicherheit zu nutzen, um den Login-Vorgang am Mailserver vor Angreifer zu schützen, da die dafür häufig verwendeten Protokolle Post Office Protocol Version 3 (POP3) und Simple Mail Transfer Protocol (SMTP) die Login-Informationen im Klartext übertragen [15].

*Transport Layer Security (TLS)* dient, genau so wie das Vorgängerprotokoll *Secure Sockets Layer (SSL)*<sup>3</sup>, als Verschlüsselungsprotokoll auf Transportebene zur Sicherung des Datenkanals. Durch den Einsatz von Zertifikaten können die Kommunikationspartner sicher authentifiziert werden [65] [40]. Die Protokolle der Anwendungsschicht im Open Systems Interconnection (OSI)-Modell können TLS zur Absicherung nutzen.

---

<sup>2</sup>Englisch: Transport based Security

<sup>3</sup>TLS 1.0 basiert auf SSL 3.1

Der große Vorteil von TLS liegt darin, dass die Protokolle der Anwendungsschicht nicht extra angepasst werden müssen, denn TLS stellt eine Erweiterung auf Ebene der Transportschicht des OSI-Modells dar.

TLS ist im Internet weit verbreitet und wird zum Sichern der Datenübertragung über Protokolle wie HTTP, POP3, Internet Message Access Protocol (IMAP), SMTP und vielen weiteren genutzt, wobei dann an den Protokollnamen meist ein „S“ angehängt wird. Somit wird aus den per TLS gesicherten Varianten dann Hypertext Transfer Protocol Secure (HTTPS), Post Office Protocol Version 3 Secure (POP3S), Internet Message Access Protocol Secure (IMAPS), Simple Mail Transfer Protocol Secure (SMTPS) und so weiter [15].

### **3.3 Sicherheitsgewinn mittels Token-basierter Sicherheit**

Das Token-basierte Sicherheitskonzept ist eine weitere wichtige Methode zur Umsetzung von Sicherheitsstrategien. Das Konzept hinter Token-basierter Sicherheit beschreibt eine Architektur zwischen Benutzer, Serverdiensten und Authentifizierungsservern. Ein Benutzer holt sich dabei per Benutzername und Passwort oder mittels einer anderen geeigneten Authentifizierungsmethode von einem zentralen Authentifizierungsserver einen sogenannten Token. Dieser Token kann zum Beispiel eine spezielle Zeichenfolge inklusive einem Gültigkeitsdatum sein. Wenn nun der Benutzer einen Dienst im Netzwerk nutzen möchte, authentifiziert er sich nicht mehr mit seinem Benutzernamen und Passwort, sondern sendet den Token an den Dienst zur Authentifizierung. Der Dienst wiederum lässt den Token beim Authentifizierungsserver überprüfen [38]. Dadurch, dass der Client sich nur gegenüber dem Authentifizierungsserver mit Benutzernamen und Passwort authentifiziert (oder anderen geeigneten Authentifizierungsmaßnahmen), nicht jedoch direkt gegenüber dem Serverdienst, wird das Gesamtsystem sicherer [58].

### 3.4 Scheinbare Sicherheit durch Verschleierung

Unter Sicherheit durch Verschleierung<sup>4</sup> versteht man das Prinzip, Sicherheit durch Geheimhaltung zu erreichen [17].

Ein Beispiel aus dem Alltag: Solange niemand weiß, dass der Türschlüssel unter der Fußmatte versteckt ist, ist die Wohnung „gesichert“. Sobald aber jemand das Versteck herausfindet - sei es durch Beobachtung oder durch Ausprobieren - ist kein Schutz mehr gegeben.

Übertragen auf die IT-Welt, kann ein Betreiber einen Serverdienst ungeschützt betreiben und darauf hoffen, dass niemand den Dienst im Netzwerk bemerkt. Um die Chancen der Geheimhaltung, vor allem gegen automatisierte Angriffe per Software, zu erhöhen, kann der Betreiber die Dienste des Servers so konfigurieren, dass nicht standardkonforme TCP-Ports verwendet werden [83]. Sicherheit durch Verschleierung sollte aber auf jeden Fall nur als Ergänzung des Gesamtsicherheitskonzepts dienen, da die Sicherheit eines Systems nicht von der Geheimhaltung spezieller Verfahren oder der Verschleierung von Diensten abhängig sein darf [18] [15].

### 3.5 Zusammenfassung der Sicherheitskonzepte in der Nachrichtenübermittelung

Bei der Nachrichtenübermittelung von schützenswerten Daten wie beispielsweise von Benutzername und Passwort während des Loginvorgangs oder beim Abrufen geschützter Daten, wie etwa Landkarten, muss gewährleistet sein, dass der Datenaustausch zwischen den Kommunikationspartner weder abgehört noch manipuliert werden kann. Grundlegende Konzepte bilden dabei die *Nachrichten-basierte Sicherheit* und die *Transport-basierte Sicherheit*. Um sich gegenüber seinem, eventuell noch unbekannten, Kommunikationspartner authentifizieren zu können, gibt es klassische Verfahren wie das Anmelden per Benutzername und Passwort. Ein sichereres Verfahren, bei dem sich der Benutzer nur bei einem ihm bekannten Authentifizierungsserver anmelden muss, um

---

<sup>4</sup>Englisch: Security by Obscurity

weitere, ihm eventuell noch unbekannte Dienste nutzen zu können, spiegelt sich in der *Token-basierte Sicherheit* wider. Das Konzept der *Sicherheit durch Verschleierung*, bei welchem Dienste auf Servern „versteckt“ werden, darf niemals als einziger Sicherheitsaspekt genutzt werden, um Daten zu schützen.

# 4

---

## Webbasierte Authentifizierungs- und Autorisierungstechnologien

### 4.1 Authentifizierungsprotokolle

In der heutigen Computerlandschaft gibt es eine Vielzahl von verschiedenen Authentifizierungs- und Autorisierungsprotokollen. Es wäre fast unmöglich und ist auch nicht die Aufgabe dieser Arbeit, alle Protokolle vorzustellen.

Die einzelnen Protokolle unterscheiden sich oft erheblich im Einsatzzweck und Verbreitung. So gibt es Protokolle, die geschaffen wurden, damit sich Benutzer per ISDN / UMTS / DSL-Modem beim Internetprovider einwählen, authentifizieren und sich so mit dem Internet verbinden können. Andere Authentifizierungsprotokolle wurden geschaffen, damit Benutzer den Internetservice ABC auf Daten desselben Benutzers beim Internetservice XYZ zugreifen lassen können. In beiden Szenarien kommen Autorisierungsprotokolle zum Einsatz, aber in sehr unterschiedlichen Ausprägungen.

Exemplarisch seien im Anschluss einige Protokolle, Standards und Frameworks beschrieben, auf die im späteren Verlauf der Arbeit noch näher eingegangen wird. Die Unterkapitel *OAuth* und *OpenID* sind besonders detailliert dargestellt, da diese Protokolle im Rahmen der prototypischen Implementierung in Kapitel 8 Verwendung finden.

### 4.1.1 Authentifizierungsmethoden im Hypertext Transfer Protokoll

Um den Zugriff auf einzelne Ressourcen eines Webservers mittels HTTP-Protokoll zu schützen, gibt es heute zwei etablierte Verfahren, welche nun genauer vorgestellt werden [65].

Das *HTTP Basic Authentifizierung*-Verfahren, welches erstmals im Request for Comments (RFC) 1945 beschrieben und in HTTP Version 1.0 implementiert wurde, bietet einen Zugriffsschutz auf einzelne Verzeichnisse eines Webservers. Da Benutzername und Passwort im Klartext vom Webbrower des Endbenutzers zum Webserver übertragen werden, bietet es nur einen geringen Schutz gegen Angreifer. Beim Einsatz der HTTP Basic Authentifizierung sollte daher zusätzlich der Übertragungskanal mit TLS gesichert werden [65].

Um die größten Sicherheitsmängel von HTTP Basic Authentifizierung zu beseitigen, wurde mit HTTP Version 1.1 im RFC 2617 die Methode *HTTP Digest Access Authentication* eingeführt, welche ein einfaches Challenge-Response-Verfahren (siehe 2.1.2) spezifiziert [65].

#### Authentifizierung mittels HTML-Formularen

Um nicht nur einzelne Ressourcen wie Webseiten oder Dokumente am Webserver zu schützen, sondern den Zugriff auf ganze Webanwendungen zu regeln, die mit Technologien wie PHP, Java, .NET oder ähnliches realisiert sind, gibt es eine weit verbreitete Form der Authentifizierung mittels HTML-Formularen. Hierbei werden Benutzername und Passwort in ein HTML-Formular eingetragen und zum Webserver gesendet. Die Authentifizierung übernimmt dann die Webanwendung selbst. Da HTTP Basic und Digest Access Authentifizierung nur zwischen dem Webbrower und dem Webserver abläuft, müsste der Webserver die Authentifizierungsdaten an die Webanwendung exportieren. Diese Abhängigkeit wird hiermit umgangen [65]. Da das Passwort meist im Klartext zur Webanwendung übertragen wird, ist auch hier eine Sicherung des Übertragungskanals mittels TLS essentiell.

### 4.1.2 Kerberos

Kerberos ist ein am „Massachusetts Institute of Technology“ entwickelter, verteilter Authentifizierungsdienst für ungesicherte TCP/IP-Netzwerke. Der Endbenutzer möchte die Dienste eines Servers nutzen, die Authentifizierung übernimmt dabei eine vertrauenswürdige dritte Partei, der Kerberos-Server [55] [67]. Ein Benutzer muss daher vom Kerberos Server zuerst eine Authentizitätsbescheinigung in Form eines sogenannten Tickets angefordert haben. Dieses Ticket übermittelt der Benutzer gemeinsam mit seiner Anfrage an den Serverdienst. Der Server löst das Ticket aus der Anfrage des Clients heraus und lässt die Korrektheit des Tickets beim Kerberos Server prüfen [56]. Der gesamte Prozess geschieht für den Anwender im Hintergrund, der von der Komplexität nichts mitbekommt. Kerberos unterstützt das Single Sign-On (SSO)-Konzept, sodass Benutzer eines Firmennetzwerkes die einzelnen Dienste bequem und doch sicher nutzen können [72].

### 4.1.3 OpenID

OpenID ist ein 2005 für das Web geschaffenes dezentrales Authentifizierungssystem vom Entwickler Brad Fitzpatrick, Gründer des LiveJournals<sup>1</sup> [59] [60]. Seit 2007 wird die Entwicklung von der OpenID Foundation<sup>2</sup> getragen [28].

Die Benutzeridentität wird in Form einer eindeutigen Uniform Resource Locator (URL) gespeichert, welche die Benutzer-ID bzw. den Benutzernamen sowie die Adresse des OpenID-Providers beinhaltet [59]. Die OpenID<sup>3</sup> des Autors hat die Form: `http://schubec.myopenid.com`.

Der Benutzer meldet sich am Beginn einer Websession bei seinem OpenID-Provider an, zum Beispiel mit einem klassischen „Benutzername und Passwort“-Verfahren. Bei den weiteren vom Benutzer verwendeten Diensten, etwa seinem Blog, meldet er sich nun mit seiner OpenID an (Abbildung 4.1). Die Dienste, die OpenID verwenden, werden dabei als Konsument bezeichnet [59].

---

<sup>1</sup><http://www.livejournal.com>

<sup>2</sup><http://openid.net/foundation/>

<sup>3</sup>Etwas verwirrend könnte der Umstand sein, dass die mit der URL verknüpfte Identität, genau so wie das gesamte Verfahren, „OpenID“ genannt wird.



Abbildung 4.1: Login beim OpenID-Provider mit persönlichem Symbol zur Abwehr von Phishing-Attacken.

Der Konsument, in diesem Beispiel der Blogdienst, kann nun den OpenID-Provider kontaktieren und nachfragen, ob der Anwender, der den Browser bedient, tatsächlich die genannte OpenID inne hat bzw. auch tatsächlich mit dem verwendeten Webbrowsereingeloggt ist. Dazu leitet der Konsument den Benutzer automatisch zum OpenID-Provider weiter (wo er sich anmelden kann, falls er noch nicht angemeldet ist). Der OpenID-Provider bittet nun den Benutzer um Erlaubnis, die Anfrage des Konsumenten positiv beantworten zu dürfen [59]. Der Benutzer kann dabei festlegen, ob er dem Konsumenten dauerhaftes Zugreifen ohne weitere Fragen erlauben möchte oder ob er es bevorzugt, bei jedem Zugriff befragt zu werden. Der Konsument kann den OpenID-Provider dabei nicht nur fragen, ob der Benutzer korrekt authentifiziert wurde, sondern diesen auch ersuchen, erweiterte Parameter wie den Namen, die Emailadresse, die Postleitzahl und ähnliche Attribute abzurufen [59]. Für die Übermittelung dieser Daten benötigt der OpenID-Provider ebenfalls die Erlaubnis des Anwenders (Abbildung 4.2).

Nun antwortet der OpenID-Provider dem Konsumenten, dass der Benutzer korrekt angemeldet ist und überträgt zusätzlich, sofern angefordert, die Daten wie die Emailadresse, Telefonnummer etc.

Dieses Verfahren hat Vorteile für den Benutzer und für den Betreiber des Konsumenten. Der Konsument muss in seiner lokalen Datenbank keine Passwörter speichern. Sollte ein Hacker die Konsumenten-Datenbank mit den Benutzerdetails in die Hände



Abbildung 4.2: Autorisierung einer Authentifizierungsanfragen beim OpenID-Provider. 1: Dienst, der Authentifizierungsinformationen abrufen möchte. 2: Verwendete OpenID. 3: Button, um die Authentifizierungsanfrage zu autorisieren. 4: Auswahl der Identität, deren Informationen preisgegeben werden. 5: Attribute und Werte der Identität, die preisgegeben werden. 6: Option, um Authentifizierungsanfragen des Dienstes dauerhaft zu erlauben. 7: Hyperlink, um die Authentifizierung nicht zu autorisieren. 8: Persönliches Symbol zur Abwehr von Phishing.

bekommen, so kann er nur die OpenIDs auslesen. Da der Konsument keine Passwörter lokal gespeichert hat, muss er keine Anstrengungen aufwenden, diese Daten speziell zu schützen [59]. Der Konsument wird jedoch nicht von der Pflicht entbunden, ein lokales Benutzerkonto für seine Anwender zu führen, wo er speichert, zu welcher Benutzergruppe der Anwender gehört, um ihm damit verschiedene Rechte einzuräumen.

Weiters kann der Konsument auf die im Internet am häufigsten benötigten Attribute

- Nickname (Spitzname)
- Email (Emailadresse)
- Full name (Voller Name des Benutzers)
- DoB (Date of Birth, Geburtsdatum)
- Gender (Geschlecht: M für männlich, F für weiblich)
- Postcode (Postleitzahl)
- Country (Land)
- Language (Sprache)
- Timezone (Zeitzone)

zugreifen (sofern vom Benutzer erlaubt) [59]. Dadurch, dass der Benutzer im OpenID System diese Daten nur an einer zentralen Stelle pflegen muss und die Daten im Falle einer Änderung nicht in jedem einzelnen verwendeten Webdienst manuell korrigiert werden müssen, ist die Wahrscheinlichkeit, dass diese Daten korrekt sind, höher als ohne zentrale Verwaltung. Durch die zentrale Verwaltung der Authentifizierungsdaten beim OpenID-Provider müssen die einzelnen Konsumenten auch keine Verfahren wie „Passwort vergessen“ oder „Passwort ändern“ implementieren, da diese Aufgaben vollständig dem OpenID-Provider überlassen werden [59].

Aus Sicht des Benutzers gibt es ebenfalls eine Menge an Vorteilen. So muss er sich nur noch eine Benutzername- und Passwortkombination merken, um sich beim OpenID-Provider anzumelden. Bei allen anderen Diensten, die OpenID verstehen (das heißt bei anderen Konsumenten), muss er nur noch seine OpenID angeben. Dadurch, dass die einzelnen Konsumenten keine Passwörter abfragen, ergibt sich automatisch auch ein großer Sicherheitsgewinn, denn Informationen, die man nicht preisgibt, können auch nicht abgefangen oder manipuliert werden [59].

Des weiteren können, je nach OpenID-Provider, zusätzliche Sicherheitsverfahren zur Authentifizierung beim OpenID-Provider verwendet werden. myOpenId<sup>4</sup> bietet zum Beispiel das Verfahren CallVerifID an, bei dem man beim Authentifizierungsvorgang auf seinem Telefon angerufen wird, um den Login zu bestätigen (Abbildung 4.3).

---

<sup>4</sup><http://www.myopenid.com>

**CALLVERIFID™**

---

For added security, myOpenID can call you on your telephone for authorization to sign in.

CallVerifID™ is not currently enabled on your account.  
Simply add your phone number below to start the service.

Your phone number   
e.g. "(503) 555-1234" or "+44 08700 100 222"



**CallVerifID**  
powered by PhoneFactor

Abbildung 4.3: Einstellungen zur starken Authentifizierung per Telefon beim Provider myOpenId.

**ADD AN SSL CLIENT CERTIFICATE**

---

Klicken Sie auf den Knopf weiter unten, um ein SSL-Zertifikat in Ihrem Browser zu installieren. Ihr Browser wird dieses Zertifikat nutzen, um Ihre Identität bei myOpenID, unter Verwendung von Transport Layer Security (TLS), zu bestätigen. Das Verwenden dieses Zertifikats vermeidet die Notwendigkeit empfindliche Daten, wie Ihr Passwort, einzugeben zu müssen.

This certificate will be installed on the hard drive of your computer. If you use more than one computer, you will need to install a certificate on each of them.

**Achtung:** Sie sollten kein Zertifikat auf einem Computer installieren, über welchen Sie keine Kontrolle haben, oder der zusammen mit anderen genutzt wird. Ihr Zertifikat ist wie Ihr Benutzername und Passwort: Schützen Sie es!

**Tip:** Add a label below to help keep track of your myOpenID certificates.

Name   
z.B. "Zuhause", "Arbeit", "Laptop",

**Zertifikat erzeugen**

Abbildung 4.4: Erstellen eines SSL-Client-Zertifikates für starke Authentifizierung.

Des Weiteren kann man ein eigenes SSL-Client-Zertifikat für seinen Webbrowser erstellen (Abbildung 4.4) sowie ein persönliches Symbol/Bild hinterlegen, welches beim Login-Vorgang angezeigt wird. Dadurch wird es Phishern<sup>5</sup> erschwert den Login-Dialog nachzuahmen, da diese das persönliche Symbol normalerweise nicht kennen (Abbildung 4.1).

<sup>5</sup>Phishing bezeichnet den Versuch, Informationen wie Benutzername und Passwort oder Kontonummer und TAN-Code vom Benutzer zu erschleichen. Dabei wird zum Beispiel die Login- und Kontoseite einer Bank nachprogrammiert und der Benutzer auf diese Seite gelockt. Der Benutzer glaubt nun auf der Webseite der Bank zu sein und gibt seine Daten preis, indem er versucht sich anzumelden. Damit haben die Phisher nun Benutzername und Passwort erschlichen und können diese Informationen missbrauchen.

<b>Bernhard Schulz</b>	
	<b>Voller Name</b> Bernhard Schulz <b>Spitzname</b> schube <b>Geschlecht</b> Male <b>Email</b> office@schubec.com <b>Web</b> <a href="http://www.schubec.com">http://www.schubec.com</a>
	<b>Postleitzahl</b> 5020 <b>Land</b> Austria <b>Language</b> German <b>Zeitzone</b> Europe/Vienna
<a href="#">Edit this persona</a> <a href="#">Delete this persona</a>	

Abbildung 4.5: OpenID Identität mit korrekten persönlichen Angaben zur Nutzung von Diensten, die auf diese Werte zugreifen dürfen, wie beispielsweise Onlineshops.

<b>Onlineplayer B.</b>	
	<b>Spitzname</b> Beans <b>Email</b> onlineplayer@gmail.com <b>Language</b> German <b>Zeitzone</b> Europe/Vienna
<a href="#">Edit this persona</a> <a href="#">Delete this persona</a>	

Abbildung 4.6: Phantasie OpenID Identität für die anonyme Nutzung von Diensten.

Provider wie myOpenId können darüber hinaus multiple Identitäten<sup>6</sup> eines Benutzers verwalten, sodass der Benutzer eine Identität für das Onlineshopping mit korrektem Namen und Postleitzahl (Abbildung 4.5) nutzen und für Dienste, wo er anonym bleiben möchte, eine Phantasie-Identität verwenden kann (Abbildung 4.6).

Da OpenID nur eine Art der Authentifizierung ist, können die Konsumenten OpenID gleichzeitig mit anderen Authentifizierungsverfahren wie Benutzername und Passwort

<sup>6</sup>Während eine Identität genau einer Person zuzuordnen ist, kann ein Benutzer multiple Identitäten annehmen [35].

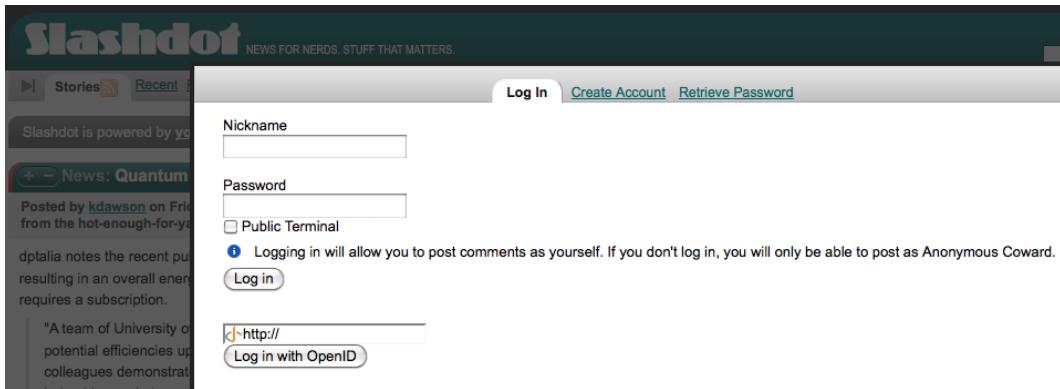


Abbildung 4.7: Authentifizierung über Benutzernname und Passwort oder OpenID bei Slashdot.org. Beide Systeme können parallel existieren.

verwenden. Diensteanbieter, die OpenID nutzen möchten, müssen daher nicht das gesamte Authentifizierungssystem wechseln, sondern können OpenID zusätzlich zu den gewohnten Verfahren anbieten, sodass bestehende Systeme einfach erweitert werden können (Abbildung 4.7).

IT-Verantwortliche können mit Hilfe von OpenSource-Software selbst OpenID-Provider für sich oder ihre Firma/Community werden [59]. Branchengrößen wie Google<sup>7</sup>, Yahoo<sup>8</sup>, AOL<sup>9</sup>, flickr<sup>10</sup>, MySpace<sup>11</sup> und weitere bekannte Namen sind ebenfalls OpenID-Provider (Abbildung 4.8). Hat ein Nutzer ein Konto bei Google, besitzt er damit automatisch eine OpenID von Google - oftmals, ohne dass ihm das bewusst ist. Durch den Support der großen Provider haben bereits mehrere hundert Millionen Anwender OpenIDs.

Auf der Seite der Konsumenten findet man tausende von Seiten, die OpenID unterstützen, in Verzeichnissen wie <https://www.myopenid.com/directory>. Große Anbieter wie Facebook<sup>12</sup> unterstützen die Authentifizierung mittels OpenID, auch wenn die Optionen dafür manchmal noch etwas versteckt sind (Abbildung 4.9). OpenID ist bereits heute weit verbreitet und wird breit unterstützt. Fast täglich kommen weitere Provider und Konsumenten hinzu. Da OpenID eine relativ junge Technologie ist und

<sup>7</sup><http://www.google.com>

<sup>8</sup><http://www.yahoo.com>

<sup>9</sup><http://www.aol.com>

<sup>10</sup><http://www.flickr.com>

<sup>11</sup><http://www.myspace.com>

<sup>12</sup><http://www.facebook.com>

## Surprise! You may already have an OpenID.

If you use any of the following services, you already have your own OpenID. Below are instructions on how to sign in with each of the following providers on an OpenID enabled website. (When you see bold text, you should replace it with your own username or screenname on that service.)



Look for the "Sign in with a Google Account" button or use your [Google Profile URL](#).



Look for the "Sign in with Yahoo" button.



Look for the "Yahoo! JAPAN IDでログイン" button.



Enter "[username.livejournal.com](#)"



Click the "Sign in with Hyves" button.



Enter your blog URL: ["blogname.blogspot.com"](#)



Look for the "Sign in with Yahoo" button or use your photostream URL



Click the "Sign in with Orange" button or enter "[orange.fr](#)"



**mixi** is a web service that allows users to communicate with their friends and acquaintances.



Look for the "Login with MySpaceID" button or enter "[www.myspace.com/username](#)"



Enter your Wordpress.com URL, for example: ["username.wordpress.com"](#)



Look for a "Sign in with AOL" button or enter "["openid.aol.com/screenname"](#)"

Abbildung 4.8: Inhaber eines Benutzerkontos bei großen Anbietern wie Google oder Yahoo haben automatisch eine OpenID, oft ohne dies zu wissen.

Abbildung 4.9: Verknüpfung eines Facebook-Kontos mit einer OpenID für eine zentrale Authentifizierung.

sich viele Anwender noch nicht mit dem Thema Authentifizierung beschäftigt haben und auf klassische Logins mit Benutzername und Passwort bei jedem Dienst setzen, sind die Optionen zur Nutzung von OpenID bei vielen Konsumenten in Detaileinstellungen verborgen und müssen erst explizit aktiviert werden, wie dies etwa bei Facebook der Fall ist. Es ist aber zu erwarten, dass OpenID bei den Anwendern mehr Aufmerksamkeit erlangt und die Technologie daher in Zukunft noch größere Verbreitung erlangt.

Hinsichtlich der technischen Spezifikationen und dem genauen Ablauf bei der Authentifizierung durch Konsumenten sei auf [24] sowie auf [59] verwiesen.

## 4.2 Autorisierungsprotokolle für verteilte Dienste

Im Folgenden werden vier Autorisierungsprotokolle beschrieben, die in verteilten Systemen zum Einsatz kommen. Obwohl diese Protokolle ähnliche Ziele verfolgen und geschaffen wurden, um Autorisierungsinformationen zu teilen, unterscheiden sie sich in den Fähigkeiten und im Einsatzzweck erheblich.

### 4.2.1 OAuth

Das offene Autorisierungsprotokoll OAuth wurde 2007 von Blaine Cook und Chris Messina geschaffen [41].

Blaine Cook arbeitete an einem Application Programming Interface (API) für den Mikrobloggingdienst Twitter<sup>13</sup>. Ziel dieses APIs war es unter anderem, dass verschiedene Web-, Desktop- und mobile Anwendungen im Namen des Benutzers Twitternachrichten senden können, ohne dass diese Benutzername und Passwort des Benutzers vorliegen haben. „Im Namen des Benutzers“ bedeutet dabei, dass der Benutzer seine Nachricht nicht direkt auf der Twitter Webseite eingibt, sondern in die jeweils genutzte Clientanwendung. Diese Anwendung nimmt die Nachricht entgegen und sendet sie an Twitter. Obwohl die Nachricht also nicht direkt vom Benutzer kommt, sondern über die Anwendung gesendet wurde, erkennt Twitter, dass die Anwendung dies im Namen des

---

<sup>13</sup><http://www.twitter.com>

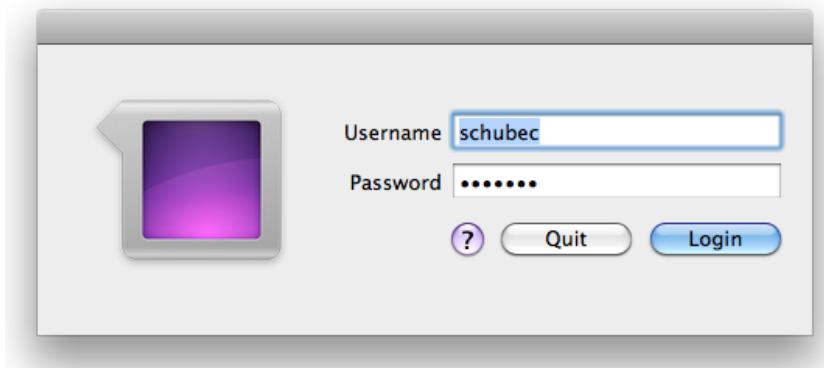


Abbildung 4.10: Klassische (unsichere) Authentifizierung eines Twitter Clients beim Twitterdienst per Benutzername und Passwort.

Benutzers gemacht hat und postet die Nachricht im entsprechenden Benutzer-Tweet [41].

Damit nun die Anwendung im Namen des Benutzers Nachrichten an Twitter senden kann, muss die Anwendung sich gegenüber Twitter authentifizieren. Dazu könnte der Benutzer die Logindaten von Twitter in seiner Anwendung hinterlegen. Das Problem dabei ist nun jedoch, dass der Benutzer der Anwendung nicht sicher weiß, was die Anwendung mit den Benutzerdaten macht. So könnte eine manipulierte Anwendung Benutzername und Passwort an einen Angreifer weiterleiten und preisgeben (Abbildung 4.10).

Eine weitere Problematik ist, dass Anwender unter Umständen eine Twitteranwendung auf ihrem Handy nutzen, eine am privaten Desktop PC, eine am Firmenlaptop und eine weitere Anwendung auf einem Netbook im Ferienhaus. Wird nun das Netbook gestohlen, muss der Anwender seine Logindaten bei Twitter ändern, denn sonst kann der Dieb im Namen des Bestohlenen unvorteilhafte Nachrichten bei Twitter posten. Eine Änderung des Passworts bewirkt nun aber, dass der Benutzer in allen anderen Clients auf seinem Desktop PC, dem Firmen Laptop und dem Handy die neuen Daten ebenfalls hinterlegen muss. Bei der Nutzung nur einer Webanwendung ist dieses Unterfangen vielleicht noch überschaubar, wird aber schnell mühsam, wenn der Benutzer mehrere Dienste und Anwendungen wie Google Mail, Facebook, Twitter, Flickr und weitere Onlinesysteme mit eigenen Clientanwendungen nutzt.



Abbildung 4.11: Der Facebook-„Freundefinder“, welcher nach Angabe von Benutzernname und Passwort die Adressbücher von Webmailbetreibern auslesen kann, um Kontakte auf Facebook zu finden. Facebook verspricht, das Passwort nicht zu speichern, der Benutzer kann dies jedoch nicht kontrollieren.

Das Hinterlegen von Logindaten ist auch unter dem Gesichtspunkt der Autorisierung problematisch, da der Benutzer mit seinem Benutzernamen und Passwort meist alle Funktionen der Webanwendung nutzen kann. Dazu ein Beispiel: Mit Benutzernname und Passwort seines Google Mail<sup>14</sup> Accounts kann man das persönliche Adressbuch, aber auch die eigenen Emails, Termine und ToDo-Listen Einträge sowie Daten von vielen weiteren Google-Diensten einsehen, ändern oder löschen. Meldet man sich beim weltgrößten Social Network Facebook an, so hat man im Zuge der Anmeldung die Möglichkeit, die Logindaten von Google Mail anzugeben. Facebook verspricht nur das Adressbuch auszulesen, um zu sehen, ob bereits Kontakte aus dem Adressbuch auf Facebook vertreten sind, um so gleich Facebookbekanntschaften zu schließen (Abbildung 4.11). Gibt der Benutzer nun Benutzernname und Passwort von Google Mail auf der Facebook Webseite an, so macht Facebook offenkundlich das, was es angibt zu machen - es sieht nach, ob bereits Freunde aus dem Adressbuch vorhanden sind. Der Benutzer kann aber nicht kontrollieren, ob Facebook nicht auch Kalendereinträge ausliest oder Emails des Benutzers speichert. Der Benutzer, der solche bequeme Funktionen nutzen will, muss Facebook daher vertrauen. Der Benutzer kann außerdem nicht kontrollieren, ob und wie lange Facebook die Logindaten von Google Mail speichert. Erst durch ein Ändern seines Passworts bei Google Mail kann er sicher sein, dass Facebook nicht mehr auf seine Daten zugreifen kann.

<sup>14</sup><http://mail.google.com>

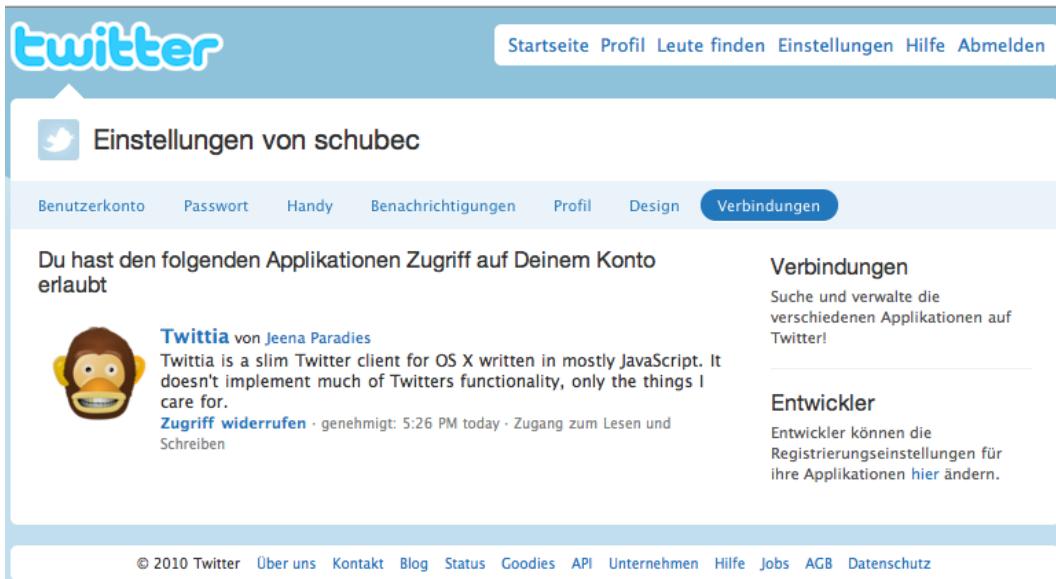


Abbildung 4.12: Verwalten der Anwendungen, die auf den eigenen Twitter Account zugreifen dürfen, inklusive der Möglichkeit, dies jederzeit zu widerrufen.

Um diese Probleme zu lösen, wurde OAuth entworfen. Das Entwurfsziel dieses Protokolls war, dass Anwender<sup>15</sup> Webanwendungen<sup>16</sup> autorisieren können, Dienste oder Daten<sup>17</sup> für andere Webanwendungen<sup>18</sup> zur Verfügung zu stellen. Dabei sollte der Endnutzer der Dienste bestimmen können, welche Daten ausgetauscht werden und wie lange der Datenaustausch möglich ist. Darüber hinaus sollte er durch Widerrufen seiner Erlaubnis in der Lage sein, diesen Datenaustausch jederzeit zu unterbinden (siehe Abbildung 4.12). Ein wichtiges Entwurfsziel war, dass der Datenaustausch stattfinden können sollte, ohne dass die Dienste untereinander die jeweiligen Logindaten kennen müssen. OAuth verwendet Tokens, die der Benutzer bestätigen muss, um dies zu erreichen (siehe Abbildung 4.13).

<sup>15</sup>Im OAuth Protokoll als Endnutzer bzw. User spezifiziert.

<sup>16</sup>Im OAuth Protokoll als Serviceanbieter bzw. Service-Provider spezifiziert.

<sup>17</sup>Im OAuth Protokoll als private Ressourcen bzw. private Resources spezifiziert.

<sup>18</sup>Im OAuth Protokoll als Konsument bzw. Consument spezifiziert.

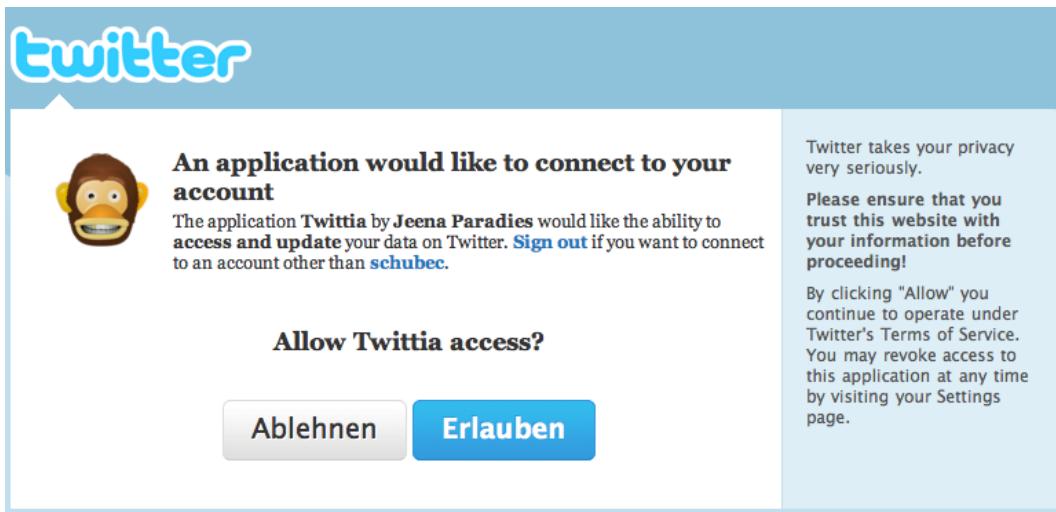


Abbildung 4.13: Nutzung von Twitter per OAuth, indem eine Clientanwendung Zugriff gewährt wird. Der Benutzer war in diesem Szenario schon bei Twitter eingeloggt. Anders als bei Abbildung 4.10 werden Benutzername und Passwort niemals an die Twitter-Clientanwendung übermittelt. Stattdessen erhält die Clientanwendung einen OAuth-Zugriffstoken, den Twitter erstellt und der Benutzer bestätigt. Die Details bleiben dem Benutzer verborgen, eine unkomplizierte Nutzung dieses Systems ist daher einfach möglich.

### Anwendungsszenarien und Spezifikationen

Für OAuth gibt es mit dem sogenannten dreibeinigen und zweibeinigen OAuth zwei typische Anwendungsszenarien<sup>19</sup>.

Beim *dreibeinigen OAuth* interagieren die drei Parteien Endnutzer, Konsument und Service-Provider miteinander - daher auch der Name. Das dreibeinige OAuth ist für den Datenaustausch zwischen Webanwendungen unter der Voraussetzung, dass der Endnutzer zustimmt, gedacht [7].

Das oben genannte Beispiel, wo Facebook auf die Kontaktdaten des Adressbuchs bei Google Mail zugreift, sofern der Endnutzer dies erlaubt, spiegelt den dreibeinigen OAuth wieder. Der Anwender repräsentiert dabei den Endnutzer, Facebook ist der Konsument und Google Mail der Service-Provider.

<sup>19</sup>Im Englischen als 3-legged und 2-legged OAuth bezeichnet.

Beim *zweibeinigen OAuth* findet ausschließlich eine Maschine-zu-Maschine-Kommunikation statt, es ist dabei keine Authentifizierung des Endnutzers nötig und die Interaktion mit dem Endnutzer entfällt - daher der Name zweibeiniges OAuth [7].

Der Zugriff zwischen den Maschinen wird mittels wissensbasierter Authentifizierung oder einer Public-Key-Infrastruktur geregelt. In diesem Szenario stellt OAuth hauptsächlich ein Authentifizierungsprotokoll dar, wobei die Autorisierung darin widergespiegelt ist, dass laut OAuth Standard eine Zeitspanne angegeben werden muss, wie lange die Dienste miteinander kommunizieren dürfen [7].

Dieser Zugriff könnte nur 5 Minuten erlaubt werden, sodass Dienst A einmalig alle benötigten Informationen von Dienst B abrufen kann, oder der Zugriff wird für 100 Jahre genehmigt, was theoretisch einem ständigen Zugriff entspricht, beispielsweise wenn Dienst A jeden Tag die Daten von Dienst B abrufen soll, um diese in einen Backupdatensatz zu schreiben.

## OAuth Versionen

*OAuth Core 1.0* wurde am 4. Dezember 2007 veröffentlicht und beinhaltete die Spezifikationen zum dreibeinigen und zum zweibeinigen OAuth. Am 24. Juni 2009 wurde *OAuth Core 1.0 Revision A* veröffentlicht, welches eine Sicherheitslücke beim dreibeinigen OAuth schloss [7]. *OAuth Version 2.0* befindet sich derzeit in der Entwicklungsphase, und liegt als Entwurf<sup>20</sup> beim IETF<sup>21</sup> vor, wird aber zum Beispiel bereits von Facebook für die Graph API<sup>22</sup> verwendet [22].

## Abgrenzung gegenüber OpenID

OpenID und OAuth werden oft verwechselt, obwohl beide Protokolle unterschiedliche Ziele verfolgen und nicht konkurrieren sondern sich ergänzen. Bei OAuth geht es um die Autorisierung, bei OpenID um die Authentifizierung.

---

<sup>20</sup><http://tools.ietf.org/html/draft-ietf-oauth-v2-02>

<sup>21</sup>The Internet Engineering Task Force, <http://www.ietf.org>

<sup>22</sup><http://developers.facebook.com/docs/authentication/>

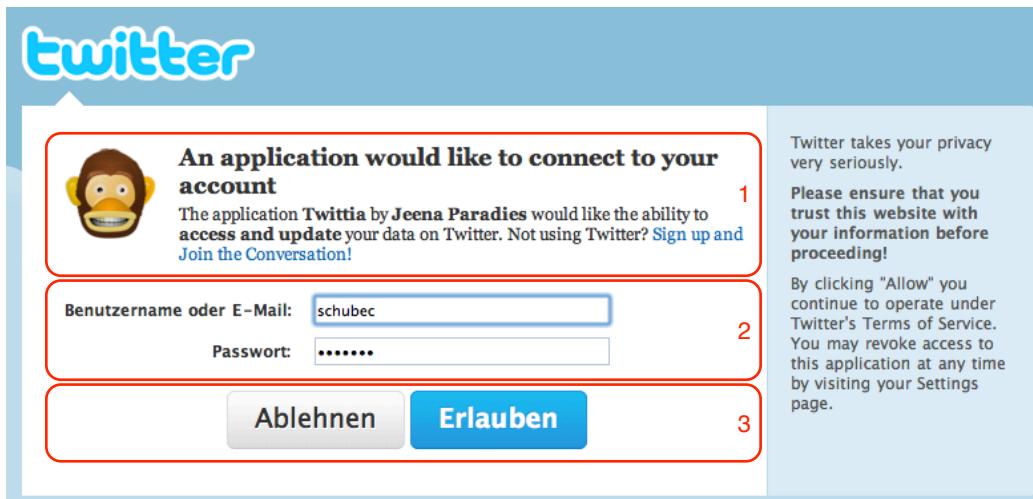


Abbildung 4.14: Autorisierung und Authentifizierung in einem Anwendungsdialog. 1. Autorisierungsanfrage durch OAuth. 2. Login per Benutzername und Passwort. Dies hat nichts mit dem OAuth Protokoll zu tun und könnte auch über andere Verfahren wie OpenID realisiert werden. 3. Button zur gleichzeitigen Autorisierung der OAuth-Anfrage und zur Authentifizierung beim Service-Provider. Dadurch, dass Authentifizierung und Autorisierung oft gleichzeitig stattfinden, ist die Unterscheidung für viele Benutzer nicht klar und wird als Einheit gesehen.

Die Authentifizierung, die oft im Rahmen der Autorisierung der Tokens beim Service-Provider notwendig ist und dem Anwender in einem Dialog präsentiert wird (Abbildung 4.14), ist nicht Bestandteil von OAuth und kann vom Anbieter frei implementiert werden - etwa per Benutzername und Passwort oder per OpenID.

### Technisches Spezifikationen von OAuth

Hinsichtlich der technischen Spezifikationen und dem genauen Ablauf bei der Autorisierung der Konsumenten sei auf [29] verwiesen.

#### 4.2.2 Security Assertion Markup Language

Die Security Assertion Markup Language (SAML) ist ein auf Internetprotokollen basierender Standard zum Austausch von Authentifizierungs- und Autorisierungsinformationen über Organisationsgrenzen hinweg [54]. Diese Übertragung ist zum Beispiel

bei firmenübergreifenden Projekten wichtig, wobei die Nutzer einer Firma den Onlinedienst einer anderen Firma nutzen möchten, beispielsweise einen externen Webmail Dienst. Damit sich die Mitarbeiter der Firma, die sich ohnehin schon im lokalen Intranet anmelden, nicht erneut beim Webmaildienst des externen Dienstleisters authentifizieren müssen, kann der lokale Intranetserver mit dem externen Webmailserver ein Vertrauensverhältnis eingehen und Authentifizierungsinformationen mit diesem teilen. Im Kontext von SAML spricht man dabei von einem „Service Provider“, der gewisse Dienste anbietet und dem „Identity Provider“, der Informationen zum Benutzer, wie zum Beispiel den Benutzernamen, zur Verfügung stellt. Greift ein Benutzer, in diesem Kontext „Subjekt“ genannt, auf eine geschützte Ressource am Service Provider zu, leitet dieser das Subjekt zum Identity Provider um, wo sich das Subjekt authentifizieren muss. Nach erfolgreicher Authentifizierung wird der Service Provider darüber in Kenntnis gesetzt und gibt den Zugriff zur geschützten Ressource frei [37].

Der Funktionsumfang von SAML ist groß und die Spezifikationen sind umfangreich. Der offizielle Standard umfasst circa 350 Seiten. Im Rahmen dieser Arbeit wurde daraus der Teil der Autorisierung näher betrachtet und im Kapitel 7.1 mit anderen Technologien verglichen, daher ist das SAML-Kapitel hier unter Autorisierungsprotokolle aufgeführt, auch wenn die Spezifikationen umfassender sind und Bereiche wie den Aufbau der XML-Nachrichten, Definition der Vertrauensverhältnisse, Protokoll-Bindings und vieles mehr beschreiben. Es würde jedoch den Rahmen dieser Einführung sprengen weitere Details näher zu beschreiben. Der interessierte Leser sei auf [1] verwiesen.

#### 4.2.3 Shibboleth

Shibboleth, ein Projekt der Internet2 Middleware Initiative<sup>23</sup>, ist ebenfalls ein System zur verteilten Authentifizierung und Autorisierung über Organisationsgrenzen hinweg [25]. Die Organisationen, die ihre Authentifizierungs- und Autorisierungsinformationen teilen möchten, schließen sich dabei zu sogenannten Föderationen zusammen. Jede Organisation verwaltet seine eigenen Benutzer und ist damit ein sogenannter Identity Provider. Bietet die Organisation Ressourcen an, auf die externe Benutzer zugreifen

<sup>23</sup><http://www.internet2.edu/middleware/index.cfm>



Abbildung 4.15: Shibboleth: 1. Schritt: Zugriff auf eine geschützte Ressource. Abbildung kopiert von [69].

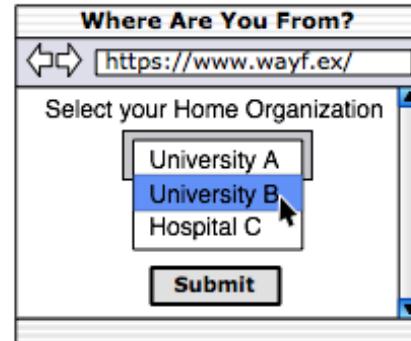


Abbildung 4.16: Shibboleth: 2. Schritt: Auswahl der Heimatuniversität zur Authentifizierung. Abbildung kopiert von [69].

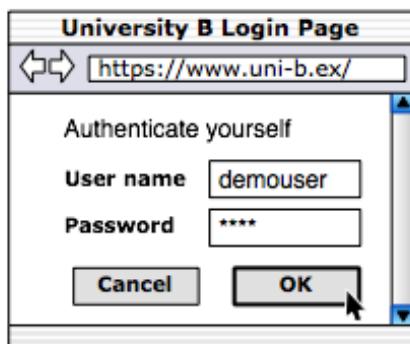


Abbildung 4.17: Shibboleth: 3. Schritt: Authentifizierung bei der Heimatuniversität. Abbildung kopiert von [69].



Abbildung 4.18: Shibboleth: 4. Schritt: Nach erfolgreicher Authentifizierung wird der Zugriff auf die geschützte Ressource erlaubt. Abbildung kopiert von [69].

möchten, tritt die Organisation auch als Service Provider auf [25].

Aus Benutzersicht tritt Shibboleth als Single Sign On-Lösung auf, technologisch basiert es auf SAML [37].

Im Rahmen dieser Arbeit wurde der Aspekt der Autorisierung näher untersucht, daher die Einordnung bei den Autorisierungsprotokollen.

Shibboleth sei nun anhand eines Beispiels erklärt: Der Benutzer versucht auf eine geschützte Ressource der Universität A zuzugreifen (Abbildung 4.15).

Da der Benutzer noch nicht authentifiziert ist, wird der Zugriff unterbunden und der Benutzer wird zu einem Lokalisierungsdienst weitergeleitet. Beim Lokalisierungsdienst wählt der Benutzer seine Heimatuniversität aus (Abbildung 4.16).

Nun authentifiziert sich der Anwender bei seiner Heimatuniversität und wird anschließend wieder zur geschützten Ressource weitergeleitet (Abbildung 4.17).

Der Server, der den Zugriff auf die geschützte Ressource regelt, kann nun entscheiden, den Zugriff auf die Ressource zu gewähren oder zu verbieten. Möchte dieser weitere Informationen zum Benutzer erhalten, etwa, ob es sich beim Benutzer um einen Studenten oder einen Professor handelt, kann er diese Informationen von der Heimatuniversität des Benutzers im Hintergrund abfragen, um seine Entscheidung anschließend zu treffen. Schlussendlich gewährt oder verbietet er im letzten Schritt den Zugriff (Abbildung 4.18).

Der genaue technische Ablauf ist komplex, wie Abbildung 4.19 zeigt. Für Details sei auf die Spezifikationen unter [33] verwiesen.

#### 4.2.4 WS-Federation

WS-Federation ist ein Teil der WS-Security Suite und hat das Ziel, eine Vertrauensdomäne über Unternehmensgrenzen hinweg, zwischen sogenannten Identity Providers und Relying Parties, zu schaffen. Identity Providers authentifizieren dabei die Benutzer und geben Security Tokens aus, mit denen die Benutzer geschützte Ressourcen auf den Relying Parties abrufen können [26] [70]. WS-Federation benutzt dabei Elemente aus WS-Security und anderen der mehr als 150 WS-\*<sup>24</sup> Spezifikationen [79]. Im Rahmen dieser Arbeit wurde der Aspekt der Autorisierung näher untersucht.

Die WS-Security Spezifikation stellt einen von Microsoft<sup>25</sup> und IBM<sup>26</sup> entwickelten Standard dar, um die Vielfalt an bestehenden Sicherheitstechnologien im Webservice-Umfeld zu standardisieren [43].

---

<sup>24</sup>Unter WS-\* versteht man die mehr als 150 mit dem Kürzel „WS-“ beginnenden Spezifikationen wie WS-Trust, WS-Security, WS-Policy, WS-SecurityPolicy, WS-Discovery etc.

<sup>25</sup><http://www.microsoft.com>

<sup>26</sup><http://www.ibm.com>

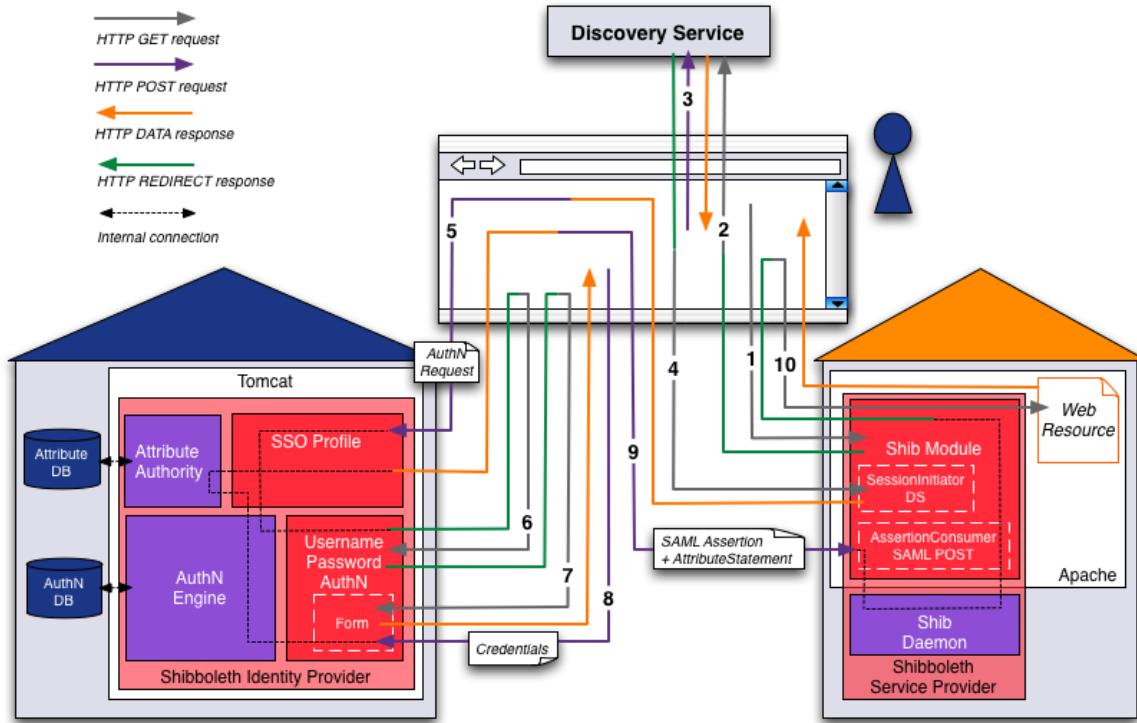


Abbildung 4.19: Shibboleth: Der Authentifizierungsvorgang per Shibboleth in seiner gesamten Komplexität. Abbildung kopiert von [69].

Hierbei wird SOAP so erweitert, dass bereits bestehende Techniken wie zum Beispiel XML-Signatur und -Verschlüsselung, Public-Key-Infrastrukturen oder auch die Security Assertion Markup Language, kurz SAML, eingesetzt werden können. Es werden also keine neuen Vorgehensweisen entwickelt, sondern lediglich der standardisierte Einsatz bewährter Verfahren vorschlagen [32, Seite 226].

WS-Security stellt somit keine Alternative zu Standards SAML dar, sondern nutzt diese [32].

Der Einleitungstext des Dokumentes [27] beschreibt WS-Federation treffend:

WS-Security, WS-Trust, and WS-SecurityPolicy provide a basic model for federation between Identity Providers and Relying Parties. These specifications define mechanisms for codifying claims (assertions) about a requestor

*as security tokens which can be used to protect and authorize web services requests in accordance with policy. WS-Federation extends this foundation by describing how the claim transformation model inherent in security token exchanges can enable richer trust relationships and advanced federation of services. This enables high value scenarios where authorized access to resources managed in one realm can be provided to security principals whose identities and attributes are managed in other realms. WS-Federation includes mechanisms for brokering of identity, attribute discovery and retrieval, authentication and authorization claims between federation partners, and protecting the privacy of these claims across organizational boundaries. These mechanisms are defined as extensions to the Security Token Service (STS) model defined in WS-Trust. In addition WS-Federation defines a mapping of these mechanisms, and the WS-Trust token issuance messages, onto HTTP such that WS-Federation can be leveraged within Web browser environments. The intention is to provide a common infrastructure for performing Federated Identity operations for both web services and browser-based applications [27, Seite 4].*

Die Spezifikationen von WS-Federation alleine sind bereits sehr umfangreich, der interessierte Leser sei auf [26] verwiesen. Da die WS-Federation Spezifikation aber in Kontext der anderen WS-\* Spezifikationen gesehen werden muss, welche viele tausende Seiten umfassen, ist eine Einarbeitung in WS-Federation nicht trivial. WS-Federation genauer vorzustellen würde den Rahmen dieser Arbeit sprengen.

### 4.3 Zusammenfassung webbasierter Authentifizierungs- und Autorisierungstechnologien

Die *HTTP-Authentifizierung* ist ein einfaches Authentifizierungsprotokoll, welches jeder moderne Webbrowser unterstützt und daher weit verbreitet ist. Die Authentifizierung regelt dabei den Zugriff auf Ressourcen am Webserver wie HTML-Dokumente oder

Grafiken. Durch die weit verbreitete, aber nicht standardisierte *Authentifizierung mittels HTML-Formularen* kann eine Zugriffsteuerung auf Web-Applikationsebene erfolgen. Damit sich Benutzer nicht direkt gegenüber Webservern authentifizieren müssen, die die Dienste beherbergen, wie dies bei der HTTP-Authentifizierung der Fall ist, wurden Protokolle wie *Kerberos* und *OpenID* entwickelt. Diese ermöglichen dem Benutzer die Authentifizierung gegenüber bekannten Authentifizierungsservern, die die Authentifizierungsinformationen auf sichere Art und Weise an die von Benutzern genutzten Webdienste und -applikationen weiterleiten. Kerberos spielt im Rahmen von über das Internet verteilte Webdienste nur eine geringe Rolle, OpenID hingegen ist bereits sehr weit verbreitet und viele Internetnutzer sind im Besitz einer sogenannten OpenID, oft ohne dies zu wissen. Allgemein wird von einem OpenID-Boom in den kommenden Jahren ausgegangen, dementsprechend ausführlich und umfangreich ist das entsprechende Kapitel 4.1.3 in dieser Einführung.

In verteilten Systemen müssen Autorisierungsinformationen zwischen verschiedenen Diensten verteilt werden. Dazu dienen komplexe und mächtige Protokolle und Standards wie *SAML*, *Shibboleth* oder *WS-Federation*, welche durch entsprechende Regeln den Zugriff auf Ressourcen genau steuern können. So ist es etwa möglich, den Zugriff auf gewisse Ressourcen nur zu bestimmten Uhrzeiten zu erlauben. Die Spezifikationen dieser Protokolle sind umfangreich und entsprechend ist der Einarbeitungsaufwand hoch. Das relativ junge Autorisierungsprotokoll *OAuth* bietet keine derartigen Regeln sondern fokussiert auf die Autorisierung zwischen Webdiensten, dementsprechend kurz sind die Spezifikationen, was eine schnelle Einarbeitung ermöglicht. Da im Rahmen der prototypischen Implementierung (siehe Kapitel 8) das OAuth-Protokoll genutzt wird, nimmt das Kapitel 4.2.1 über OAuth den meisten Platz unter der Beschreibung Autorisierungsprotokolle für verteilte Dienste ein.

# 5

---

## Verteilte Geoinformationssysteme

Moderne Geoinformationssysteme sind meist verteilte webbasierte Systeme und bestehen aus vielen Komponenten und Diensten. Sie geben ihre Daten über standardisierte Webservice Schnittstellen preis. Im Folgenden werden die am häufigsten verwendeten Geoinformationsdienste vorgestellt.

### 5.1 Webbasierte Geoinformationsdienste

Sämtliche hier vorgestellten Geoinformationssystem (GIS)-Dienste wurden vom Open Geospatial Consortium (OGC) spezifiziert. Das OGC beschreibt in den einzelnen Standards die Parameter der Anfrage sowie das jeweilige Format der Antwort.

#### 5.1.1 Informationen als Grafik vom Web Map Services

Ein Web Map Service (WMS), implementiert von einem Kartenserver, liefert als Antwort auf eine HTTP GET oder POST Anfrage einen Kartenausschnitt in einem gerasterten Grafikformat wie einem JPEG-Bild oder Vektordaten im Scalable Vector Graphics (SVG)-Format zurück [10]. Abbildung 5.1 skizziert die Nutzung eines WMS-Dienstes. Ein WMS liefert, im Gegensatz zum Web Feature Service (WFS), keine direkten Geometriedaten, sondern deren visuelle Präsentation [10]. Insofern beinhaltet das Vektorformat SVG keine detaillierten Geometriedaten, sondern nur so viele Informationen, dass das darstellende System das Bild entsprechend der Antwort rastern kann.

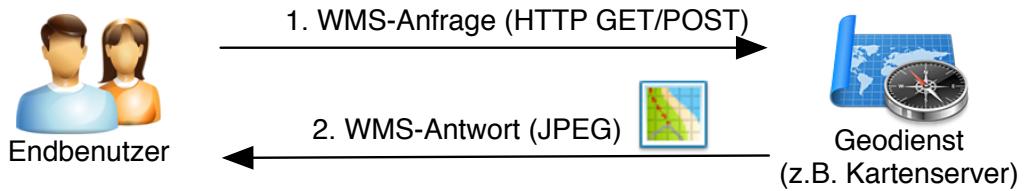


Abbildung 5.1: Einfacher Ablauf bei direkter Nutzung eines webbasierten Geodienstes: HTTP Anfrage an eine WMS-Schnittstelle und Antwort im Bildformat.

Der Nutzer des Dienstes kann verschiedene Anfrageparameter wie verwendete Kartenebene und deren Darstellungseigenschaften, das Grafikformat und die Größe des Ergebnisbildes, den Kartenausschnitt und das zugrunde liegende Koordinatensystem sowie weitere Parameter angeben [10].

Abbildung 5.2 zeigt das Ergebnis der Anfrage `http://localhost/geoserver/wms?bbox=-130,24,-66,50&Format=image/png&request=GetMap&layers=topp:states&width=550&height=250&srs=EPSG:4326`. Hierbei liefert das System entsprechend der Anfrage ein PNG-Bild mit einer Breite von 550 Pixel und einer Höhe von 250 Pixel als Antwort aus. Der Kartenausschnitt wird vom Koordinatensystem EPSG:4326 und den Koordinaten  $-130, 24, -66, 50$  bestimmt. Als Kartenebene wurde der Layer mit dem Namen `topp:states` ausgewählt. Mit dem Befehl `GetMap` hat der Aufrufer angegeben, dass er den entsprechenden Kartenausschnitt als Antwort erhalten möchte [10].

Um herauszufinden, welche Kartenebene und weitere Attribute verfügbar sind, kann der Dienst auch mit dem Befehl `getCapabilities` aufgerufen werden (`http://localhost/geoserver/wms?request=getCapabilities`). Als Ergebnis erhält der Aufrufer ein XML-Dokument mit einer Beschreibung der WMS-Fähigkeiten des Kartenservers [10].

Die hier dargestellte Representational State Transfer (REST)-Anfrage im HTTP-GET Format kann alternativ auch als HTTP-POST-Anfrage gestellt werden, wobei die einzelnen Parameter in diesem Fall in Form einer XML-Struktur übergeben werden müssen.

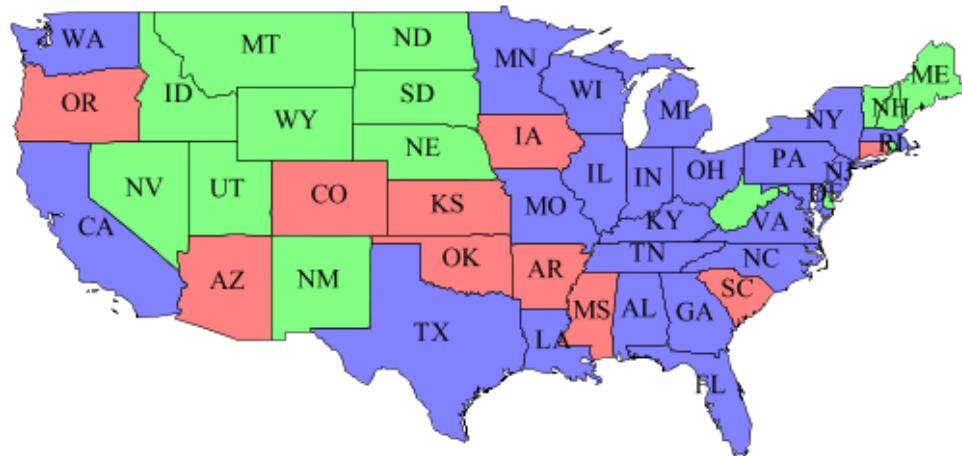


Abbildung 5.2: Gerasterte Grafik als Ergebnis einer einfachen WMS-Anfrage.

Wie die einzelnen Kartenserver die Datenhaltung intern organisieren, also die Daten als Rasterdaten speichern und ausliefern oder Vektordaten zur Laufzeit rastern, ist für den Nutzer der WMS-Schnittstelle unerheblich. Ziel des OGC war es, eine system- und herstellerunabhängige Schnittstelle zu schaffen [73].

### 5.1.2 Geografische Vektordaten vom Web Feature Services

Ein Web Feature Service (WFS) liefert über seine standardisierte Schnittstelle sogenannte Features im XML-basierten Geography Markup Language (GML)-Format aus [78]. Diese Features repräsentieren dabei geometrische Strukturen wie Flüsse, Straßen, Bundesländer und ähnliches und können mit weiteren Attributen, zum Beispiel der Einwohneranzahl, angereichert sein [78]. Durch diese zusätzlichen Daten und genaue Beschreibung der geometrischen Strukturen im Vektorformat kann ein Konsument einer WFS-Schnittstelle die Daten in der Regel sehr viel besser weiterverarbeiten, als dies bei WMS-Anfragen möglich wäre [78].

Ein Beispiel: Die WFS-Anfrage `http://localhost/geoserver/wfs?request=GetFeature&version=1.1.0&typeName=topp:states&propertyName=STATE_NAME,PERSONS&bbox=-75.102613,40.212597,-72.361859,41.512517,EPNG:4326` führt zur Ausgabe der Bevölkerungsanzahl sowie der Namen der Staaten, wie in Listing 5.1 zu sehen ist. (Geometrische Informationen zum Verlauf der

Staatsgrenzen wurden hier aus Gründen der Übersichtlichkeit nicht abgefragt. Dieselbe Anfrage, um Geometriedaten ergänzt, ist im Anhang als Listing B.1 zu finden.)

Aufgrund des Ergebnisses im Vektorformat ist es nun einfach möglich, alle amerikanischen Staaten mit einer Bevölkerungsanzahl unter 5.000.000 Einwohnern festzustellen (in diesem Beispiel Connecticut, siehe Zeile 12 in Listing 5.1 ).

**Listing 5.1: XML-Struktur als Ergebnis einer einfachen WFS-Anfrage**

---

```

1 <wfs:FeatureCollection numberOfFeatures="4" timeStamp="2010-06-03T21:46:26.683+02:00" xsi:>
  schemaLocation="http://www.openplans.org/topp http://localhost/geoserver/wfs?service=,
    WFS&version=1.1.0&request=DescribeFeatureType&typeName=topp:states http ,
    ://www.opengis.net/wfs http://localhost/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc ,
    ="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http ,
    ://www.w3.org/2001/XMLSchema-instance" xmlns:topp="http://www.openplans.org/topp" ,
    xmlns:tiger="http://www.census.gov" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns: ,
    ows="http://www.opengis.net/ows" xmlns:wfs="http://www.opengis.net/wfs">

```

```

3   <gml:featureMembers>
4     <topp:states gml:id="states.39">
5       <topp:STATE\_NAME>New York</topp:STATE\_NAME>
6         <topp:PERSONS>1.8235907E7</topp:PERSONS></topp:states>
7       <topp:states gml:id="states.40">
8         <topp:STATE\_NAME>Pennsylvania</topp:STATE\_NAME>
9           <topp:PERSONS>1.1881643E7</topp:PERSONS></topp:states>
10      <topp:states gml:id="states.41">
11        <topp:STATE\_NAME>Connecticut</topp:STATE\_NAME>
12          <topp:PERSONS>3287116.0</topp:PERSONS>
13        </topp:states><topp:states gml:id="states.43">
14          <topp:STATE\_NAME>New Jersey</topp:STATE\_NAME>
15            <topp:PERSONS>7484736.0</topp:PERSONS>
16          </topp:states>
17    </gml:featureMembers>
18  </wfs:FeatureCollection>

```

---

### 5.1.3 Datenpflege per Transactional Web Feature Service

Transactional Web Feature Service (WFS-T) stellt eine Erweiterung der WFS-Schnittstelle dar. Über sie werden schreibende Zugriffe auf das Kartenmaterial spezifiziert. Dabei können Features neu erstellt, geändert oder auch komplett gelöscht werden [78]. Über die spezielle LockFeature-Operation können einzelne Features mit einem Schreibschutz versehen werden [78].

### 5.1.4 Raum- und Zeitdimensionen durch den Web Coverage Service

Der Web Coverage Service (WCS) liefert, wie der WMS, Rasterdaten aus, jedoch angereichert um detaillierte Beschreibungen und unter Bewahrung der ursprünglichen Semantik in einem raumbasierten Datenmodell, welches bis zu drei Raumdimensionen und eine Zeitdimension umfassen kann [80]. Dazu liefert der WCS die Daten in speziellen Formaten wie Geo Tagged Image File Format (GeoTIFF)<sup>1</sup>, Digital Terrain Elevation Data (DTED) oder GML aus [80] [21]. Während ein WMS also fertiges Kartennmaterial im Rasterformat ausliefert, stellt ein WCS das Ausgangsmaterial dafür bereit [57].

### 5.1.5 Online Datenverarbeitung mit dem Web Processing Service

Web Processing Service (WPS) stellt einen Standard dar, um geospatale Daten online und über definierte Schnittstellen zu verarbeiten [64]. Unter Verarbeitung versteht man dabei die Berechnung von Fakten aufgrund der geospatialen Daten sowie das Ausführen verschiedener Algorithmen auf Basis dieser Daten [64]. So kann man mit einem WPS beispielsweise die Anzahl der an Grippe erkrankten Personen zweier Jahre in einer Region vergleichen oder Daten, die ein Straßennetz repräsentieren, glätten, um damit einfachere Berechnungen durchzuführen. Abbildung 5.3 zeigt in grüner Farbe einen Ausschnitt aus dem galizischen Straßennetz (Spanien) und in roter Farbe die geglättete Version desselben Straßennetzes.

Ziel dieser Glättung könnte die schnellere Weiterverarbeitung der Daten sein, denn durch einfachere Polygone können anschließende Berechnungen mit weniger Aufwand ausgeführt werden. In Abbildung 5.4 wurde das geglättete Straßennetz mit den Informationen eines Waldbrandes überlagert. Um die betroffenen Straßenabschnitte automatisch identifizieren zu können, zum Beispiel um der Feuerwehr die Informationen für die Brandbekämpfung zukommen zu lassen, wurden mittels WPS die Vereinigung des

---

<sup>1</sup>GeoTIFF ist eine Erweiterung des Tagged Image File Format (TIFF)-Dateiformats um georeferenzierte Daten und kann in der Regel von moderner GIS-Software verarbeitet werden [57].

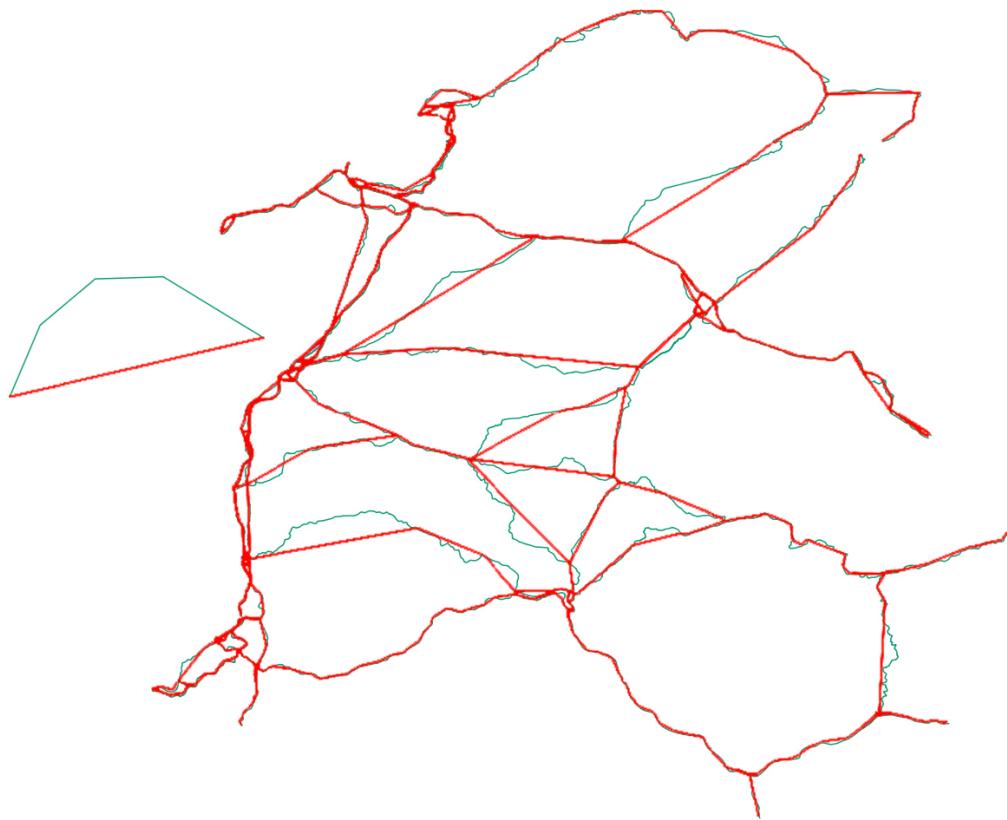


Abbildung 5.3: Teil des galizischen Straßennetzes (Spanien) nach [63]. Die dünne grüne Linie repräsentiert das tatsächliche Straßennetz, die rote Linie eine geglättete Vereinfachung des Netzes.

Straßennetzes mit dem Brand gebildet und in Abbildung 5.5 blau dargestellt [63]. In der Praxis können dabei weitere geospatiale Parameter wie Windrichtung, Trockenheit, Vegetation usw. berücksichtigt werden, um eine präzisere Vorhersage zu treffen.

Anders, als bei den zuvor vorgestellten OGC Webservices, verwendet der Anwender bei der Benutzung von WPS nicht nur diesen Service selbst, sondern indirekt gegebenenfalls weitere Services, aufgrund dessen der WPS seine Berechnungen anstellen kann (Abbildung 5.6) [64].

WPS Dienste können sowohl Vektordaten als auch gerasterte Daten verarbeiten [64].

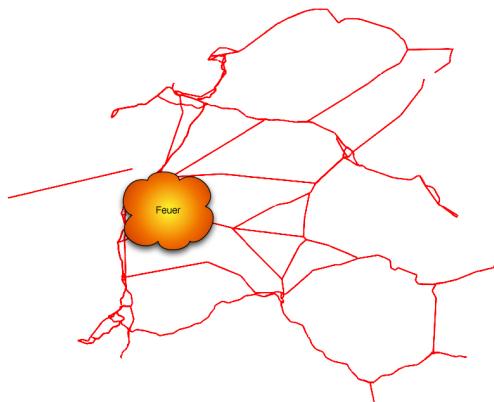


Abbildung 5.4: WMS-Darstellung mit zwei Kartenebenen. Geglätteten Straßennetzes in der Hintergrundebene, überlagert von einer einen Waldbrand darstellenden Ebene.

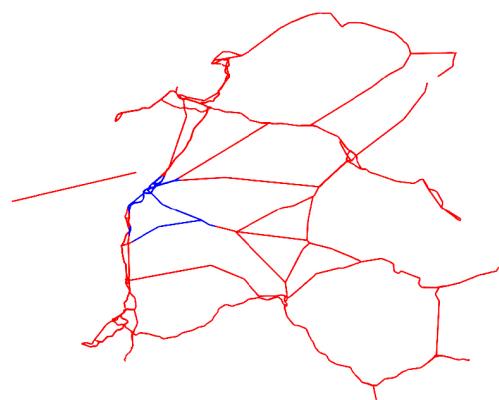


Abbildung 5.5: Ergebnis der WPS-Anfrage zur Berechnung der brennenden Straßenabschnitte, welche in blauer Farbe dargestellt sind.

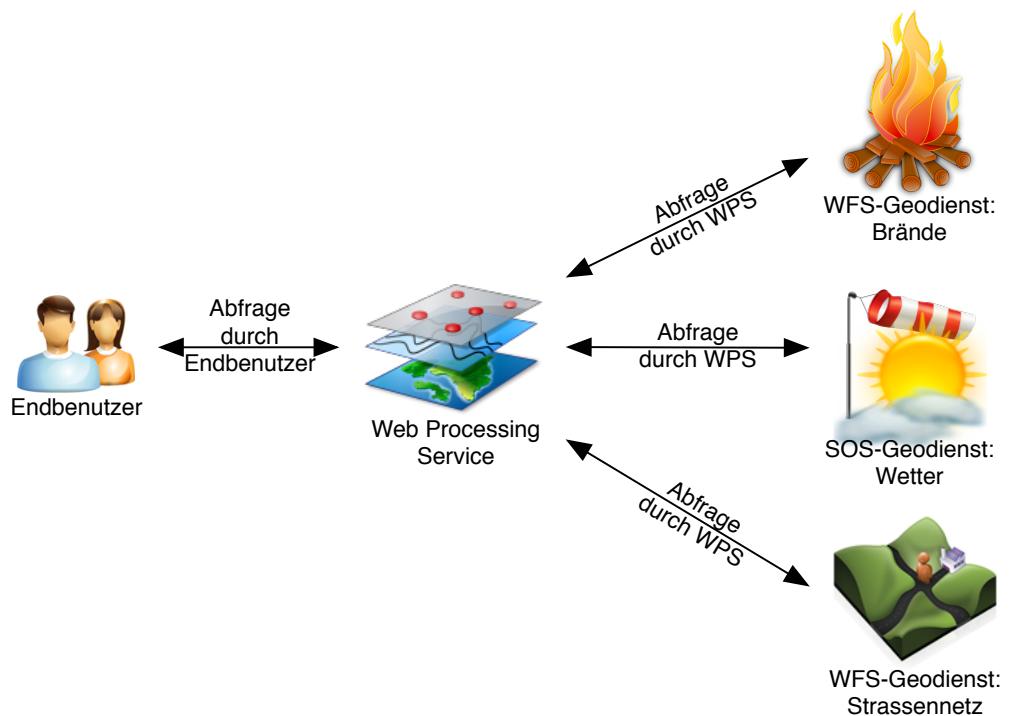


Abbildung 5.6: Ein WPS nutzt andere Geoinformationsdienste als Basis für seine Berechnungen.



Abbildung 5.7: „Sun SPOT“ als Beispiel eines Embedded Devices. Weitere Informationen unter <http://www.sunspotworld.com>. Abbildung kopiert von [http://blogs.sun.com/cparty/entry/tom\\_presents\\_a\\_talk\\_about](http://blogs.sun.com/cparty/entry/tom_presents_a_talk_about).

### 5.1.6 Sensordaten aus dem Sensor Observation Service und Sensor Alert Service

Der *Sensor Observation Service (SOS)* wird eingesetzt, um Sensoren über definierte Schnittstellen abzufragen [51]. Diese Sensoren messen Werte wie Temperatur, Niederschlag, Windstärke und -richtung, Luftverschmutzung und vieles mehr und werden in der Praxis oftmals in Form von Embedded Devices realisiert [68] (siehe Abbildung 5.7). Die Sensoren können dabei neben dem derzeitig gemessenen Wert auch historische Werte zurückliefern, um so etwa die Temperatur vor einer Stunde oder vor einem Tag abrufen zu können [51]. Da diese Sensoren oft, wie eingangs erwähnt, auf limitierter Hardware laufen, können Messwerte nur bis zu einem gewissen Umfang am Sensor selbst gespeichert werden und müssen von diesem von Zeit zu Zeit gelöscht werden.

Über einen weiteren Standard, den *Sensor Alert Service (SAS)* können die Sensoren Alarm schlagen, wenn bestimmte Grenzwerte überschritten werden [51]. Steigt der Pegel eines Flusses über einen vordefinierten Wert, könnten Abonnenten des SAS darüber automatisch benachrichtigt werden.

Eine Anfrage zur Lufttemperatur eines bestimmten Sensors liefert dabei ein umfangreiches XML-Dokument mit den genauen Messwerten und vielen Metadaten zurück, wie Listing 5.2 zeigt. Die tatsächlichen Messdaten – 13,3 Grad Celsius um 19:29 und 13,1 Grad von 19:49 bis 20:19 – sind erst auf Zeile 86 zu finden. Die Zeilen 51 bis 84 beschreiben, welche Werte pro Messung zurückgegeben werden - in diesem Beispiel den

Sensorname, Uhrzeit und Datum, Längen- und Breitengrad, Höhe über dem Meeresspiegel und die gemessenen Grad Celsius Lufttemperatur.

Über verschiedene Filterparameter könnte man nun die Werte einer bestimmten Zeitspanne auslesen oder nur Werte, die zum Beispiel 20 Grad Celsius überschreiten und ähnliches [77].

Listing 5.2: Lufttemperatur als SOS-Antwort im XML Format auf die Anfrage `http://sensor.tld/sos?request=GetObservation&offering=Bad_Ischl&observedProperty=AirTemperature`

---

```

<om:ObservationCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:om="http://www.opengis.net/om/1.0" xmlns:gml="http://www.opengis.net/gml" xmlns:swe="http://www.opengis.net/swe/1.0.1" xsi:schemaLocation="http://www.opengis.net/om/1.0 http://schemas.opengis.net/om/1.0.0/observation.xsd" gml:id="RSA\_sensor">
  <gml:description>AIRTEMPERATURE measurements from RSA Bad\Ischl</gml:description>
  <gml:name>AIRTEMPERATURE measurements from RSA Bad\Ischl</gml:name>
  22  <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.5:4326">
      24    <gml:lowerCorner>12.1 46.547.71667 13.63278 0</gml:lowerCorner>
      <gml:upperCorner>13.5 49.1047.71667 13.63278 0</gml:upperCorner>
    26  </gml:Envelope>
    </gml:boundedBy>
  28  <om:member>
    <om:Observation gml:id="RSA\_Bad\Ischl">
      <gml:description>AIRTEMPERATURE measurements from RSA Bad\Ischl</gml:>
      30      description>
      <gml:name>AIRTEMPERATURE measurements from RSA Bad\Ischl</gml:name>
      <gml:boundedBy>
        <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.5:4326">
          34        <gml:lowerCorner>47.71667 13.63278 0</gml:lowerCorner>
        <gml:upperCorner>47.71667 13.63278 0</gml:upperCorner>
      36        </gml:Envelope>
        </gml:boundedBy>
      38        <om:samplingTime><gml:TimePeriod><gml:beginPosition>2010-06-18T19:29:00Z</gml:>
        beginPosition><gml:endPosition>2010-06-18T20:19:00Z</gml:endPosition></gml:>
        TimePeriod></om:samplingTime>
        <om:procedure xlink:href="urn:iSPACE.org:source.postgres\_OOE\#Bad\Ischl"/>
      40        <om:observedProperty>
        <swe:CompositePhenomenon dimension="1" gml:id="WEATHER\_OBSERVABLES">
          <gml:name>Weather Station Observables</gml:name>
          <swe:component xlink:href="urn:x-ogc:def:phenomenon:OGC:AirTemperature"/></swe:>
          CompositePhenomenon>
        </om:observedProperty>
      44        <om:featureOfInterest xlink:href="urn:iSPACE:Air"/>
    
```

```
46    <om:result>
47        <swe:DataArray>
48            <swe:elementCount>
49                <swe:Count>
50                    <swe:value>6</swe:value>
51                </swe:Count>
52            </swe:elementCount>
53            <swe:elementType name="Bad\_IschlObservations">
54                <swe:DataRecord>
55                    <swe:field name="PlatformName">
56                        <swe:Quantity definition="urn:mmisw.org\#platform"/>
57                    </swe:field>
58                    <swe:field name="time">
59                        <swe:Time definition="urn:ogc:phenomenon:time:iso8601"/>
60                    </swe:field>
61                    <swe:field name="latitude">
62                        <swe:Quantity definition="urn:ogc:phenomenon:latitude">
63                            <swe:uom code="deg"/>
64                        </swe:Quantity>
65                    </swe:field>
66                    <swe:field name="longitude">
67                        <swe:Quantity definition="urn:ogc:phenomenon:longitude">
68                            <swe:uom code="deg"/>
69                        </swe:Quantity>
70                    </swe:field>
71                    <swe:field name="altitude">
72                        <swe:Quantity definition="http://mmisw.org/cf\#altitude" referenceFrame="urn:ogc:def:crs:EPSG:6.15:5778">
73                            <swe:uom code="m"/>
74                        </swe:Quantity>
75                    </swe:field>
76                    <swe:field name="observedProperty1">
77                        <swe:Quantity definition="urn:x-ogc:def:phenomenon:OGC:\#AirTemperature">
78                            <swe:uom code="degC"/>
79                        </swe:Quantity>
80                    </swe:field></swe:DataRecord>
81                </swe:elementType>
82                <swe:encoding>
83                    <swe:TextBlock blockSeparator=" " decimalSeparator="." tokenSeparator=","/>
84                </swe:encoding>
85                <swe:values>
86                    Bad\_Ischl,2010-06-18T19:29:00Z,47.71667,13.63278,460,13.3
87                    Bad\_Ischl,2010-06-18T19:39:00Z,47.71667,13.63278,460,13.1
88                    Bad\_Ischl,2010-06-18T19:49:00Z,47.71667,13.63278,460,13.1
89                    Bad\_Ischl,2010-06-18T19:59:00Z,47.71667,13.63278,460,13.1
90                    Bad\_Ischl,2010-06-18T20:09:00Z,47.71667,13.63278,460,13.1
```

---

```

Bad\Ischl,2010-06-18T20:19:00Z,47.71667,13.63278,460,13.1
92      </swe:values>
        </swe:DataArray>
94    </om:result>
      </om:Observation>
96  </om:member>
</om:ObservationCollection>

```

---

## 5.2 Kartenausschnitt per Bounding Box

Eine Bounding Box beschreibt einen Ausschnitt aus einer Karte und ist in der einfachsten Form ein Rechteck, kann aber auch komplexere Formen wie umfangreiche Polygonzüge annehmen [10]. Bounding Boxes werden sowohl bei der Anfrage durch den Endbenutzer als auch zur serverseitigen Zugriffsteuerung verwendet [50].

Bei einer WMS-Anfrage spezifiziert der Endbenutzer den Kartenausschnitt, den er als Antwort erhalten möchte, indem er eine Bounding Box in der WMS-Anfrage übergibt [10]. Um die Karte vom Bundesland Salzburg als Antwort zu erhalten, spezifiziert er ein Rechteck mit entsprechender Größe, welches Salzburg umschließt. Dass dabei auch Bereiche aus anderen Gebieten wie Tirol, Kärnten, Oberösterreich und Bayern in der Antwort enthalten sind, da sie Teil der Bounding Box sind, stören den Endbenutzer in der Regel nicht.

Anders verhält es sich bei der Zugriffsteuerung per Bounding Box. Bekommt ein Kunde per Vertrag nur Daten von Österreich, wäre eine den Zugriff steuernde Bounding Box wie in Abbildung 5.8, vermutlich zu grob. Hier eignet sich ein feinerer Polygonzug wie in Abbildung 5.9 meist besser. Jedoch ist die Berechnung des Ergebnisses um so komplizierter und aufwändiger, je feiner die Bounding Box definiert ist [50]. Hier ist in der Praxis ein Mittelweg zwischen Datensicherheit und Berechenbarkeit zu wählen.

Als Ergebnis erhält der Benutzer die Vereinigung der anfragenden Bounding Box und der den Zugriff steuernden Bounding Box zurück. Im Falle einer WMS-Anfrage bleiben die Bereiche, auf die der Benutzer keinen Zugriff hat, einfach leer. In Abbildung 5.10 ist der leere Bereich zur besseren Visualisierung rot gefärbt.

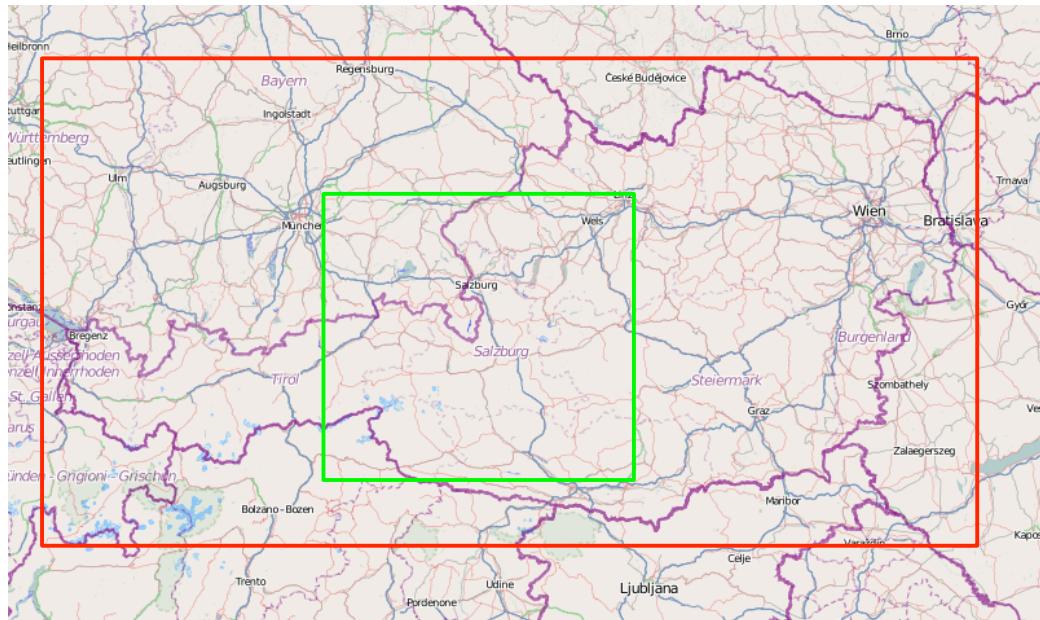


Abbildung 5.8: Abfrage der grünen, das Bundesland Salzburg umschließenden Bounding Box innerhalb des erlaubten, roten Bereiches.

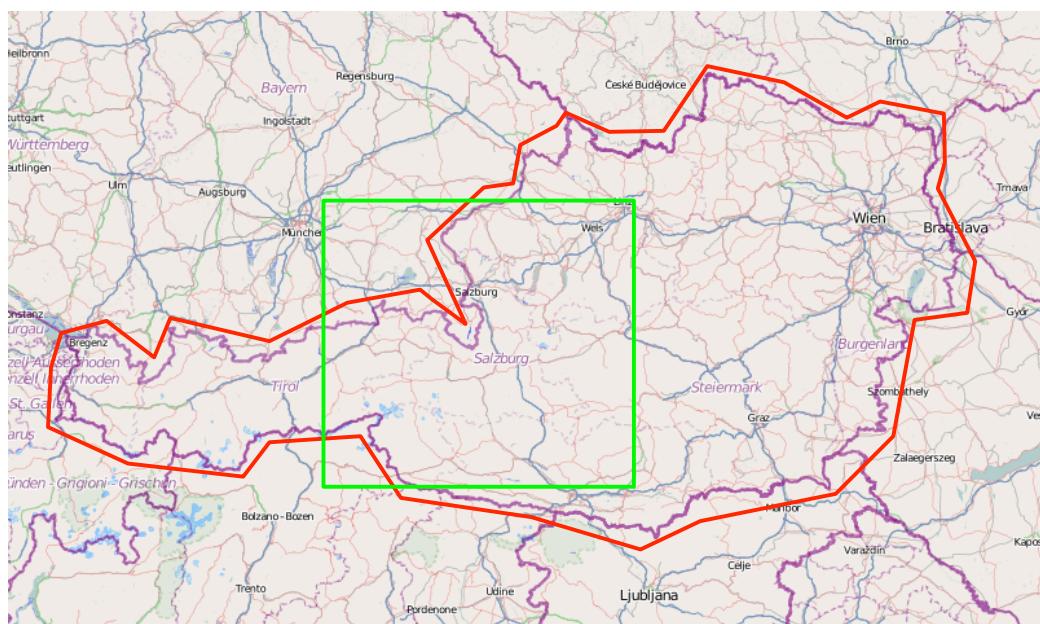


Abbildung 5.9: Gleiche Abfrage des Bundeslandes Salzburg wie in Abbildung 5.8, jedoch innerhalb einer komplexeren, den Zugriff steuernden Bounding Box.

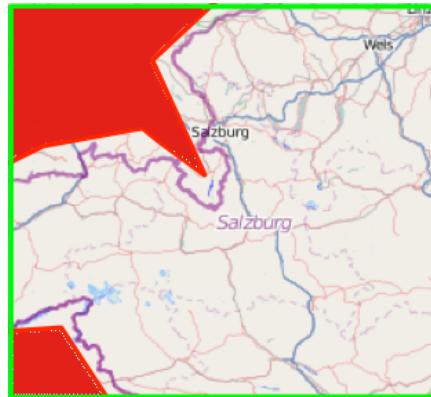


Abbildung 5.10: Ergebnis einer WMS-Anfrage, wobei die Bounding Box der Anfrage die Bounding Box der erlaubten Regionen überlappt. Rot hervorgehobene Bereiche bleiben bei der WMS-Antwort leer.

### 5.3 Kartenaufbau aus einzelnen Ebenen

Anders als ausgedruckte Landkarten, wo Berge, Straßen, Eisenbahnlinien, Flüsse und so weiter auf einem Blatt Papier gedruckt sind und nur gemeinsam betrachtet werden können, sind die Informationen in digitalen Karten in sogenannten Kartenebenen, auch Kartenlayer genannt, aufgeteilt [82] (Abbildung 5.11). Die Ebenen können einzeln eingeblendet und beliebig kombiniert werden. Dadurch kann sich der Benutzer auf die Informationen in der Karte konzentrieren, die für ihn gerade wichtig sind [82].

Weiters können Kartenebenen von vielen verschiedenen Anbietern auf unterschiedlichen Servern am Client zu einer Karte kombiniert werden, sodass zum Beispiel die Landesregierung die Kartenebene der Waldbrände vom Server der Feuerwehr mit der Ebene der Gasleitungen, zur Verfügung gestellt vom Energieversorger, mit einer Ebene der Krankenhäuser, abgerufen von einem internen Server, kombinieren kann, um Evakuierungspläne auszuarbeiten.

Dabei können Rasterdaten von WMS-Diensten mit Vektordaten aus WFS-Diensten, die jeweils in eigenen Kartenebenen abgebildet sind, zu einer Gesamtkarte vereint werden [82].



Abbildung 5.11: Karten werden aus verschiedenen Ebenen aufgebaut. 1: Straßennetz von Manhattan. 2: Landmarks (Charakteristika der Landschaft) von Manhattan. 3: Ausgewählte Sehenswürdigkeiten. 1+2+3: Zusammengesetzte Karte mit allen Ebenen.

## 5.4 Schützenswerte Aspekte von Geodaten

Nachdem der Endbenutzer authentifiziert und autorisiert wurde, kann er auf die einzelnen Geoinformationsdienste zugreifen. Durch die Autorisierung wird bestimmt, welche der verfügbaren Daten er abrufen darf. Diese Einschränkungen können von verschiedenen Parametern bzw. einer Kombination dieser Parameter abhängig sein [44].

### **Einschränkung des Zugriffs auf bestimmte Kartenausschnitte durch Bounding Boxes**

Durch die Einschränkung auf gewisse Bounding Boxes bekommt der Endbenutzer nur Zugriff zu gewissen geografischen Regionen [50]. Die Bounding Boxes können dabei einfache Rechtecke oder komplizierte Polygone darstellen, um so etwa Landesgrenzen zu modellieren [50]. Hat ein Endbenutzer durch die Autorisierung das Recht erlangt, Daten aus dem Bundesland Salzburg abzurufen, so bekommt er nur Informationen aus dem Bundesland Salzburg, auch wenn er Daten aus einer größeren Region wie Mitteleuropa abruft.

### **Einschränkung des Zugriffs auf bestimmte Kartenebenen**

Ein Geoinformationsdienst kann vielfältige, detaillierte und teilweise exklusive Daten wie Standorte von Atomkraftwerken, Militärbasen oder auch Wohngegenden vermögender Bevölkerungsschichten in einer einzigen Datenbank speichern und bei Bedarf ausliefern. Die Geodaten werden dabei in der internen Datenbank in einzelnen Kartenebenen organisiert. Durch die Autorisierung wird gewährleistet, dass der Benutzer nur die Kartenebenen und somit ausschließlich die Informationen erhält, für die er autorisiert wurde.

### **Einschränkung des Zugriffs auf Daten gewisser Genauigkeit**

Der Wert von Kartenmaterial richtet sich ganz erheblich nach der Exaktheit der Daten. Es liegt auf der Hand, dass eine Straßenkarte, die auf 1 Meter genau ist, mehr Wert hat als eine Karte, die nur eine Genauigkeit von 100 Metern aufweist. Die verschiedenen Genauigkeitsgrade werden heute in verschiedenen Kartenebenen abgebildet, sodass der Benutzer für die entsprechenden Kartenebenen autorisiert werden muss.

### **Einschränkung des Zugriffs auf Daten gewisser Aktualität**

Der Wert einer Karte richtet sich sehr oft auch nach dem Alter der Daten. Aktuelle Daten zur Bevölkerungsdichte und Autobahnanschlüssen eignen sich besser, um den

optimalen Standort eines neuen Shopping Centers zu finden, als Material, das 15 Jahre alt ist. Dementsprechend ist für neueres Kartenmaterial in der Regel mehr zu bezahlen. Eine Ausnahme bilden historische Karten, die oft auch von erheblichem Wert sind. Es ist üblich, dass die Aktualität der Informationen in verschiedenen Kartenebenen abgebildet ist, sodass der Benutzer sich für die korrekten Kartenebenen autorisieren muss.

### **Einschränkung des Zugriffs auf Daten gewisser Zeitspannen**

Im Kontext von SOS-Daten ist die Zeitspanne, auf die der Endbenutzer zugreifen darf, ein wichtiges Merkmal. Temperaturdaten vom heutigen Tag bis ein Jahr zurückreichend sind mehr wert als Temperaturdaten desselben Sensors der letzten beiden Wochen. Die Abfrage der Zeitspanne ist in den SOS-Spezifikationen standardisiert<sup>2</sup> [51].

### **Unterschiedliche Schreib- und Leserechte**

In den meisten Fällen werden die Informationen vom Kartenanbieter lesend abgerufen und anschließend vom Benutzer lokal weiterverarbeitet. Möchte der Kartenanbieter aber ausgewählten Benutzern auch Schreibzugriff ermöglichen, wie dies etwa für registrierte Benutzer im Communityprojekt OpenStreetMap<sup>3</sup> möglich ist, so muss der schreibende Zugriff autorisiert werden, damit nicht jeder Benutzer beliebig Daten verändern bzw. manipulieren kann [44]. Der Schreibzugriff könnte außerdem auf einzelne Kartenebenen oder Bounding Boxes eingeschränkt sein.

### **Digital Rights Management in der Geoinformatik**

Hat der Benutzer die Daten einmal abgerufen und in seinem System gespeichert, kann der Datenanbieter üblicherweise mit technischen Hilfsmitteln nicht mehr darüber entscheiden, was der Endanwender damit macht. Der Endanwender könnte alle Daten abspeichern und selbst widerrechtlich verkaufen. Mit Digital Rights Management (DRM)

---

<sup>2</sup>Standardisiert in den SOS-Spezifikationen durch den Befehl GetFeatureOfInterestTime

<sup>3</sup><http://www.openstreetmap.org>

bzw. Geospatial Digital Rights Management (GeoDRM) versucht man hier die Verwendung der Daten auch nach der Auslieferung an den Endanwender zu kontrollieren, um so technisch zu gewährleisten, dass das Kartenmaterial etwa maximal 24 Stunden lang genutzt werden kann [76]. GeoDRM ist ein junges Forschungsfeld und nicht Thema dieser Arbeit, es ist jedoch leicht vorstellbar, dass Kartenmaterial mit verschiedenen GeoDRM-Schutzmechanismen unterschiedlich viel kostet - Material, das man nur 24 Stunden nutzen kann, kostet somit weniger als dasselbe Material mit einer Nutzungsdauer von 72 Stunden. Es erscheint auf den ersten Blick sinnvoll, auch diesen Parameter in verschiedenen Kartenebenen zugänglich zu machen, aufgrund fehlender Spezifikationen wird dieser Aspekt jedoch vorerst außer Acht gelassen.

### **Komplexität und Exklusivität von Algorithmen**

Algorithmen, die im Rahmen von WPS genutzt werden, sind von verschiedener Komplexität bzw. Exklusivität und somit verschiedenen Wertes. Ein Algorithmus, der aufgrund von Straßen und Flussdaten automatisch Brücken erkennt, könnte einfacher und daher weniger wert sein als ein Algorithmus, der aufgrund von demographischen Daten den zu erwartenden Umsatz eines neuen Shopping Centers berechnet. Durch die entsprechende Autorisierung ist gewährleistet, dass der Endanwender nur die Algorithmen nutzen kann, die er auch benutzen darf. Da die Bewertung verschiedener Algorithmen nicht standardisiert ist, werden Überlegungen diesbezüglich beim Filtern und Autorisieren der WPS-Anfragen im Folgenden ignoriert.

## **5.5 Existierende Schutzmaßnahmen für Geoinformationsdienste**

Ein Zugriffsschutz der Webservice-Schnittstellen ist in den jeweiligen Standards der OGC nicht vorgesehen. Um die Daten trotzdem schützen zu können, implementieren die meisten Softwareprodukte, die WMS-, WFS-, WPS- oder SOS-Dienste anbieten, einen rudimentären Zugriffsschutz. Für den in der Praxis weit verbreiteten GeoServer<sup>4</sup>

---

<sup>4</sup><http://geoserver.org/>

kann der Administrator Benutzername, Passwort, die Kartenebenen und eine Bounding Box festlegen, auf die der Benutzer zugreifen darf [4] (siehe Kapitel 7.3). Die Authentifizierung des Benutzers geschieht dabei über HTTP-Authentifizierung [4]. Eine umfangreiche Geodienste-Infrastruktur lässt sich damit aufgrund des hohen Administrationsaufwandes kaum bewerkstelligen.

Um komplexe Geoinfrastruktur schützen zu können, gibt es kommerzielle Lösungen. Die Firma ESRI<sup>5</sup> bietet mit der Softwarefamilie ArcGIS Serverprodukte an, um unbefugte Zugriffe zu unterbinden [20] [6]. Die entsprechenden Clients desselben Herstellers ermöglichen das Einloggen am Server, welcher die Daten nur nach der erfolgreichen Authentifizierung und Autorisierung bereitstellt. Aufgrund der hohen Kosten sowie der Abhängigkeit von einem Hersteller und seinen proprietären Schnittstellen sind derartige kommerzielle Lösungen für Geodiensteanbieter nicht optimal, selbst wenn diese auf technischer Basis überzeugen können.

Im Bereich von kostenloser OpenSource-Software ist die „52°North Initiative for Geospatial Open Source Software GmbH“ mit dem „Security & Geo-Rights Management“-Modul zu nennen, die mit dem Unterprojekt 52n Web Enforcement Service (WSS) ebenfalls versucht WMS-, WFS-, WPS- oder SOS-Dienste vor unbefugtem Zugriff zu schützen [3]. Die Authentifizierung geschieht dabei entweder über HTTP-Authentifizierung oder SAML-Tickets<sup>6</sup> [3]. Die HTTP-Authentifizierung eignet sich dabei wiederum aufgrund des hohen Administrationsaufwandes nur für den Schutz einfacher Geoinfrastruktur. Die Authentifizierung über SAML-Tickets hingegen setzt eine entsprechend komplexe SAML-Infrastruktur beim Benutzer und Geodiensteanbieter voraus, was ebenfalls nicht optimal ist. Da der Quellcode von WSS relativ umfangreich und die Dokumentation nicht vollständig ist, muss man mit erhöhter Einarbeitungszeit rechnen, wenn man die Software an eigene Bedürfnisse anpassen möchte [3]. Als der Verfasser dieser Arbeit versuchte, die Quellcodes vom offiziellen Server herunterzuladen und zu kompilieren, gelang dies nicht, da einige vom Projekt benötigten Bibliotheken falsch verlinkt waren. Nach Kontaktaufnahme mit dem Team von 52°North wurde

---

<sup>5</sup><http://www.esri.com>

<sup>6</sup>Als dritte Möglichkeit kann der Benutzer über das proprietäre WSS-Protokoll authentifiziert werden.

der Fehler bestätigt und eine Korrektur zugesichert. Diese Email ist im Anhang unter C.1 zu finden. Aufgrund dieser widrigen Erfahrung und der verbesserungsfähigen Dokumentation wurde jedoch von weiteren Versuchen, die Software zu kompilieren, abgesehen.

## 5.6 Zusammenfassung verteilter Geoinformationssysteme

Verteilte Geoinformationssysteme liefern aufgrund von HTTP-Anfragen über Webservice-Schnittstellen ihre Daten im Raster- oder XML-Format aus. Das Schützen der Daten der durch die OGC spezifizierten Dienste wie *Web Map Service (WMS)*, *Web Feature Service (WFS)*, *Transactional Web Feature Service (WFS-T)*, *Web Processing Service (WPS)*, *Web Coverage Service (WCS)*, *Sensor Observation Service (SOS)* und *Sensor Alert Service (SAS)* ist dabei mit dem Einsatz von erheblichen Ressourcen in Form von kostspieliger kommerzieller Software oder der nicht trivialen Einarbeitung in OpenSource-Software verbunden. Damit der Benutzer genau die geografischen Daten erhält, an denen er interessiert ist, kann er vom Kartenserver den gewünschten Kartenausschnitt per *Bounding Box* sowie die gewünschten *Kartenebenen* angeben. Der Anbieter der geografischen Daten gliedert seine Informationen unter anderem nach *Genaugigkeit* und *Aktualität* der Daten, unterscheidet zwischen *Lese-* und *Schreibzugriffen* und gewährt den Zugriff nur, wenn der Kunde für die Daten entsprechend bezahlt hat und sein *Konto ausreichende Deckung* aufweist.

# 6

---

## Ableitung einer serviceorientierten Sicherheitsarchitektur für Geoinformationsdienste

Im Folgenden wird aufgrund der Anforderungen von GIS-Anbietern und -Kunden schrittweise eine serviceorientierte Sicherheitsarchitektur für Geoinformationsdienste entwickelt. Durch das Paradigma der „Separation of Concerns“ ergeben sich am Ende der Überlegungen klar voneinander abgetrennte Komponenten, die jeweils einen Teilaспект des Gesamtsystems erfüllen und im Summe die gesamte serviceorientierte Sicherheitsarchitektur repräsentieren.

### 6.1 Anforderungen der GIS-Anbieter und -Kunden

Geoinformationsdienste-Anbieter möchten die zur Verfügung stehenden Daten über einheitliche Webserviceschnittstellen mit ihren Kunden teilen, welche die Daten stets aktualisiert von den Anbietern abrufen können, ohne diese selbst pflegen zu müssen [47]. Um den Zugriff steuern und das Abrufen der Daten berechnen zu können, fordern Geoinformationsdiensteanbieter, unter anderem das Research Studio iSpace aus Salzburg, die Entwicklung eines Triple-A-Systems zum Schutz von Geoinformationsdiensten auf Basis von Internetprotokollen und Technologien, ohne die etablierten, auf den OGC-Spezifikationen aufbauenden Geoinformationsdienste selbst verändern zu müssen. Die

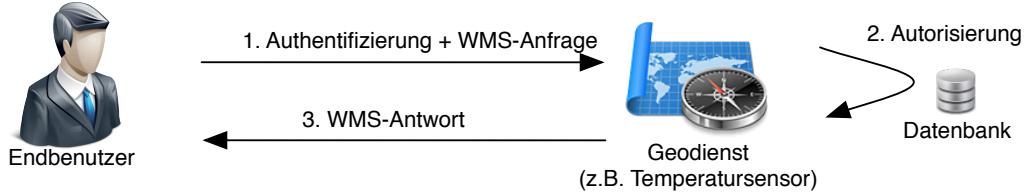


Abbildung 6.1: Anfrage an eine WMS-Schnittstelle mit Authentifizierung, Autorisierung und Abrechnung.

Autorisierung soll auf bestimmte Kartenebenen oder Kartenausschnitte begrenzt werden können, ebenso soll die Anzahl der Zugriffe für spätere Abrechnungszwecke protokolliert werden. Zuletzt sollte die Verwaltung der Benutzer-Accounts optional zu den Kunden ausgelagert werden können, um Behörden und großen Organisationen mit mehreren hundert oder tausend Benutzern eine bequeme und sichere Nutzung der Dienste zu ermöglichen.

## 6.2 Authentifizierung, Autorisierung und Abrechnung am Geodienst selbst

Am Anfang stehen dabei die Geoinformationsdienste SOS, WMS, WFS, WFS-T und WCS, welche vor unbefugtem Zugriff geschützt werden müssen. Die genannten Dienste sollen nur nach erfolgreicher Authentifizierung und entsprechender Autorisierung die angemessenen Daten ausliefern. Andererseits sollen die Zugriffe des Benutzers protokolliert werden, um eine Abrechnung im Sinne eines Triple-A-Systems zu ermöglichen. Ein vereinfachter Ablauf ist in Abbildung 6.1 dargestellt.

Dieses simple Schema ist einfach zu implementieren und könnte für Szenarien mit nur einem Dienst umgesetzt werden. In der Regel ist dieses Szenario aber zu einfach und unflexibel, da es mehrere eklatante Nachteile mit sich bringt, wie folgende Beispiele anführen.

Ein sehr großer Nachteil liegt darin, dass das Paradigma der „*Separation of Concerns*“ im Szenario von Abbildung 6.1 nicht eingehalten wird. Ein WMS-Dienst müsste neben

seiner eigentlichen Aufgabe, dem Ausliefern von Rastergrafiken, auch Authentifizierungs-, Autorisierungs- und Abrechnungsfunktionen unterstützen und diese Daten lokal speichern sowie abrufen können, in Abbildung 6.1 schematisch dargestellt durch das Datenbanksymbol.

Dies mag zwar für Geoinformationsdienste, die auf leistungsfähigen Servern laufen, möglich sein, ist aber auf Embedded Devices, wie sie oft für Sensoren verwendet werden, technisch oder wirtschaftlich nicht machbar. Würde ein Embedded Device neben dem Messen der Temperatur und dem Aussenden der Daten auch Authentifizierung per Benutzername und Passwort oder per Kerberos unterstützen, so stiege der Implementierungsaufwand auf dem Embedded Device in nicht zu rechtfertigende Höhen. Selbst wenn man diesen Aufwand betreiben würde, wäre der Energiehunger solcher Systeme zu groß, um effizient eingesetzt werden zu können.

Darüber hinaus hat man es in der Praxis meist nicht nur mit einem Sensor zu tun, sondern mit mehreren dutzend, hunderten oder gar tausenden von Sensoren. Es wäre wenig effektiv, würde jeder Sensor alle Benutzerinformationen für die Authentifizierung selbst speichern - auch der Administrationsaufwand wäre immens.

Es steht daher außer Zweifel, dass sich die Geoinformationsdienste auf ihre eigentliche Aufgabe, zum Beispiel das Messen der Temperatur, konzentrieren sollen und die Authentifizierung, Autorisierung und die Abrechnung von anderen Komponenten übernommen werden müssen.

### 6.3 Einführung eines Proxydienstes

Bei der Überlegung im nächsten Szenario, dargestellt in Abbildung 6.2, werden Authentifizierung, Autorisierung und Abrechnung von einem Proxydienst übernommen. Sobald der Benutzer autorisiert ist, leitet der Proxydienst die eigentliche Anfrage an den Geoinformationsdienst weiter. Dieser Dienst antwortet im Folgenden dem Client. Ob die Antwort direkt an den Client gesendet oder über den Proxydienst weitergeleitet wird, ist für die grundlegenden Überlegungen dieses Szenarios irrelevant und müsste bei einer technischen Umsetzung genauer diskutiert werden.

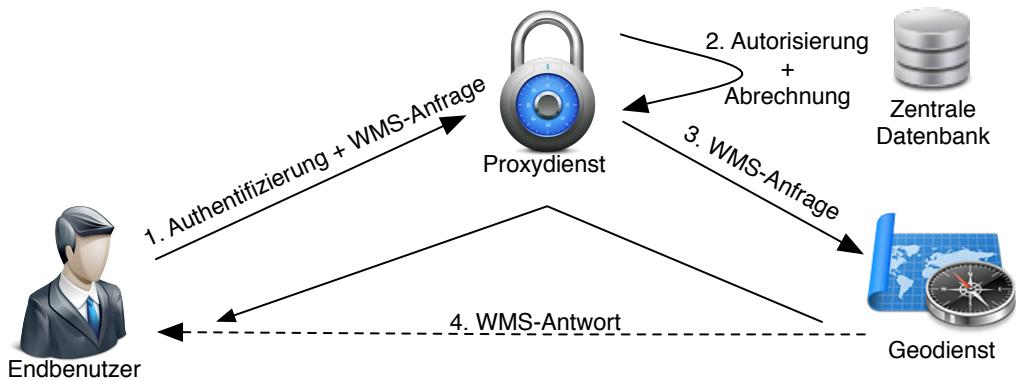


Abbildung 6.2: Anfrage an eine WMS-Schnittstelle über einen Proxydienst für Authentifizierung, Autorisierung und Abrechnung.

Der Proxydienst stellt hierbei eine separate Komponente dar, die auf einem eigenen Server im Rechenzentrum des GIS-Anbieters seinen Dienst verrichtet, wodurch Embedded Devices, sofern diese zum Einsatz kommen, entlastet werden. Bei Diensten wie WMS, WCS oder WFS, die auf leistungsfähiger Hardware gehostet werden, kann der Proxydienst auf demselben Rechner installiert werden.

Das Szenario aus Abbildung 6.2 entlastet den Geoinformationsdienst, der nun nur seine eigentliche Aufgabe, das Ausliefern von Kartenmaterial, erfüllen muss. Der Nachteil dieser Architektur zeigt sich, wenn man ein Szenario mit mehreren Diensten betrachtet, wie in Abbildung 6.3 dargestellt: Jeder Proxydienst müsste eine lokale Datenbank für die Authentifizierung, Autorisierung und die Abrechnung führen.

Der Administrationsaufwand hierfür wäre enorm und ist daher wohl nur in Ausnahmefällen oder beim Einsatz sehr weniger Benutzer und Dienste praktikabel.

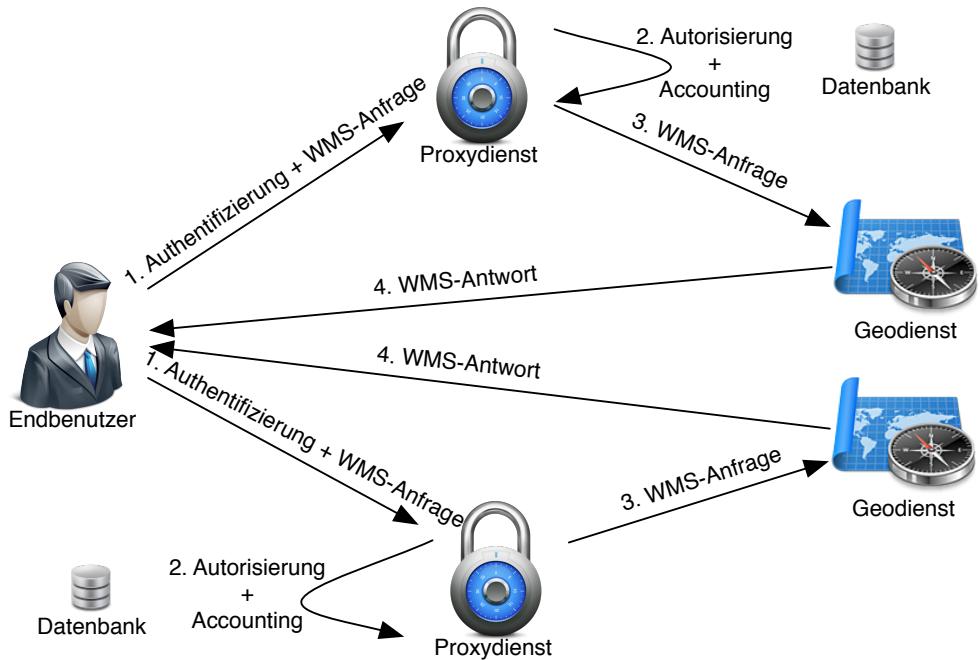


Abbildung 6.3: Anfrage an zwei WMS-Schnittstellen und doppelten Benutzerdatenbanken.

## 6.4 Einführung einer zentralen Datenbank für Triple-A-Systeme

Besser als das Führen einer separaten Datenbank für jeden Proxydienst ist daher der Einsatz einer zentralen Datenbank, wo sämtliche Werte, welche für die Authentifizierung, Autorisierung und Abrechnung notwendig sind, gespeichert werden. Schematisch dargestellt ist dies in Abbildung 6.4.

Im gegenwärtigen Szenario müsste jeder einzelne Proxydienst die Authentifizierung und Autorisierung selbst durchführen, kann dabei jedoch auf die Daten der zentralen Datenbank zugreifen.

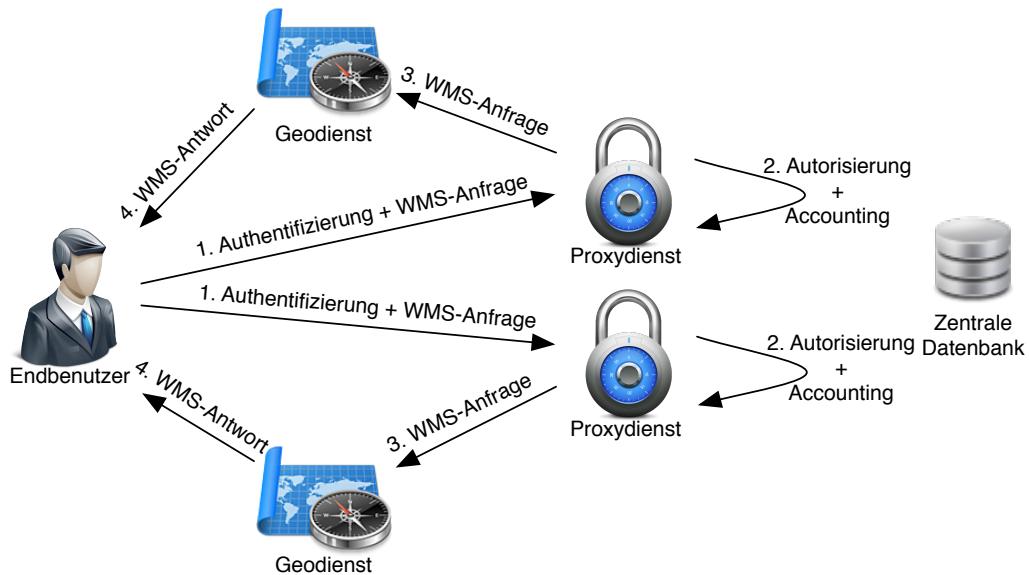


Abbildung 6.4: Zentrale Datenbank für Authentifizierung, Autorisierung und Abrechnung.

## 6.5 Einführung eines Authentifizierungs-, Autorisierungs- und Abrechnungsdienstes

Im nächsten Abstraktionsschritt wird der Zugriff auf die zentrale Datenbank in Form eines Webservices realisiert, welcher die Authentifizierung, Autorisierung und Abrechnung über Webprotokolle für die einzelnen Proxydienste zur Verfügung stellt. Der Proxydienst muss dazu an diesen zentralen Triple-A-Dienst, im Folgenden Authentifizierungs-, Autorisierungs- und Abrechnungsdienst (AAAD) genannt, die Authentifizierungsinformationen des Benutzers sowie die vom Benutzer gestellte Anfrage weiterleiten und erhält von diesem die Autorisierungsentscheidung, ob der Benutzer zugreifen darf oder die Anfrage abgelehnt werden muss. Der gesamte Entscheidungsprozess findet zentral am AAAD statt. Wie der AAAD diese Entscheidung trifft, ist für den Proxydienst irrelevant. Der Ablauf ist, aus Gründen der Übersichtlichkeit vorerst nur mit einem Dienst, in Abbildung 6.5 dargestellt.

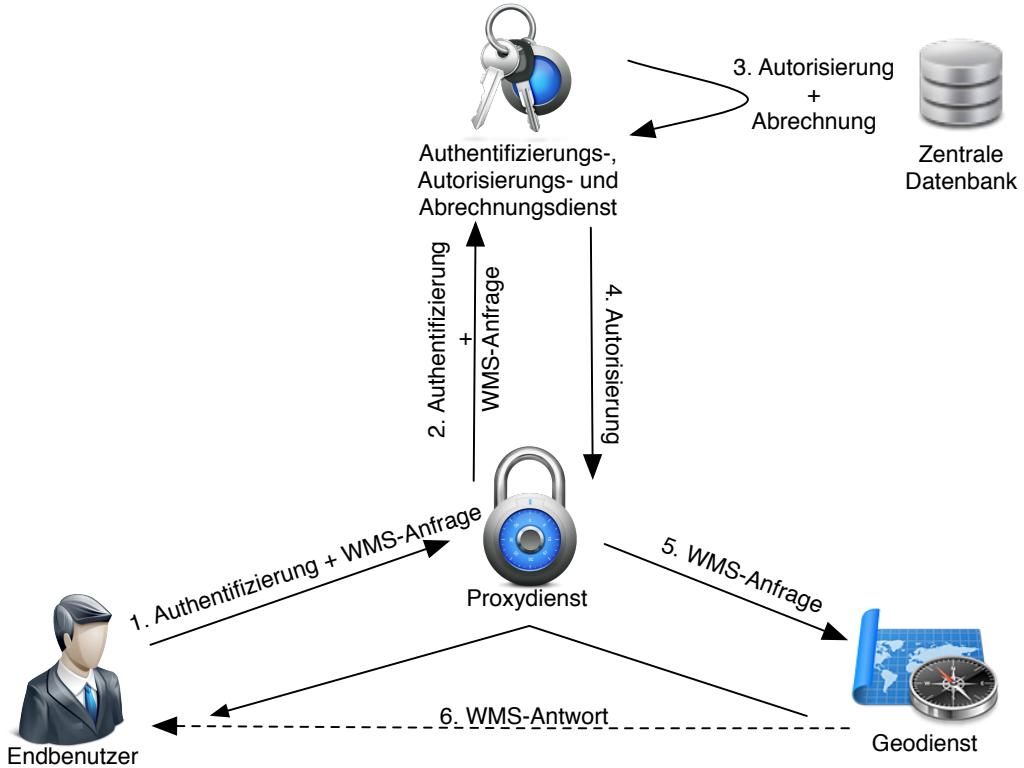


Abbildung 6.5: Serviceorientierte Systemarchitektur mit zentralem Authentifizierungs-, Autorisierungs- und Abrechnungsdienst für Webservices.

Im Szenario aus Abbildung 6.5 kann sich der Geoinformationsdienst seiner eigentlichen Aufgabe, dem Ausliefern von Kartenmaterial, widmen. Geschützt wird der Geoinformationsdienst durch den Proxydienst. Die Authentifizierung, Autorisierung und Protokollierung zur Abrechnung übernimmt vollständig der AAAD. Der Proxydienst kann durch die Auslagerung von Authentifizierung, Autorisierung und Abrechnung einfach gehalten und dadurch im Normalfall auch auf Embedded Devices implementiert werden, da der Proxydienst nur die Anfrage des Endbenutzers abfangen und an den AAAD weiterleiten muss. Der AAAD entscheidet aufgrund verschiedener Kriterien, zum Beispiel korrektem Benutzername und Passwort, ob er die Anfrage autorisiert. Seine Entscheidung teilt der AAAD dem Proxydienst mit. Im einfachsten Fall übermittelt der AAAD ein einfaches „Ja, zulassen“ bzw. „Nein, ablehnen“ an den Proxydienst, der nun die Anfrage des Endbenutzers an den Geoinformationsdienst weiterleitet oder direkt ablehnt.

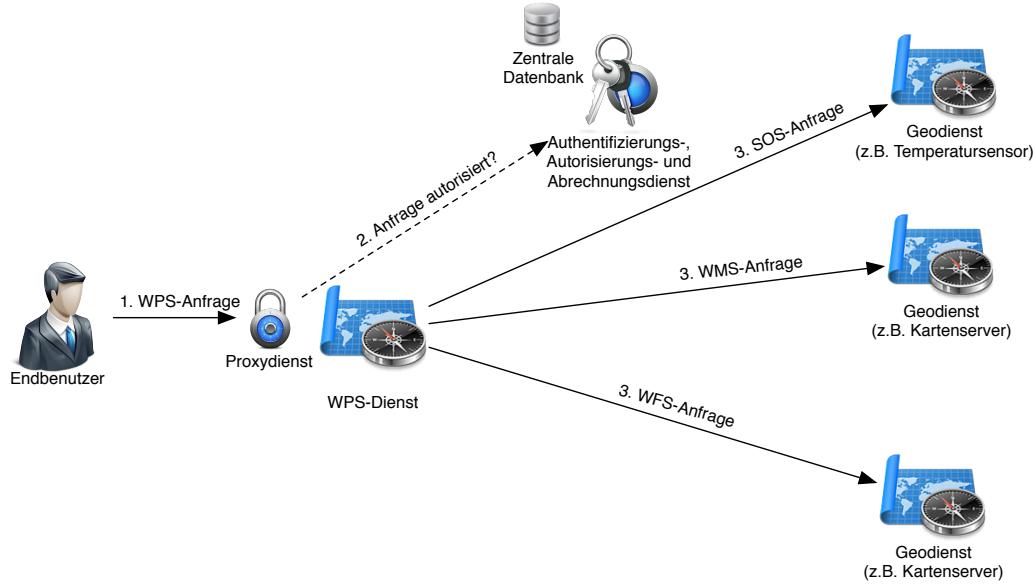


Abbildung 6.6: WPS-Anfrage mit einer zentralen Autorisierungsprüfung beim AAAD.

## 6.6 Betrachtung komplexer Szenarien mit Web Processing Services

Bei der Betrachtung des komplexeren Geoinformationsdienste WPS, der wiederum andere Geodienste nutzt, kommen weitere Überlegungen zum Systementwurf hinzu. In einem einfachen Szenario, wie in Abbildung 6.6 dargestellt, fängt ein Proxydienst die erste WPS-Anfrage ab und lässt diese vom AAAD autorisieren. Die nun folgenden Anfragen an die einzelnen vom WPS verwendeten Geoinformationsdienste finden nun ohne weitere Prüfung statt, da diese nicht mehr durch einen Proxydienst geschützt sind.

Dies mag für einzelne Spezialfälle eine annehmbare Architektur sein, im Normalfall soll jedoch jeder einzelne Geoinformationsdienst geschützt werden.

Aus diesem Grund ist eine Architektur wie in Abbildung 6.7 zu erwarten, wo jeder Geoinformationsdienst einzeln durch einen davor geschalteten Proxydienst geschützt wird, der wiederum beim AAAD um die Autorisierung der Anfrage bittet. Aus Gründen der

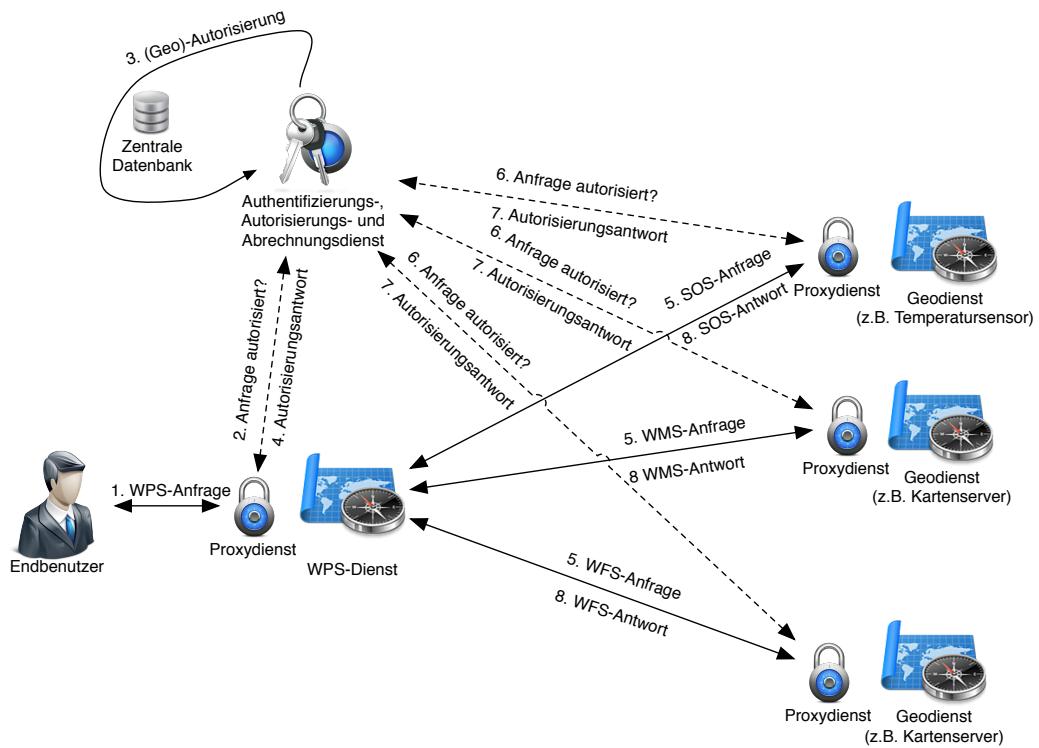


Abbildung 6.7: WPS-Anfrage geschützter Geoinformationsdienste und deren jeweilige Autorisierungsanfragen beim AAAD.

Übersichtlichkeit wurden in der Abbildung viele Kommunikationszwischenschritte vereinfacht dargestellt oder weggelassen, die in vorherigen Darstellungen noch abgebildet waren.

## 6.7 Einführung von Token-basierter Sicherheit

Damit der AAAD weiß, um welchen Benutzer es sich handelt, muss der Proxydienst entsprechende Authentifizierungsinformationen zur Verfügung stellen. Dazu könnte der Benutzer neben seiner Geodienst-Anfrage auch Benutzername und Passwort zum Proxydienst senden, der die entsprechenden Informationen an den AAAD weitersendet, wie in Abbildung 6.8 dargestellt.

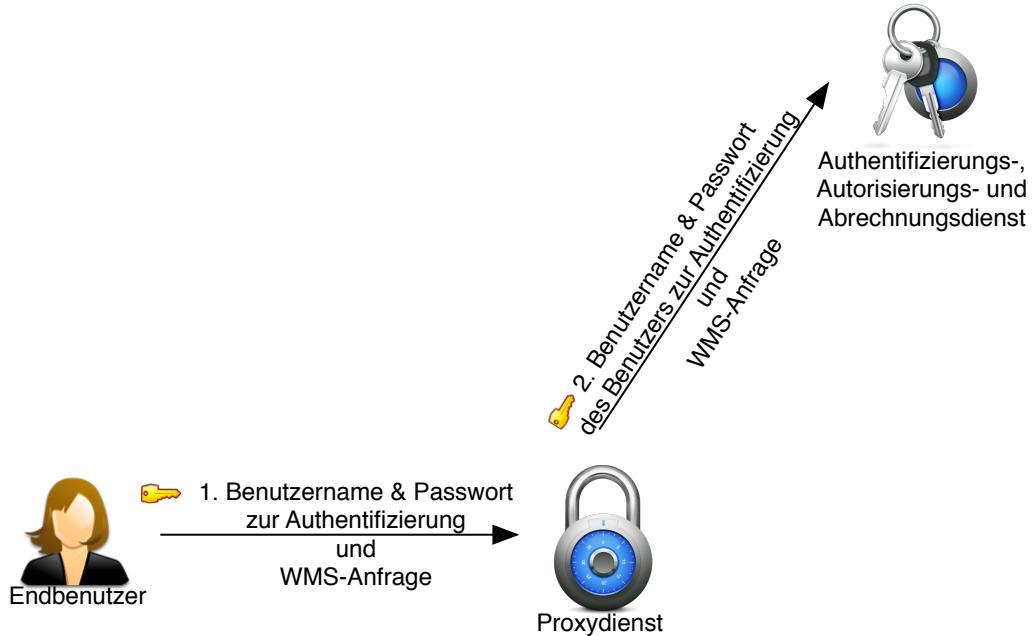


Abbildung 6.8: Unsichere Authentifizierung mittels Benutzername und Passwort.

Dies zieht jedoch gravierende Sicherheitsbedenken nach sich, da ein manipulierter Proxydienst Benutzername und Passwort des Benutzers abfangen und missbrauchen könnte.

Besser ist daher der Einsatz von Tokens. Der Benutzer sendet mit seiner Anfrage einen Token, welchen der Proxydienst an den AAAD weiterleitet, wie in Abbildung 6.9 dargestellt ist. Aufgrund des Tokens ist der AAAD in der Lage, den Benutzer zu authentifizieren und seine Anfrage entsprechend zu autorisieren.

Fängt ein Angreifer den Zugriffstoken ab, ist er nicht in der Lage, daraus Rückschlüsse über den Benutzer oder das Passwort des Benutzers zu ziehen, was die Sicherheit signifikant erhöht.

Implementiert man weitere Sicherheitsmerkmale wie zum Beispiel eine eingeschränkte Token-Gültigkeitsdauer<sup>1</sup> und hält diese gering, hat ein Angreifer nur einen kurzen zeitlichen Spielraum für einen Missbrauch, was die Sicherheit weiter erhöht.

<sup>1</sup>Im Sinne von Anzahl von Zugriffen (z.B.: Zugriffstoken für 3 Anfragen) oder im Sinne von Zeit-einheiten (z.B.: Zugriffstoken für die nächsten 5 Minuten)

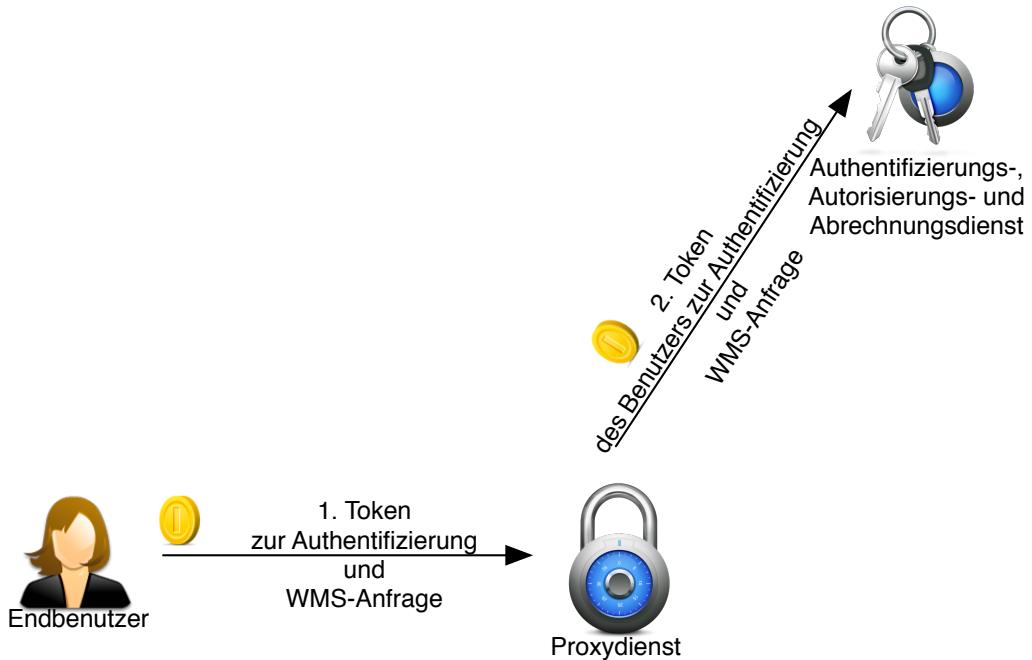


Abbildung 6.9: Sichere Authentifizierung mittels Token.

In einem *Hochsicherheitsszenario* wäre denkbar, dass der AAAD dem Endbenutzer direkt oder indirekt zusätzlich nach jeder Anfrage einen zufälligen Autorisierungscode übermittelt, welcher der Endbenutzer bei der nächsten Anfrage gemeinsam mit dem Zugriffstoken senden muss. Stimmt der Autorisierungscode der Anfrage nicht mit dem hinterlegten Autorisierungscode beim AAAD überein, so wird der gesamte Zugriffstoken sofort verworfen und der Benutzer muss sich neu authentifizieren. Dadurch wird zwar nicht verhindert, dass ein Angreifer einen Token abfängt und diesen zur Kommunikation mit dem AAAD verwendet. Würde der Endbenutzer aber eine neue Anfrage an den AAAD senden, könnte dieser die Anfrage mit der Begründung eines ungültigen Autorisierungscode ablehnen. Der Benutzer würde dann zumindest, dass ein Angriff stattgefunden haben muss. Man könnte dadurch Angriffe erkennen.

Des weiteren sollten die Nachrichtenpakete sowie der Nachrichtenkanal durch Transportbasierte und Nachrichten-basierte Sicherheit geschützt werden (siehe Kapitel 3).

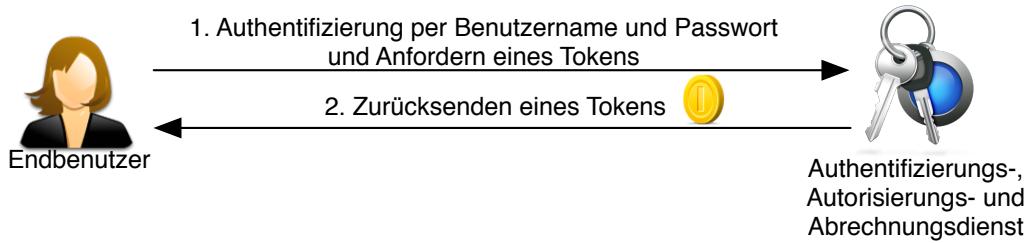


Abbildung 6.10: Anfordern eines Tokens beim AAAD bei lokaler Authentifizierung.

### 6.7.1 Bereitstellung des Tokens

Der Benutzer muss in Besitz eines Tokens gelangen, damit er diesen mit seiner eigentlichen Geoinformationsanfrage absenden kann. Den Token kann er dadurch gewinnen, dass er sich beim AAAD authentifiziert und dafür einen Token bekommt, wie dies in Abbildung 6.10 dargestellt ist.

In diesem Szenario meldet sich der Benutzer beim AAAD direkt an, um den Token zu erhalten. Der AAAD muss daher Benutzername und Passwort des Benutzers kennen.

In größeren Installationen ist jedoch seitens der Geoinformationsdienst-Kunden gewünscht, deren eigene Authentifizierungsserver nutzen zu können. Die Mitarbeiter der Behörde oder Organisation melden sich dabei an dem firmeneigenen Authentifizierungsserver mit starker Authentifizierung (siehe Kapitel 2.1), beispielsweise einem OpenID-System, an und können danach den Geoinformationsdienst des GIS-Anbieters nutzen, wie in Abbildung 6.11 dargestellt. Der AAAD muss dabei dem externen Authentifizierungssystem entweder uneingeschränkt vertrauen und allen dort authentifizierten Benutzern den Zugriff erlauben oder autorisiert nach der Authentifizierung durch den externen Server nochmals zusätzlich den Zugriff mit seiner internen Benutzerdatenbank, welche nun jedoch nur noch die Kennung des Benutzers beinhaltet, nicht mehr jedoch dessen Passwort. Im Katastrophenfall, wo man einer ganzen Behörde oder Organisation wie der Feuerwehr den Zugriff auf wichtige Daten genehmigen möchte, scheint das Szenario, dem externen Authentifizierungsserver zu vertrauen, zielführend. Im Regelbetrieb, wo Informationen einer zahlenden Organisation wie einem Straßenbaukonzern oder einem Shoppingcenter-Betreiber zur Verfügung gestellt werden sollen, kann

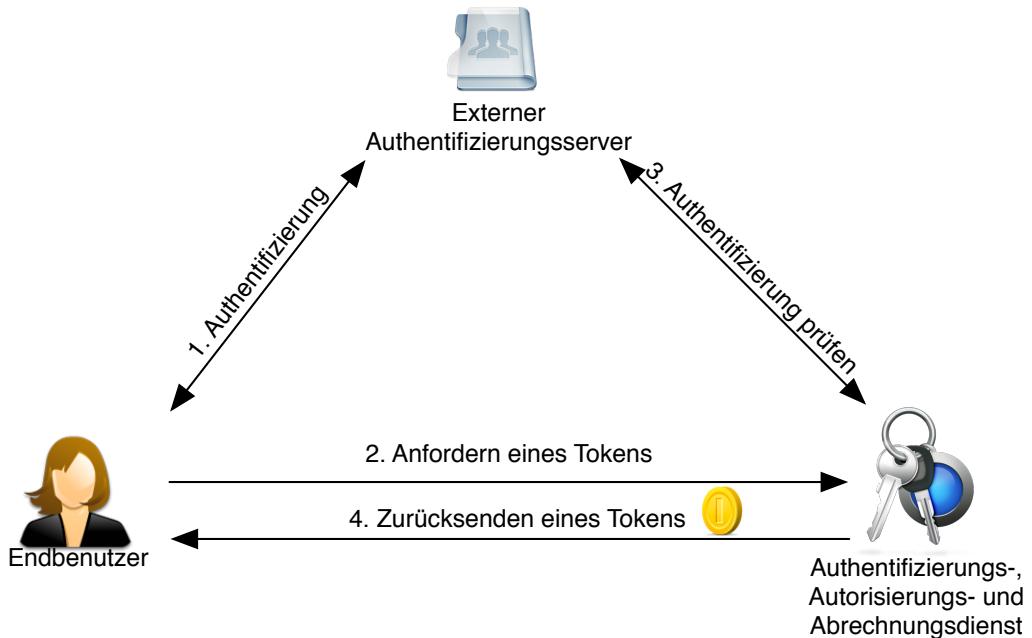


Abbildung 6.11: Anfordern eines Tokens beim AAAD bei externer Authentifizierung.

die Verwaltung der Benutzer bei der externen Organisation erfolgen, die Autorisierung jedoch beim AAAD zusätzlich gesteuert werden.

## 6.8 Zusammenfassung zur serviceorientierten Geoinformationsarchitektur

Durch das Paradigma der „Separation of Concerns“ leiten sich die beteiligten Rollen und Komponenten eindeutig ab.

Einerseits gibt es die Komponente *Geoinformationsdienst*, welcher die Daten zur Verfügung stellt, ohne sich um Triple-A-Aspekte<sup>2</sup> kümmern zu müssen.

Eine weitere Komponente bzw. Rolle in diesem System ist der *Benutzer*, der Geoinformationen abrufen möchte. Der Benutzer ist bereit, sich dafür zu authentifizieren und gegebenenfalls einen angemessenen Betrag zu bezahlen.

Die Triple-A-Aspekte im Rahmen der serviceorientierten Geoinformationsarchitektur

<sup>2</sup>Authentifizierung, Autorisierung und Abrechnung, siehe 2.

werden von einem zentralen Webservice, dem *Authentifizierungs-, Autorisierungs- und Abrechnungsdienst (AAAD)* bereitgestellt. Er authentifiziert den Benutzer, autorisiert den Zugriff auf den Geoinformationsdienst und übernimmt die Protokollierung der Zugriffe für die Abrechnung. In komplexeren Szenarien kann die *Authentifizierung durch externe Server* übernommen werden.

Die Kommunikation zwischen dem Endbenutzer, dem Geoinformationsdienst und dem AAAD werden von einem *Proxydienst* gesteuert.

Somit sind die beteiligten Rollen und Komponenten vollständig identifiziert:

- Benutzer
- Geoinformationsdienst
- Authentifizierungs-, Autorisierungs- und Abrechnungsdienst (AAAD)
- Proxydienst
- (Externer Authentifizierungsserver in komplexen Szenarien)

Der Proxydienst kann dabei direkt auf dem Server oder Embedded Device des Geoinformationsdienstes implementiert werden oder auf eigener Hardware laufen. Im letztgenannten Fall muss gewährleistet sein, dass der Geoinformationsdienst nur Anfragen des Proxydienst beantwortet, damit der Endbenutzer die Autorisierung nicht umgehen kann, indem er direkt auf den Dienst zugreift. Dies kann zum Beispiel mittels eines IP-Adressen-Filters realisiert werden, wobei der Geoinformationsdienst so konfiguriert wird, dass er nur mit dem Proxydienst kommuniziert, der die hinterlegte IP-Adresse nutzt.

# 7

---

## Evaluierung bestehender Protokolle und Implementierungen zum Schutz von Geoinformationssystemen

Nachdem im vorherigen Kapitel eine Sicherheitsarchitektur zum Schutz von Geoinformationsdiensten entworfen wurde, wird in diesem Kapitel untersucht, mit welchen Protokollen oder Produkten eine derartige Infrastruktur realisiert werden kann.

Vier zentrale Fragestellungen stehen dabei im Vordergrund:

1. Welches Authentifizierungsprotokoll bzw. welches Produkt bietet den erfolgversprechendsten Ansatz für eine Implementierung?
2. Welches Autorisierungsprotokoll bzw. welches Produkt bietet den erfolgversprechendsten Ansatz für eine Implementierung?
3. In welchem Format sollen die Geoauthorisierungsregeln gespeichert werden?
4. Wo und wann werden die Geoauthorisierungsregeln zwischen Endbenutzer, AAAD, Proxydienst und Geoinformationsdienst übertragen?

Auf diese Fragestellungen gehen die nächsten Abschnitte näher ein, wobei zuerst die Problematik des für den Anwendungsfall am besten geeigneten Autorisierungsprotokolls behandelt wird.

## 7.1 Begründete Auswahl eines Autorisierungsprotokolls für verteilte Dienste

Zum Schutz der verteilten Geoinfrastruktur stehen mehrere Technologien, Standards, Protokolle und Implementierungen zur Verfügung. Im Folgenden werden die für die Implementierung relevanten Kriterien dieser Technologien miteinander verglichen, um eine begründete Technologiewahl zu treffen, die als Basis für die Realisierung dient.

Zur Auswahl stehen OAuth, Shibboleth, SAML und WS-Federation, da diese Protokolle auf Webstandards basieren, Dokumentation vorliegt, von großen Unternehmen unterstützt werden und unter anderem geschaffen wurden, Autorisierungsinformationen in verteilten Systemen zu übermitteln.

Shibboleth verwendet intern SAML, so mag ein Vergleich dieser beiden Technologien eigenartig anmuten, jedoch stellt sich die Frage, ob ein fertiger Baustein wie Shibboleth als Basis für die Implementierung verwendet werden soll oder nur einzelne Komponenten daraus, wie eben SAML. Aus dieser Sicht macht ein Vergleich Sinn.

### Gewichtung

Die einzelnen Kriterien unterliegen unterschiedlicher Gewichtung, wobei die Gewichtung vom Verfasser dieser Arbeit in Zusammenarbeit mit Mitarbeitern vom Research Studio iSpace vorgenommen wurde. Besonders wichtig ist dabei der *Implementierungsaufwand am Client*, denn alle von den Kunden verwendete Clients müssen so angepasst werden, dass sie vom AAAD Tokens anfordern und diese mit ihrer WMS-Anfrage an den Proxydienst weiterleiten können. Je einfacher die Implementierung, desto größer die Akzeptanz beim Kunden. Dabei spielt als zweitwichtigstes Kriterium die *Komplexität* der verwendeten Technologie eine tragende Rolle. Das verwendete Protokoll muss so mächtig sein, dass es die Anforderungen der Sicherheitsarchitektur erfüllt, den Benutzer sowie den Programmierer, der die Technik am Client implementiert, jedoch nicht durch eine zu große Komplexität abschreckt. Der *Funktionsumfang* sollte daher für den Anwendungsfall angemessen sein und nicht Details regeln, die hier nicht verwendet werden. Dies stellt den dritt wichtigsten Faktor dar. Da für einen großflächigen

Einsatz wichtig ist, dass viele Geoinformationsdienste ohne viel *Konfigurationsaufwand* geschützt werden können, wurde als nächstes Gewicht der Einrichtungs- und Konfigurationsaufwand zwischen schützendem Proxydienst und dem AAAD herangezogen. Sowohl auf Kundenseite als auch auf Geodiensteanbieter-Seite ist die *Verfügbarkeit von fertigen und kostenlosen oder günstigen Programmierbibliotheken* wichtig, um nicht jedes Detail im Kommunikationsablauf selbst implementieren zu müssen, diesem Aspekt kommen somit 10 % Gewicht zu. Weniger wichtig sind Faktoren wie derzeitige *Verbreitung, Datenoverhead* sowie der *Implementierungsaufwand am AAAD und am Proxydienst*, denn wenn die Software einmal entwickelt wurde, kann diese auf allen weiteren AAAD und Proxydiensten kopiert werden.

## Verfügbare Bibliotheken

Für OAuth gibt es eine Vielzahl an kostenlosen Bibliotheken für fast jede moderne Programmiersprache sowie umfangreiche Beispiele und Dokumentation [2]. Auch für Shibboleth stehen einige APIs und eine Dokumentation zum freien Download zur Verfügung [5]. Im Bereich SAML und WS-Federation lassen sich vor allem im .NET und Java-Umfeld einige Bibliotheken finden<sup>1</sup>, jedoch nicht im selben Umfang wie für OAuth.

## Implementierungsaufwand am Client, Proxydienst und AAAD

Der Implementierungsaufwand am Client ist bei OAuth relativ gering. Um OAuth unterstützen zu können, muss das HTTP-Protokoll, welches die Clients ohnehin bereits unterstützen, um die OAuth Attribute erweitert werden, ähnlich wie das bei der HTTP-Authentifizierung bereits der Fall ist. Der Aufwand dafür ist überschaubar. Dies trifft ebenso für den Implementierungsaufwand am Proxydienst als auch am AAAD zu. Bei Shibboleth ist der Aufwand aufgrund des komplexeren Protokolls höher, jedoch mit den verfügbaren Bibliotheken und Dokumentation in vertretbarem Aufwand machbar. Bei SAML und WS-Federation ist der Implementierungsaufwand aufgrund der umfangreichen Standards, auf denen diese Spezifikationen basieren, höher.

---

<sup>1</sup>Kommerzielle SAML-Bibliothek für .NET <http://www.componentsoft.net/component/saml.net/>

## **Einrichtungsaufwand Vertrauensverhältnis zwischen Proxydienst und AAAD**

Das Vertrauensverhältnis zwischen Konsument und Service-Provider im OAuth Protokoll wird durch ein geteiltes Geheimnis in Form eines kurzen Textes<sup>2</sup> eingerichtet [41]. Das Herstellen eines Vertrauensverhältnisses im Falle von Shibboleth, SAML und WS-Security ist komplexer, da diese Protokolle private und öffentliche Schlüssel verwenden, welche zuerst erstellt und dann verteilt werden müssen<sup>3</sup> [1] [26] [33].

## **Verbreitung**

OAuth ist trotz seines jungen Alters durch das Web 2.0 und das damit zusammenhängende Bedürfnis einer sicheren Nutzung mehrerer Webdienste bereits sehr weit verbreitet. Shibboleth hat im Umfeld von Hochschulen breiten Fuß gefasst und gilt dort als De-Facto Standard [46]. WS-Federation ist durch die Unterstützung von Microsoft und IBM zwar weit verbreitet, kommt jedoch außerhalb von Produkten dieser Hersteller eher seltener vor. Aufgrund der Komplexität hat auch SAML noch nicht die von den Erschaffern angestrebte Verbreitung erfahren.

## **Datenoverhead**

Der Datenoverhead von OAuth ist relativ gering, da nur einige Daten in Form von HTTP-Headern oder eingebettet in HTTP-GET/POST Anfragen ausgetauscht werden. Die anderen auf XML basierenden Protokolle zeigen hier einen weit größeren Overhead in der Kommunikation. Der Datenoverhead spielt bei Geodiensten wie WMS, WPS, WCS und WFS, die zumeist auf leistungsfähiger Hardware in Rechenzentren gehostet werden, kaum eine Rolle. Bei Diensten wie SOS, die oftmals auf Embedded Devices implementiert werden, bedeutet ein großer Overhead jedoch oftmals langsame Verarbeitung sowie mehr Energiehunger des Gesamtsystems.

---

<sup>2</sup>Beispiel für ein Konsumentengeheimnis in OAuth: djr9rjt0jd78jf88.

<sup>3</sup>Allerdings steigt dadurch auch die Sicherheit.

## Akzeptable Komplexität

Die Komplexität von OAuth ist gering und die Spezifikationen im Vergleich zu den anderen Protokollen sind kurz. Ein mit Autorisierungsprotokollen unerfahrener Programmierer kann sich in die Thematik rasch einlesen und wird den einfachen Kommunikationsablauf vermutlich innerhalb kurzer Zeit verstehen. Shibboleth und SAML sind hier weit komplexer und der Einarbeitungsaufwand ist viel höher als bei OAuth. WS-Federation, als Teil der umfangreichen WS-\* Spezifikationen, erfordert den größten Einarbeitungsaufwand aufgrund des enormen Umfangs der Spezifikationen und der damit verbundenen Komplexität.

## Angemessener Funktionsumfang für den Anwendungsfall

Technologien wie SAML bieten mehr Features als einfache Protokolle wie OAuth. Mit SAML könnte man gewisse Vorgaben bei der Authentifizierung am externen Authentifizierungsserver erzwingen (zum Beispiel Login per biometrischer Authentifizierung). Bei OAuth hingegen muss der Benutzer authentifiziert sein, es besteht keine Möglichkeit, den Authentifizierungsprozess zu beeinflussen. Im vorgegebenen Szenario gibt es jedoch keine solchen Anforderungen, daher ist ein Protokoll wie OAuth besser für den Anwendungsfall geeignet als eine Technologie, welche viele weitere, nicht benötigte Features unterstützt.

## Evaluierungsergebnis der Autorisierungsprotokolle

Nach Betrachtung und Vergleich aller Kriterien stellt OAuth die beste Wahl für eine Implementierung dar (siehe Tabelle 7.1). Shibboleth belegt den zweiten Platz, da es sich hier bereits um eine fertige Implementierung handelt, die viele Details der intern verwendeten Technologien wie SAML vor dem Programmierer verbirgt. Im Vergleich von WS-Federation und SAML belegt SAML den besseren Platz, da die Spezifikationen weniger umfangreich sind als die WS-Federation Spezifikationen.

### 7.1.1 Vergleichstabelle der Technologien zum Schutz der GIS-Infrastruktur

	Gewichtung	OAuth 1.0a	Shibboleth 2.3	SAML 2.0	WS-Federation 1.1
Verfügbare Bibliotheken	10 %	+	10 Pkt.	+	10 Pkt.
Implementierungsaufwand Client	25 %	+	25 Pkt.	o	15 Pkt.
Implementierungsaufwand Proxydienst	5 %	+	5 Pkt.	o	3 Pkt.
Implementierungsaufwand AAAD	5 %	+	5 Pkt.	o	3 Pkt.
Einrichtungsaufwand Vertrauensverhältnis	10 %	+	10 Pkt.	o	6 Pkt.
Verbreitung	5 %	+	5 Pkt.	+	5 Pkt.
Datenoverhead	5 %	+	5 Pkt.	o	3 Pkt.
Akzeptable Komplexität	20 %	+	20 Pkt.	o	12 Pkt.
Funktionsumfang angemessen für Anwendungsfall	15 %	+	15 Pkt.	o	9 Pkt.
Ergebnis	100 %	1.Wahl	100 Pkt.	2.Wahl	66 Pkt.
				3.Wahl	53 Pkt.
				4.Wahl	49 Pkt.

Tabelle 7.1: Gewichtete Bewertung der Technologien zum Schutz der verteilten Geoinfrastruktur. „+“ bedeutet eine positive Beurteilung, „-“ eine negative Beurteilung und „o“ eine neutrale Beurteilung. Der Vergleich der Technologien ist als relativ zu den anderen zu interpretieren, ein „-“ bedeutet daher nicht prinzipiell eine absolute negative Bewertung der Technologie, sondern eine weniger gute Bewertung im Beziehung zu den anderen angeführten Protokollen. Um eine Auswahl treffen zu können, wurde „+“ mit 100 Punkten, „o“ mit 60 Punkten und „-“ mit 40 Punkten bewertet und die Punktzahl mit der jeweiligen Gewichtung multipliziert.

## 7.2 Auswahl des Authentifizierungsprotokolls

Die zweite, eingangs erwähnte, zentrale Fragestellung betrifft die Authentifizierung des Benutzers.

Hierbei fiel die Wahl des Autors auf eine klassische Benutzername/Passwort-Authentifizierung, um das Szenario einer einfachen lokalen Benutzerdatenbank zu implementieren.

Um die erweiterten Anforderungen, dem Einsatz eines externen Authentifizierungsservers, zu prüfen, wurde OpenID als leichtgewichtiges „Schwesterprotokoll“ zu OAuth vom Autor dieser Arbeit ausgewählt.

Eine umfangreiche Evaluierung der verfügbaren Authentifizierungsprotokolle fand nicht statt, denn in der Praxis gibt in den meisten Fällen der Kunde das Authentifizierungssystem vor. Verwendet der Kunde in seinem Unternehmen das Lightweight Directory Access Protocol (LDAP) zur Authentifizierung, so muss man den AAAD um ein LDAP-Modul erweitern, wenn man das Unternehmen als Kunden gewinnen möchte. Verwendet das Unternehmen hingegen ActiveDirectory, so ist eine Anpassung dafür notwendig. Es würde daher wenig Sinn machen, das „beste“ Authentifizierungssystem zu evaluieren, wenn man als Anbieter ohnehin meist das vom Kunden genutzte Authentifizierungssystem in die serviceorientierte Geoinformationsarchitektur einzubinden hat.

## 7.3 Format zum Speichern der Geoauthorisierungsregeln

Die dritte, eingangs gestellte, Frage behandelt das Speichern der Geoauthorisierungsregeln. Durch die verschiedenen zu sichernden Attribute (siehe Kapitel 5.4) in Geoinformationssystemen muss die Erlaubnis für den Zugriff auf die einzelnen Ressourcen pro Benutzer flexibel konfigurierbar sein und einfach gespeichert werden können, damit die entsprechenden Geoinformationssysteme prüfen können, welche Informationen der Benutzer abrufen bzw. schreiben darf.

## Speichern einfacher Regeln in INI-Dateien

Um diese Attribute speichern zu können, werden in der einfachsten Form Schlüssel/Wert Paare in einer Konfigurationsdatei oder Datenbank abgespeichert.

Im GeoServer<sup>4</sup> werden hierfür etwa einfache Konfigurationsdateien, verwendet, um Benutzer, Passwörter und Gruppen zu definieren (siehe Listing 7.1).

---

Listing 7.1: GEOSERVER\_DATA\_DIR/security/user.properties

---

```
98 admin=geoserver,ROLE\ADMINISTRATOR
99 wfst=wfst,ROLE\WFS\_READ,ROLE\WFS\_WRITE
100 wfs=wfs,ROLE\WFS\_READ
```

---

Die Konfigurationsdatei in Listing 7.2 zeigt die Zugriffsrechte der Benutzer und Gruppen für einzelne Kartenebenen. Die genaue Bedeutung der Parameter ist [4] zu entnehmen.

---

Listing 7.2: GEOSERVER\_DATA\_DIR/security/layers.properties

---

```
*.*.r=*
102 *.*.w=NO\_ONE
private.*.r=TRUSTED\_ROLE
104 private.*.w=TRUSTED\_ROLE
topp.congress\_district.w=STATE\_LEGISLATORS
```

---

## Abbildung geografischer Regeln in proprietären XML-Formaten

Um Konfigurationsdateien syntaktisch besser überprüfbar zu machen<sup>5</sup>, liegt es nahe, diese in Form von XML-Dateien zu speichern.

Die 52n Web Enforcement Service (WSS)-Komponente des Security Servers 52°North<sup>6</sup> verwendet in der einfachsten Form sein eigenes, proprietäres XML-Format namens „Simple Permissions XML“, welches in Listing 7.3 dargestellt ist.

---

<sup>4</sup><http://www.geoserver.org>. Referenzimplementierung des OGCs der WMS-, WFS- und WCS-Schnittstellen.

<sup>5</sup>Der Aufbau von XML-Dokumenten kann mit XML-Schemata überprüft werden.

<sup>6</sup><http://www.52north.org>

Listing 7.3: SERVER\_HOME/webapps/wss/WEB-INF/classes/permissions.xml

```

106 <?xml version="1.0" encoding="UTF-8"?>
108 <SimplePermissions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
109   xmlns="http://www.52north.org/security/simple-permission/1.0">
110   <PermissionSet name="WMS1 Permission">
111     <ResourceDomain value="http://localhost/wss/*/demiswms/" />
112     <ActionDomain value="http://localhost/wss/*/demiswms/" />
113     <SubjectDomain value="urn:n52:security:subject:role" />
114     <Permission name="alice\_all">
115       <Resource value="layers/*" />      <!-- Any layers -->
116       <Action value="operations/*" />    <!-- Any operations -->
117       <Subject value="alice" />
118     </Permission>
119     <Permission name="bobAndGuest\_most\_GetMap\_GetCaps">
120       <Resource value="layers/Cities" />
121       <Resource value="layers/Builtup%20areas" />
122       <Resource value="layers/Hillshading" />
123       <Resource value="layers/Borders" />
124       <Resource value="layers/Countries" />
125       <Action value="operations/GetCapabilities" />
126       <Action value="operations/GetMap" />
127       <Subject value="bob" />
128       <Subject value="guest" />
129     </Permission>
130     <Permission name="bob\_Countries\_GetFeatureInfo">
131       <Resource value="layers/Countries" />
132       <Action value="operations/GetFeatureInfo" />
133       <Subject value="bob" />
134     </Permission>
135     <Permission name="guest\_countries\_GetFeatureInfo\_obliged">
136       <Resource value="layers/Countries" />
137       <Action value="operations/GetFeatureInfo" />
138       <Subject value="guest" />
139       <Obligation name="obligation:wms:extent:boundingbox">
140         <Attribute name="srs">EPSG:4326</Attribute>
141         <Attribute name="box">-170,-56,-36,83</Attribute>
142       </Obligation>
143     </Permission>
144   </PermissionSet>
145 </SimplePermissions>

```

In der XML-Konfigurationsdatei in Listing 7.3 werden verschiedene Zugriffe auf einzelne Kartenebenen und Bounding Boxes erlaubt. Naheliegender, als eigene XML-Dialekte zu verwenden, scheint es jedoch, auf etablierte Standards zurückzugreifen.

## Formulierung von Zugriffsregeln mittels der standardisierten OGC-Filter Spezifikationen

Das OGC definiert mit der „OpenGIS Filter Encoding Implementation Specification“ einen XML-Standard zum Spezifizieren von Filterausdrücken [77]. Dieser offizielle Standard findet in vielen Spezifikationen der OGC wie WMS, WFS, WPS oder SOS Verwendung [77] [51] [64]. Produkte, die beispielsweise einen WMS-Dienst standardkonform implementieren, können daher automatisch auch Open Geospatial Consortium-Filter (OGC-Filter) auswerten und berücksichtigen.

Um geografische Aspekte wie Punkte, Linien, Rechtecke oder Polygone zu beschreiben, stützen sich OGC-Filter auf das etablierte Format GML [77]. Ein einfacher Filter, der eine Bounding Box definiert, ist in Listing 7.4 zu sehen.

**Listing 7.4:** Einfacher OGC-Filter mit Definition einer Bounding Box im GML-Format.

---

```

146 <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" >
    xmlns:gmgml="http://www.intergraph.com/geomedia/gml">
    <ogc:BBOX>
148      <ogc:PropertyName>the\_\_geom</ogc:PropertyName>
      <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml\#4326">
150        <gml:coordinates>
          -73.9,40.9 -74.05,40.8
152        </gml:coordinates>
      </gml:Box >
154    </ogc:BBOX>
  </ogc:Filter>

```

---

Durch den einfachen Aufbau und die breite Unterstützung der OGC scheinen OGC-Filter eine gute Möglichkeit zum Formulieren der geografischen Zugriffsregeln zu sein. Anfragen vom Endbenutzer müssen nur um die hinterlegten Filterregeln ergänzt und zum ausführenden Service gesendet werden, welcher diese dann selbstständig auswertet. Sendet der Endbenutzer bereits OGC-Filter-Ausdrücke mit seiner Anfrage, so ist der Filter des Benutzers und der hinterlegte Filter am AAAD mit einem logischen AND zu verknüpfen. Weitere Anforderungen als diese AND-Verknüpfung durchzuführen werden nicht an den AAAD gestellt. Die zwei durch AND verknüpften Filter kann der ausführende Geoinformationsdienst in weiterer Folge selbstständig auswerten.

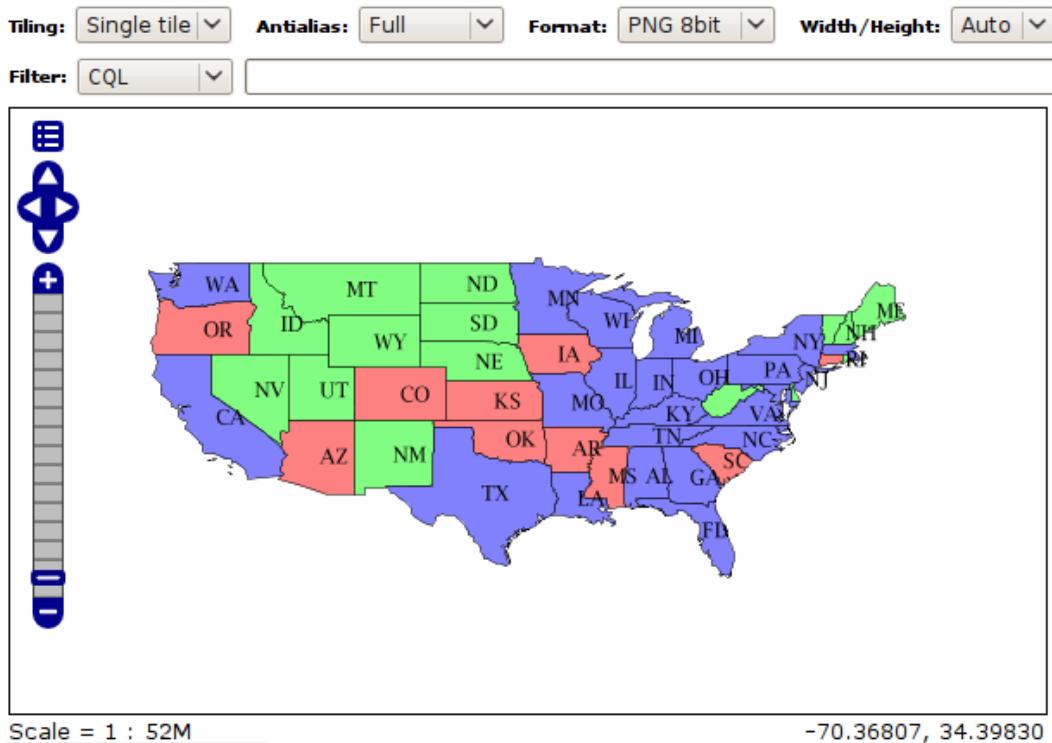


Abbildung 7.1: Ergebnis einer ungefilterten WFS-Anfrage zum Auslesen der amerikanischen Bundesstaaten.

## Filtern von GIS-Anfragen mit der OGC Common Catalogue Query Language

Die OGC Common Catalogue Query Language (CQL) ist eine weitere, vom OGC geschaffene Filtersprache für Catalogue Service-Web (CSW)-Dienste wie WMS oder WFS [75]. Ähnlich, wie Daten aus relationalen Datenbanken mit der Abfragesprache SQL abgerufen werden können, ist es mit der CQL möglich, geografische Daten abzurufen bzw. zu filtern [52].

Wendet man etwa den Filterausdruck MALE >FEMALE auf die Daten an, die Abbildung 7.1 zu Grund liegen, so resultiert dies in Abbildung 7.2.

Ein komplexeres Beispiel mit einer Berechnung im Filter ist in Abbildung 7.3 dargestellt, wo sämtliche Bundesstaaten weggefiltert werden, deren Arbeitslosenquote unter 7 % liegt.



Abbildung 7.2: Ergebnis einer mit dem Attribut MALE > FEMA-LE gefilterten WFS-Anfrage, um darzustellen, wo der Männeranteil der Bevölkerung größer ist als der Frauenanteil.

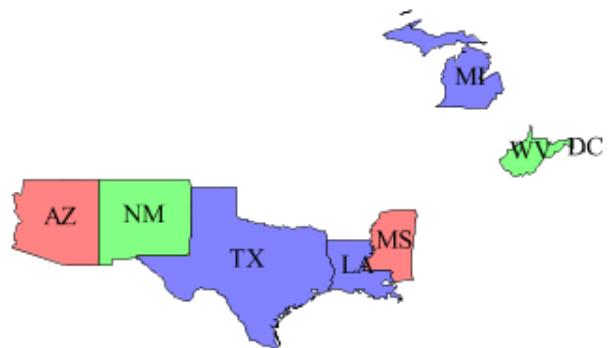


Abbildung 7.3: Ergebnis einer WFS-Anfrage, gefiltert nach Bundesstaaten mit einer Arbeitslosenquote unter 7 %.

Weitere Beispiele wären die Filterung nach Bounding Boxes oder alle geografischen Objekte innerhalb eines bestimmten Radius und vieles mehr. Das Filtern nach Karte-nebenen ist mit der CQL jedoch nicht möglich.

CQL Ausdrücke wie etwa HEIGHT between 1500 and 2500, zum Filtern aller Objekte (Berge) mit einer Höhe zwischen 1500 und 2500 Meter, lassen sich direkt in XML basierte OGC-Filter transformieren, wie dies in Listing 7.5 dargestellt ist.

Listing 7.5: CQL-Statement in XML-Darstellung

---

```

156 <Filter>
  <PropertyIsBetween>
    <PropertyName>HEIGHT</PropertyName>
    <LowerBoundary><Literal>1500</Literal> </LowerBoundary>
    <UpperBoundary><Literal>2500</Literal> </UpperBoundary>
  </PropertyIsBetween>
162 </Filter>

```

---

Der Vorteile der CQL ist das für Menschen relativ gut lesbare Format und die einfache Transformierung nach OGC-Filter [52].

## Komplexe Infrastruktur zum Schutz von Geodiensten mit GeoXACML

GeoXACML betitelt sich selbst mit „Access Control for Geospatial Data“ und ist daher eine nähere Betrachtung wert. Geospatial eXtensible Access Control Markup Language (GeoXACML) ist dabei eine vom OGC standardisierte Erweiterung des komplexen eXtensible Access Control Markup Language (XACML)-Standards um Geometrien und dazugehörige Datentypen wie Punkt, Linie, Polygon und mathematische Funktionen wie Überlappen, Schneiden, Beinhalten und ähnlichen Funktionen. Diese Geometrien können in der GML Version 2 oder Version 3 definiert werden [44] [50]. Die Funktionen selbst sind nicht festgelegt und können vom jeweiligen Produkt individuell implementiert werden, wobei ein breiter de-facto Standard an häufig benutzten Funktionen in den einzelnen Implementierungen zu finden ist. Um das Konzept hinter GeoXACML verstehen zu können, sei zuerst der übergeordnete Standard XACML näher beschrieben.

Der Organization for the Advancement of Structured Information Standards (OASIS)-Standard XACML beschreibt ein Vokabular und eine Grammatik, um Regeln in XML abzubilden, die als Grundlage für Autorisierungsentscheidungen dienen [54] [44]. XACML definiert dabei Regeln, um den Zugriff auf einzelne Ressourcen auf Basis der Zugriffsart, der Eigenschaften der zugreifenden Einheit (ein Benutzer oder eine Maschine) und der Art der Authentifizierung zu steuern. So könnte eine XACML-Regel lauten, dass nur Benutzer der Benutzergruppe „Personalmanagement“ die Ressourcen „Lohnzettel“ editieren dürfen, und dies auch nur, wenn das Netzwerkprotokoll für den Zugriff per TLS gesichert ist und sich der Benutzer mit einem digitalen Zertifikat authentifiziert hat [54]. Der XACML-Standard beschreibt hierbei eine Architektur aus mehreren Komponenten. Abbildung 8.11 zeigt eine XACML-Architektur, wobei die XACML Komponenten in blauer Farbe beschriftet sind. Über einen sogenannten Policy Administration Point (PAP) werden die einzelnen Regeln administriert. Stellt nun ein Benutzer eine Anfrage an einen per XACML geschützten Dienst, so wird die Anfrage vom Policy Enforcement Point (PEP) abgefangen und an den Policy Decision Point (PDP) weitergeleitet. Der PDP analysiert die Anfrage des Benutzers und arbeitet die hinterlegten Regeln ab, um zu bestimmen, ob der Benutzer Zugriff auf

die gewünschte Ressourcen erhält oder nicht. Dabei antwortet der PDP mit PERMIT, DENY, INDETERMINATE oder NOT APPLICABLE an den PEP, der aufgrund der Antwort des PDPs die Anfrage nun an den eigentlichen Dienst weiterleitet oder dem Benutzer mitteilt, dass der Zugriff verweigert wurde. Die PERMIT-Antwort bedeutet dabei, dass der Zugriff gestattet ist, mit DENY hingegen antwortet der PDP, wenn der Zugriff verweigert wurde. Bei einem Verarbeitungsfehler der Anfrage antwortet der PDP mit INDETERMINATE. NOT APPLICABLE lautet die Antwort für den Fall, dass keine hinterlegte Regel auf die Anfrage des Benutzers zutrifft. Je nach Implementierung des PEP wird er in den beiden zuletzt genannten Fällen den Zugriff des Benutzers ablehnen oder zulassen [50]. Eine vierte Komponente, der Policy Information Point (PIP), stellt dem PDP gewisse Informationen zur Verfügung, die dieser benötigt, um seine Entscheidungen zu treffen [54]. Ist der Zugriff auf eine bestimmte Ressource etwa nur zu den „gewöhnlichen Bürozeiten“ erlaubt, kann diese Regel am PDP hinterlegt werden. Der PDP kann dann bei der Abarbeitung seiner Regeln den PIP fragen, welcher Zeitraum denn nun genau die „gewöhnlichen Bürozeiten“ betrifft. Die Kommunikation zwischen PAP, PIP, PDP und PEP findet dabei im XML-Format auf Simple Object Access Protocol (SOAP)-Basis statt.

GeoXACML erweitert, wie oben bereits erwähnt, den XACML-Standard um Geometrien, sodass der Zugriff beispielsweise auf einzelne Kartenausschnitte nur gewährt wird, wenn der Benutzer zur üblichen Bürozeit über ein per TLS gesichertes Netzwerkprotokoll zugreift [44]. Die Beschreibung eines einfachen Filters ähnelt dem OGC-Filter Format, wie in Listing 7.6 zu sehen ist. Definiert man komplexere Zugriffsregeln, wobei man Zugriffszeiten oder Authentifizierungsarten berücksichtigt, werden die Filter erheblich komplexer.

Listing 7.6: GeoXACML

---

```

164 <Function FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-contains"/>
    <AttributeValue DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry">
166     <gml:Polygon gid="North and South America" srsName="xmlns:gml="http://www.opengis.net/ ,
        gml">
            <gml:outerBoundaryIs><gml:LinearRing>
168         <gml:coordinates cs="," ts=" ">
            -180,60 -180,47 -137,55 -125,35 -110,17 -80,5 -87,-5
170         -74,2 -78,-53 -67,-58 -60,-48 -35,-23 -27,-3 -55,12 -60,22
            -75,30 -67,42 -50,45 -60,85 -60,85 -85,85 -120,80 -130,75
172         -160,75 -168,70 -180,60
        </gml:coordinates>
174     </gml:LinearRing></gml:outerBoundaryIs >
    </gml:Polygon>
176   </AttributeValue>
    http://www.opengis.net/gml/srs/epsg.xml#\#4326"
178 <AttributeSelector RequestContextPath="//gml:boundedBy/gml:Box" DataType="urn:ogc:def: ,
    dataType:geoxacml:1.0:geometry"/>

```

---

### 7.3.1 Vergleich der Formate zum Speichern von geografischen Zugriffsregeln

Da, wie im vorherigen Abschnitt dargestellt, verschiedene Möglichkeiten existieren, die Zugriffsregeln zu speichern, werden diese Optionen im Folgenden miteinander verglichen, um die beste Variante für eine anschließende Implementierung zu finden.

#### Gewichtung der Evaluierungskriterien

Die einzelnen Kriterien wurden gewichtet und verglichen. Die Gewichtung wurde vom Autor dieser Arbeit vorgeschlagen und mit dem Research Studio iSpace abgestimmt. Am bedeutsamsten ist dabei die *direkte Anwendbarkeit* der Filter bei der Abfrage von Geoinformationsdiensten. Filterausdrücke, die die Dienste bereits verstehen, erlauben eine schnelle und wenig aufwändige Implementierung. Dieser Aspekt wurde daher mit 35 % gewichtet. Mit 25 % Gewicht als zweitwichtigstes Kriterium ist der *Umfang der Filtermöglichkeiten* in die Evaluierung eingeflossen. Ein Filterausdruck sollte möglichst viele in Kapitel 5.4 vorgestellten Aspekte der Geoinformationen filtern können. Eine

geringe *Komplexität der Gesamtarchitektur* wurde mit 15 % gewichtet. Je geringer die Komplexität, desto einfacher und schneller kann die Implementierung umgesetzt werden. Die letzten beiden Aspekte betrifft die *Standardisierung* sowie die *Verbreitung* der Technologie.

### **Standardisierung**

OGC-Filter, CQL und GeoXACML wurden durch das OGC standardisiert [77] [52] [44]. GeoXACML befindet sich momentan in der Standardisierungsphase. Die proprietären XML-Formate und INI-Dateien sind nicht standardisiert.

### **Verbreitung**

Dadurch, dass Schnittstellen wie WMS, WFS, WCS, WPS und SOS per Standard OGC-Filter unterstützen müssen, ist diese Technologie am weitesten verbreitet. Die CQL wird durch die Unterstützung des OGCs ebenfalls gut am Markt akzeptiert und dementsprechend von den Herstellern der Geoinformationsdienste getragen. GeoXACML erfuhr noch nicht die Verbreitung von OGC-Filter oder der CQL. Häufig trifft man auch eigene XML-Dialekte, so etwa bei 52°North. Der Einsatz von INI-Dateien im GIS-Umfeld ist relativ selten.

### **Direkte Filterbarkeit von Geoinformationsdienste**

OGC-Filter und CQL werden von den meisten GIS-Diensten direkt verstanden und können einfach an die entsprechende Anfrage als Filterkriterium angehängt werden. GeoXACML, proprietäre XML-Formate und INI-Dateien werden von Geodiensten nicht direkt verstanden und müssen zuerst nach OGC-Filter oder CQL transformiert werden.

### **Umfang der Filtermöglichkeiten**

Durch die komplexen Möglichkeiten von OGC-Filter und GeoXACML kann im Prinzip jede beliebige Autorisierungsinformation in Form eines Filterausdruckes oder einer

Regel abgelegt werden. Die CQL bietet hier weniger Möglichkeiten, wurde sie auch eigentlich als Abfragesprache und weniger als Filtersprache konzipiert. Ein Vergleich mit eigenen XML- oder INI-Dateien ist nicht möglich, da über diese kaum eine allgemeine Aussage getroffen werden kann.

### **Komplexität der Gesamtarchitektur**

CQL und OGC-Filter benötigen keine aufwändige Architektur. Der Filterausdruck wird einfach an die Anfrage des Benutzers angehängt - wie dies bewerkstelligt wird, obliegt dem Entwickler des Clients oder, in diesem Szenario, dem Entwickler des AAAD bzw. des Proxydienstes. Die beiden proprietären Formate machen auch keine Vorgabe zur Architektur, jedoch ist eine Transformation der Daten in OGC-Filter oder CQL notwendig, damit die GIS-Dienste die Filter verstehen. Es wird daher eine Transformationskomponente benötigt. GeoXACML gibt durch das übergeordnete XACML eine komplexe Architektur mit PIP, PEP, PDP und PAP vor.

### **Evaluierungsergebnis des Speicherformats für geografische Zugriffsregeln**

Der OGC-Filter-Standard wird von den Geoinformationsdiensten der OGC direkt unterstützt und basiert auf bewährten Formaten wie XML und GML. Nichts spricht gegen einen Einsatz dieses Formates. Die CQL wird weniger gut unterstützt und bietet auch nicht den selben Umfang an Möglichkeiten, stellt daher also nur die 2. Wahl dar. GeoXACML ist für den Anwendungsfall zu komplex und daher nur dritte Wahl. Die Entwicklung auf Basis eines proprietären Formates erscheint nicht sinnvoll, daher belegen diese beiden Varianten Platz 4 und 5.

### 7.3.2 Gewichtete Vergleichstabelle der Formate zum Speichern von geografischen Zugriffsregeln

	Gewichtung	OGC-Filter	CQL	GeoXACML	Proprietäres XML	INI-Datei
Standardisierung	10%	+	10 Pkt.	+	6 Pkt.	-
Verbreitung	10%	+	10 Pkt.	0	4 Pkt.	-
Direkte Filterbarkeit von Geoinformationsdienste	35%	+	35 Pkt.	-	6 Pkt.	-
Umfangreiche Filtermöglichkeiten	25%	+	30 Pkt.	o	18 Pkt.	-
Komplexität der Architektur	15%	+	15 Pkt.	+	15 Pkt.	-
Ergebnis	100%	1.Wahl	100 Pkt.	2.Wahl	84 Pkt.	3.Wahl
					4.Wahl	60 Pkt.
					5.Wahl	33 Pkt.*
					o	9 Pkt.
					n/a	9 Pkt.
						31 Pkt.*

Tabelle 7.2: Gewichtete Bewertung der Formate zum Speichern von geografischen Zugriffsregeln. „+“ bedeutet eine positive Beurteilung, „-“ eine negative Beurteilung und „o“ eine neutrale Beurteilung. Der Vergleich der Formate ist als relativ zu den anderen zu interpretieren, ein „-“ bedeutet daher nicht prinzipiell eine absolute negative Bewertung des Formates, sondern eine weniger gute Bewertung in Beziehung zu den anderen angeführten Formaten. Um eine Auswahl treffen zu können, wurde „+“ mit 100 Punkten, „o“ mit 60 Punkten und „-“ mit 40 Punkten bewertet und die Punktzahl mit der jeweiligen Gewichtung multipliziert. Die beiden mit \* markierten Summen sind unvollständig, da keine Punktevergabe für das Kriterium „Umfangreiche Filtermöglichkeiten“ möglich war.

## 7.4 Übertragung von Geoauthorisierungsregeln

Die letzte zentrale Fragestellung, wann und wo die Geoauthorisierungsregeln zwischen den beteiligten Parteien ausgetauscht werden, ist durch alle vorhergehenden Überlegungen einfach zu beantworten: Sah es am Anfang dieser Arbeit, bevor konzentrierte Überlegungen in die Architektur und Protokollimplementierung flossen, noch so aus, als müssten die Geoauthorisierungsparameter mit den einzelnen Geodienstanfragen<sup>7</sup> gemeinsam übertragen werden, sei es im HTTP-Header oder in einem OAuth-Token, SAML-Token oder in einem SOAP-Envelope, so hat sich am Ende der Überlegungen ergeben, dass diese Information den AAAD nicht verlassen müssen und somit optimal geschützt sind. Dies liegt darin begründet, dass der Proxydienst ohnehin jede Anfrage beim AAAD prüfen lässt, da dieser die Gültigkeitsprüfung der Zugriffstokens und das Protokollieren der Zugriffe für die Abrechnung zentral übernimmt. Dadurch kann der Proxydienst sehr einfach gehalten werden und so auch auf Embedded Devices implementiert werden.

Da der AAAD jede Geoinformationsanfrage vom Proxydienst weitergeleitet bekommt, kann er im Zuge dessen die Anfrage entsprechend der hinterlegten geografischen Zugriffsregeln modifizieren und um entsprechende Filterausdrücke ergänzen. Der Proxydienst erhält die modifizierte Anfrage vom AAAD zurück, welche er schlussendlich an den Geoinformationsdienst weiterleitet. Der Proxy muss daher keine geografischen Zugriffsregeln verarbeiten können und bleibt so entsprechend den Anforderungen sehr einfach.

Im Sinne der „Separation of Concerns“ könnte man das Modifizieren der Anfragen des Benutzers an eine weitere Komponente, zum Beispiel einen Filterdienst, weiter delegieren. Dies sind Implementierungsdetails, die die grundlegende Architektur nicht ändern. Auf diesen Aspekt wird daher nicht näher eingegangen.

---

<sup>7</sup>WMS-Anfragen, WFS-Anfrage, WCS-Anfrage, WPS-Anfragen und SOS-Anfragen

## 7.5 Abrechnung der Zugriffe

Jede einzelne Anfrage des Endbenutzers wird zum AAAD weitergeleitet, welcher die Anfrage entsprechend der hinterlegten geografischen Zugriffsregeln filtert. Wie in Kapitel 7.3.1 dargestellt, werden die geografischen Zugriffsregeln als OGC-Filter am AAAD hinterlegt. Bei dieser Gelegenheit kann und muss der AAAD die Abrechnung des Zugriffs durchführen. Dazu könnte er die Komplexität der Anfrage bewerten und einen entsprechenden Betrag dem Guthaben des Benutzers abziehen. Die Komplexität könnte sich beispielsweise nach Anzahl der abgefragten Kartenebenen, Größe des Kartenausschnitts oder im Falle von WPS nach der Komplexität des Algorithmus richten.

Je nach Implementierung und Vertrag mit den Kunden muss der AAAD die Anfrage des Endbenutzers bewerten und entsprechend abrechnen oder er bewertet das Ergebnis der Anfrage, etwa aufgrund der Anzahl der Vektorelemente im Ergebnis.

Die einzelnen Faktoren, die den Wert einer Antwort bestimmen, wurden in Kapitel 5.4 ausführlich dargestellt. Ziel dieser Arbeit ist es jedoch nicht die Anfragen des Endbenutzers oder die Ergebnisse zu bewerten. Im Rahmen der prototypischen Implementierung wird jeder Zugriff des Benutzers mit € 1 bewertet, unabhängig davon welche Daten er abruft.

Greift der Endbenutzer schreibend auf den Kartenserver zu, um neue Informationen zu hinterlegen, könnte das Guthaben des Benutzers erhöht werden, aber auch dies sind Details, die hier nicht weiter betrachtet werden.

Aus diesen Überlegungen leiten sich folgende Informationen ab, die pro Benutzer am AAAD, etwa in einer Datenbank, hinterlegt sein müssen:

- Authentifizierungsdaten des Benutzers (Benutzername und Passwort bei lokaler Authentifizierung oder BenutzerID bei externer Authentifizierung)
- OGC-Filter für lesende Zugriffe
- OGC-Filter für schreibende Zugriffe
- Guthaben in € für die Abrechnung der Zugriffe

## 7.6 Zusammenfassung der Evaluierung bestehender Protokolle und Implementierungen zum Schutz von Geoinformationssystemen

Nach Vergleich aller zur Verfügung stehenden Möglichkeiten zeigte sich, dass *OAuth* die erste Wahl darstellt, um Autorisierungsinformationen innerhalb der verteilten, in Kapitel 6 aufgestellten, Geoinfrastruktur auszutauschen. Ausschlaggebend dafür waren vor allem der geringe Implementierungsaufwand an der Clientsoftware, eine akzeptable Komplexität des Gesamtsystems sowie ein für den Anwendungsfall angemessener Funktionsumfang. Die Technologien *Shibboleth*, *SAML* und *WS-Federation* repräsentieren bei diesem Vergleich in gleicher Reihenfolge zweite, dritte und vierte Wahl (Details siehe 7.1).

*Authentifizierungsprotokolle* wurden nicht verglichen, da komplexe Authentifizierungssysteme wie LDAP oder ActiveDirectory zumeist vom Endkunden vorgegeben bzw. einfache Authentifizierungsvorgänge ohnehin mit einem klassischen Login per Benutzername und Passwort am AAAD abgewickelt werden.

*Geoautorisierungsregeln* werden, wie ein Vergleich mehrerer Alternativen ergab (siehe 7.2), als OGC-Filter gespeichert, da Filterbefehle in diesem Format direkt von den OGC-Webservices, wie beispielsweise dem WMS oder dem WFS, verstanden werden. Durch die direkte Unterstützung müssen die Benutzeranfragen nur vom Proxydienst zum AAAD weitergeleitet werden, welcher den Filterausdruck ergänzt. Die so modifizierte Anfrage wird von den geografischen Webservices direkt und unter Berücksichtigung des Filters ausgeführt.

Diese *Geoautorisierungsregeln* werden dabei niemals zum Benutzer übertragen, denn der gesamte Filterprozess findet zwischen dem Absenden seiner Anfrage und dem Rückerhalt der Antwort statt.

Der Benutzer kann maximal am Endergebnis feststellen, dass eine Filterung stattfand, wenn er weniger Informationen erhält, als eigentlich zu erwarten wäre. Wenn er zum Beispiel die Straßenkarte der gesamten Stadt Salzburg per WMS abruft und im Stadtteil „Gnigl“ und „Parsch“ keine Straßen abgebildet sind, könnte er Rückschlüsse darauf ziehen, dass er für dieses Gebiet keine Zugriffsberechtigung hat.

Der AAAD führt zum Zwecke der *Abrechnung* Protokoll über alle Benutzeranfragen.

# 8

---

## Prototypische Implementierung zum Schutz von Geoinfrastruktur

Die in Kapitel 6 erarbeitete Architektur sowie der in Abbildung 6.5 dargestellte Kommunikationsfluss wurden als Basis für eine prototypische Implementierung verwendet. Ein Proxydienst sollte dabei Anfragen des Benutzers entgegennehmen und an den AAAD weiterleiten, damit dieser die Anfrage autorisieren oder ablehnen kann. Aus diesem Grund muss der Benutzer beim AAAD authentifiziert sein und die eingehende Anfrage des Proxydienstes dem Benutzer zuordnen können, damit er eine Autorisierungsentscheidung treffen kann. Um dies zu bewerkstelligen, könnte der Benutzer gemeinsam mit seiner Anfrage Benutzername und Passwort mitsenden, jedoch könnte ein manipulierter Proxydienst die Daten abzweigen und missbrauchen. Alternative Möglichkeiten wurden in Kapitel 6.7 aufgezeigt.

### 8.1 Implementierung von OAuth

Die Entscheidung, welches Autorisierungsprotokoll für eine prototypische Implementierung am besten geeignet ist, fiel im Kapitel 7.1 auf OAuth. Die in Kapitel 6 erarbeitete Architektur sowie der in Abbildung 6.5 dargestellte Kommunikationsfluss wurden daher so erweitert, dass die Autorisierung über OAuth stattfindet. Der so erweiterte Kommunikationsfluss ist in Abbildung 8.1 zu sehen und im Folgenden näher erklärt,

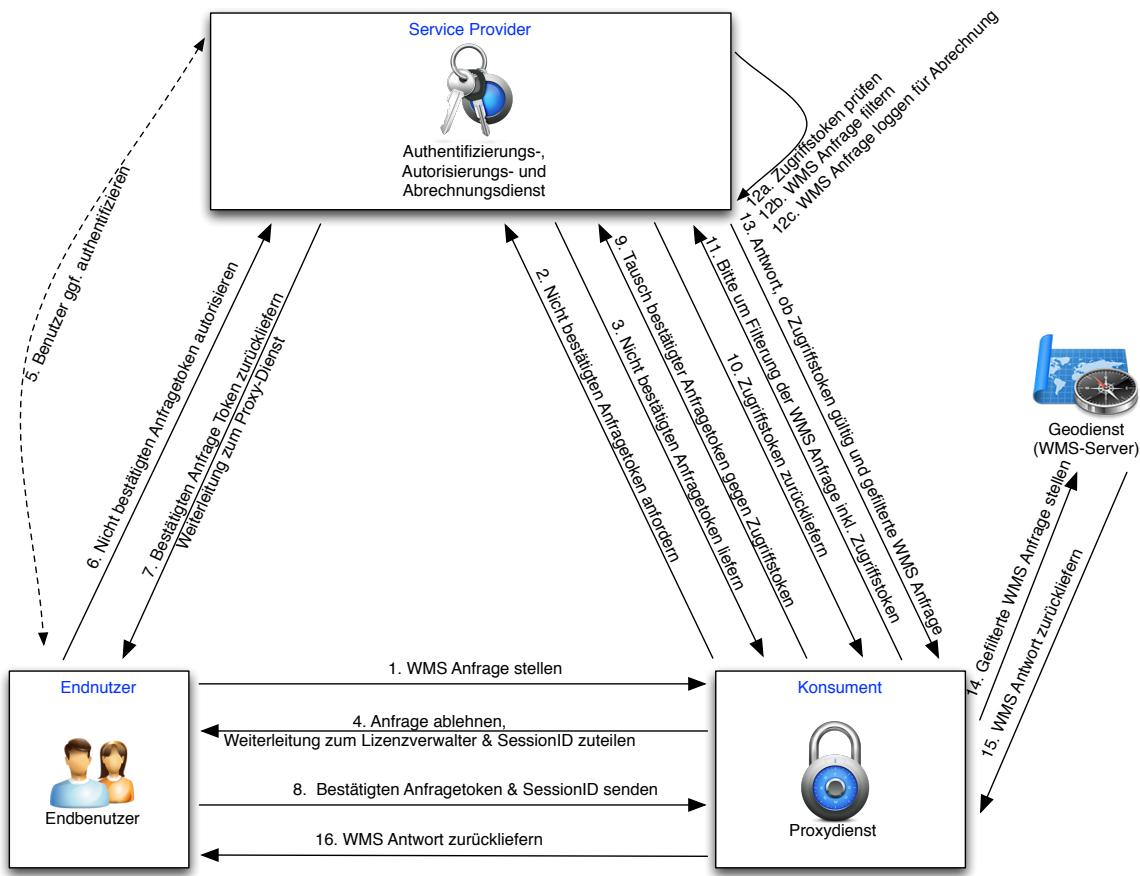


Abbildung 8.1: Darstellung des Ablaufes beim erstmaligen Zugriff auf eine geschützte WMS-Ressource.

wobei die einzelnen Nummern den Schritten der Abbildung 8.1 entsprechen. Fachbegriffe aus den OAuth-Spezifikationen sind dabei mit Schreibmaschinenschrift geschrieben, um diese klar erkennen zu können.

- Der Endbenutzer, im OAuth Jargon **Endnutzer** genannt, stellt seine WMS-Anfrage per HTTP-GET an den WMS-Geoinformationsdienst. Dem **Endnutzer** ist dabei nicht bewusst, dass er nicht direkt auf den WMS-Dienst, sondern auf einen vorgeschalteten Proxydienst zugreift.
- Der Proxydienst, **Konsument** bei OAuth genannt, fängt die WMS-Anfrage ab. Er erkennt durch Analyse des HTTP-Headers, dass der Benutzer nur eine WMS-Anfrage, aber keine SessionID gesendet hat und erkennt dadurch, dass es sich um einen neuen,

noch nicht authentifizierten Benutzer handelt. Aus diesem Grund bittet der Proxydienst den AAAD um einen sogenannten nicht bestätigten Anfragetoken. Der Konsument sendet dabei sein Konsumentengeheimnis und seine Konsumentenkennung mit.

3. Der AAAD, bei OAuth Service-Provider genannt, erkennt über die Konsumentenkennung und das Konsumentengeheimnis, um welchen Konsumenten es geht und stellt für ihn den angeforderten nicht bestätigten Anfragetoken aus.

4. Der Proxydienst antwortet dem Endnutzer mit einer Ablehnung seiner Anfrage, da er nicht authentifiziert ist. Er sendet ihm außerdem den nicht bestätigten Anfragetoken sowie eine SessionID. Die eindeutige Kennung der SessionID muss der Endnutzer nun bei jeder zukünftigen Kommunikation mit dem Proxydienst mitsenden, damit der Proxydienst weiß, mit welchem Endnutzer er zu tun hat. Der Konsument in Form des Proxydienstes bittet den Benutzer den nicht bestätigten Anfragetoken beim Service-Provider in Form des AAAD gegen einen bestätigten Anfragetoken zu tauschen. Dazu ersucht der Konsument den Endnutzer per Hyperlink oder HTTP-Weiterleitung, zum Service-Provider zu wechseln.

5. Sofern der Benutzer noch nicht eingeloggt war, muss er sich vor dem Autorisierungsvorgang beim Service-Provider authentifizieren. Dies kann durch Benutzername und Passwort oder einen anderen Authentifizierungsmechanismus erreicht werden. Die Authentifizierung selbst hat nichts mit OAuth zu tun und wurde in Abbildung 8.1 daher strichliert eingezeichnet.

6. Der Endnutzer präsentiert dem Service-Provider den nicht bestätigten Anfragetoken und bestätigt diesen. Dadurch wird er zum bestätigten Anfragetoken.

7. Der Service-Provider teilt dem Endnutzer den bestätigten Anfragetoken mit und bittet ihn per Hyperlink oder HTTP-Weiterleitung, wieder zum Konsumenten zu gehen.

8. Der Endnutzer greift wieder auf den Konsumenten zu. Diesmal übermittelt er den bestätigten Anfragetoken sowie seine SessionID.
9. + 10. Der Konsument tauscht beim Service-Provider den bestätigten Anfragetoken gegen einen Zugriffstoken aus.
11. Der Konsument, der nun im Besitz des Zugriffstoken ist, sendet die WMS-Anfrage des Endnutzers aus dem ersten Schritt an den AAAD. Er authentifiziert sich über den Zugriffstoken.
- 12.a. Der Service-Provider in Form des AAAD prüft, ob der Zugriffstoken (noch) gültig ist.
- 12.b. + 12.c. Der AAAD filtert die WMS-Anfrage laut den geografischen Zugriffsregeln des Benutzers und protokolliert den Zugriff, um eine spätere Abrechnung der Anfragen zu ermöglichen.
13. Der AAAD antwortet dem Proxydienst mit einer gefilterten WMS-Anfrage, sofern der Zugriffstoken gültig war.
14. + 15. Nun kann der Proxydienst die gefilterte WMS-Anfrage an den eigentlichen WMS-Dienst weiterleiten. Der WMS Dienst antwortet dem Fragesteller - dem Proxydienst.
16. Der Proxydienst leitet die Antwort an den Endnutzer weiter.

Einfacher wäre es, wenn der Endnutzer gleich einen bestätigten Anfragetoken vom Service-Provider anfordern und bekommen würde und diesen Token dann mit samt seiner WMS-Anfrage an den Proxydienst senden könnte. Dies widerspräche jedoch dem OAuth-Protokoll, da jeder Zugriffstoken nur von exakt einem Konsumenten genutzt werden kann und dieser individuell, und mit dem Konsumentengeheimnis signiert, für den Konsumenten ausgestellt wird. Aus diesem Grund muss der Konsument den nicht autorisierten Zugriffstoken anfordern, der Kommunikationsablauf lässt sich daher nicht vereinfachen.

Möchte der Benutzer eine weitere Anfrage stellen, vereinfacht sich der Kommunikationsfluss erheblich, wie in Abbildung 8.2 dargestellt ist. Da der Konsument bereits im

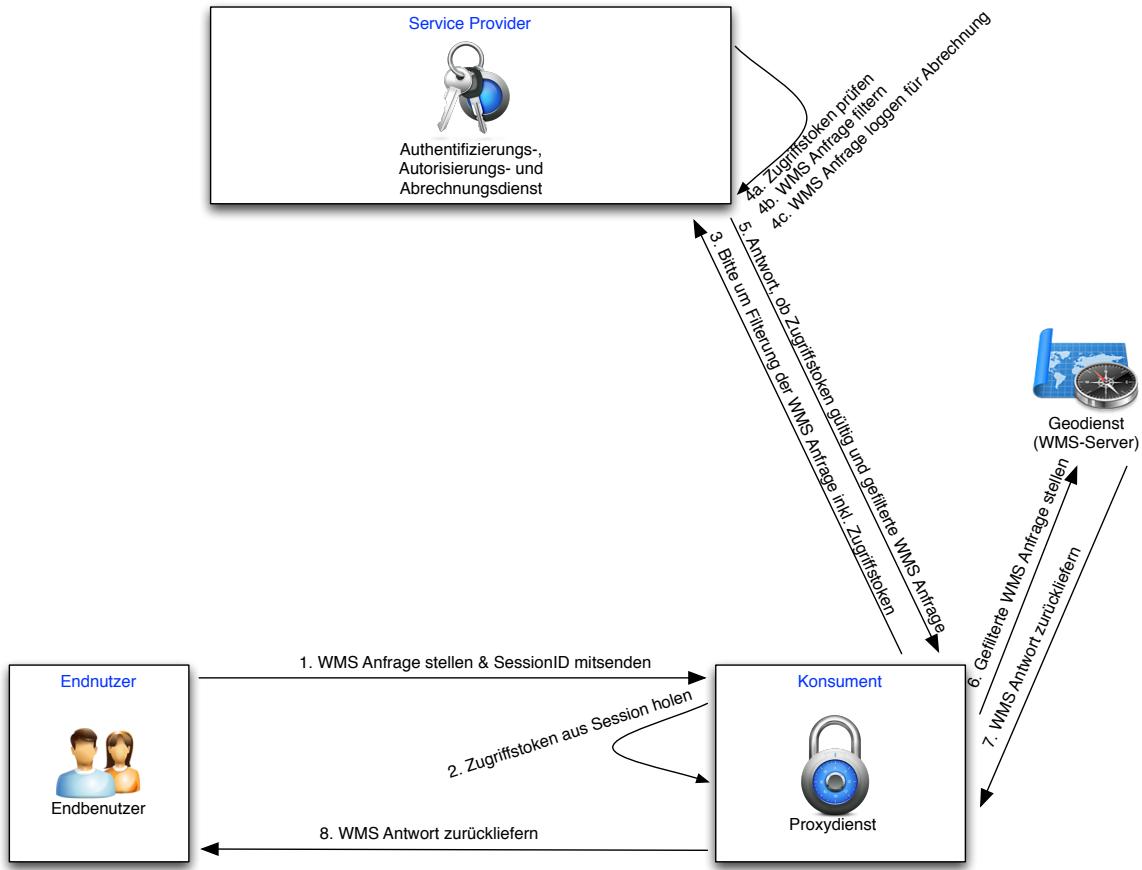


Abbildung 8.2: Darstellung des Ablaufes bei weiteren Zugriffen auf eine geschützte WMS-Ressource mit vorhandener SessionID und gültigem Zugriffstoken.

Besitz des Zugriffstokens ist, kann er direkt im Namen des Benutzers entsprechende Filteranfragen beim AAAD stellen.

Im Folgenden ist der Ablauf im Detail geschildert:

1. Der Endnutzer stellt seine WMS-Anfrage und sendet seine eindeutige SessionID mit seiner Anfrage.
2. Der Konsument holt aus dem Arbeitsspeicher (oder aus einer Datenbank) den Zugriffstoken zur dazu passenden SessionID.
3. Der Konsument schickt die WMS-Anfrage gemeinsam mit dem Zugriffstoken an den AAAD.
- 4.a. Dieser prüft, ob der Zugriffstoken (noch) gültig ist.

4.b. + 4.c. Der AAAD filtert die WMS-Anfrage gemäß der hinterlegten Regeln und protokolliert den Zugriff.

5. Der AAAD antwortet dem Proxydienst mit der gefilterten WMS-Anfrage.

6. + 7. + 8. Der Proxy Dienst leitet die gefilterte Anfrage an den WMS-Dienst weiter, ebenso wie er dessen Antwort an den Endnutzer durchschleust.

Zu implementieren waren drei beteiligte Komponenten - der Endbenutzer, der Proxydienst sowie der AAAD. Der WMS-Dienst wurde in Form des GeoServers als gegeben angenommen. Als Programmiersprache wurde für alle selbst implementierten Komponenten PHP gewählt, da der Autor bereits über fundierte PHP Kenntnisse verfügt sowie PHP kostenlos und plattformunabhängig eingesetzt werden kann.

Um nicht jedes Detail selbst implementieren zu müssen, wurden fertige OpenSource-Bibliotheken für PHP ausgesucht und die Implementierung darauf aufgebaut.

## 8.2 Softwarekomponenten des Endnutzers

Der Endbenutzer nutzt im einfachsten Fall einen gewöhnlichen Webbrowser mit aktivierte Cookies, um darin die SessionID zu speichern. Damit der derjenige, der den Webbrowser bedient, ein klein wenig Komfort und Information erhält, wurde ein rudimentäres Eingabeinterface mit erklärender Willkommensseite geschaffen, wie Abbildung 8.3 zeigt.

Der Endbenutzer gibt dabei seine WMS-Anfrage in das Eingabefeld „URL“ ein und schickt die Anfrage an den eingebetteten HTML-iFrame.

## 8.3 Softwarekomponenten des Konsumenten

Der Konsument hat mehrere Aufgaben. Zunächst muss er die Sessions der Endbenutzer verwalten. Dieser Mechanismus ist in PHP bereits eingebaut und musste daher nur aktiviert werden. Für die Abwicklung der Kommunikation im Rahmen des

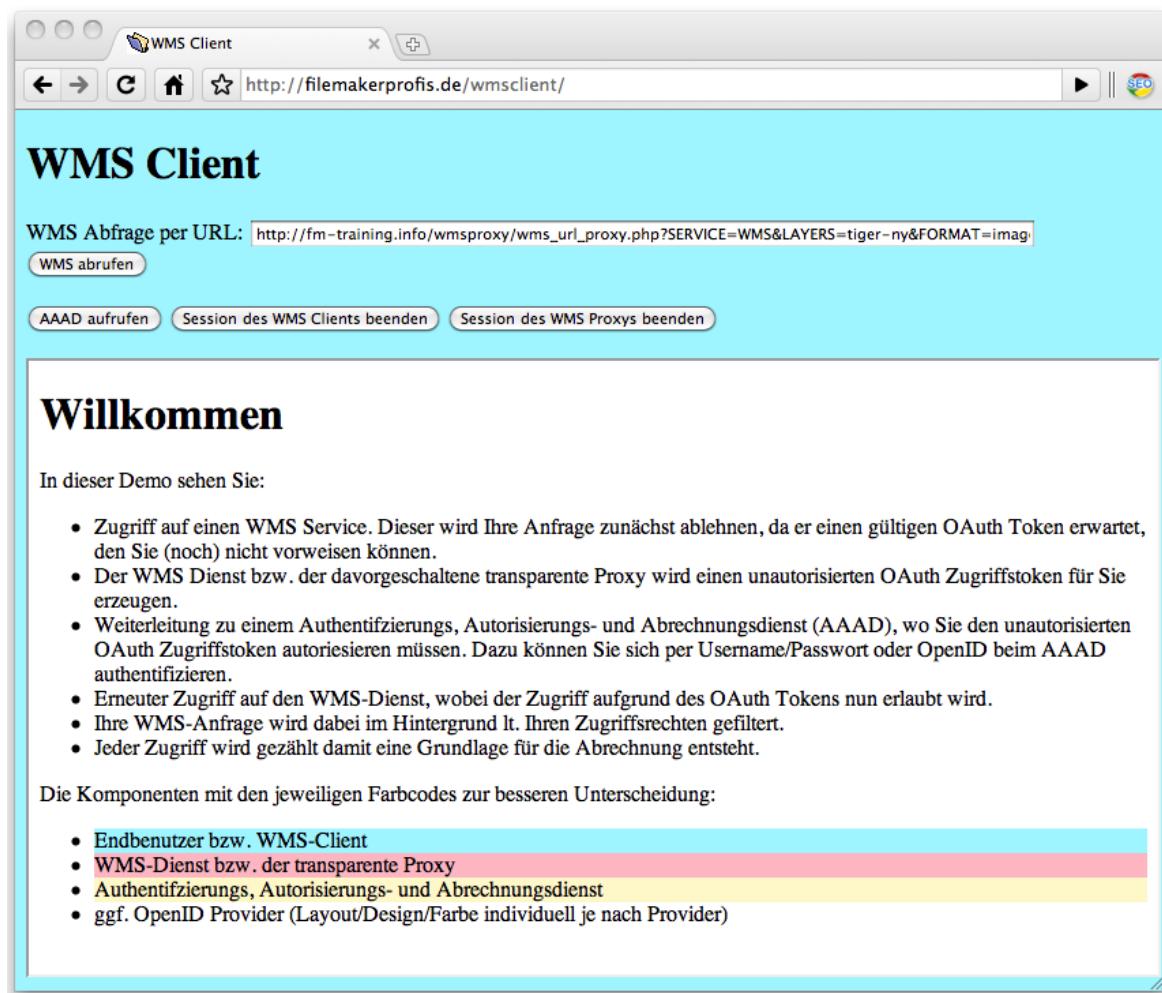


Abbildung 8.3: Screenshot der Willkommensseite des WMS-Clients mit kurzen Erklärungen.

Token-Austausches sowie für den Zugriff auf den Service-Provider wurde die „Zend<sup>1</sup> OAuth“-Bibliothek verwendet. Um die WMS-Anfragen an den WMS-Dienst in Form des GeoServers weiterzuleiten und dessen Antwort auszuwerten, wurde die „Zend Http-Client“-Bibliothek verwendet.

<sup>1</sup><http://www.zend.com>

## 8.4 Softwarekomponenten des Service-Providers

Der Service-Provider stammt zum Großteil von Arjan Scherpenisse's<sup>2</sup> Bibliothek und Beispielimplementierung „OAuth Consumer And Server Library For PHP“. Sein Programm wurde um eine Benutzerdatenbank erweitert, sodass der Service-Provider von mehreren Benutzern mit unterschiedlichen Rechten gleichzeitig benutzt werden kann. Der OAuth-Service in Arjan Scherpenisse's Beispielimplementierung bestand aus einer Echo-Demoanwendung, die alle Eingabewerte einfach wieder ausgab. Diese wurde so abgeändert, dass sie WMS-Anfragen entgegennimmt, aus dem Zugriffstoken die Benutzer-ID ausliest und nach hinterlegten Filtern in der Datenbank sucht. Danach wird die eingehende WMS-Anfrage, um die in der Datenbank hinterlegten Filterkriterien ergänzt, der Zugriff protokolliert und zurück an den Aufrufer gesendet.

Neben der Login-Möglichkeit mit Benutzername und Passwort wurde eine Authentifizierung per OpenID realisiert. Um dies zu bewerkstelligen, kam die „Zend OpenID“-Bibliothek zum Einsatz.

## 8.5 Anwendung der Servicearchitektur aus Benutzersicht

Nachdem alle Komponenten mit Hilfe der verfügbaren Bibliotheken implementiert wurden und lokale Tests auf einem Webserver wie erwartet funktionierten, installierte der Autor die drei Komponenten auf drei verschiedenen Servern unter drei dem Autor zur Verfügung stehenden Domains und färbte jede Komponenten mit einer individuellen Farbe, wie in den Abbildungen 8.3, 8.4, 8.5, 8.6 und 8.7 zu sehen ist.

- Der WMS-Client wurde unter der URL `http://www.filemakerprofis.de/wmsclient` installiert und mit einem türkisen Hintergrund versehen.
- Der WMS-Proxy wurde unter `http://www.fm-training.info/wmsproxy` installiert und trägt rosa als Hintergrundfarbe.
- Der AAAD unter `http://www.schubec.com/aaad` wurde gelb gefärbt.

---

<sup>2</sup><http://scherpenisse.net>

Der Ablauf aus Benzersicht ist in den Abbildungen 8.3, 8.4, 8.5 und 8.6 zu sehen und im Folgenden näher erklärt.

Um das System zu nutzen, ruft der Anwender den Client unter `http://www.filemakerprofis.de/wmsclient` auf, um über diesen eine WMS-Anfrage wie `http://fm-training.info/wmsproxy/wms_url_proxy.php?SERVICE=WMS&LAYERS=tiger-ny&FORMAT=image/png&transparent=true&REQUEST=GetMap&BBOX=-74.03273,40.6987,-73.92765,40.80657&WIDTH=860&HEIGHT=883&SR=EPSG:4326&VERSION=1.3.0` zu stellen. Da man im iFrame des Clients jedoch nicht die URLs sieht, die im Rahmen des Kommunikationsflusses aufgerufen werden, wurde bei den folgenden Screenshots auf die Nutzung des Clients mit iFrame verzichtet und die URL mit der WMS-Anfrage direkt in den Browser eingegeben.

Im Schritt 1 in Abbildung 8.4 stellt der Client seine WMS-Anfrage, welche vom Proxydienst abgefangen wird. Im Hintergrund holt er vom AAAD einen nicht autorisierten Zugriffstoken und möchte den Endbenutzer zum Service-Provider weiterleiten. Im Produktivbetrieb würde dies automatisch per HTTP-Weiterleitung geschehen, zu Demonstrationszwecken wird in der Beispielimplementierung jedoch ein Hyperlink mit erklärendem Text gezeigt. Geschulte Leser erkennen in der URL sowohl den OAuth Token sowie die Callback-URL, also die URL, die der Service-Provider aufrufen soll, sobald der Benutzer den Anfragetoken autorisiert hat.

Der Benutzer klickt auf den Hyperlink und wird im Schritt 2 vom AAAD empfangen, der den Zugriff verweigert, da der Benutzer nicht authentifiziert ist. Der Benutzer kann nun auswählen, ob er sich mit OpenID oder per Benutzername und Passwort authentifizieren möchte.

Entscheidet sich der Benutzer für die Authentifizierung per Benutzername und Passwort, kann er diese im Schritt 3a angeben und gelangt direkt zum Schritt 6a in Abbildung 8.5.

Bei einer Authentifizierung per OpenID gibt der Benutzer im Schritt 3b seine OpenID an und wird dann im Schritt 4 zu seinem OpenID-Provider weitergeleitet, wo er sich authentifizieren muss.

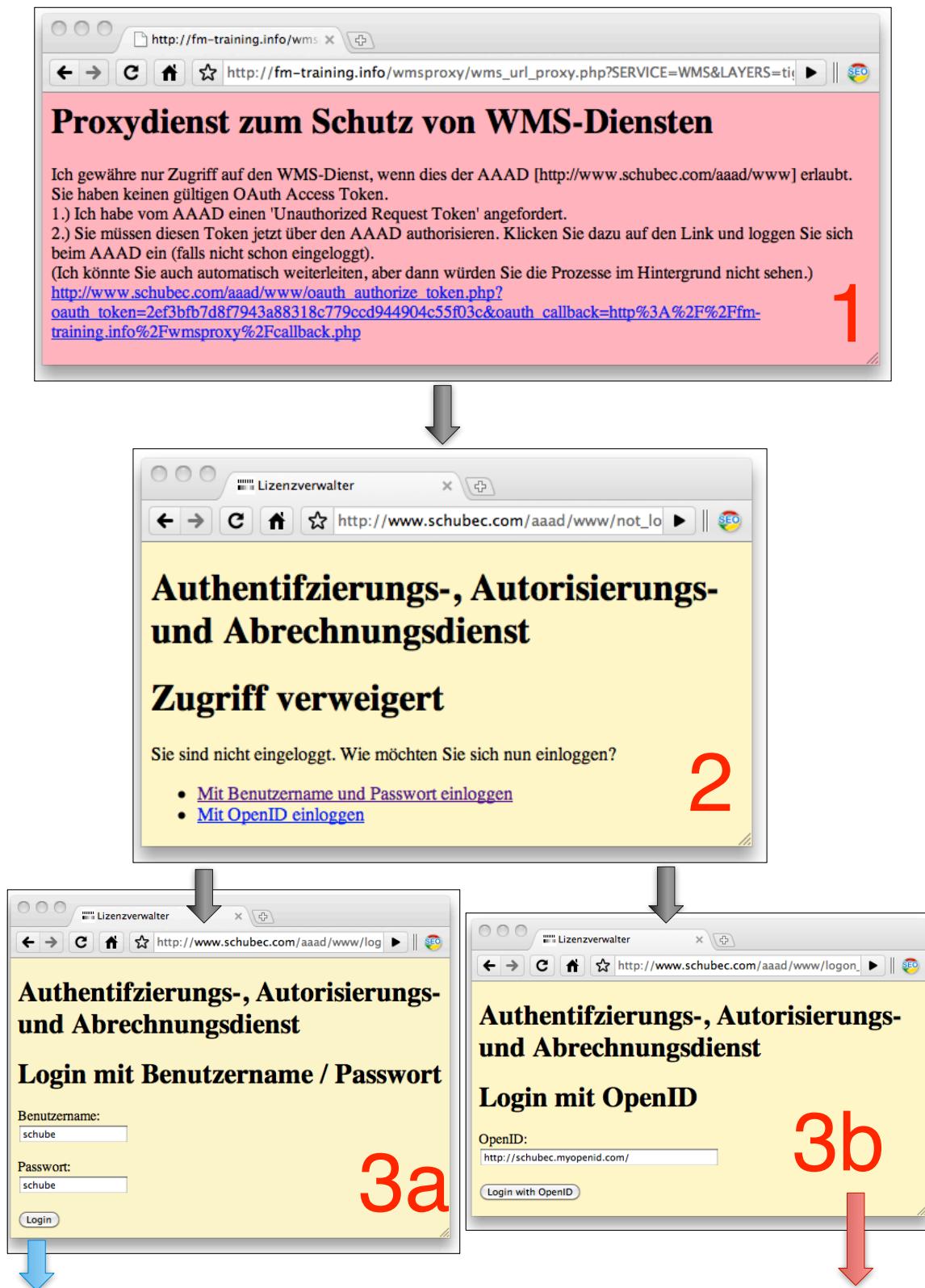


Abbildung 8.4: Abbildung 1 von 3 mit Screenshots beim Benutzerzugriff auf einen geschützten WMS-Dienst. Genaue Erklärung des Ablaufs unter 8.5.

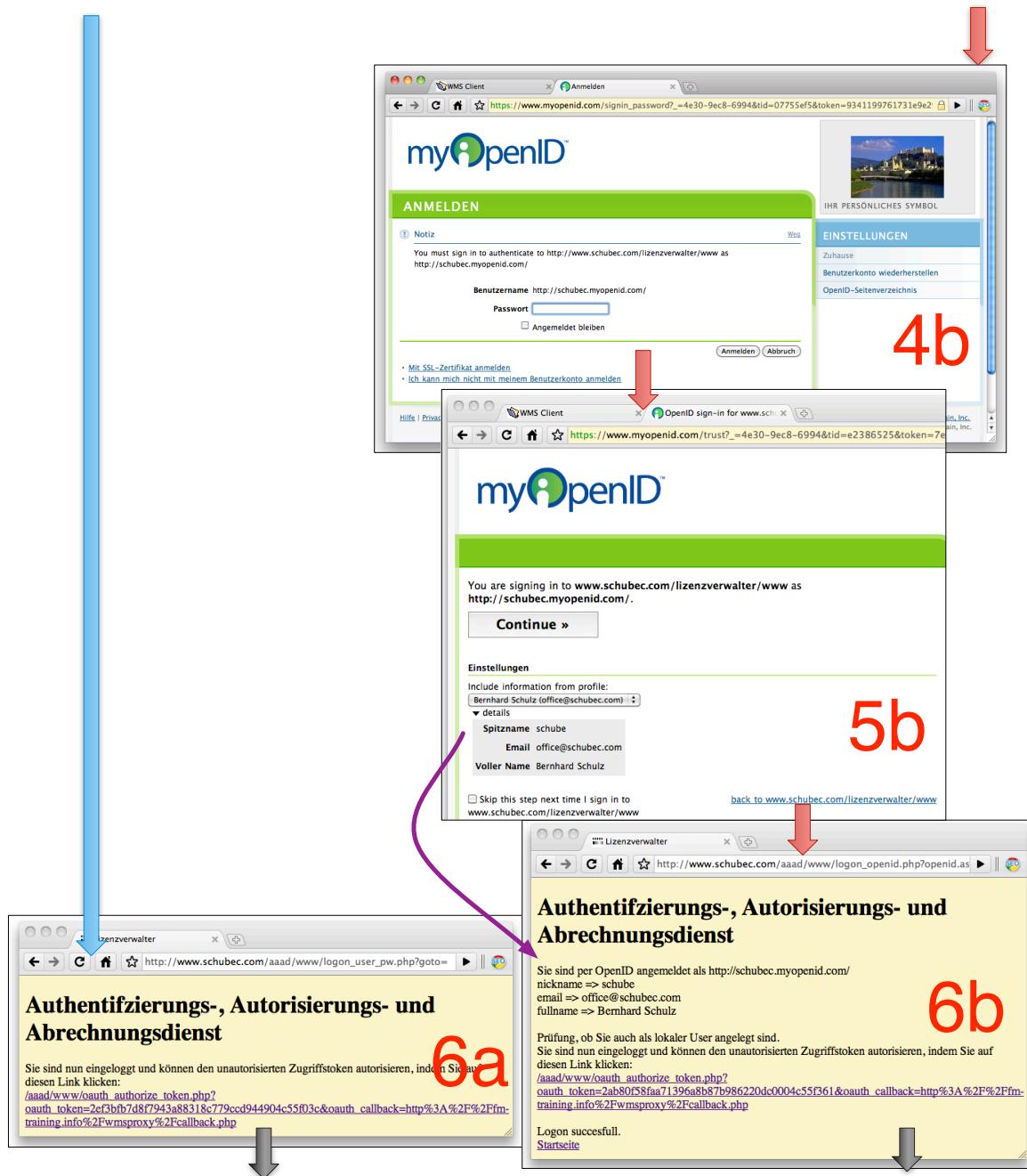


Abbildung 8.5: Abbildung 2 von 3 mit Screenshots beim Benutzerzugriff auf einen geschützten WMS-Dienst. Genaue Erklärung des Ablaufs unter 8.5.

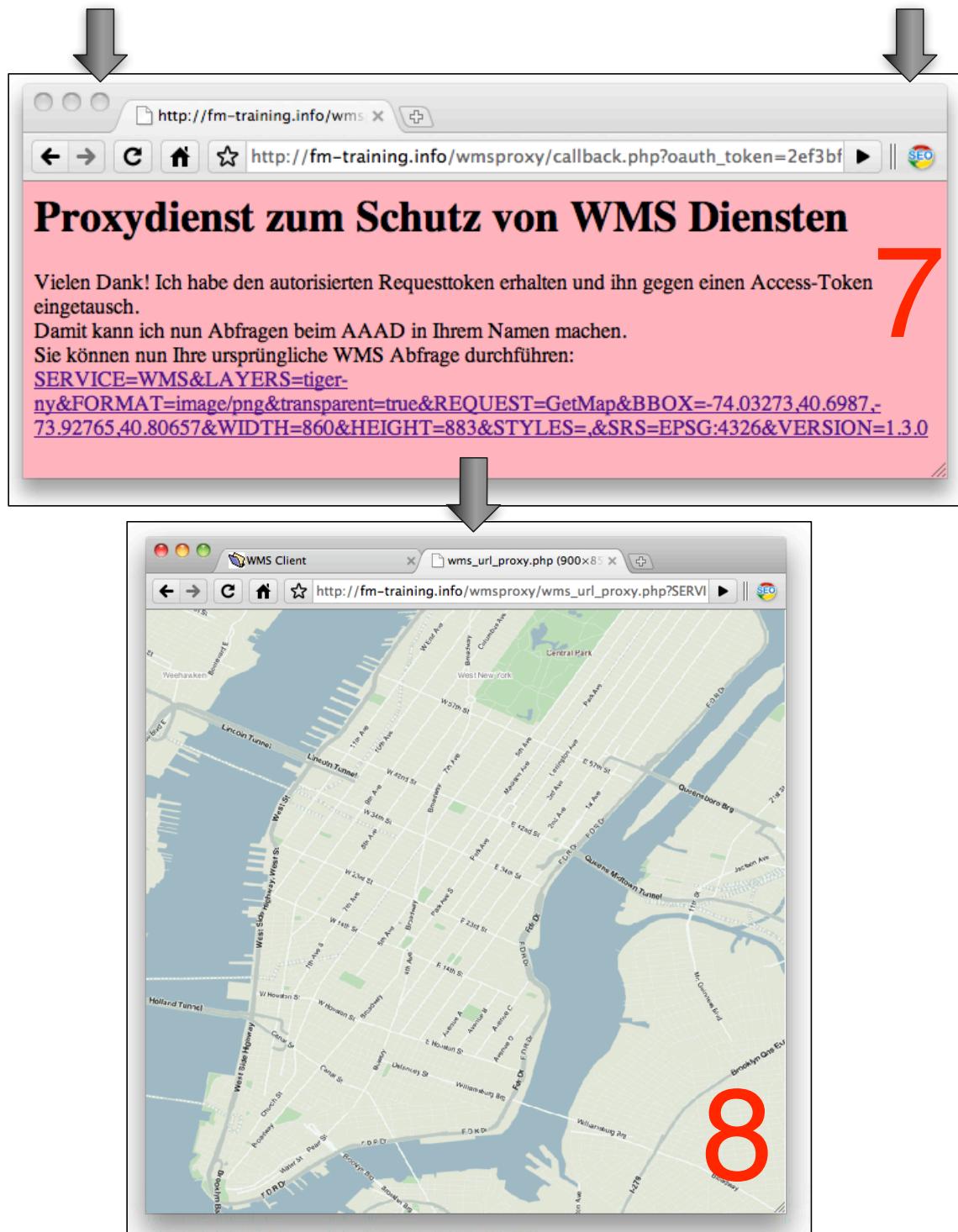


Abbildung 8.6: Abbildung 3 von 3 mit Screenshots beim Benutzerzugriff auf einen geschützten WMS-Dienst. Genaue Erklärung des Ablaufs unter 8.5.



Abbildung 8.7: Screenshot der Zugriffsverweigerung wegen ungültigem Zugriffstoken.

Im Schritt 5b kann er dem OpenID-Provider bekannt geben, dass er dem AAAD die angeforderten Authentifizierungsinformationen mitteilen darf. Im Rahmen der Beispielimplementierung fordert der AAAD noch zusätzlich Informationen wie den Spitznamen, die Emailadresse und den vollen Namen des Benutzers vom OpenID-Provider an. Diese werden nach der Autorisierung der Authentifizierungsinformationen ebenfalls zum AAAD übertragen.

Im Schritt 6a bzw. 6b ist der Benutzer gegenüber dem AAAD authentifiziert und kann mit einem Klick auf den entsprechenden Hyperlink den nicht bestätigten Anfragetoken in einen bestätigten Anfragetoken wechseln und zum WMS-Proxy weiterleiten.

Wie im Schritt 7 in Abbildung 8.6 dargestellt, hat der WMS-Proxy den autorisierten Anfragetoken erhalten und im Hintergrund gegen einen Zugriffstoken getauscht. Damit ist er nun in der Lage, die ursprüngliche Anfrage durchzuführen, indem der Benutzer auf den Hyperlink klickt. Im Hintergrund passiert die Prüfung des Zugriffstokens sowie die Filterung der WMS-Anfrage, wie in Abbildung 8.1 dargestellt.

Schlussendlich erhält der Benutzer die angeforderte Karte im Schritt 8.

Wird ein Zugriffstoken ungültig, sei es, dass seine Gültigkeitszeit abgelaufen ist oder dass der Benutzer beim Service-Provider den Zugriffstoken gelöscht hat, werden die Filteranfragen des WMS-Proxys beim Service-Provider mit einer Fehlermeldung beantwortet und der WMS-Proxy verbietet den Zugriff auf die geschützte Ressource (Abbildung 8.7). In diesem Fall muss der Benutzer die aktuelle Session beenden, eine

neue starten und den gesamten Autorisierungsvorgang (und gegebenenfalls auch den Authentifizierungsvorgang) erneut durchgehen.

## 8.6 Filtern der Anfragen

Die vom Benutzer gestellten Anfragen werden vom Proxydienst zum AAAD weitergeleitet, welcher die Anfragen um die hinterlegten geografischen Filterkriterien im OGC-Filter-Format ergänzt. Hat der Benutzer seine Anfrage ohne OGC-Filter-Befehl abgesetzt, ergänzt der AAAD die Anfrage des Benutzers mit dem hinterlegten, benutzerspezifischen OGC-Filter. Falls der Benutzer selbst bereits einen OGC-Filter-Befehl mit seiner Anfrage mitgesandt hat, so muss der AAAD den Filter vom Benutzer sowie den hinterlegten Filter mit einem logischen AND-Befehl verknüpfen. Hierbei müssen die zwei OGC-Filter, die im XML-Format vorliegen, mit entsprechenden XML-Tools verbunden werden. Aus zeitlichen Gründen wurde im Rahmen dieser Arbeit kein Modul zum Verknüpfen der Filter entworfen. Es wird daher davon ausgegangen, dass der Benutzer keinen OGC-Filter-Befehl mitsendet.

Listing 8.1: Komplexer OGC-Filter mit zwei per logischem OR-Befehl verknüpften Bounding Boxes.

---

```

<ogc:Filter xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" >
    xmlns:gmgml="http://www.intergraph.com/geomedia/gml">
180    <ogc:Or>
        <ogc:BBOX>
182        <ogc:PropertyName>the\_geom</ogc:PropertyName>
        <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#\#4326">
184            <gml:coordinates>-73.985085,40.76355 -73.94613,40.803</gml:coordinates>
        </gml:Box >
186    </ogc:BBOX>
        <ogc:BBOX>
188        <ogc:PropertyName>the\_geom</ogc:PropertyName>
        <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#\#4326">
190            <gml:coordinates>-74.02402,40.69801 -73.98478,40.723484</gml:coordinates>
        </gml:Box >
192    </ogc:BBOX>
    </ogc:Or>
194 </ogc:Filter>

```

---

Um den prinzipiellen Nachweis zu erbringen, dass das Verknüpfen von einzelnen OGC-Filter-Ausdrücken in einer Geodienst-Anfrage funktioniert, wurden zwei einfache Bounding Boxes durch zwei OGC-Filter-Audrücke definiert und mit einem logischen OR-Befehl verknüpft. Daraus ergab sich der in Listing 8.1 dargestellte, komplexere OGC-Filter-Befehl. Die beiden Bounding Boxes definieren die beiden in Abbildung 8.9 grün gefärbten Flächen.

Die ungefilterte Anfrage `http://localhost/geoserver/wms?SERVICE=WMS&LAYERS=tiger-ny&FORMAT=image/png&transparent=true&REQUEST=GetMap&BBOX=-74.032730,40.69870,-73.92765,40.80657&WIDTH=860&HEIGHT=883&STYLES=&SRS=EPSG:4326&VERSION=1.3.0` des Benutzers resultiert in der Karte von Manhattan, welche in Abbildung 8.8 dargestellt ist. Hinterlegt man am AAD den Filter aus Listing 8.1, so resultiert dies in einer gefilterten Karte, welche in Abbildung 8.10 zu sehen ist. Wie man der Abbildung 8.10 entnehmen kann, werden beide Bounding Boxes berücksichtigt und der Benutzer erhält ein entsprechend gefiltertes Ergebnis.

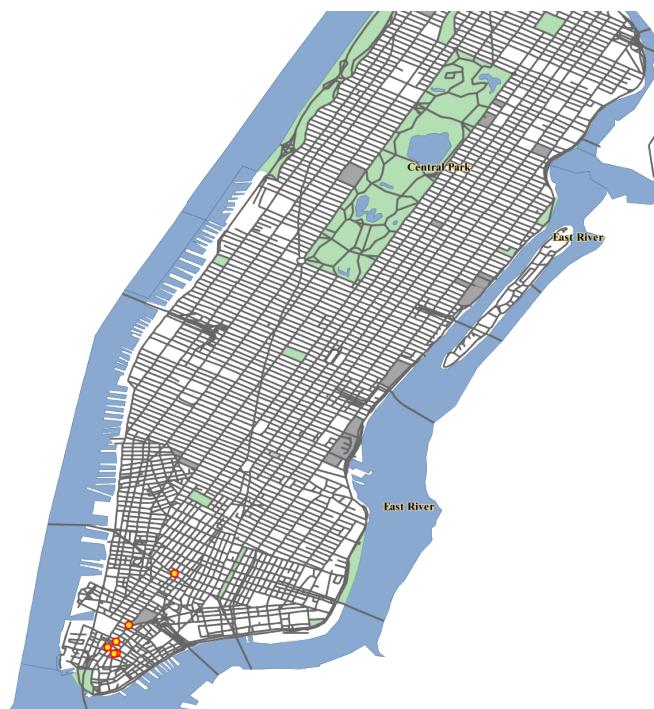


Abbildung 8.8: Ergebnis der ungefilterten WMS-Anfrage.

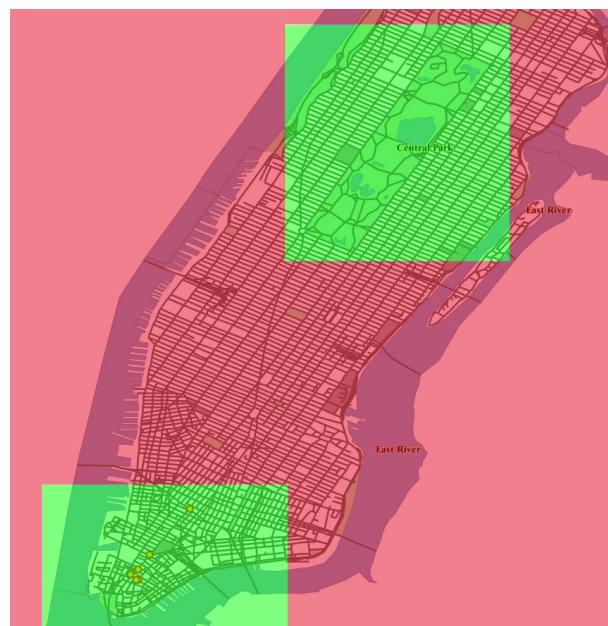


Abbildung 8.9: Definition zweier Bounding Box-Filter. Der Benutzer erhält Zugriff auf die beiden grün markierten Bereiche. Der Zugriff auf den roten Bereich wird unterbunden.

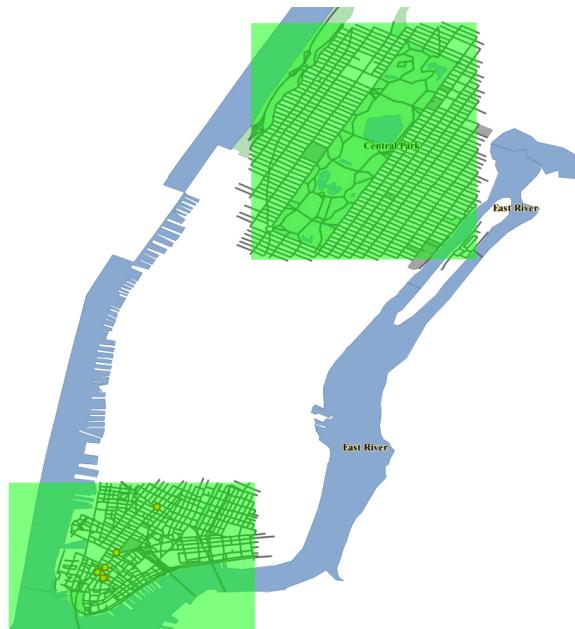


Abbildung 8.10: Ergebnis der gefilterten WMS-Anfrage. Zusätzlich wurden die Flächen, auf die der Benutzer Zugriff hat, mit einem Bildbearbeitungsprogramm zur Visualisierung nachträglich grün eingefärbt. Elemente, die die erlaubten Bereiche schneiden, werden nicht ausgefiltert, daher sind die Wasserflächen durchgehend sichtbar.

Es fällt auf, dass geografische Strukturen wie der „East River“ und der „Hudson“, die eine oder beide Bounding Boxes schneiden, vollständig im Ergebnis enthalten sind. Dies ist kein Fehler, sondern entspricht dem OGC-Filter-Standard [77].

## 8.7 Diskussion der Umsetzung

Die prototypische Implementierung mit der Programmiersprache PHP und die anschließenden Tests dieser Umsetzung ergaben, dass die in Kapitel 6 erarbeitete verteilte serviceorientierte Sicherheitsarchitektur mit den in Kapitel 7 ausgewählten Protokollen und Technologien prinzipiell funktioniert. Unautorisierte Zugriffe auf den Geoinformationsdienst werden zuverlässig unterbunden. Erst nach Authentifizierung per Benutzername und Passwort oder per OpenID werden die Anfragen des Benutzers vom Geoinformationsdienst ausgeführt, nicht jedoch ohne zuvor die hinterlegten Geoautorisierungsregeln in Form von OGC-Filter zu berücksichtigen.

Die derzeitige Implementierung geht davon aus, dass der Endbenutzer keine eigenen

OGC-Filter-Befehle in seiner Anfrage mitsendet. Es wäre problemlos möglich, die Anfrage des Benutzers genauer zu analysieren und die OGC-Filter-Befehle des Benutzers, sofern vorhanden, mit den als Geozugriffsregeln hinterlegten OGC-Filter-Ausdrücken zu kombinieren, jedoch wurde dies aus zeitlichen Gründen nicht umgesetzt. Um dies zu bewerkstelligen, müssten beide OGC-Filter-Befehle als XML-Struktur eingelesen und verknüpft werden. Könnte man sich darauf verlassen, dass der Benutzer dieselben XML-Namespace verwendet, wie die am AAAD hinterlegten Regeln, so könnte man beide OGC-Filter mit einfachen String-Operationen verknüpfen. Da der Benutzer jedoch durch die Flexibilität von XML die Möglichkeit hat, eigene XML-Namespace zu definieren, müssen beide OGC-Filter durch entsprechende XML-Verarbeitungsroutinen zusammengeführt werden, welche die XML-Namespacekollisionen oder -überlappungen erkennen und auflösen können.

Die prototypische Implementierung zeigt zu Demonstrationszwecken viele einzelne Schritte im Webbrowser des Benutzers an und wartet auf das Aktivieren von Hyperlinks, um weitere Schritte auszuführen. In einer realen Anwendung würde man hier statt Hyperlinks direkte HTTP-Weiterleitungen verwenden, um die komplexen Schritte vor dem Benutzer zu verstecken.

Als Programmiersprache wurde PHP gewählt, da diese Programmierumgebung dem Autor bekannt ist und die für die Implementierung benötigten Bibliotheken für OAuth und OpenID kostenlos zur Verfügung stehen. Nicht optimal beim Einsatz dieser Bibliotheken ist jedoch, dass die OAuth Bibliothek von Zend nur den Konsumententeil am Proxydienst abdecken konnte, da es zum Zeitpunkt dieser Arbeit keine OAuth-Service-Provider Implementierung von Zend gab. Weiter zu bemängeln ist, dass die OpenID Bibliothek von Zend die Version 2.0 von OpenID noch nicht unterstützt und somit ein Login über einen Google-Account momentan nicht möglich ist. Hier gibt es Potential für Verbesserungen, wie in Kapitel 9 dargestellt ist.

Dem gefilterten Ergebnis der WMS-Anfrage des Benutzers ist momentan nicht zu entnehmen, ob Bereiche herausgefiltert wurden oder ob an den blanken Stellen tatsächlich keine Daten vorliegen. Bei Karten bekannter Orte wie Manhattan fällt die Filterung auf, wie in Abbildung 8.10 zu sehen ist, doch in unbekannten oder weniger besiedelten Gebieten ist es für den Benutzer nicht ersichtlich, aus welchem Grund in gewissen

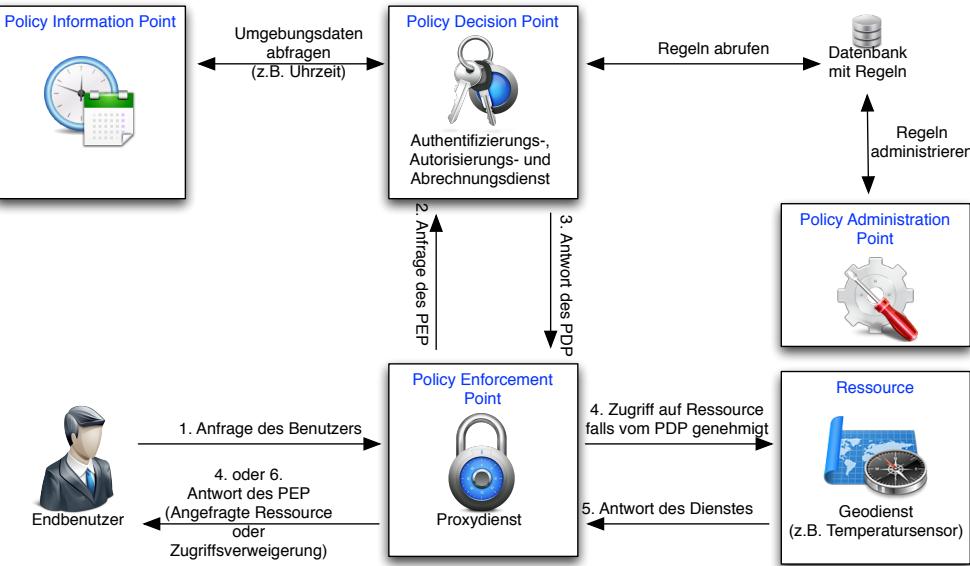


Abbildung 8.11: Vergleich der abgeleiteten Sicherheitsarchitektur mit GeoXACML. Die blaue Farbe repräsentiert die Bezeichnung der Komponenten in einer XACML Infrastruktur, die schwarze Beschriftung der Komponenten entspricht den Namen der in dieser Arbeit abgeleiteten serviceorientierten Sicherheitsarchitektur.

Regionen keine Informationen vorliegen. Hier wäre zu überlegen, ob man die weggeföhlerten Bereiche markiert, etwa durch eine rote Schraffierung, um den Benutzer auf die für ihn gesperrten Zonen hinzuweisen.

Vergleicht man die im Rahmen dieser Arbeit abgeleitete Architektur mit der Architektur von XACML bzw. GeoXACML mit PDP und PEP, so ergeben sich interessante Parallelen. Der XACML-PEP entspricht dem Proxydienst und der XACML-PDP dem AAAD. Die Administration der geografischen Zugriffsregeln wurde im Rahmen dieser Arbeit direkt in die Datenbankverwaltungssoftware phpMyAdmin<sup>3</sup> eingegeben, welcher daher dem PAP entspricht. Im Prinzip wurde die Basis für eine GeoXACML Infrastruktur gelegt, ohne jedoch die strikten Formalismen des GeoXACML Standards einzuhalten, wie man Abbildung 8.11 entnehmen kann.

<sup>3</sup><http://www.phpmyadmin.net>

## 8.8 Zusammenfassung der prototypischen Implementierung

Mit frei zur Verfügung stehenden Bibliotheken wie etwa dem Zend-Framework<sup>4</sup> wurde die in Kapitel 6 entwickelte serviceorientierte Sicherheitsarchitektur für Geoinformationsdienste mit Hilfe von OAuth, welches sich in Kapitel 7.1 als das für den Anwendungsfall geeignete Autorisierungsprotokoll für verteilte Dienste herausgestellt hat, mit der Programmiersprache PHP für WMS-Anfragen prototypisch umgesetzt. Die Authentifizierung erfolgt dabei mit einem Login per Benutzername und Passwort oder alternativ per OpenID. Der AAAD filtert die WMS-Anfragen des Benutzers aufgrund der im OGC-Filter-Format hinterlegten geografischen Zugriffsregeln.

Tests der prototypischen Implementierung ergaben, dass der WMS-Dienst zuverlässig gegen nicht autorisierte Zugriffe geschützt wird.

Die momentane prototypische Implementierung geht davon aus, dass der Benutzer keine eigenen OGC-Filter-Befehl in seiner Anfrage mitsendet. Im Rahmen einer Weiterentwicklung für einen produktiven Einsatz sollte diese Einschränkung durch entsprechende Programmierung aufgehoben werden. Des weiteren könnte der AAAD sowie der Proxydienst um weitere Protokolle wie WFS, WCS und SOS ergänzt werden.

---

<sup>4</sup><http://framework.zend.com>

# 9

---

## Resümee, Weiterentwicklung und Ausblick

Nachdem die dieser Arbeit zugrundeliegenden theoretischen Überlegungen im Zuge der praktischen Implementierung erfolgreich umgesetzt wurden, werden nun am Ende dieser Arbeit ein Resümee und Verbesserungsvorschläge für zukünftige Erweiterungen gegeben.

### 9.1 Resümee

Diese Arbeit behandelt, geprägt durch die Zusammenarbeit mit dem Research Studio iSpace Salzburg, verteilte Geoinformationssysteme. Das Ziel, diese Geoinformationssysteme zu schützen und die Zugriffe zu protokollieren, wurde erfüllt. Neben der begründeten Ableitung einer serviceorientierten Sicherheitsarchitektur für Geoinformationssysteme wurden passende Protokolle und Technologien gefunden, um Geozugriffsregeln abzuspeichern und einfach anwenden zu können. Es ist dabei möglich, dem Benutzer nur gewisse Regionen oder Kategorien zugänglich zu machen. Eine zentrale Komponente deckt dabei die drei Aspekte Authentifizierung, Autorisierung und Abrechnung eines Triple-A-Systems ab.

Durch den Einsatz des lizenzkostenfreien und in der Komplexität überschaubaren Protokolls OAuth wurde eine technische Grundlage geschaffen, die mit vertretbaren Ressourcen in bestehende Systeme eingebaut werden kann. Eine prototypische Implementierung unter einer OpenSource-Lizenz in der Programmiersprache PHP liegt vor und wurde verwendet, um die Ergebnisse zu verifizieren.

Das Einbinden externer Authentifizierungssysteme wurde durch die Implementierung von OpenID erfolgreich verifiziert, sodass auch größere Unternehmen oder Behörden derart geschützte Geoinfrastruktur durch den Einsatz ihrer eigenen Authentifizierungssysteme mit minimalen administrativen Kosten nutzen können.

Die im Rahmen dieser Arbeit abgeleitete serviceorientierten Sicherheitsarchitektur ermöglicht durch den modularen Aufbau auch den Schutz und gegebenenfalls die Abrechnung anderer webbasierter Dienste.

## 9.2 Weiterentwicklung und Ausblick

Die prototypische Implementierung im Rahmen dieser Arbeit wurde aus zeitlichen Gründen ausschließlich für den WMS-Dienst umgesetzt. Die Antworten von WMS-Diensten in Form von Grafiken erlauben eine direkte visuelle Überprüfung des Ergebnisses durch Betrachten der Bilder. Um die Ergebnisse von Filterbefehlen maschinell prüfen zu können, wäre die Unterstützung von WFS-Anfragen wünschenswert, da das Ergebnis in Form von Vektordaten im XML-Format besser automatisierbar auswertbar ist als Bilddaten.

Der nächste Schritt wäre daher eine Erweiterung des Proxydienstes und des AAAD, um WFS-Anfragen bearbeiten zu können. WFS-Anfragen, genau so wie WCS-Anfragen, unterscheiden sich von WMS-Anfragen syntaktisch kaum, eine Unterstützung dieser Formate sollte daher nach Meinung des Autors mit relativ wenig Aufwand umsetzbar sein. Dieselbe Annahme trifft für die Sensordienste SOS und SAS zu. Hierbei müssen jedoch andere OGC-Filter Befehle am AAAD hinterlegt werden, da die Ergebnisse nach anderen Kriterien, etwa der Zeitspanne der Sensordaten, nicht jedoch nach Kartenebenen, gefiltert werden.

Deutlich aufwändiger ist die Unterstützung von WPS-Diensten, da beim Einsatz von WPS unter Umständen mehrere Geoinformationsdienste indirekt genutzt werden, die der WPS-Dienst im Namen des Benutzers aufruft, wie Abbildung 5.6 zeigt. Dabei ist der Autorisierungsprozess bei jedem indirekt verwendetem Geoinformationsdienst entsprechend zu berücksichtigen. Der von der OGC verabschiedete WPS-Standard selbst müsste erweitert werden, um die Autorisierungsinformationen am WPS an die von ihm genutzten Dienste weiterleiten zu können. Am prinzipiellen Ablauf mit Proxydiensten, einem zentralem AAAD und dem Verteilen der Autorisierungsinformationen mit dem OAuth-Protokoll würde sich hierbei jedoch nichts ändern.

Besonderer Aufmerksamkeit gilt der OpenID Bibliothek von Zend. Sobald eine neue Version dieser Bibliothek erscheint, wäre zu prüfen, ob diese OpenID Version 2.0 unterstützt, damit eine Authentifizierung mit einem Google-Account möglich wird, denn die derzeitige Version unterstützt dies nicht.

Der Aspekt der Abrechnung der Zugriffe wurde im Rahmen dieser Arbeit nicht vertieft. Momentan werden alle Zugriffe am AAAD protokolliert. Hier wäre denkbar, die Zugriffe je nach Umfang, Komplexität oder Exklusivität der Daten unterschiedlich zu bewerten und den Kunden entsprechend in Rechnung zu stellen.

Da sich, wie oben erwähnt, die abgeleitete serviceorientierten Sicherheitsarchitektur prinzipiell eignet, webbasierte Dienste zu schützen und daher keine Einschränkung auf Geo-Webservices besteht, wäre es denkbar, ein Framework zu entwickeln, welches eine Basisimplementierung des AAADs und des Proxydienstes anbietet. Entwickler könnten dann darauf aufbauend beliebige Webdienste schützen.

# Literaturverzeichnis

- [1] *SAML Specifications.* <http://saml.xml.org/saml-specifications>, März 2005. Letzter Zugriff: 1.8.2010.
- [2] *OAuth - Code.* <http://oauth.net/code/>, 2010. Letzter Zugriff: 1.8.2010.
- [3] *Security & Geo-Rights Management Community.* <http://52north.org/maven/project-sites/security/>, Juli 2010. Letzter Zugriff: 1.8.2010.
- [4] *Security subsystem in GeoServer.* <http://docs.geoserver.org/stable/en/user/security/index.html>, Juli 2010. Letzter Zugriff: 1.8.2010.
- [5] *Shibboleth - Downloads.* <http://shibboleth.internet2.edu/downloads.html>, 2010. Letzter Zugriff: 1.8.2010.
- [6] *Ways to implement security in ArcGIS Server.* [http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis\\_server\\_dotnet\\_help/0093/0093000000p600000.htm](http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/0093/0093000000p600000.htm), Juli 2010. Letzter Zugriff: 1.8.2010.
- [7] Allamaraju, A.: *RESTful Web Services Cookbook*. O'Reilly, Sebastopol, CA, 2010.
- [8] Barret, D. J., Silverman, R. E. und Byrnes, R. G.: *SSH - The Secure Shell*. O'Reilly, Sebastopol, CA, 2. Auflage, 2005.
- [9] Bartelme, N.: *Geoinformatik - Modelle, Strukturen, Funktionen*. Springer-Verlag, Berlin, 4. Auflage, 2005.
- [10] Beaujardiere, J.: *OGC Web Map Service Interface, Version 1.3.0*. Open Geospatial Consortium, Wayland, MA, 2004.

- [11] Bertsch, A.: *Digitale Signaturen*. Springer-Verlag, Berlin, 2002.
- [12] Beutelspacher, A., Schwenk, J. und Wolfenstetter, K. D.: *Moderne Verfahren der Kryptographie: Von RSA zu Zero-knowledge*. Vieweg+Teubner Verlag, Wiesbaden, 7. Auflage, 2006.
- [13] Birch, D.: *Digital Identity Management*. Gower Publishing, Aldershot, 2007.
- [14] Buchmann, J.: *Einführung in die Kryptographie*. Springer-Verlag, Berlin, 3. Auflage, 2004.
- [15] Cheswick, W. R., Bellovin, S. M. und Rubin, A. D.: *Firewalls and Internet security*. Pearson Education, Boston, MA, 2. Auflage, 2003.
- [16] Ciampa, M.: *Security+ Guide to Network Security Fundamentals*. Course Technology, Boston, MA, 3. Auflage, 2008.
- [17] Daswani, N., Kern, C. und Kesavan, A.: *Foundations of security: what every programmer needs to know*. Springer-Verlag, New York, NY, 1. Auflage, 2007.
- [18] Eckert, C.: *IT-Sicherheit, Konzepte-Verfahren-Protokolle*. Oldenbourg Wissenschaftsverlag, München, 6. Auflage, 2008.
- [19] Ertel, W.: *Angewandte Kryptographie*. Carl Hanser Verlag, München, 3. Auflage, 2007.
- [20] ESRI, Redlands, CA: *ArcGIS®Enterprise Security: Delivering Secure Solutions. An ESRI® White Paper*, 2005.
- [21] Evans, J. D.: *Web Coverage Service (WCS) Implementation Standard, Version 1.0.0*. Open Geospatial Consortium, Wayland, MA, 2003.
- [22] Facebook: *Facebook Developers: Authentication*. <http://developers.facebook.com/docs/authentication/>, April 2010. Letzter Zugriff: 1.8.2010.
- [23] Faith, L. und Simson, G.: *Security and usability. Designing secure systems that people can use*. O'Reilly, Sebastopol, CA, 2005.

- [24] Foundation, OpenID: *OpenID Authentication 2.0 - Final.* [http://openid.net/specs/openid-authentication-2\\_0.txt](http://openid.net/specs/openid-authentication-2_0.txt), Dezember 2007. Letzter Zugriff: 1.8.2010.
- [25] Fumy, W. und Sauerbrey, J.: *Enterprise security: IT security solutions: Concepts, Practical Experiences, Technologies.* Publicis Corporate Publishing, Erlangen, 2006.
- [26] Goodner, M. und Goodner, M. A.: *Web Services Federation Language (WS-Federation) Version 1.2.* <http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html>, Mai 2009. Letzter Zugriff: 1.8.2010.
- [27] Goodner, M., Hondo, M., Nadalin, A., McIntosh, M und Schmidt, D.: *Understanding WS-Federation.* <http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-FederationSpec05282007.pdf>, Mai 2007. Letzter Zugriff: 1.8.2010.
- [28] Gupta, J. N. D., Sharma, S. K. und Rashid, M. A.: *Handbook of Research on Enterprise Systems.* IGI Global, Hershey, PA, 2009.
- [29] Hammer-Lahav, E.: *RFC 5849: The OAuth 1.0 Protocol.* <http://tools.ietf.org/html/rfc5849>, April 2010. Letzter Zugriff: 1.8.2010.
- [30] Hilton, B. N.: *Emerging Spatial Information Systems and Applications.* Idea Group Publishing, London, 2007.
- [31] Hosbach, S.: *Vergleich aktueller Authentifizierungsverfahren und deren Bewertung - Studienarbeit.* Carl von Ossietzky Universität Oldenburg, 2005.
- [32] I., Melzer *et al.:* *Service-orientierte Architekturen mit Web Services.* Spektrum Akademischer Verlag, Heidelberg, 4. Auflage, 2010.
- [33] Internet2: *Shibboleth Technical Specs.* <https://spaces.internet2.edu/display/SHIB2/TechnicalSpecs>, März 2010. Letzter Zugriff: 1.8.2010.
- [34] Jain, A. K., Ross, A. und Prabhakar, S.: *An introduction to biometric recognition.* IEEE Trans. on Circuits and Systems for Video Technology, 14:4–20, 2004.

- [35] Jones, S. G.: *Cybersociety 2.0*. SAGE Publications, Thousand Oaks, CA, 1998.
- [36] Kahler, T. und Werner, S.: *Electronic Banking und Datenschutz*. Springer-Verlag, Berlin, 2008.
- [37] Kampe, I.: *Studienarbeit SAML/Shibboleth*. <http://www2.informatik.hu-berlin.de/~kampe/termpaper/doc/0-Input/7-related/saml.pdf>, Februar 2010. Letzter Zugriff: 1.8.2010.
- [38] Kumar, P.: *J2EE security for servlets, EJBs and Web services*. Prentice Hall, Upper Saddle River, NJ, 1. Auflage, 2004.
- [39] Lehtinen, R., Russell, D. und Gangemi, G. T.: *Computer Security Basics*. O'Reilly, Sebastopol, CA, 2. Auflage, 2007.
- [40] Lipp, M.: *VPN - Virtuelle Private Netzwerke*. Addison-Wesley Verlag, München, 2001.
- [41] Makice, K.: *Twitter API: up and running*. O'Reilly, Sebastopol, CA, 2009.
- [42] Maltoni, D., Maio, D., Jain, A.K. und Prabhakar, S.: *Handbook of Fingerprint Recognition*. Springer-Verlag, London, 2. Auflage, 2009.
- [43] Masak, D.: *Digitale Ökosysteme*. Springer-Verlag, Berlin, 2009.
- [44] Matheus, A. und Herrmann, J.: *Geospatial eXtensible Access Control Markup Language (GeoXACML), Version 1.0.0*. Open Geospatial Consortium, Wayland, MA, 2008.
- [45] Menezes, A. J., Van Oorschot, P. C. und Vanstone, S. A.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 2. Auflage, 1996.
- [46] Mercuri, M.: *Beginning information cards and CardSpace*. Springer-Verlag, New York, NY, 2007.
- [47] Mitchell, T.: *Web Mapping mit Open Source-GIS-Tools*. O'Reilly, Köln, 2008.
- [48] Mitnick, K. und Simon, W.: *Die Kunst der Täuschung*. Mitp-Verlag, Heidelberg, 2006.

- [49] Mogollon, M.: *Cryptography and Security Services: Mechanisms and Applications*. CyberTech Publishing, Hershey, PA, 2007.
- [50] Müller, C.: *GeoXACML im J2EE Environment*. Diplomarbeit, Paris Lodron-Universität Salzburg, 2008.
- [51] Na, A. und Priest, M.: *Sensor Observation Service, Version 1.0.0*. Open Geospatial Consortium, Wayland, MA, 2007.
- [52] Nebert, D., Whiteside, A. und P., Vretanos: *OpenGIS® Catalogue Services Specification*. Open Geospatial Consortium, Wayland, MA, 2007.
- [53] Nichols, R. K. und Lekkas, P. C.: *Wireless security: models, threats, and solutions*. McGraw-Hill Professional, Columbus, OH, 2002.
- [54] O'Neill, M.: *Web Service Security*. McGraw-Hill/Osborne, Berkeley, CA, 2003.
- [55] Peikari, C. und Chwakin, A.: *Kenne Deinen Feind - Fortgeschrittene Sicherheitstechniken*. O'Reilly, Köln, 1. Auflage, 2004.
- [56] Poguntke, W.: *Basiswissen IT-Sicherheit*. W3L, Herdecke, 1. Auflage, 2007.
- [57] Qu, J. et al.: *Earth Science Satellite Remote Sensing*. Springer Verlag, Berlin, 2006.
- [58] Reagan, D.: *Twitter Application Development For Dummies*. Wiley Publishing, Indianapolis, IN, 2010.
- [59] Rehman, R. U.: *OpenID: The OpenID book, Draft Version Revision 15*. Conformix Books, Columbus, OH, 2007.
- [60] Rittinghouse, J. und Ransome, J. F.: *Cloud Computing: Implementation, Management, and Security*. CRC Press, Boca Raton, FL, 2010.
- [61] Rockford, L.: *Expert C# 2005 Business Objects*. Springer-Verlag, New York, NY, 2. Auflage, 2006.

- [62] Schaeffer, B., Baranski, B. und Foerster, T.: *Licensing OGC Geoprocessing Services as a Foundation for Commercial Use in SDIs*. Advanced Geographic Information Systems, Applications, and Services, International Conference on, 0:111–116, 2010.
- [63] Schäffer, B.: *Standardized Geoprocessing with 52°North Open Source Software - Web Processing Service Tutorial*. 52°North, Münster, 2009.
- [64] Schut, P.: *OpenGIS® Web Processing Service, Version 1.0.0*. Open Geospatial Consortium, Wayland, MA, 2007.
- [65] Schwenk, J.: *Sicherheit und Kryptographie im Internet*. Vieweg+Teubner Verlag, Wiesbaden, 3. Auflage, 2010.
- [66] Senk, C.: *Biometrische Authentifizierung im Kontext hochflexibler Geschäftsprozesse*. Bayerischer Forschungsverbund forFLEX - Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse, Bamberg, 2009.
- [67] Stein, E.: *Taschenbuch Rechnernetze und Internet*. Carl Hanser Verlag, München, 3. Auflage, 2008.
- [68] Stojmenović, I.: *Handbook of Sensor Networks: Algorithms and Architectures*. John Wiley & Sons, Hoboken, NJ, 2005.
- [69] SWITCH: *Shibboleth Architecture*. <http://switch.ch/aai/demo/>, 2010. Letzter Zugriff: 1.8.2010.
- [70] Systems, BEA, Software, BMC, Inc., CA, IBM, Technologies, Layer 7, Microsoft, Novell und VeriSign: *Web Services Federation Language*. <http://www.ibm.com/developerworks/library/specification/ws-fed/>, Mai 2007. Letzter Zugriff: 1.8.2010.
- [71] Tannenbaum, A. S.: *Moderne Betriebssysteme*. Pearson Education, München, 3. Auflage, 2009.
- [72] Tierling, E.: *Windows Server 2008 R2; Einrichtung, Verwaltung, Referenz*. Addison-Wesley Verlag, München, 1. Auflage, 2010.

- [73] Tscherkasski, E.: *Konzeption und Implementierung eines erweiterten Clients für den Web-Map-Service (WMS) der Stadt Bochum.* Diplomarbeit, Fachhochschule Bochum, 2006.
- [74] Velte, T. J. und Velte, A. T.: *Cisco - A Beginner's Guide.* McGraw-Hill, New York, NY, 4. Auflage, 2007.
- [75] Voges, U. und Senkler, K.: *OpenGIS® Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile.* Open Geospatial Consortium, Wayland, MA, 2007.
- [76] Vowles, G.: *Geospatial Digital Rights Management Reference Model (GeoDRM RM), Version 1.0.0.* Open Geospatial Consortium, Wayland, MA, 2006.
- [77] Vretanos, P. A.: *OpenGIS® Filter Encoding Implementation Specification, Version 1.1.0.* Open Geospatial Consortium, Wayland, MA, 2005.
- [78] Vretanos, P. A.: *Web Feature Service Implementation Specification, Version 1.1.0.* Open Geospatial Consortium, Wayland, MA, 2005.
- [79] Weerawarana, S., Curbera, F., Leymann, F., Storey, T. und Ferguson, D. F.: *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More.* Prentice Hall, Upper Saddle River, NJ, 1. Auflage, 2005.
- [80] Whiteside, A. und Evans, J. D.: *Web Coverage Service (WCS) Implementation Standard, Version 1.1.2.* Open Geospatial Consortium, Wayland, MA, 2008.
- [81] Wind, M. und Kröger, D.: *Handbuch IT in der Verwaltung.* Springer-Verlag, Berlin, 2006.
- [82] Yeung, A. K. W. und Hall, G. B.: *Spatial database systems: design, implementation and project management.* Springer, Dordrecht, 1. Auflage, 2007.
- [83] Zwick, E. D., Cooper, S. und Chapman, D. B.: *Building Internet Firewalls.* O'Reilly, Sebastopol, CA, 2. Auflage, 2000.

# Abkürzungsverzeichnis

- AAAD** Authentifizierungs-, Autorisierungs- und Abrechnungsdienst
- API** Application Programming Interface
- CQL** OGC Common Catalogue Query Language
- CSW** Catalogue Service-Web
- DRM** Digital Rights Management
- DSL** Digital Subscriber Line
- DTED** Digital Terrain Elevation Data
- GeoDRM** Geospatial Digital Rights Management
- GeoTIFF** Geo Tagged Image File Format
- GeoXACML** Geospatial eXtensible Access Control Markup Language
- GIS** Geoinformationssystem
- GML** Geography Markup Language
- HTTPS** Hypertext Transfer Protocol Secure
- IMAP** Internet Message Access Protocol
- IMAPS** Internet Message Access Protocol Secure
- JPEG** Joint Photographic Experts Group
- LDAP** Lightweight Directory Access Protocol
- OASIS** Organization for the Advancement of Structured Information Standards
- OGC-Filter** Open Geospatial Consortium-Filter
- OGC** Open Geospatial Consortium
- OSI** Open Systems Interconnection
- PAP** Policy Administration Point
- PDP** Policy Decision Point
- PEP** Policy Enforcement Point

- PIP** Policy Information Point
- PNG** Portable Network Graphics
- POP3** Post Office Protocol Version 3
- POP3S** Post Office Protocol Version 3 Secure
- REST** Representational State Transfer
- RFC** Request for Comments
- SAML** Security Assertion Markup Language
- SAS** Sensor Alert Service
- SMTP** Simple Mail Transfer Protocol
- SMTPS** Simple Mail Transfer Protocol Secure
- SOAP** Simple Object Access Protocol
- SOS** Sensor Observation Service
- SSL** Secure Sockets Layer
- SSO** Single Sign-On
- SVG** Scalable Vector Graphics
- TAN** Transaktionsnummer
- TIFF** Tagged Image File Format
- TLS** Transport Layer Security
- URL** Uniform Resource Locator
- WCS** Web Coverage Service
- WFS-T** Transactional Web Feature Service
- WFS** Web Feature Service
- WMS** Web Map Service
- WPS** Web Processing Service
- WSS** 52n Web Enforcement Service
- XACML** eXtensible Access Control Markup Language

# **Anhang**

# A

---

## Aufgabenstellungen und Anforderungen durch Research Studio iSpace

11. März 2010, 12:15 Uhr bis 14:30 Uhr, Schillerstraße 25, 5020 Salzburg

Anwesende:

- Dr. Resch Bernd (Research Studio iSpace)
- Mag. Mittlböck Manfred (Research Studio iSpace)
- DI (FH) Schulz Bernhard

Ergebnisprotokoll<sup>1</sup> in Stichworten:

- Diese Masterarbeit entsteht in Zusammenarbeit von Fachhochschule Salzburg und Research Studio iSpace.
- Ziel der Arbeit: Absicherung über raum-/zeitspezifische Zugangskontrolle der standardisierten Geo-Webservices wie WMS, WFS, WCS, WPS, SOS und SAS.
- Diese standardisierten Webservices sollen nicht verändert werden. Bernhard Schulz schlägt daher den Einsatz von Proxy-Diensten vor. Dr. Resch und Mag. Mittlböck sind einverstanden.
- Die Zugriffserlaubnis soll pro Benutzer konfiguriert werden können.
- Die Zugriffssteuerung für Kartendienste soll per Kartenebene und Kartenausschnitt regulierbar sein.
- Bernhard Schulz soll evaluieren, in welchem Format die Zugriffsregeln abgespeichert und angewandt werden können.

---

<sup>1</sup>Gedächtnisprotokoll, erstellt am 26.8.2010 durch Bernhard Schulz und geprüft durch Mag. Manfred Mittlböck.

- Die Evaluierung soll auf Grund verschiedener gewichteter Kriterien erfolgen und tabellarisch festgehalten werden. Die Kriterien und Gewichte werden von Bernhard Schulz in Absprache mit iSpace zu einem späteren Zeitpunkt festgelegt werden.
- Die Zugriffssteuerung soll sowohl aufgrund interner Benutzerdatenbanken als auch auf Grund externer Authentifizierungsserver erfolgen.
- Externe Authentifizierung: Angestrebte Unabhängigkeit von konkreten Authentifizierungsverfahren wie LDAP, ActiveDirectory, OpenID und so weiter.

Ziel ist es, großen Organisationen wie zum Beispiel der Salzburger Landesregierung die Nutzung der Geoinfrastruktur zu ermöglichen, ohne dass sämtliche Benutzerdaten zwischen den Authentifizierungsservern von iSpace und der Landesregierung abgeglichen werden müssen. Stattdessen sollen direkt die Authentifizierungsserver der Kunden genutzt werden können.

- Wunsch von iSPACE nach einer „Light-Weight“ Security Lösung basierend auf lizenzkostenfreien und betriebssystemunabhängigen Webprotokollen. Wenn möglich Verzicht auf SOAP, da dies aufgrund der Protokolloverheads auf Embedded Systems zu hohem Ressourcenverbrauch führt, so Mag. Mittlböck.
- Durch den Einsatz von Webprotokollen könnte die Architektur später einmal in der „Cloud“ laufen.
- Bernhard Schulz soll evaluieren, welches Protokoll/welche Technologie sich am besten eignet, um Autorisierungsinformationen in der Geoinfrastruktur zu übertragen.
- Auch diese Evaluierung soll auf Grund verschiedener gewichteter Kriterien erfolgen und tabellarisch festgehalten werden. Die Kriterien und Gewichte werden von Bernhard Schulz in Absprache mit iSpace zu einem späteren Zeitpunkt festgelegt werden.
- Sämtliche Zugriffe von Benutzern auf die Webservices sollen protokolliert werden.
- Es soll nur eine bestimmte Anzahl von Zugriffen möglich sein, bevor der Benutzer sich erneut authentifizieren muss. Nach langer Diskussion einigen sich die Gesprächspartner dies so auszulegen, dass der Benutzer nur ein gewisses Guthaben

an Zugriffen hat. Jeder Zugriff verringert das Guthaben. Sobald das Guthaben verbraucht ist, werden weitere Zugriffe vom System unterbunden.

- Den Gesprächspartnern wurde durch die Diskussion bewusst, dass sämtliche Aspekte eines Tripe-A-Systems umgesetzt werden sollten.
- Die prototypische Implementierung soll auf Wunsch von iSpace in der Programmiersprache „Phyton“ umgesetzt werden. Es wird Bernhard Schulz aber frei gestellt, auch „PHP“ für die Implementierung zu verwenden.

# B

---

## WFS-Anfrage mit Geometriedaten

```
http://localhost:8080/geoserver/wfs?request=GetFeature&version=1.1.0&typeName=topp:states&propertyName=STATE_NAME,PERSONS,the_geom&BBOX=-75.102613,40.212597,-72.361859,41.512517,EPSC:4326
```

Listing B.1: Ergebnis einer einfachen WFS-Anfrage mit Geostrukturen

```
<wfs:FeatureCollection numberOfFeatures="4" timeStamp="2010-06-03T22:09:02.030+02:00" xsi:schemaLocation="http://www.openplans.org/topp http://localhost:8080/geoserver/wfs?service=WFS&version=1.1.0&request=DescribeFeatureType&typeName=topp:states http://www.opengis.net/wfs http://localhost:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:topp="http://www.openplans.org/topp" xmlns:tiger="http://www.census.gov" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ows="http://www.opengis.net/ows" xmlns:wfs="http://www.opengis.net/wfs">
196    <gml:featureMembers>
197        <topp:states gml:id="states.39">
198            <topp:the_geom>
199                <gml:MultiSurface srsName="urn:x-ogc:def:crs:EPSG:4326">
200                    <gml:surfaceMember>
201                        <gml:Polygon>
202                            <gml:exterior>
203                                <gml:LinearRing>
204                                    <gml:posList>
205                                        42.267269 -79.763466 42.41930400000001 -79.444252 42.493404 -79.355118
206                                            42.57455799999996 -79.142471 42.69918799999999 -79.043991 42.792686 -78.859444
207                                                42.97417400000005 -78.93679
208                                            43.022301 -78.883034 43.06657000000001 -78.925835 43.09054900000001 -79.061348
209                                                43.14468400000001 -79.039558 43.26816199999999 -79.062469 43.371937 -78.464905
210                                                43.36551299999999 -77.992271
211                                            43.335109999999986 -77.745277 43.24148600000001 -77.575989 43.27565000000001 -77.377602
212                                                43.27852999999999 -76.914841 43.342667000000006 -76.737152 43.323375999999996
213                                                -76.718796 43.414085
214                                            -76.619957 43.500652 -76.454994 43.55408499999986 -76.223114 43.633129 -76.184921
215                                                43.68263200000001 -76.206017 43.83506399999999 -76.240341 43.91243 -76.194069
216                                                43.93214800000001 -76.129417
217                                            44.013172 -76.134872 44.06554399999999 -76.201889 44.04196200000001 -76.297226
218                                                44.09830099999999 -76.363213 44.390209 -75.848351 44.51747499999999 -75.758972
219                                                44.81057000000001 -75.329201 44.948578
220
```

210 -74.968819 44.99286699999999 -74.736481 44.990803 -74.021935 45.0061 -73.345146 ↗  
44.98193399999995 -73.351181 44.93256400000001 -73.336838 44.847893 -73.382729 ↗  
44.81907699999999 -73.369476  
44.799251999999996 -73.327209 44.72419400000001 -73.373581 44.68032500000001 -73.358574 ↗  
44.66123200000001 -73.37352 44.63430399999986 -73.37056 44.61976200000001 ↗  
-73.382248 44.57912099999999  
212 -73.371719 44.55392499999999 -73.348236 44.54428100000001 -73.334877 44.432804000000004 ↗  
-73.293747 44.40548300000004 -73.300423 44.36734000000001 -73.330215 ↗  
44.26008999999999 -73.305756  
44.20119500000001 -73.377762 44.172054 -73.382492 44.136173000000014 -73.408295 ↗  
44.10655600000001 -73.409187 44.0638429999999 -73.435646 44.045624000000004 ↗  
-73.436432 44.01816600000001  
214 -73.408684 43.98813999999999 -73.417839 43.914749 -73.405769 43.885918000000004 ↗  
-73.375557 43.80444700000001 -73.385178 43.778366000000005 -73.359436 43.756496 ↗  
-73.357109 43.71421799999999  
-73.371429 43.63204999999999 -73.423401 43.582413 -73.418762 43.5690769999999 ↗  
-73.388557 43.61493300000001 -73.364128 43.624649000000005 -73.303978 ↗  
43.619586999999996 -73.294548 43.593121  
216 -73.282181 43.57496599999999 -73.291847 43.559315 -73.260429 43.512764000000004 ↗  
-73.238838 43.31077999999994 -73.250519 42.9402159999999 -73.276451 ↗  
42.83702500000001 -73.280029 42.803471  
-73.296616 42.74740199999994 -73.269722 42.745979000000005 -73.258507 42.504673 ↗  
-73.351273 42.07737 -73.499275 42.047337 -73.484665 41.665592000000004 -73.517578 ↗  
41.522651999999994 -73.530823  
218 41.365204000000006 -73.544724 41.293526000000014 -73.55069 41.2106589999999 -73.478554 ↗  
41.100258 -73.725662 41.01251999999995 -73.654152 40.99829500000001 -73.65358 ↗  
40.88658899999999 -73.780464  
40.832233 -73.796768 40.80270400000006 -73.92028 40.70389900000001 -74.006599 40.737629 ↗  
-74.006676 40.7973859999999 -73.977478 40.885941 -73.922813 40.9272159999999 ↗  
-73.909386 40.9607729999999  
220 -73.896568 40.9984320000001 -73.897118 41.12351599999995 -74.213448 41.13753099999996 ↗  
-74.242767 41.1957549999999 -74.372383 41.35047900000001 -74.700455 41.374966 ↗  
-74.705666 41.40154300000004  
-74.740829 41.42196699999995 -74.740433 41.43005400000001 -74.755219 41.42982499999994 ↗  
-74.79319 41.44706300000014 -74.864456 41.44458 -74.895668 41.461803 -74.898949 ↗  
41.48426100000004 -74.932953  
222 41.48351299999999 -74.972176 41.539467 -75.015274 41.5657119999999 -75.025475 41.604389 ↗  
-75.070251 41.61299099999994 -75.072838 41.637226 -75.051712 41.71474799999986 ↗  
-75.065788 41.72662  
-75.057358 41.770172 -75.061707 41.778954 -75.097542 41.79711900000001 -75.097214 ↗  
41.8140600000001 -75.080231 41.8368989999999 -75.118164 41.8490940000001 ↗  
-75.125137 41.85570100000001 -75.148666  
224 41.86775199999996 -75.171669 41.868786 -75.254898 41.947517000000005 -75.284073 ↗  
41.9611889999999 -75.324448 41.992760000000004 -75.346039 41.99827199999986 ↗  
-75.383194 41.996284 -75.48011  
41.999424000000005 -76.105186 42.000580000000014 -76.14537 42.002937 -76.564247 ↗  
42.00246000000001 -76.928711 42.0029069999999 -76.968887 41.998760000000004 ↗  
-77.613136 41.997265 -77.745293

```
226  41.99813499999999 -78.204529 41.99935500000001 -78.305351 41.999786 -78.918777 ,
     42.00109900000001 -79.059723 42.00053 -79.612595 42.003052 -79.761887 42.267269 ,
     -79.763466
  </gml:posList>
228          </gml:LinearRing>
230          </gml:exterior>
  </gml:Polygon>
232          <gml:surfaceMember>
  <gml:surfaceMember>
    <gml:Polygon>
234          <gml:exterior>
            <gml:LinearRing>
236  <gml:posList>
  40.594482 -73.752632 40.55754500000005 -73.928009 40.618134 -73.761818 ,
    40.63682900000006 -73.765358 40.65249600000001 -73.846481 40.63102699999999 ,
    -73.916153 40.59016399999999
238  -73.880234 40.581154 -74.004456 40.63923299999999 -74.028137 40.73928100000005 ,
    -73.956024 40.79701600000001 -73.899437 40.78874999999999 -73.754341 40.844898 ,
    -73.749901 40.90304599999999
    -73.598701 40.87964600000001 -73.478523 40.922459 -73.431084 40.90094400000001 ,
    -73.214966 40.95129800000001 -73.141426 40.968334 -73.021721 40.98118600000001 ,
    -72.632034 41.149235000000004
240  -72.317505 41.142437 -72.281624 41.11010400000001 -72.354752 41.02593999999999 ,
    -72.416817 40.96608000000005 -72.551437 40.90520100000005 -72.605278 ,
    40.92004800000001 -72.476578 41.02391800000001
    -72.293556 41.03527500000001 -72.203827 41.000473 -72.077492 41.08476999999999 ,
    -71.924187 41.07440600000001 -71.870476 41.030472 -71.919189 40.814941000000005 ,
    -72.521629 40.661224000000004
242  -73.422958 40.594482 -73.752632
  </gml:posList>
244          </gml:LinearRing>
246          </gml:exterior>
  </gml:Polygon>
248          <gml:surfaceMember>
  <gml:surfaceMember>
    <gml:Polygon>
250          <gml:exterior>
            <gml:LinearRing>
252  <gml:posList>
  40.626282 -73.293495 40.63297700000001 -73.29158 40.63297700000001 -73.240875 ,
    40.67506800000001 -73.051468 40.73724699999996 -72.877357 40.76403400000001 ,
    -72.782654
254  40.76786000000001 -72.757782 40.75829300000001 -72.764481 40.69994 -72.956764 ,
    40.67124200000001 -73.03138 40.62532400000006 -73.249489 40.626282 -73.293495
  </gml:posList>
256          </gml:LinearRing>
258          </gml:exterior>
  </gml:Polygon>
```

```
260           </gml:surfaceMember>
261           <gml:surfaceMember>
262             <gml:Polygon>
263               <gml:exterior>
264                 <gml:LinearRing>
265                   40.505898 -74.23735 40.53780000000004 -74.23735 40.624393 -74.16671 40.64946000000005 ,
266                   -74.07328 40.60160400000001 -74.059608 40.544636 -74.123413 40.510456000000005
267               </gml:posList>
268             </gml:LinearRing>
269           </gml:exterior>
270         </gml:Polygon>
271       </gml:surfaceMember>
272     </gml:MultiSurface>
273   </topp:the_geom>
274   <topp:STATE_NAME>New York</topp:STATE_NAME>
275   <topp:PERSONS>1.8235907E7</topp:PERSONS>
276 </topp:states>
277 <topp:states gml:id="states.40">
278   <topp:the_geom>
279     <gml:MultiSurface srsName="urn:x-ogc:def:crs:EPSG:4326">
280       <gml:surfaceMember>
281         <gml:Polygon>
282           <gml:exterior>
283             <gml:LinearRing>
284           <gml:posList>
285             39.719528 -77.476082 39.725368 -78.096222 39.724007 -78.334816 39.723659999999995 ,
286             -78.385048 39.72302999999994 -78.818016 39.722252 -78.930428 39.719234
287             -79.396851 39.72019599999999 -79.481209 39.72172900000001 -79.765358 39.721588 ,
288             -79.918488 39.71976100000005 -80.429283 39.721126999999996 -80.524467 ,
289             39.95834400000001
290             -80.524857 40.022751 -80.525169 40.16244900000001 -80.526253 40.40296599999999 ,
291             -80.523773 40.47871799999986 -80.524567 40.63713799999999 -80.522209 40.854107 ,
292             -80.520515
293             40.897209000000004 -80.521133 41.12957 -80.52314 41.48923500000001 -80.520203 ,
294             41.495048999999995 -80.524132 41.850719 -80.52285 41.986816000000005 -80.520798 ,
295             42.267269
296             -79.763466 42.003052 -79.761887 42.00053 -79.612595 42.00109900000001 -79.059723 ,
297             41.999786 -78.918777 41.99935500000001 -78.305351 41.99813499999999 -78.204529 ,
298             41.997265
299             -77.745293 41.99876000000004 -77.613136 42.00290699999999 -76.968887 42.00246000000001 ,
300             -76.928711 42.002937 -76.564247 42.00058000000014 -76.14537 41.999424000000005
301             -76.105186 41.996284 -75.48011 41.99827199999986 -75.383194 41.99276000000004 ,
302             -75.346039 41.96118899999999 -75.324448 41.94751700000005 -75.284073 41.868786 ,
303             -75.254898
304             41.867751999999996 -75.171669 41.85570100000001 -75.148666 41.84909400000001 -75.125137 ,
305             41.83689899999999 -75.118164 41.81406000000001 -75.080231 41.79711900000001
```

```

-75.097214 41.778954 -75.097542 41.770172 -75.061707 41.72662 -75.057358 ,
41.71474799999986 -75.065788 41.637226 -75.051712 41.61299099999994 -75.072838 ,
41.604389

294 -75.070251 41.5657119999999 -75.025475 41.539467 -75.015274 41.4835129999999 ,
-74.972176 41.48426100000004 -74.932953 41.461803 -74.898949 41.44458 -74.895668
41.447063000000014 -74.864456 41.42982499999994 -74.79319 41.43005400000001 -74.755219 ,
41.42196699999995 -74.740433 41.40154300000004 -74.740829 41.374966 -74.705666

296 41.3504790000001 -74.700455 41.311871 -74.792053 41.29512 -74.794533 41.282612 ,
-74.825584 41.2267229999999 -74.866798 41.2066759999999 -74.863274 ,
41.14100999999994

-74.915154 41.11175900000006 -74.950378 41.09928500000001 -74.984978 41.0816649999999 ,
-74.989265 41.08258100000005 -74.966675 41.06238899999996 -75.001534

298 41.028107000000006 -75.035637 41.0106199999999 -75.070358 41.00015300000001 -75.116997 ,
40.9774319999999 -75.139679 40.962841 -75.135902 40.90325200000001 -75.080116
40.88452899999986 -75.074066 40.871948 -75.05687 40.85557600000001 -75.055 ,
40.83918800000001 -75.099922 40.821293 -75.089592 40.791561 -75.100922 ,
40.77260999999986

300 -75.130699 40.77471199999994 -75.170692 40.747906 -75.194023 40.723759 -75.188133 ,
40.6859629999999 -75.205734 40.66969299999995 -75.184441 40.65063900000001 ,
-75.209297

40.634106 -75.198318 40.614643 -75.200768 40.5836679999999 -75.193893 40.570583 ,
-75.197952 40.5566980000001 -75.182663 40.56469699999995 -75.124908 40.545254

302 -75.08017 40.5209010000001 -75.064064 40.45615000000001 -75.070503 40.4200669999999 ,
-75.057838 40.401218 -75.021652 40.4085159999999 -75.000862 40.40434300000001
-74.973228 40.3453670000001 -74.950577 40.3336679999999 -74.932892 40.31392700000001 ,
-74.921539 40.29948400000001 -74.88105 40.248344 -74.843155 40.177616 -74.73922

304 40.1491969999999 -74.725876 40.1242489999999 -74.746689 40.116051 -74.829414 40.077946 ,
-74.872314 40.0579030000001 -74.956589 40.033962 -74.983871 40.0075229999999 ,
-75.046059 39.9852789999999 -75.068428 39.9756199999999 -75.084969 39.9765779999999 ,
-75.111343 39.9558069999999 -75.140244 39.934628000000004 -75.147537

306 39.8967739999999 -75.136185 39.8814889999999 -75.14328 39.8772930000001 -75.185982 ,
39.8502919999996 -75.247368 39.84542500000006 -75.254112 39.8484039999999
-75.346298 39.79886999999994 -75.42083 39.82643500000004 -75.470345 39.84000800000001 ,
-75.583794 39.8381960000001 -75.644341 39.820347 -75.695114 39.77481800000001

308 -75.745934 39.7244420000001 -75.775269 39.72375500000001 -75.791435 39.72212200000001 ,
-76.139549 39.7217479999999 -76.233444 39.7201609999999 -76.570145 ,
39.7211529999999

-76.790794 39.7207909999999 -76.997108 39.7205810000001 -77.221344 39.719978 ,
-77.464722 39.719528 -77.476082

310 </gml:posList>
                                     </gml:LinearRing>
                                     </gml:exterior>
                                     </gml:Polygon>
                                     </gml:surfaceMember>
                                     </gml:MultiSurface>
</topp:the_geom>
<topp:STATE_NAME>Pennsylvania</topp:STATE_NAME>
<topp:PERSONS>1.1881643E7</topp:PERSONS>

312
314
316
318

```

```

        </topp:states>
320      <topp:states gml:id="states.41">
          <topp:the_geom>
322          <gml:MultiSurface srsName="urn:x-ogc:def:crs:EPSG:4326">
            <gml:surfaceMember>
324              <gml:Polygon>
                <gml:exterior>
326                  <gml:LinearRing>
                    <gml:posList>
328        41.522651999999994 -73.530823 41.665592000000004 -73.517578 42.047337 -73.484665 ↘
                     42.03621699999999 -73.046082 42.03591499999999 -73.006546 42.033412999999996
-72.81691 41.99708899999999 -72.818138 42.002071 -72.768036 42.03375199999999 -72.756355 ↘
                     42.030440999999996 -72.609993 42.022705 -72.608292 42.021511000000004 -72.582375
330        42.03002900000001 -72.571693 42.030669999999986 -72.508041 42.026306000000005 -72.136826 ↘
                     42.02570299999999 -72.095451 42.01787899999999 -71.802826 42.004177 -71.798317
41.72150400000001 -71.788734 41.641659000000004 -71.793091 41.601208000000014 -71.79068 ↘
                     41.415729999999996 -71.80323 41.40375499999999 -71.846481 41.34186199999999 ↘
                     -71.837357
332        41.325249000000014 -71.848259 41.32266999999999 -71.867165 41.281048 -72.281891 ↘
                     41.28954300000001 -72.326828 41.35825 -72.378883 41.27800400000001 -72.378624 ↘
                     41.26360299999999
-72.52771 41.26996600000001 -72.907188 41.160942000000006 -73.104866 40.99829500000001 ↘
                     -73.65358 41.01251999999995 -73.654152 41.100258 -73.725662 41.21065899999999
334        -73.478554 41.293526000000014 -73.55069 41.365204000000006 -73.544724 41.52265199999994 ↘
                     -73.530823
</gml:posList>
336                  </gml:LinearRing>
                    </gml:exterior>
338                  </gml:Polygon>
                    </gml:surfaceMember>
340                  </gml:MultiSurface>
                    </topp:the_geom>
342          <topp:STATE_NAME>Connecticut</topp:STATE_NAME>
          <topp:PERSONS>3287116.0</topp:PERSONS>
344          </topp:states>
          <topp:states gml:id="states.43">
346            <topp:the_geom>
              <gml:MultiSurface srsName="urn:x-ogc:def:crs:EPSG:4326">
                <gml:surfaceMember>
348                  <gml:Polygon>
                    <gml:exterior>
350                      <gml:LinearRing>
352            <gml:posList>
39.71474499999999 -75.489639 39.71997099999999 -75.476334 39.74171799999999 -75.475128 ↘
                     39.763248000000004 -75.460754 39.778130000000004 -75.428009 39.789658
354        -75.412117 39.79886999999994 -75.42083 39.84840399999999 -75.346298 39.845425000000006 ↘
                     -75.254112 39.85029199999996 -75.247368 39.87729300000001 -75.185982 ↘
                     39.88148899999999

```

-75.14328 39.89677399999999 -75.136185 39.934628000000004 -75.147537 39.95580699999999 ↗  
 -75.140244 39.97657799999999 -75.111343 39.97561999999999 -75.084969 ↗  
 39.9852789999999  
 356 -75.068428 40.00752299999999 -75.046059 40.033962 -74.983871 40.05790300000001 ↗  
 -74.956589 40.077946 -74.872314 40.116051 -74.829414 40.12424899999999 -74.746689  
 40.14919699999999 -74.725876 40.177616 -74.73922 40.248344 -74.843155 40.29948400000001 ↗  
 -74.88105 40.31392700000001 -74.921539 40.33366799999999 -74.932892 ↗  
 40.34536700000001  
 358 -74.950577 40.40434300000001 -74.973228 40.40851599999999 -75.000862 40.401218 ↗  
 -75.021652 40.42006699999999 -75.057838 40.45615000000001 -75.070503 ↗  
 40.52090100000001  
 -75.064064 40.545254 -75.08017 40.56469699999995 -75.124908 40.55669800000001 ↗  
 -75.182663 40.570583 -75.197952 40.58366799999999 -75.193893 40.614643 -75.200768 ↗  
 40.634106  
 360 -75.198318 40.65063900000001 -75.209297 40.66969299999995 -75.184441 40.68596299999999 ↗  
 -75.205734 40.723759 -75.188133 40.747906 -75.194023 40.77471199999994 -75.170692  
 40.77260999999986 -75.130699 40.791561 -75.100922 40.821293 -75.089592 ↗  
 40.83918800000001 -75.099922 40.85557600000001 -75.055 40.871948 -75.05687 ↗  
 40.88452899999986  
 362 -75.074066 40.90325200000001 -75.080116 40.962841 -75.135902 40.97743199999999 ↗  
 -75.139679 41.00015300000001 -75.116997 41.01061999999999 -75.070358 ↗  
 41.02810700000006  
 -75.035637 41.06238899999996 -75.001534 41.08258100000005 -74.966675 41.08166499999999 ↗  
 -74.989265 41.09928500000001 -74.984978 41.11175900000006 -74.950378 ↗  
 41.14100999999994  
 364 -74.915154 41.20667599999999 -74.863274 41.22672299999999 -74.866798 41.282612 ↗  
 -74.825584 41.29512 -74.794533 41.311871 -74.792053 41.35047900000001 -74.700455 ↗  
 41.19575499999999  
 -74.372383 41.13753099999996 -74.242767 41.12351599999995 -74.213448 40.99843200000001 ↗  
 -73.897118 40.96077299999999 -73.896568 40.92721599999999 -73.909386 40.885941  
 366 -73.922813 40.79738599999999 -73.977478 40.737629 -74.006676 40.70389900000001 ↗  
 -74.006599 40.64696900000001 -74.129471 40.705524 -74.115944 40.675377 -74.14727 ↗  
 40.59108399999995  
 -74.208664 40.51419799999999 -74.27932 40.46364199999999 -74.269325 40.443478 -74.224915 ↗  
 40.45135099999999 -74.122299 40.32350500000001 -73.978859 40.10172700000001 ↗  
 -74.03965  
 368 40.115971 -74.091843 40.088066 -74.08419 40.091251 -74.034714 40.05673999999999 ↗  
 -74.050209 40.051445 -74.122597 40.04215600000006 -74.077751 39.87848700000001 ↗  
 -74.159645  
 39.71815499999996 -74.171844 39.62392800000001 -74.238091 39.571968 -74.323753 ↗  
 39.52350999999999 -74.329262 39.54250300000001 -74.412796 39.50251 -74.401527 ↗  
 39.42663999999999  
 370 -74.460823 39.38095899999999 -74.447906 39.28713600000004 -74.65863 39.28151700000001 ↗  
 -74.622925 39.25071299999999 -74.624985 39.02626000000001 -74.802681 ↗  
 38.98973100000006  
 -74.879639 38.95657 -74.876686 38.97162599999986 -74.968506 39.11374699999999 ↗  
 -74.890587 39.170525 -74.917038 39.19825 -75.014786 39.18457799999999 -75.120331

```
372    39.37485899999999 -75.416031 39.49040199999999 -75.553116 39.56645599999999 -75.517044 <
373        39.61762200000001 -75.570587 39.71474499999999 -75.489639
374    </gml:posList>                                </gml:LinearRing>
375                                    </gml:exterior>
376    </gml:Polygon>
377        </gml:surfaceMember>
378    </gml:MultiSurface>
379        </topp:the_geom>
380    <topp:STATE_NAME>New Jersey</topp:STATE_NAME>
381    <topp:PERSONS>7484736.0</topp:PERSONS>
382        </topp:states>
383    </gml:featureMembers>
384 </wfs:FeatureCollection>
```

---

# C

---

## Bestätigungsemail bzgl. defekten Projektsetups der 52North Security Suite

Listing C.1: Bestätigungsemail bzgl. defektem Projektsetup der 52north Security Suite

```
Return-path: <J.Drewnak@conterra.de>
2 Received: from beta.fh-salzburg.ac.at ([193.170.110.60])
   by edna.core.fh-salzburg.ac.at
4  (Sun Java System Messaging Server 6.2-3.04 (built Jul 15 2005))
   with ESMTP id <OL1J001VZRVQHF00@edna.core.fh-salzburg.ac.at> for
6  bernhard.schulz@fh-salzburg.ac.at; Tue, 27 Apr 2010 20:11:50 +0200 (CEST)
Received: from localhost (localhost [127.0.0.1]) by beta.fh-salzburg.ac.at
8  (Postfix) with ESMTP id 1AE359002C3 for <bernhard.schulz@fh-salzburg.ac.at>;
   Tue, 27 Apr 2010 20:11:50 +0200 (CEST)
10 Received: from beta.fh-salzburg.ac.at ([127.0.0.1])
    by localhost (beta.fh-salzburg.ac.at [127.0.0.1]) (amavisd-new, port 10024)
12  with ESMTP id zTNRvIDE0b-d for <bernhard.schulz@fh-salzburg.ac.at>; Tue,
   27 Apr 2010 20:11:39 +0200 (CEST)
14 Received: by beta.fh-salzburg.ac.at (Postfix, from userid 502)
   id 4DD4290078A; Tue, 27 Apr 2010 20:11:39 +0200 (CEST)
16 Received: from gate00.it-plus.net (gate00.it-plus.net [217.64.164.70])
   by beta.fh-salzburg.ac.at (Postfix) with ESMTP id 11FA69002C3 for
18  <bernhard.schulz@fh-salzburg.ac.at>; Tue, 27 Apr 2010 20:11:31 +0200 (CEST)
Received: from WASAT.esri-de.com (172.20.110.23)
20  by kakra.esri-de.com (172.20.110.24) with Microsoft SMTP Server (TLS)
   id 8.1.393.1; Tue, 27 Apr 2010 20:11:31 +0200
22 Received: from rana.esri-de.com ([172.20.110.21])
   by wasat.esri-de.com ([172.20.110.23]) with mapi; Tue,
24  27 Apr 2010 20:11:31 +0200
Date: Tue, 27 Apr 2010 20:07:11 +0200
26 From: Jan Drewnak <J.Drewnak@conterra.de>
Subject: RE: [52N Security] Problem with initial setup / Maven - unable to run
28 tests/unable to compile
In-reply-to: <134D378D-F606-44BA-9063-2F68FA970B1F@fh-salzburg.ac.at>
30 To: Bernhard Schulz <bschulz.itsb-m2008@fh-salzburg.ac.at>
Message-id: <448DBED62823FC49959D65461AF92C0932F98B2448@rana.esri-de.com>
32 MIME-version: 1.0
```

Content-type: text/plain; charset=us-ascii  
34 Content-language: en-US  
Content-transfer-encoding: quoted-printable  
36 Accept-Language: de-DE, en-US  
Thread-topic: [52N Security] Problem with initial setup / Maven - unable to run  
38 tests/unable to compile  
Thread-index: AcrmI420p8qyZ3YGRXuO1OOuhciUAABN1KA  
40 acceptlanguage: de-DE, en-US  
X-Virus-Scanned: amavisd-new at fh-salzburg.ac.at  
42 X-Spam-Checker-Version: SpamAssassin 3.3.0 (2010-01-18) on  
beta.fh-salzburg.ac.at  
44 X-Spam-FHS-Level:  
X-Greylist: from auto-whitelisted by SQLgrey-1.7.6  
46 X-policyd-weight: using cached result; rate: -5.5  
X-MS-Has-Attach:  
48 X-MS-TNEF-Correlator:  
X-TM-AS-Product-Ver: SMEX-8.0.0.1285-6.000.1038-17346.000  
50 X-TM-AS-Result: No--27.273100-8.000000-31  
X-TM-AS-User-Approved-Sender: No  
52 X-TM-AS-User-Blocked-Sender: No  
References: <F83044D7-7AB9-45E1-859E-D2FE7231A712@fh-salzburg.ac.at>  
54 <448DBED62823FC49959D65461AF92C0932F98B2436@rana.esri-de.com>  
<134D378D-F606-44BA-9063-2F68FA970B1F@fh-salzburg.ac.at>  
56 Original-recipient: rfc822;bernhard.schulz@fh-salzburg.ac.at  
X-Spam-Status: No, score=-1.9 required=5.0 tests=BAYES\_00 autolearn=ham  
58 version=3.3.0  
X-Spam-Level:  
60  
Bernhard,  
62  
I'm facing the same error if starting with a new local repository from scratc  
64 tch. So, you seem to have found a "project setup bug" ;) I will investigate  
on that issue and let the list know as soon as I have got a solution.  
66  
  
68 Jan  
  
70 > -----Original Message-----  
> From: security-bounces@52north.org  
72 > [mailto:security-bounces@52north.org] On Behalf Of Bernhard Schulz  
> Sent: Dienstag, 27. April 2010 18:06  
74 > To: security@52north.org  
> Subject: Re: [52N Security] Problem with initial setup /  
76 > Maven - unable to run tests/unable to compile  
>  
78 > Hello and thank you for your response!  
>  
80 > Here are my answers:

```
>  
82 > > this sounds like some artifacts in your local maven  
> repository ~/.m2/repository are broken.  
84 > > What's the content of  
> ~/~/.m2/repository/commons-fileupload/commons-fileupload/1.2/com  
86 > mons-fileupload-1.2.pom?  
> It is a HTML file with a 302 - Moved permanently header.  
88 >  
> #####  
90 > <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
> <html><head>  
92 > <title>301 Moved Permanently</title>  
> </head><body>  
94 > <h1>Moved Permanently</h1>  
> <p>The document has moved <a  
96 > href=3D"http://download.java.net/maven/1/commons-fileupload/poms  
> /commons-fileupload-1.2.pom">here</a>. </p>  
98 > <hr>  
> <address>Apache Server at maven-repository.dev.java.net Port  
100 > 443</address>  
> </body></html>  
102 > #####  
>  
104 >  
> Okay - that explains the problem - thanks. And how can I solve that?  
106 >  
>  
108 > > Does it look sensible (pom xml style) or is it html?  
> > Are you connecting to the internet from behind a proxy?  
110 > No. Direct connection without proxy.  
>  
112 >  
> > If so, you need to specify the proxy in the settings.xml.  
114 > You could also send a copy of your settings.xml. If all  
> connection problems are ruled out move ~/.m2/repository to  
116 > ~/~/.m2/repository.orig or similar and run 'mvn clean install'  
> in the main folder. This will make maven to download all  
118 > external artifacts again.  
> I use the original maven settings file and added the  
120 > 52n-start profile as described on your homepage.  
>  
122 > This is the settings file:  
>  
124 > #####  
> <?xml version=3D"1.0" encoding=3D"UTF-8"?>  
126 >  
> <!--  
128 > Licensed to the Apache Software Foundation (ASF) under one
```

```
> or more contributor license agreements. See the NOTICE file
130 > distributed with this work for additional information
> regarding copyright ownership. The ASF licenses this file
132 > to you under the Apache License, Version 2.0 (the
> "License"); you may not use this file except in compliance
134 > with the License. You may obtain a copy of the License at
>
136 >     http://www.apache.org/licenses/LICENSE-2.0
>
138 > Unless required by applicable law or agreed to in writing,
> software distributed under the License is distributed on an
140 > "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
> KIND, either express or implied. See the License for the
142 > specific language governing permissions and limitations
> under the License.
144 > -->
>
146 > <!--
> | This is the configuration file for Maven. It can be
148 > specified at two levels:
> |
150 > | 1. User Level. This settings.xml file provides
> configuration for a single user,
152 > | and is normally provided in
> ${user.home}/.m2/settings.xml.
154 > |
> | NOTE: This location can be overridden with
156 > the CLI option:
> |
158 > | -s /path/to/user/settings.xml
> |
160 > | 2. Global Level. This settings.xml file provides
> configuration for all Maven
162 > | users on a machine (assuming they're all
> using the same Maven
164 > | installation). It's normally provided in
> | ${maven.home}/conf/settings.xml.
166 > |
> | NOTE: This location can be overridden with
168 > the CLI option:
> |
170 > | -gs /path/to/global/settings.xml
> |
172 > | The sections in this sample file are intended to give you
> a running start at
174 > | getting the most out of your Maven installation. Where
> appropriate, the default
176 > | values (values used when the setting is not specified) are
```

```
> provided.  
178 > |  
> |-->  
180 > <settings xmlns=3D"http://maven.apache.org/SETTINGS/1.0.0"  
>           xmlns:xsi=3D"http://www.w3.org/2001/XMLSchema-instance"  
182 >  
    > xsi:schemaLocation=3D"http://maven.apache.org/SETTINGS/1.0.0  
184 > http://maven.apache.org/xsd/settings-1.0.0.xsd">  
    >     <!-- localRepository  
186 >     | The path to the local repository maven will use to store  
    > artifacts.  
188 >     |  
    >     | Default: ~/.m2/repository  
190 >     <localRepository>/path/to/local/repo</localRepository>  
    > -->  
192 >  
    >     <!-- interactiveMode  
194 >     | This will determine whether maven prompts you when it  
    > needs input. If set to false,  
196 >     | maven will use a sensible default value, perhaps based  
    > on some other setting, for  
198 >     | the parameter in question.  
    > |  
200 >     | Default: true  
    >     <interactiveMode>true</interactiveMode>  
202 >     -->  
    >  
204 >     <!-- offline  
    >     | Determines whether maven should attempt to connect to  
206 > the network when executing a build.  
    >     | This will have an effect on artifact downloads, artifact  
208 > deployment, and others.  
    > |  
210 >     | Default: false  
    >     <offline>false</offline>  
212 >     -->  
    >  
214 >     <!-- pluginGroups  
    >     | This is a list of additional group identifiers that will  
216 > be searched when resolving plugins by their prefix, i.e.  
    >     | when invoking a command line like "mvn prefix:goal".  
218 > Maven will automatically add the group identifiers  
    >     | "org.apache.maven.plugins" and "org.codehaus.mojo" if  
220 > these are not already contained in the list.  
    >     |-->  
222 >     <pluginGroups>  
    >     <!-- pluginGroup  
224 >     | Specifies a further group identifier to use for plugin lookup.
```

```
>      <pluginGroup>com.your.plugins</pluginGroup>
226 >      -->
>      </pluginGroups>
228 >
>      <!-- proxies
230 >      | This is a list of proxies which can be used on this
> machine to connect to the network.
232 >      | Unless otherwise specified (by system property or
> command-line switch), the first proxy
234 >      | specification in this list marked as active will be used.
>      |-->
236 >      <proxies>
>      <!-- proxy
238 >      | Specification for one proxy, to be used in connecting
> to the network.
240 >      |
>      <proxy>
242 >          <id>optional</id>
>          <active>true</active>
244 >          <protocol>http</protocol>
>          <username>proxyuser</username>
246 >          <password>proxypass</password>
>          <host>proxy.host.net</host>
248 >          <port>80</port>
>          <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
250 >      </proxy>
>      -->
252 >      </proxies>
>
254 >      <!-- servers
>      | This is a list of authentication profiles, keyed by the
256 > server-id used within the system.
>      | Authentication profiles can be used whenever maven must
258 > make a connection to a remote server.
>      |-->
260 >      <servers>
>      <!-- server
262 >          | Specifies the authentication information to use when
> connecting to a particular server, identified by
264 >          | a unique name within the system (referred to by the
> 'id' attribute below).
266 >          |
>          | NOTE: You should either specify username/password OR
268 > privateKey/passphrase, since these pairings are
>          | used together.
270 >          |
>          <server>
272 >              <id>deploymentRepo</id>
```

```
>      <username>repouser</username>
274 >      <password>repopwd</password>
>      </server>
276 >      -->
>
278 >      <!-- Another sample, using keys to authenticate.
>      <server>
280 >          <id>siteServer</id>
>          <privateKey>/path/to/private/key</privateKey>
282 >          <passphrase>optional; leave empty if not used.</passphrase>
>          </server>
284 >      -->
>      </servers>
286 >
>      <!-- mirrors
288 >          | This is a list of mirrors to be used in downloading
>          artifacts from remote repositories.
290 >          |
>          | It works like this: a POM may declare a repository to
292 > use in resolving certain artifacts.
>          | However, this repository may have problems with heavy
294 > traffic at times, so people have mirrored
>          | it to several places.
296 >          |
>          | That repository definition will have a unique id, so we
298 > can create a mirror reference for that
>          | repository, to be used as an alternate download site.
300 > The mirror site will be the preferred
>          | server for that repository.
302 >      |-->
>      <mirrors>
304 >          <!-- mirror
>          | Specifies a repository mirror site to use instead of a
306 > given repository. The repository that
>          | this mirror serves has an ID that matches the mirrorOf
308 > element of this mirror. IDs are used
>          | for inheritance and direct lookup purposes, and must
310 > be unique across the set of mirrors.
>          |
312 >      <mirror>
>          <id>mirrorId</id>
314 >          <mirrorOf>repositoryId</mirrorOf>
>          <name>Human Readable Name for this Mirror.</name>
316 >          <url>http://my.repository.com/repo/path</url>
>          </mirror>
318 >      -->
>      </mirrors>
320 >
```

```
>      <!-- profiles
322 >      | This is a list of profiles which can be activated in a
> variety of ways, and which can modify
324 >      | the build process. Profiles provided in the settings.xml
> are intended to provide local machine-
326 >      | specific paths and repository locations which allow the
> build to work in the local environment.
328 >      |
>      | For example, if you have an integration testing plugin -
330 > like cactus - that needs to know where
>      | your Tomcat instance is installed, you can provide a
332 > variable here such that the variable is
>      | dereferenced during the build process to configure the
334 > cactus plugin.
>      |
336 >      | As noted above, profiles can be activated in a variety
> of ways. One way - the activeProfiles
338 >      | section of this document (settings.xml) - will be
> discussed later. Another way essentially
340 >      | relies on the detection of a system property, either
> matching a particular value for the property,
342 >      | or merely testing its existence. Profiles can also be
> activated by JDK version prefix, where a
344 >      | value of '1.4' might activate a profile when the build
> is executed on a JDK version of '1.4.2_07'.
346 >      | Finally, the list of active profiles can be specified
> directly from the command line.
348 >      |
>      | NOTE: For profiles defined in the settings.xml, you are
350 > restricted to specifying only artifact
>      | repositories, plugin repositories, and free-form
352 > properties to be used as configuration
>      | variables for plugins in the POM.
354 >      |
>      |-->
356 >  <profiles>
>      <!-- profile
358 >      | Specifies a set of introductions to the build process,
> to be activated using one or more of the
360 >      | mechanisms described above. For inheritance purposes,
> and to activate profiles via <activatedProfiles/>
362 >      | or the command line, profiles have to have an ID that
> is unique.
364 >      |
>      | An encouraged best practice for profile identification
366 > is to use a consistent naming convention
>      | for profiles, such as 'env-dev', 'env-test',
368 > 'env-production', 'user-jdcasey', 'user-brett', etc.
```

```
>      | This will make it more intuitive to understand what
370 > the set of introduced profiles is attempting
>      | to accomplish, particularly when you only have a list
372 > of profile id's for debug.
>      |
374 >      | This profile example uses the JDK version to trigger
> activation, and provides a JDK-specific repo.
376 >      <profile>
377 >          <id>jdk-1.4</id>
378 >
379 >          <activation>
380 >              <jdk>1.4</jdk>
381 >          </activation>
382 >
383 >          <repositories>
384 >              <repository>
385 >                  <id>jdk14</id>
386 >                  <name>Repository for JDK 1.4 builds</name>
387 >                  <url>http://www.myhost.com/maven/jdk14</url>
388 >                  <layout>default</layout>
389 >                  <snapshotPolicy>always</snapshotPolicy>
390 >              </repository>
391 >          </repositories>
392 >      </profile>
393 >      -->
394 >
395 >      <!--
396 >          | Here is another profile, activated by the system
397 >          | property 'target-env' with a value of 'dev',
398 >          | which provides a specific path to the Tomcat instance.
399 >          To use this, your plugin configuration
400 >          | might hypothetically look like:
401 >          |
402 >          | ...
403 >          | <plugin>
404 >              |   <groupId>org.myco.myplugins</groupId>
405 >              |   <artifactId>myplugin</artifactId>
406 >              |
407 >              |   <configuration>
408 >                  |       <tomcatLocation>\${tomcatPath}</tomcatLocation>
409 >                  |   </configuration>
410 >              | </plugin>
411 >              |
412 >              | ...
413 >              | NOTE: If you just wanted to inject this configuration
414 > whenever someone set 'target-env' to
415 >      |      anything, you could just leave off the <value/>
416 > inside the activation-property.
```

```
>      |
418 >      <profile>
>          <id>env-dev</id>
420 >
>          <activation>
422 >              <property>
>                  <name>target-env</name>
424 >                  <value>dev</value>
>              </property>
426 >          </activation>
>
428 >          <properties>
>              <tomcatPath>/path/to/tomcat/instance</tomcatPath>
430 >          </properties>
>      </profile>
432 >      -->
>      <profile>
434 >          <id>52n-start</id>
>          <repositories>
436 >              <repository>
>                  <id>n52-releases</id>
438 >                  <name>52n Releases</name>
>                  <url>http://52north.org/maven/repo/releases/</url>
440 >              <releases>
>                  <enabled>true</enabled>
442 >              </releases>
>              <snapshots>
444 >                  <enabled>false</enabled>
>              </snapshots>
446 >          </repository>
>          <repository>
448 >              <id>n52-snapshots</id>
>              <name>52n Snapshots</name>
450 >              <url>http://52north.org/maven/repo/snapshots/</url>
>              <releases>
452 >                  <enabled>false</enabled>
>              </releases>
454 >              <snapshots>
>              </snapshots>
456 >          </repository>
>      </repositories>
458 >      <pluginRepositories>
>          <pluginRepository>
460 >              <id>n52-releases</id>
>              <name>52n Releases</name>
462 >              <url>http://52north.org/maven/repo/releases/</url>
>          </pluginRepository>
464 >      </pluginRepositories>
```

```
>
466 > </profile>
>
468 >   </profiles>
>
470 >   <!-- activeProfiles
        | List of profiles that are active for all builds.
472 >   |
        >   <activeProfiles>
474 >     <activeProfile>alwaysActiveProfile</activeProfile>
        >     <activeProfile>anotherAlwaysActiveProfile</activeProfile>
476 >   </activeProfiles>
        >   -->
478 >   <activeProfiles>
        >     <activeProfile>52n-start</activeProfile>
480 >   </activeProfiles>
        >
482 > </settings>
        > ######
484 >
        >
486 >
        > Thank you!
488 >
        >
490 > >>
        > >>
492 >
        >
494 >
        > _____
496 > Security mailing list
        > Security@52north.org
498 > http://www2.52north.org/mailman/listinfo/security
        >
```

---