Web-Services-Security
Reicht der REST?

Hans-Joachim Knobloch (Secorvo)
hans-joachim.knobloch@secorvo.de
Kai Jendrian (Secorvo)
kai.jendrian@secorvo.de

OWASP
4. German OWASP DAY
17.11.2011

The OWASP Foundation
http://www.owasp.org

Orientierung:

•Anwender - vor allem aus dem KMU-Umfeld - die Web-Services einsetzen wollen

- Evtl. als „Spiegelbild“ einervorhandenen Web-Applikation
- Evtl. als Ersatz für „veraltete“ Technologien wie DB-Queues, Message-Queues, EDIFACT etc.
- Evtl. als Frontend für vorhandene Systeme

•Entwickler aus dem KMU-Umfeld, die Web-Services entwickeln sollen

•Ziel dieses Vortrags: Denkanstöße, Orientierung geben

Inhalt

- Überblick Ansätze für Web-Services
- Sicherheitsmechanismen!? (Blickwinkel: Technik)
- Einsatzszenarien Web-Services
- Sicherheitsmechanismen?! (Blickwinkel: Einsatz)
- Fazit

OWASP 3

Inhalte und Zielsetzung des Vortrags:

•Technische Rahmenbedingungen und Einsatzgebiete von SOAP vs. REST
•Sicherheitsmechanismen von SOAP vs. REST
•Typische Einsatzszenarien mit sicherheitsrelevanten Aspekten

SOAP ist ein ausgefeiltes Framework:

- XML-basiert
  - Schemata
  - WSDL
- Datenformate und Protokolle von Grund auf neu entworfen
- Libraries und Produkte für viele Plattformen und Sprachen
  - Unterstützung für Versionen und Optionen ist im Einzelfall zu prüfen
- Geeignet für komplexe Architekturen und abgestimmtes Zusammenspiel
  - Strenge Vorgaben durch Frameworks und Standards

SOAP-Beispiel: PayPal SOAP API

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xsi= "  http://www.w3.org/2001/XMLSchema-instance"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
><SOAP-ENV:Header>
            <RequesterCredentials xmlns="urn:ebay:api:PayPalAPI">
                <Credentials xmlns="urn:ebay:apis:eBLBaseComponents">,
                                <Username>api_username</Username>
                                <Password>api_password</Password>
                                <Signature/>
                                <Subject/>
                </Credentials>
            </RequesterCredentials>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
            <specific_api_name_Req xmlns="urn:ebay:api:PayPalAPI">
                <specific_api_name_Request>
                                <Version xmlns=urn:ebay:apis:eBLBaseComponents">service_version
                                </Version>
                                <required_or_optional_fields xsi:type="some_type_here">data
                                </required_or_optional_fields>
                </specific_api_name_Request>
            </specific_api_name_Req>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
            xmlns:SOAP-ENV= "http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xsx="http://www.w3.org/2001/XMLSchema"
            xmlns:ccc="urn:ebay:apis:CoreComponentTypes"
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
            xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
            xmlns:ebl="urn:ebay:apis:eBLBaseComponents"
            xmlns:ns="urn:ebay:api:PayPalAPI">
            <SOAP-ENV:Header>
                <Security
                                xmlns="http://schemas.xmlsoap.org/ws/2002/12/secext"
                                xsi:type="wsse:SecurityType"
                />
                <RequesterCredentials xmlns="urn:ebay:api:PayPalAPI"
                                xsi:type="ebl:CustomSecurityHeaderType">
                                <Credentials
                                                xmlns="urn:ebay:apis:eBLBaseComponents"
                                                xsi:type="ebl:UserIDPasswordType"
                                />
                </RequesterCredentials>
            </SOAP-ENV:Header>
            <SOAP-ENV:Body id="_0">
                <specific_api_name_Response xmlns="urn:ebay:api:PayPalAPI">
                                <Timestamp xmlns="urn:ebay:api:PayPalAPI"> dateTime_in_UTC/GMT
                                </TIMESTAMP>
                                <Ack xmlns="urn:ebay:apis:eBLBaseComponents">Success</Ack>
                                <Version xmlns="urn:ebay:apis:eBLBaseComponents">
                                                serviceVersion
                                </Version>
                                <CorrelationId xmlns="urn:ebay:apis:eBLBaseComponents">
                                                applicationCorrelation
                                </CorrelationID>
                                <Build xmlns="urn:ebay:apis:eBLBaseComponents">
                                                api_build_number
                                </Build>
                                <elements_for_specific_api_response>          data
                                </elements_for_specific_api_response>
                </specific_api_name_Response>
            </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- WSDL-Definition:
  ca. 4.500 Zeilen
- XML-Schema-Definition:
  ca. 14.000 Zeilen
- SDK für PHP:
  ca. 13 MB (unkomprimiert)
- Developer Reference:
  ca. 380 Seiten PDF
- Unterstützte Plattformen:
  - Windows Server 2000, 2003
  - Red Hat Linux 9.0
  - Sun Solaris 9.0
  - .NET 1.1 Service Pack 1
  - .NET 2.0
  - JDK 1.6.x
  - PHP 4.3 or later
  - Tomcat 5.0 or later
  - ColdFusion MX 7

OWASP

REST (Representational State Transfer)

•ist eine Philosophie und kein festes Protokoll (oder Protokollfamilie)

- Im Prinzip wie bei HTTP
- Nutze URLs, HTTP-Kommandos und Status-Codes mit „passender“ Semantik

•Man kennt sich und kann "spontan" zusammen spielen

- Datenaustauschformate werden frei gewählt
  - Kann XML sein („Plain Old XML“, POX), muss aber nicht

REST-Beispiel: PayPal NVP API

Request:

```
USER=someone@unknowncompany.com
&PWD=mypassword
&METHOD=GetExpressCheckoutDetails
&TOKEN=EC-23T233ZP3DFB...
```

Response:

```
ACK=Success
&TIMESTAMP=date/timeOfResponse
&CORRELATIONID=debuggingToken
&VERSION=2.300000
&BUILD=buildNumber
&TOKEN=EC-3DJ78083ES565113B
&EMAIL=abcdef@anyemail.com
&PAYERID=95HR9CM6D56Q2
&PAYERSTATUS=verified
&FIRSTNAME=John
&LASTNAME=Smith...
```

- Beispielcode für PHP:
  ca. 270 KB (unkomprimiert)
- Developer Guide:
  ca. 300 Seiten PDF
- Beispielcode bereit gestellt für:
  - Java
  - ASP.NET
  - Ruby
  - Classic ASP
  - PHP
  - ColdFusion

OWASP

Es ist absehbar, was die meisten Entwickler vorziehen werden...

85% der Nutzung von Amazon Web-Services lief 2003 über das REST interface, der Rest über SOAP
(http://oreilly.com/pub/wlg/3005)

Aber: REST bietet doch viel weniger vordefinierte Sicherheitsmechanismen – ist das nicht ein eindeutiges Argument für SOAP?

Siehe:

https://cms.paypal.com/us/cgi-bin/?cmd=_render-content&content_ID=developer/library_download_sdks

REST-Beispiel: PayPal NVP API

Request:

```
USER=someone@unknowncompany.com
&PWD=mypassword
&METHOD=
&TOKEN=E
```

Response:

```
ACK=Succ
&TIMESTA
&CORRELATIONID=debugging|oken
&VERSION=2.300000
&BUILD=buildNumber
&TOKEN=EC-3DJ78083ES565113B
&EMAIL=abcdef@anyemail.com
&PAYERID=95HR9CM6D56Q2
&PAYERSTATUS=verified
&FIRSTNAME=John
&LASTNAME=Smith...
```

- Beispielcode für PHP: ca. 270 KB (unkomprimiert)
- Developer Guide:

lt für:

- ColdFusion

"PayPal recommends that you use the PayPal NVP interface to the PayPal API unless you are already familiar with using SOAP web services."

OWASP

Es ist absehbar, was die meisten Entwickler vorziehen werden...

85% der Nutzung von Amazon Web-Services lief 2003 über das REST interface, der Rest über SOAP
(http://oreilly.com/pub/wlg/3005)

Aber: REST bietet doch viel weniger vordefinierte Sicherheitsmechanismen – ist das nicht ein eindeutiges Argument für SOAP?

Siehe:

https://cms.paypal.com/us/cgi-bin/?cmd=_render-content&content_ID=developer/library_download_sdks

Sicherheitsmechanismen für REST

- HTTP-Authentifikation
  - Authentifikation des Clients
- SSL/TLS
  - Authentifikation des Services
  - Authentifikation des Clients (optional)
  - Vertraulichkeit und Integrität der Übertragung (Punkt-zu-Punkt!)
- ???
  - Identitäts-Management
  - Autorisierung
  - Nicht-Abstreitbarkeit
  - Verfügbarkeit / Bandbreitenbeschränkung

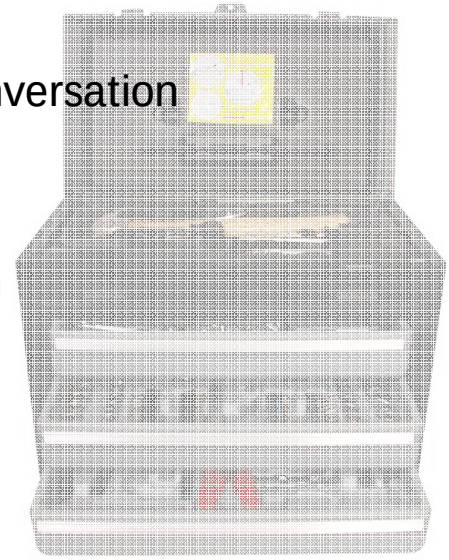
OWASP

Getreu der Philopsophie: "Man nehme, was vorhanden ist...":

- HTTP-Authentication
  - Basic, Digest, NTLM, SPNEGO (Kerberos), ...
- SSL/TLS
  - Ohne Client-Authentication
  - Mit Client-Authentication
- Andere, explizit ausgewählte Sicherheitsmechanismen

Sicherheitsmechanismen für SOAP

- WS-Security
- WS-SecureConversation
- WS-Policy
- WS-Trust
- WS-Federation
- WS-Privacy
- …



OWASP

WS-Security

Web Services Security: SOAP Message Security 1.1

OASIS Open, 1 February 2006

- Ende-zu-Ende(!) Integrität und Nicht-Abstreitbarkeit
  - XML Digital Signature
- Ende-zu-Ende(!) Vertraulichkeit
  - XML Encryption
- Sender-Authentifikation/Autorisierung
  - „Username Token“ (euphemistisch für: Klartext-Passwort)
  - X.509 Zertifikate
  - SAML-Token
  - Kerberos-Token
  - Rights Expression Language Token (DRM)

```
<S11:Header>
  ...
  <wsse:Security>
    <wsse:UsernameToken>
      <wsse:Username>Zoe</ws
      <wsse:Password>IloveD
    </wsse:UsernameToken>
  ...
</S11:Header>
  ...
</S11:Envelope>
```

OWASP

WS-SecureConversation

WS-SecureConversation 1.4

OASIS Open, 2 February 2009

- Ziel: effizientere Absicherung von mehreren aufeinanderfolgenden SOAP-Requests
- Security Context / Sitzungsschlüssel ähnlich SSL/TLS
  - Erstellen
    - Ausgehandelt
    - Vom Sender erstellt
    - Über Security Token Service
  - Erneuern
  - Erweitern
  - Löschen

```
<S11:Header>
  <wsse:Security>
    <wsc:SecurityContextToken ws
      <wsc:Identifier>uuid:...
    </wsc:SecurityContextToken>
    <wsc:DerivedKeyToken wsu:Id=
      <wsse:SecurityTokenRefer
        <wsse:Reference URI=
      </wsse:SecurityTokenRefe
      <wsc:Nonce>KJHFRE...</ws
    </wsc:DerivedKeyToken>
    ...
```

OWASP

WS-Policy

Web Services Policy 1.5 - Framework

W3C Recommendation, 04 September 2007

- Spezifikation von Policy-Anforderungen
  - Hinsichtlich Security, Quality-of-Service, ...
  - Typischerweise Anforderungen des Services (kann in WSDL verankert werden)
  - Ggf. auch Anforderungen des Clients
  - Vereinbarkeit von Service- und Client-Policy?
- Unterspezifikationen des Frameworks u.a.:
  - WS-Policy Assertions
  - WS-Policy Attachments
  - WS-SecurityPolicy

```
<wsp:Policy>
  <sp:AlgorithmSuite>
    <wsp:Policy>
      <wsp:ExactlyOne>
        <sp:Basic256Rsa15 />
        <sp:TripleDesRsa15 />
      </wsp:ExactlyOne>
    </wsp:Policy>
  </sp:AlgorithmSuite>
  <sp:TransportToken>
    <wsp:Policy>
      <sp:HttpsToken>
```

OWASP

WS-Trust

WS-Trust 1.4

OASIS Open, 2 February 2009

- Security Token Service (STS)
  - Web-Service zum Ausstellen, Erneuern und Validieren von Security Tokens für WS-Security / WS-SecureConversation
- Eigenschaften u. a.
  - Service(-Endpunkt) kann angeben, welcher STS akzeptiert wird
  - Anforderung von Security Tokens kann an Dritte delegiert werden
- Ähnlichkeiten mit dem Kerberos-Konzept

OWASP

WS-Federation

Web Services Federation Language Version 1.1

BEA, BMC, IBM, Layer7, Microsoft, Novell, VeriSign, December 2006

Web Services Federation Language Version 1.2

OASIS Open Committee Specification 01, March 4 2009

- Erweiterung von WS-Trust zur Nutzung föderierter Identitäten
  - Ziele: vorhandene IDs nutzen, Single-Sign-On, ...
- Teile der Spezifikation u. a.:
  - „Federation Metadata“ als Erweiterung von WSDL/WS-Policy
  - Sign-Out
  - Attribute, Pseudonyme
  - Authorization

OWASP

„WS-Privacy“

The Platform for Privacy Preferences 1.0 (P3P1.0)

W3C Recommendation, 16 April 2002

- XML-Beschreibung von Datenschutzrichtlinien
  - Primär für Browser-basierte Web-Anwendungen
  - Ggf. für Web-Services: Policy-Beschreibung als Erweiterung von WS-Policy, WS-Trust
- Unabhängig von der Definition von Privacy-Attributen in WS-Federation

```
<ACCESS><nonident/></ACCESS>
<DISPUTES-GROUP>
  <DISPUTES resolution-type="inde
    service="http://www.PrivacySe
    short-description="PrivacySea
    IMPACTED <correct/></REMEDI
  </DISPUTES>
</DISPUTES-GROUP>
<STATEMENT>
  <PURPOSE><admin/><develop/></PU
  <RECIPIENT><ours/></RECIPIENT>
  <RETENTION><stated-purpose/></R
```

OWASP

WS-* Standards insgesamt: Viele Mechanismen, aber auch viele Baustellen:

•Mehrere beteiligte Gremien

•Keine öffentlichen Trust-/Federation-Services

•Manche Standards sehen nach Sackgasse aus (Bsp.: P3P für Web-Services)

•Aktuelle Angriffe auf

- XML-Signature (Signature Wrapping)
- XML-Encryption (CBC Padding Oracle)

Wer kümmert sich um Verfügbarkeit???

•Keiner der Standards!

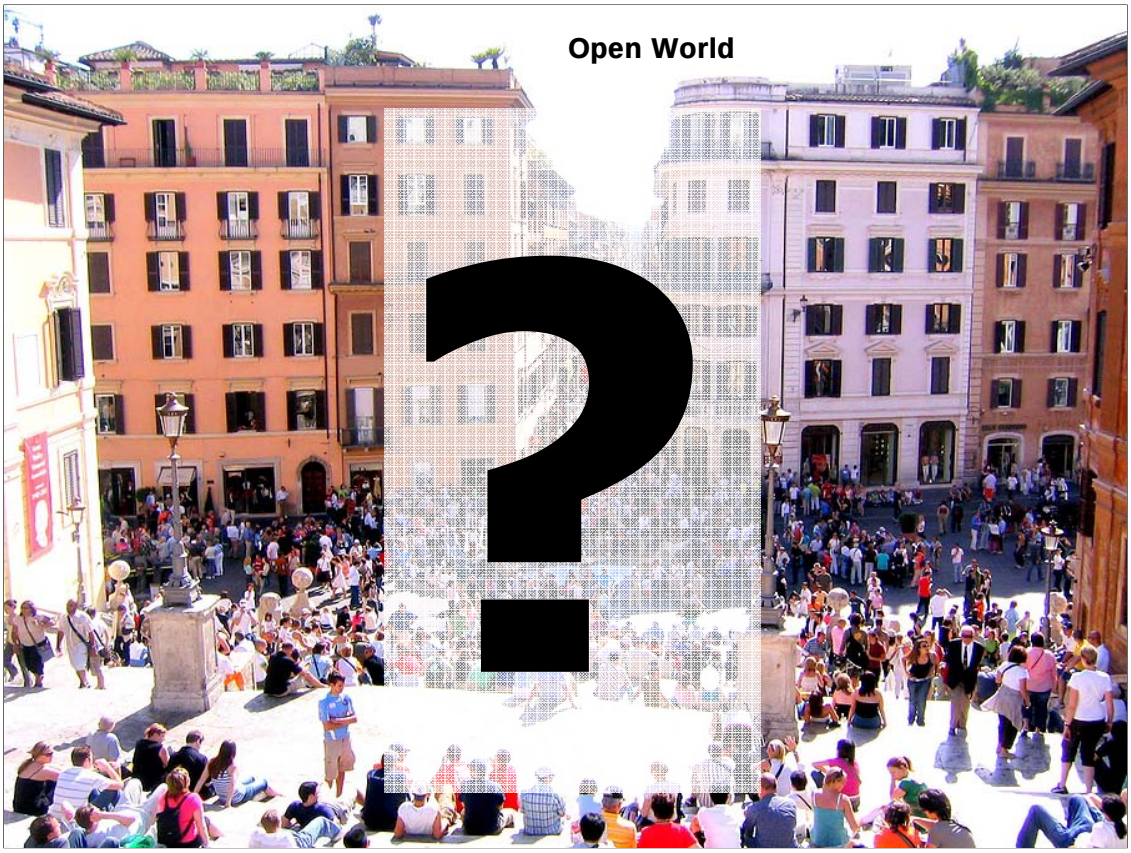
•Nur protprietäre Erweiterungen für SOAP

- Bsp.: Apache Synapse ESB (http://synapse.apache.org/)
  - als erweiterte Attribute von WS-Policy

Technische Sicherheitsmechanismen

SOAP	-	REST
5	:	2

OWASP



Szenario (A): Offene Anwendungslandschaft

•Offene Architektur

- Suche nach beliebigen Services soll möglich sein
- Mitspieler müssen zueinander finden
- Mitspieler kennen und vertrauen sich nicht

SOAP bietet Antworten auf die Fragen:

•Wie finde ich Service-Partner?

•Wie etabliere ich ein Vertrauen?

- Identitifkation & Authentifizierung
- Autorisierung
- Vertrauen
- Vertraulichkeit

Szenario (B): Closed User Group

•Geschlossene Architektur

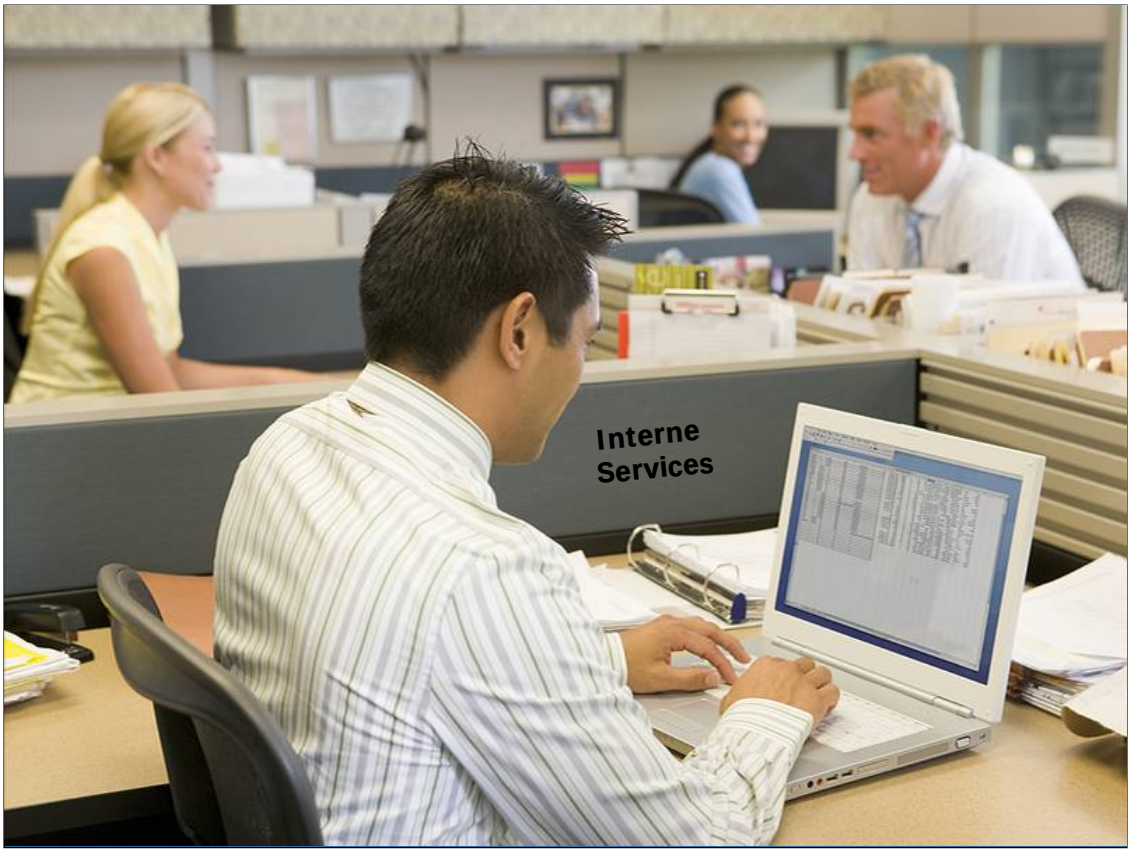
- Services stehen vorab fest
- Mitspieler sind einander bekannt
  - Neue Mitspieler können hinzukommen
- Mitspieler kennen und vertrauen sich begrenzt
  - Vertrauen per se erst einmal nur gegenüber "Croupier" (trusted entity / service provider)

Vertrauen per se nicht vorhanden - muss etabliert werden

Szenario (C): Bilaterale Anbindung

•Geschlossene Architektur

- Services stehen vorab fest
- Mitspieler sind einander bekannt
- Mitspieler kennen und vertrauen sich untereinander
- Vertrauen gegenüber der anderen Partei muss etabliert werden

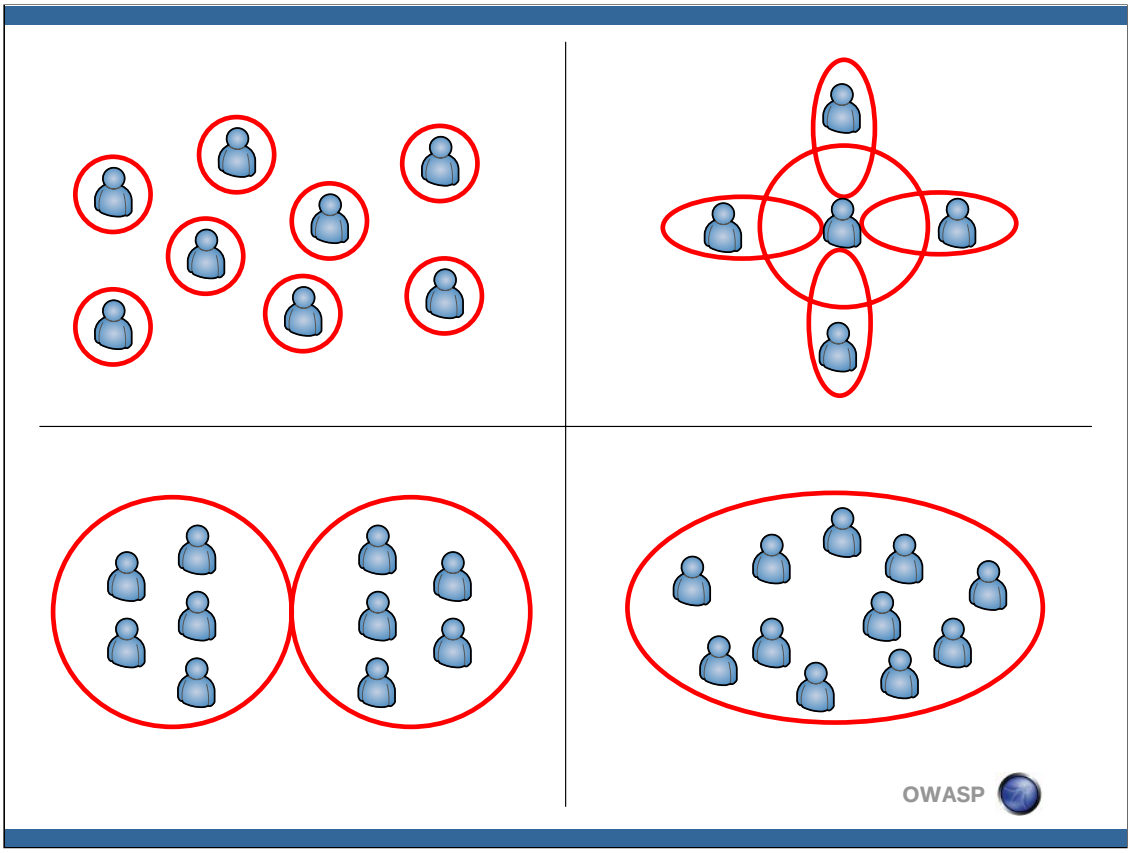


Szenario (D): Interne Kopplungen

•Geschlossene Architektur

- Services stehen vorab fest
- Mitspieler sind einander bekannt
- Mitspieler kennen und vertrauen sich untereinander

Vertrauensgrenzen sind ein wesentliches Konzept für Sicherheitsbetrachtungen von Web-Services.



Die Szenarien unterscheiden sich darin,

•wie viele Vertrauensgrenzen müssen überbrückt werden.
•ob es etablierte Vertrauensübergänge es gibt.
•welche Anforderungen an die Kommunikation.
•welche Dynamik die Vertrauensübergänge erfordern.

Die Kommunikation zwischen Web-Services bei Übergang zwischen Vertrauensgrenzen müssen die Spielregeln festgelegt werden.

Spielregeln: Verträge und Spezifikationen

Spielregeln:

•Umfang des Dienstes

•Verpflichtungen, Rechte und Vergütung

•Haftung

•Datenaustauschformate und -protokolle

•...

Alle organisatorischen Aspekte sind für REST und SOAP explizit festzulegen.

Für SOAP existieren rudimentäre unterstützende Mechanismen auf der technischen Ebene (WSDL, WS-Policy)

Identifikation / Authentifizierung

Bei SOAP:

•WS-Security

•WS-Trust

•WS-SecureConversation

•WS-Federation

aber auch

•Client-Authentifzierung mit SSL/TLS

Bei REST:

•HTTP-Authentifzierung

•Client-Authentifzierung mit SSL/TLS

•Weitere Mechanismen

- Bsp.: OAuth (http://hueniverse.com/oauth/guide/)
- ...

Vertraulichkeit und Integrität:
Schutz des Inhaltes von Nachrichten vor unbefugter Kenntnisnahme und Veränderung

Bei SOAP:
•WS-Security mit XML-Encryption und XML-Signature (Ende zu Ende)
aber auch
•SSL/TLS (Nur für Kommunikationsendpunkte)

Bei REST:
•SSL/TLS
aber auch
•Nachrichtenverschlüsselung (Bsp.: PGP)

Inhaltskontrolle:

Check von Inhalten bei Übergang zwischen Vertrauensgrenzen!

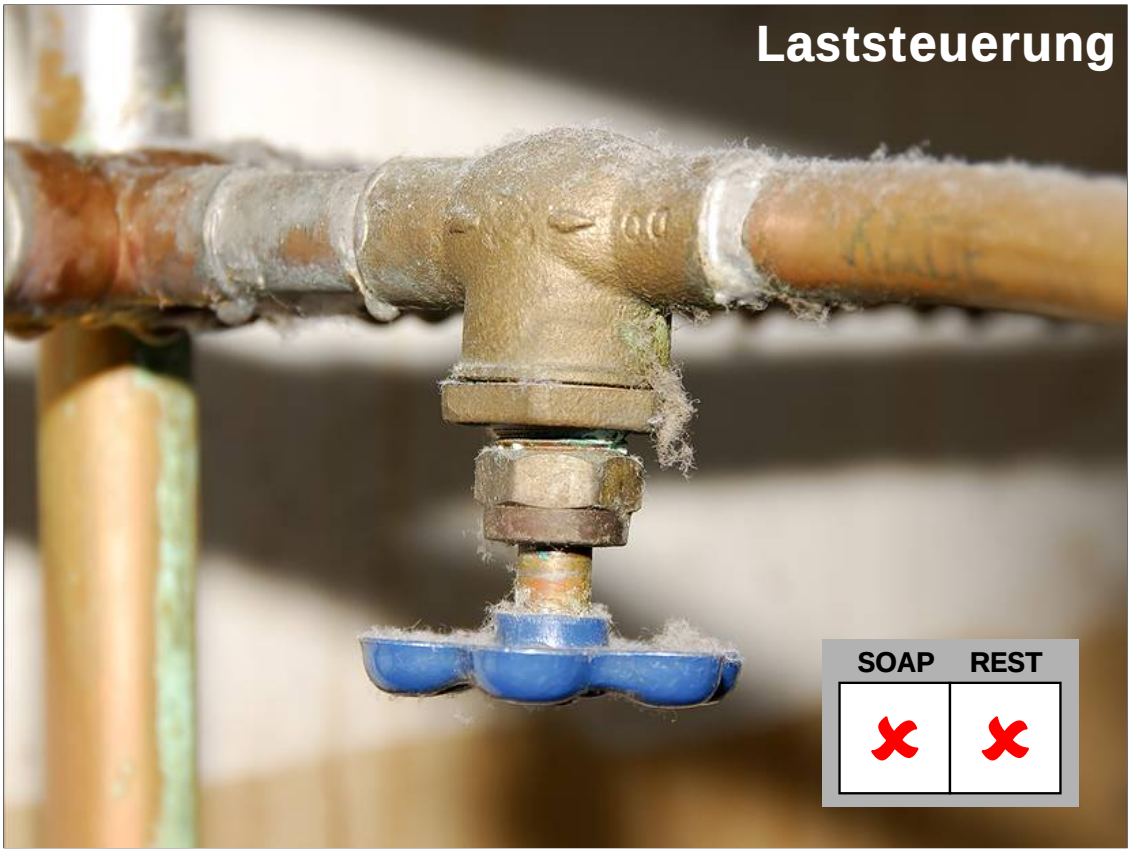
Elementar wichtiger Sicherheitsmechanismus!!!

Bei SOAP:

- Schema-Validierung

Bei REST:

- Manuelle Prüfung von Datentypen



Verfügbarkeit/Laststeuerung:

Regulierung von Anfragen
Durchflusssteuerung

Implementierung muss manuell erfolgen.

Weitere Mechanismen bei Bedarf

- Signatur und –Verschlüsselung von Daten
  - PGP-Signatur und –Verschlüsselung
  - CMS / PKCS#7
  - PDF-Signatur
  - XML Digital Signature, XML Encryption
- Nutzung von Authentifikationsdiensten
- Content-Scan / Viren-Scan
- DRM-Systeme
- …

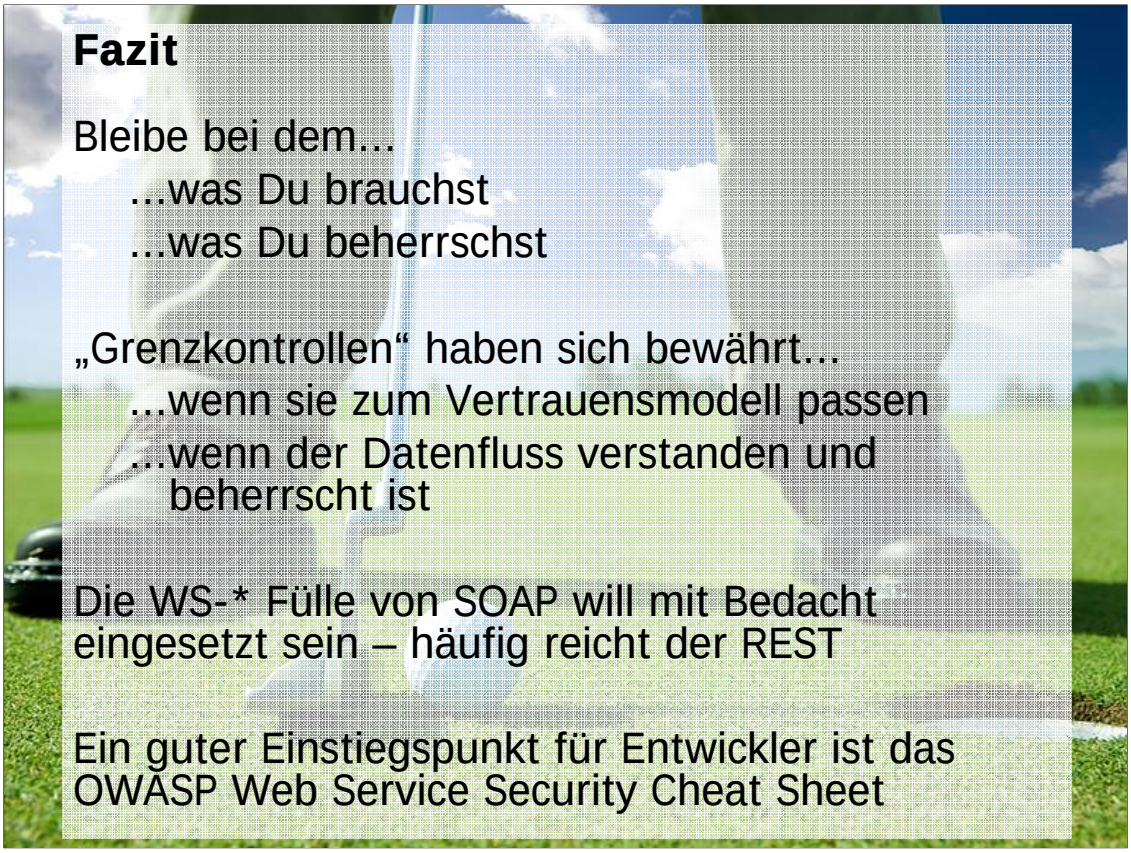
SOAP	REST
✓	✗

OWASP

Benötigte Sicherheitsmechanismen für einfache, praxisrelevante Einsatzszenarien

SOAP	-	REST
3	:	2

OWASP



•KISS ist besser als „blind“ auf ein komplexes Framework zu vertrauen

•Es ist wichtig, sich von der Anwendung her klar zu machen, welche Mechanismen wirklich gebraucht werden

•Vielen heutigen Anwendungsszenarien liegen einfache, klar strukturierte Vertrauensmodelle zu Grunde

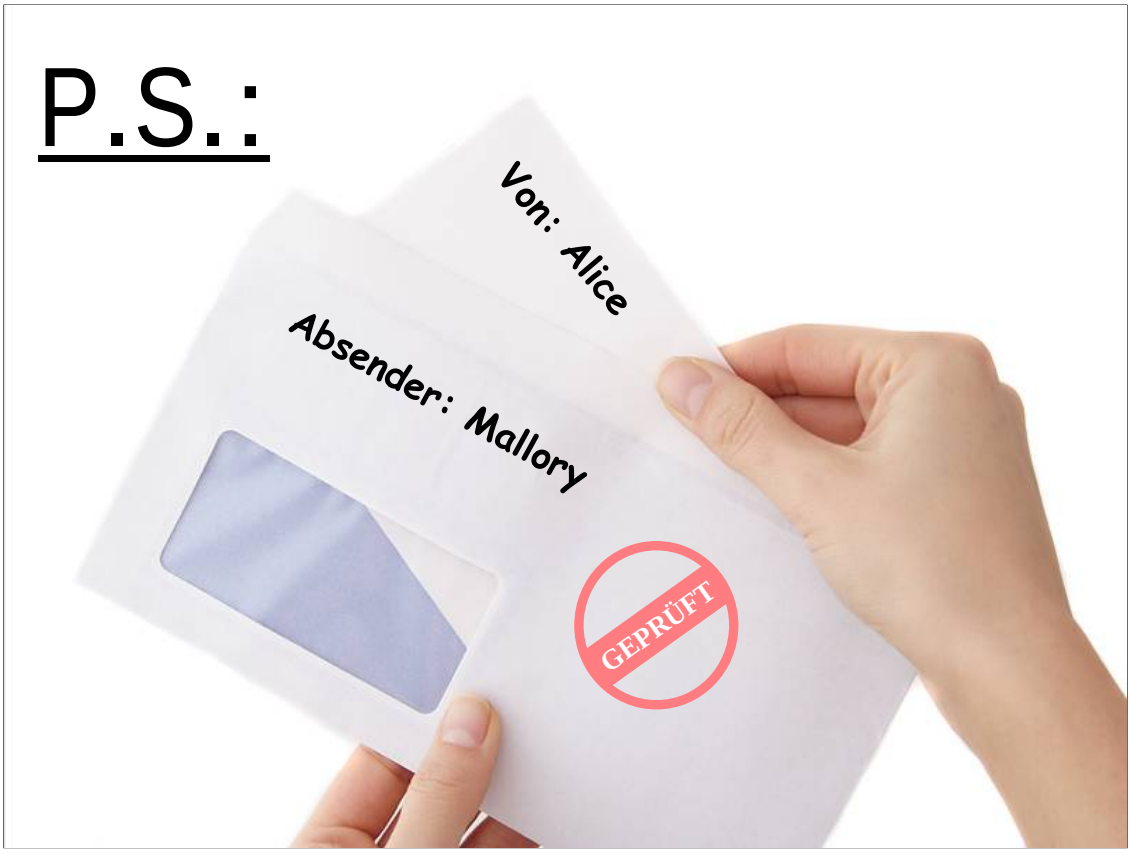
- Dafür reichen meist wenige, grundlegende Sicherheitsmechanismen an den „Vertrauensgrenzen“
  - Das hat sich im Netzwerkbereich seit langem bewährt: Firewalls
  - Hier ist es aber nicht ganz so einfach, die Vertrauensgrenzen zu finden
    - Entspricht der Entwicklung: nur Firewall -> DMZ -> interne Firewalls -> lokale Firewalls/Device Control
- Dazu braucht es ein klares Verständnis des Datenflusses und der Verbindungswege zwischen Anwendungen

•Für diese einfachen Vertrauensmodelle reichen auch die Sicherheitsmechanismen, die man bei REST „vorfindet“ aus

- Fülle an Sicherheitsfunktionen ist also oft kein wirkliches Argument in der Entscheidung SOAP vs. REST

Siehe:

https://www.owasp.org/index.php/Web_Service_Security_Cheat_Sheet (Oktober 2011)



Aus dem richtigen Leben: Eine Straight-Forward SOAP-Implementierung mit SSL/TLS mit Client-Authentifikation...

...schon ganz ohne XML-Signature-Wrapping-Attacken

Bildquellen

Folie 02:	© [2007] Oleksandr Staroseltsev, bigstockphoto.com
Folie 04:	© [2011] Pavel Losevsky, bigstockphoto.com
Folie 06:	© [2011] Rob Marmion, bigstockphoto.com
Folie 09:	© [2007] Ruslan Khabirov, bigstockphoto.com
Folie 10:	© [2005] Graça Victoria, bigstockphoto.com
Folie 17:	© [2007] Robert Asento, bigstockphoto.com
Folie 18:	© [2007] Pavel Losevsky, bigstockphoto.com
Folie 20:	© [2007] Steven Sakata, bigstockphoto.com
Folie 21:	© [2009] charles knox, bigstockphoto.com
Folie 22:	© [2008] Cindy Farmer, bigstockphoto.com
Folie 23:	© [2008] Cathy Yeulet, bigstockphoto.com
Folie 24:	© [2011] 1photo, bigstockphoto.com
Folie 26:	© [2005] martin workman, bigstockphoto.com
Folie 27:	© [2008] anweber, bigstockphoto.com
Folie 28:	© [2009] Ex13, Wikimedia Commons (CC-BY-SA-2.5)
Folie 29:	© [2011] 1photo, bigstockphoto.com
Folie 30:	© [2006] Dana Rothstein, bigstockphoto.com
Folie 31:	© [2007] Andrzej Tokarski, bigstockphoto.com
Folie 33:	© [2008] tyler olson, bigstockphoto.com
Folie 34:	© [2009] Daniel Kaesler, bigstockphoto.com

OWASP