RUHR-UNIVERSITÄT BOCHUM

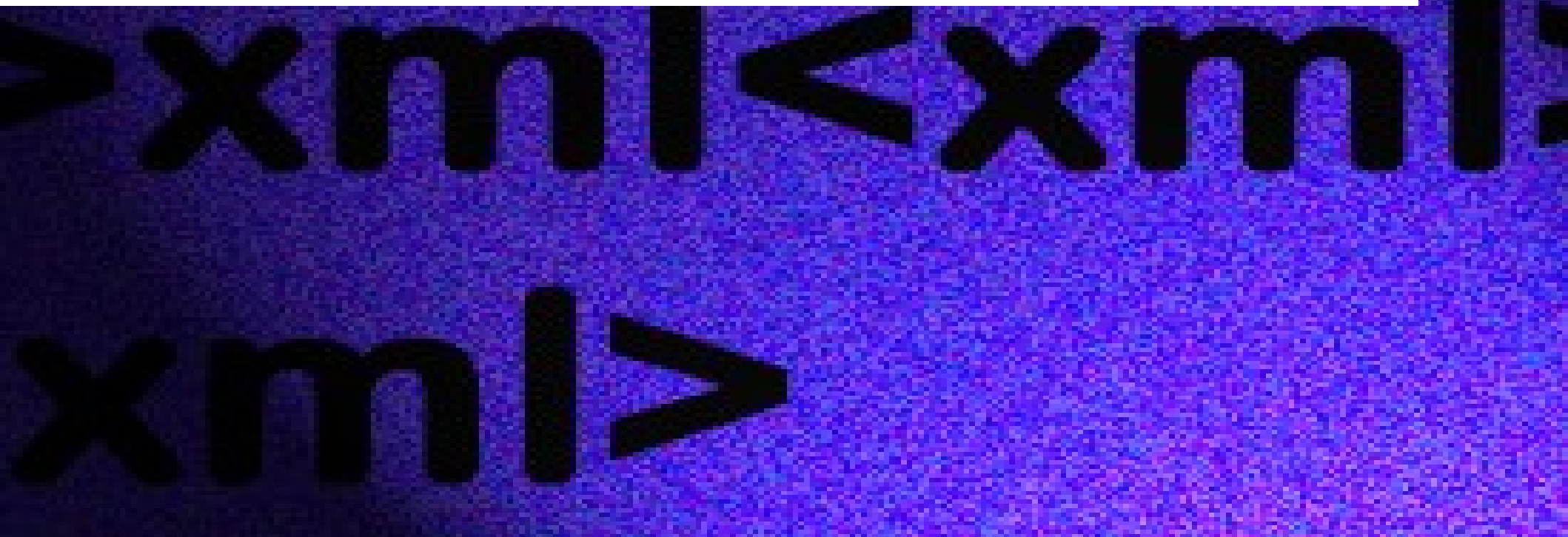
RUB

IWSSC 2011

On the Effectiveness of XML Schema Validation for Countering XML Signature Wrapping Attacks

07.09.2011

Horst Görtz Institute for IT-Security
Chair for Network and Data Security

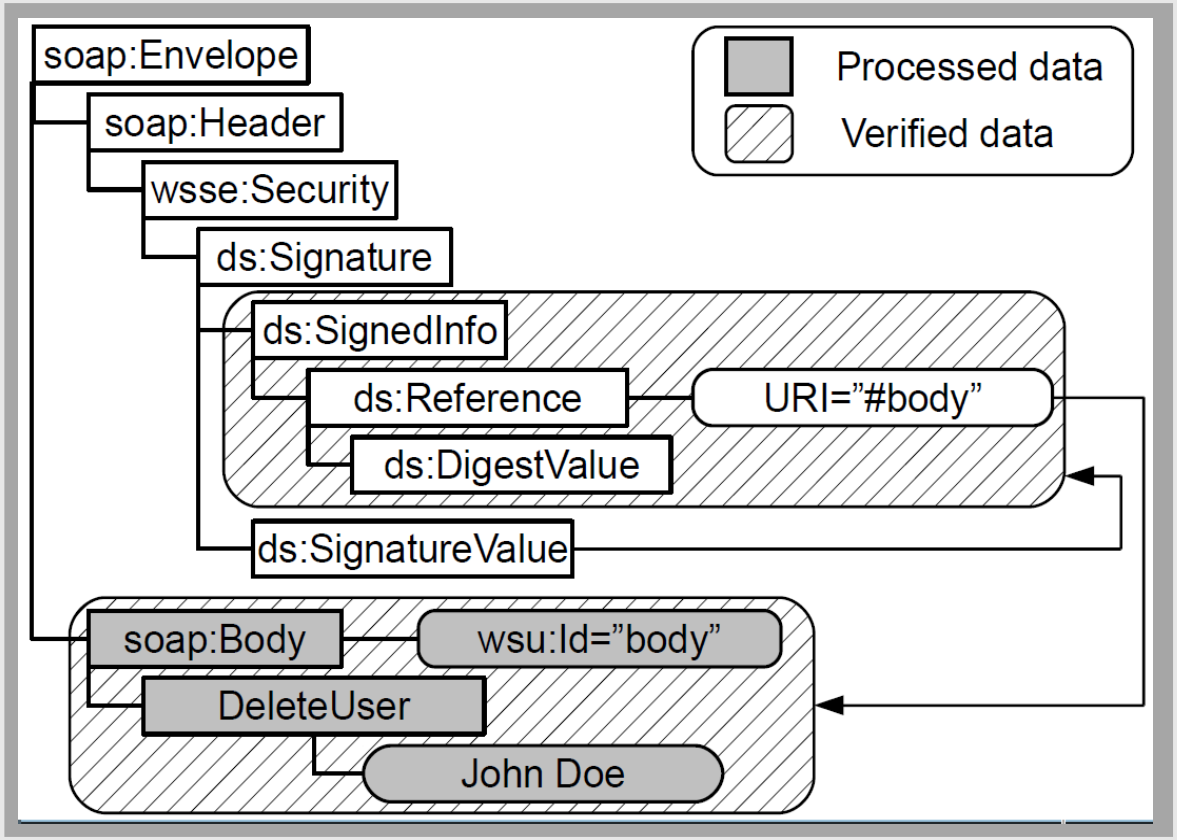


XML Signature
A quick overview

- Standard to enable digital signatures on XML documents
- Multiple ways of application
  - *Enveloped*
  - *Enveloping*
  - *Detached*

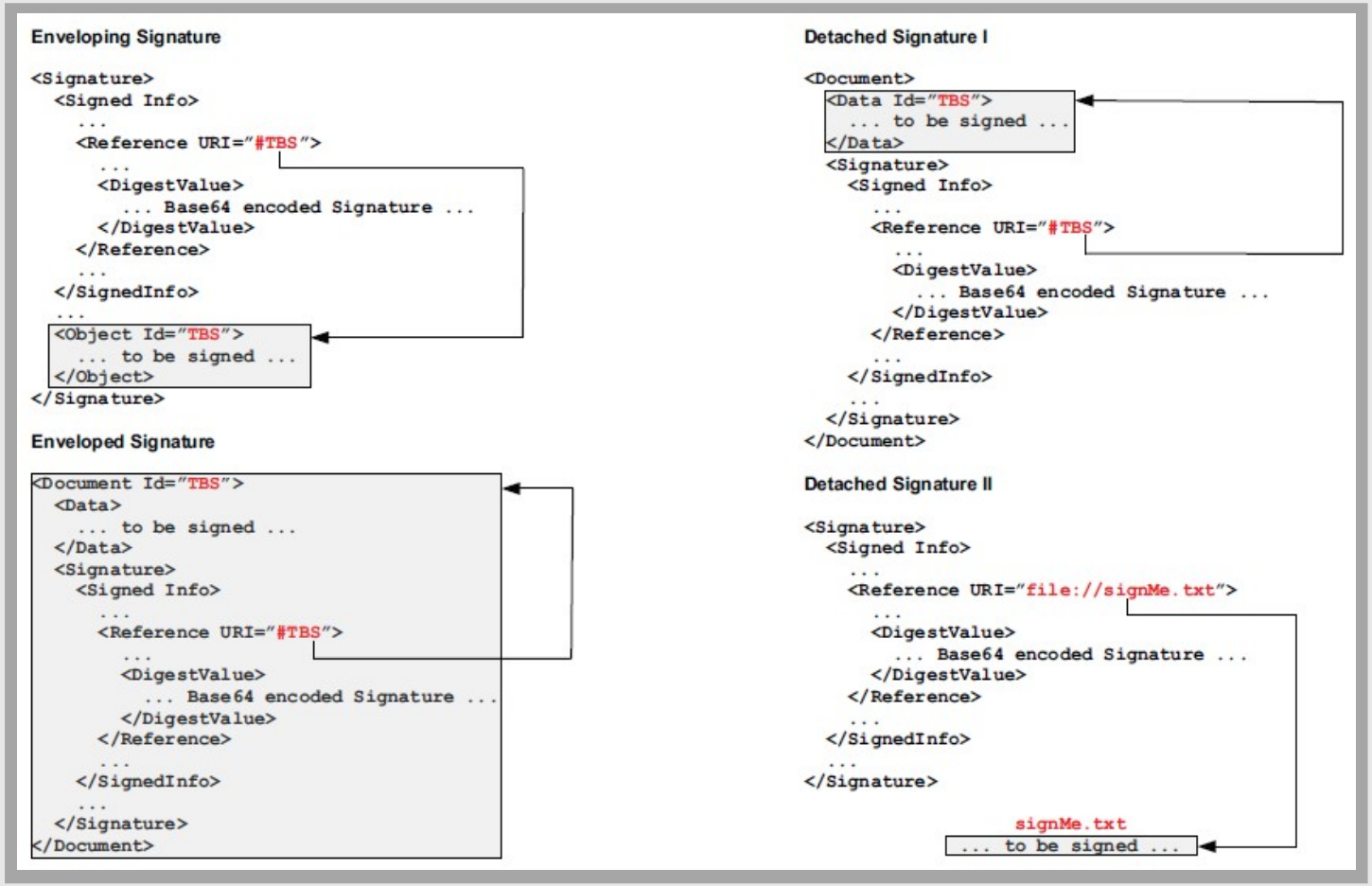
XML Signature
A quick overview

- Standard to enable digital signatures on XML documents
- Multiple ways of application
  - *Enveloped*
  - *Enveloping*
  - *Detached*



XML Signature

Different signature types

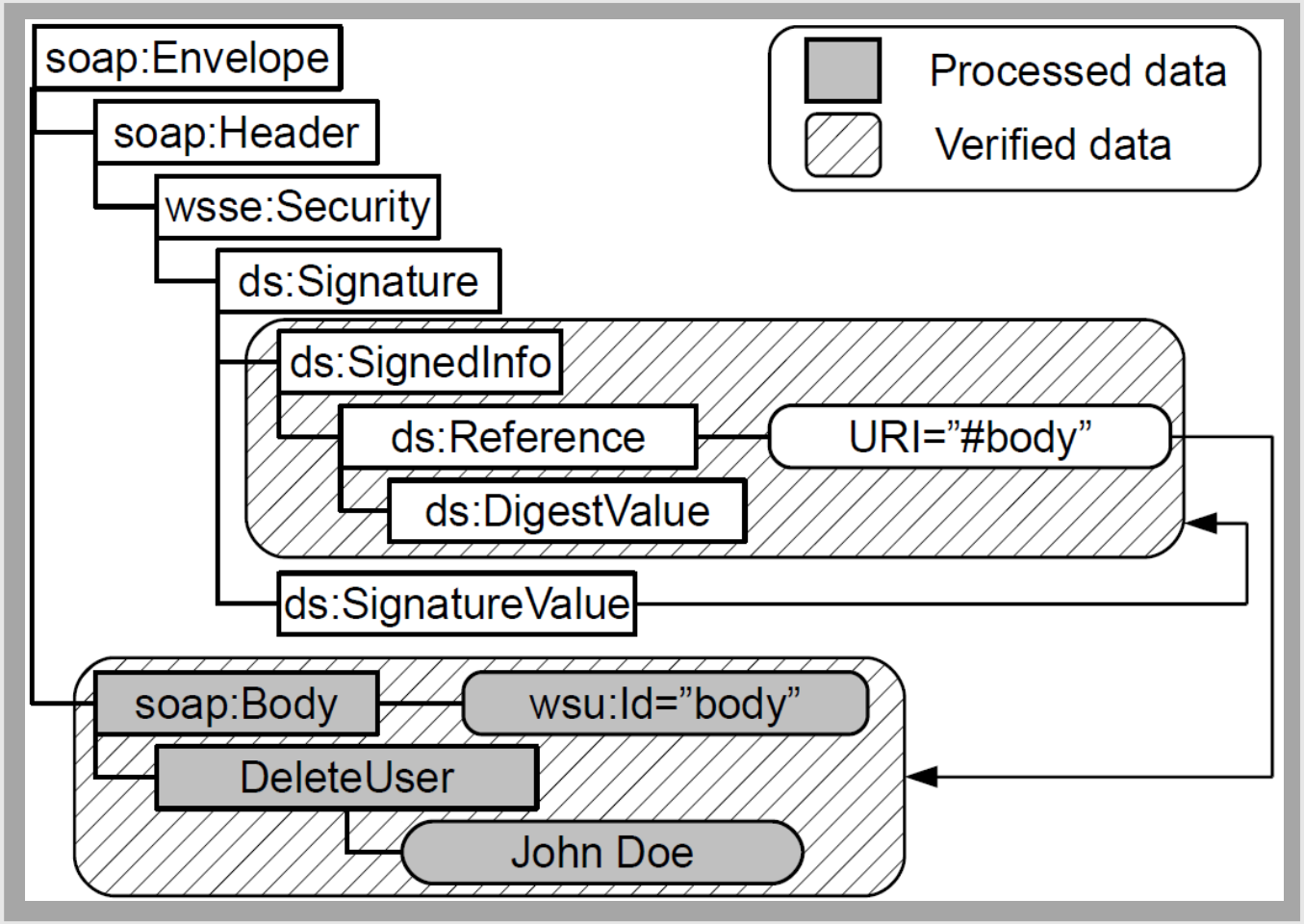


XML Signature
An example

- Action should be secured against manipulation
- Signature aims to provide integrity

XML Signature
An example

- Action should be secured against manipulation
- Signature aims to provide integrity



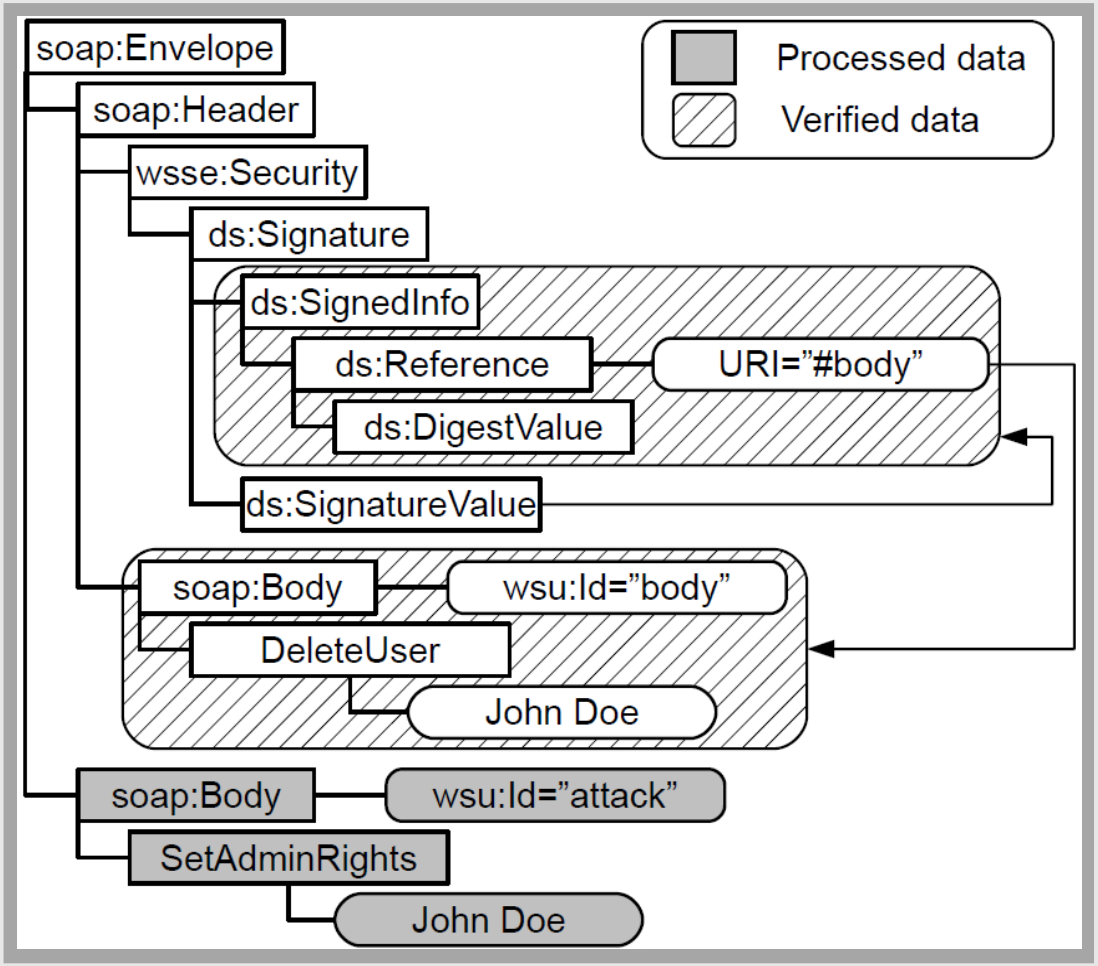
XML Signature Wrapping
A quick overview

- Tricking the executing code to process different data than the validated
- Move verified data
- Add new data

XML Signature Wrapping

A quick overview

- Tricking the executing code to process different data than the validated
- Move signed data
- Add new arbitrary data



XML Schema
A quick overview

- Describes XML structures
- Can be used to verify well-formedness of documents
- Great flexibility
- Itself written in XML

XML Schema

An example snippet

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://schemas.xmlsoap.org/soap/envelope/"
      targetNamespace="http://schemas.xmlsoap.org/soap/envelope/">
                        ...
 <!-- Envelope, header and body -->
 <xs:element name="Envelope" type="tns:Envelope"/>
 <xs:complexType name="Envelope">
  <xs:sequence>
   <xs:element ref="tns:Header" minOccurs="0"/>
   <xs:element ref="tns:Body" minOccurs="1"/>
   <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
          processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
 </xs:complexType>
                        ...
</xs:schema>
```

Source: [http://schemas.xmlsoap.org/soap/envelope/]

XML Schema
Fending the classics

```
<soapenv:Envelope>
    <soapenv:Header>
        <!--
            same header as original message
        -->
    </soapenv:Header>
    <soapenv:Body wsu:Id="id-17547166">
        <ec2:RunInstances>
            … Newly added Body ...
        </ec2:RunInstances>
    </soapenv:Body>
    <soapenv:Body wsu:Id="id-17547166">
        <ec2:DescribeImages>
            … Original Body ...
        </ec2:DescribeImages>
    </soapenv:Body>
</soapenv:Envelope>
```

Source:
[Gruschka, Lo Iacono - Vulnerable Cloud: SOAP Message Security Revisited]

Source:
[http://schemas.xmlsoap.org/soap/envelope/]

XML Schema

Fending the classics

```
<soapenv:Envelope>
      <soapenv:Header>
            <!--
                  same header as original message
            -->
      </soapenv:Header>
      <soapenv:Body wsu:Id="id-17547166">
            <ec2:RunInstances>
                        … Newly added Body ...
            </ec2:RunInstances>
      </soapenv:Body>
      <soapenv:Body wsu:Id="id-17547166">
            <ec2:DescribeImages>
                        … Original Body ...
            </ec2:DescribeImages>
      </soapenv:Body>
</soapenv:Envelope>
```

Source:
[Gruschka, Lo Iacono - Vulnerable Cloud: SOAP Message Security Revisited]

```
<xs:complexType name="Envelope">
  <xs:sequence>
    <xs:element ref="tns:Header"
      minOccurs="0"/>
    <xs:element ref="tns:Body"
      minOccurs="1"/>
    <xs:any namespace="##other"
      minOccurs="0"
      maxOccurs="unbounded"
      processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other"
      processContents="lax"/>
</xs:complexType>
```

Source:
[http://schemas.xmlsoap.org/soap/envelope/]

XML Schema

Breaking the chains by exploiting weakness indicators

```
<xs:complexType name="Header">
      <xs:annotation>
            ...
      </xs:annotation>
      <xs:sequence>
            <xs:any namespace="##any"
                  processContents="lax"
                  minOccurs="0"
                  maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute
            namespace="##other"
            processContents="lax"/>
</xs:complexType>
```

Source: [http://schemas.xmlsoap.org/soap/envelope/]

XML Schema

Breaking the chains by exploiting weakness indicators

```
<xs:complexType name="Header">
      <xs:annotation>
            ...
      </xs:annotation>
      <xs:sequence>
            <xs:any namespace="##any"
                  processContents="lax"
                  minOccurs="0"
                  maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute
            namespace="##other"
            processContents="lax"/>
</xs:complexType>
```

```
<soap:Header>
      .....
      <myownnamespace:wrapper
            xmlns:myownnamespace="http://.....">
            <soap:Body
                  id="referencedBySignature">
                        ...
            </soap>
      </myownnamespace:wrapper>
</soap:Header>

<soap:Body
            id="newBodyProcessedByAppLogic">
</soap:Body>
```

Source:
[http://schemas.xmlsoap.org/soap/envelope/]

XML Schema
Weakness Indicators 1/2

- Element: <xs:any>

  Enables the document to contain elements at this position which are not defined by the schema
- Attribute: namespace="##any"

  Allows the usage of elements from any namespace(s)
- Attribute: namespace="##other"

  Allows the usage of elements from any namespace(s) except the namespace of the parent element

XML Schema
Weakness Indicators 2/2

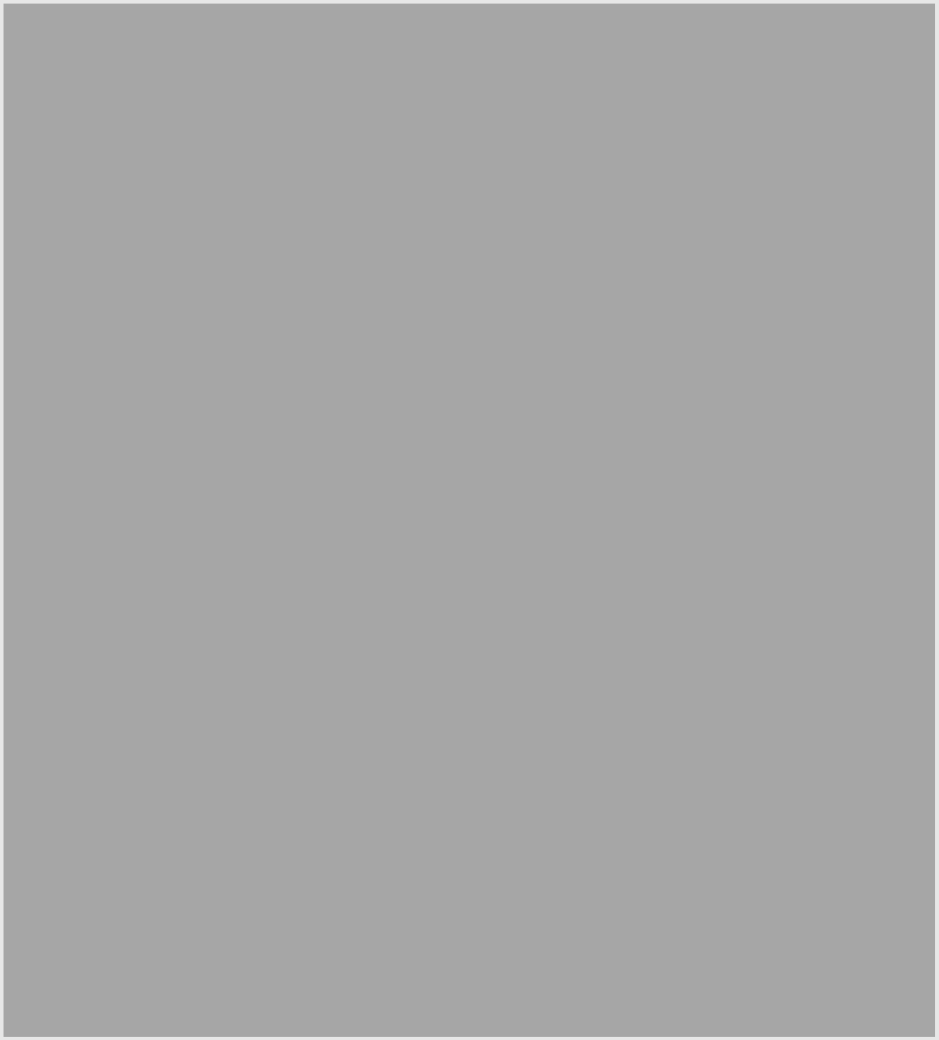
- Attribute: processContents="lax"

  Only validate the content if a valid schema could be obtained
- Attribute: processContents="skip"

  No validation will take place on the namespace(s) referenced by the current element

XML Schema
Schema Hardening

- Bound exclusively to content of the current Schema
- Prohibit any content that is not part of the Schema
- Remove any extension points
- Strictly remove all weakness indicators



XML Schema

Schema Hardening – an example

```
<xs:element name="Header" type="tns:Header"/>
<xs:complexType name="Header">
      <xs:all>
            <!-- WS-Security -->
            <xs:element ref="wsse:Security" minOccurs="0"/>
            <!-- WS-Addressing -->
            <xs:element ref="wsa:MessageID" minOccurs="0"/>
            <xs:element ref="wsa:RelatesTo" minOccurs="0"/>
            <xs:element ref="wsa:To" minOccurs="0"/>
            <xs:element ref="wsa:Action" minOccurs="0"/>
            <xs:element ref="wsa:From" minOccurs="0"/>
            <xs:element ref="wsa:ReplyTo" minOccurs="0"/>
            <xs:element ref="wsa:FaultTo" minOccurs="0"/>
      </xs:all>
      <xs:anyAttribute namespace="##other"
            processContents="lax"/>
</xs:complexType>
```

Hardened XML Schemas
Downsides

- Extremely increased processing time
- All possible successors of all elements have to be determined
- Access to only the determined successors has to be enforced
- All extensions have to be embedded into the hardened scheme

Discussion

Time for your questions

Your questions are welcome.
Please don't hesitate to ask!

Meiko Jensen
meiko.jensen@rub.de

Juraj Somorovsky
juraj.somorovsky@rub.de

Christopher Meyer
christopher.meyer@rub.de

Jörg Schwenk
joerg.schwenk@rub.de