On Breaking SAML: Be Whoever You Want to Be

Juraj Somorovsky and Christian Mainka

Horst-Görtz Institute for IT-Security
Ruhr-University Bochum

7.11.2012
Motivation – XML Security

• W3C Standards: XML Signature and XML Encryption
• Describe various methods for applying cryptographic algorithms to XML documents

<?xml version='1.0'?><PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
Motivation – XML Security

• Usage:
  • Web Services: Method for machine-to-machine communication over networks
  • Used in commerce, finance, government, military, ...
Motivation – XML Security

• New standards, new attacks

• Last year:
  • Signature Wrapping attacks on Amazon and Eucalyptus cloud interfaces
  • Attacks on XML Encryption

• Today:
  • Attacks on SAML-based Single Sign-On systems
  • WS-Attacker: first automated penetration testing tool for XML Security in Web Services
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
Motivation – Single Sign-On

- Too many identities / passwords
- Solution: Single Sign-On

- Advantages: one password for users, no password management for Service Providers
Motivation – Single Sign-On

• OpenID
• OAuth
• Security Assertion Markup Language (SAML)
  • OASIS
  • Web Services or browser-based Single Sign-On
  • Authentication Statements stored in *Assertions*
Motivation – Single Sign-On

• How do we secure the messages?
• Does SSL / TLS help?

• Messages secured only during transport!
Motivation – Single Sign-On

- Does SSL / TLS help?

- Need for message level security!
Motivation – Single Sign-On

• Message level security?

• Realized using XML Signatures
• Are we secure?
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
SAML Assertion

```xml
<saml:Assertion ID="123">
  <saml:Issuer>www.SecureIdP.com</saml:Issuer>
  <saml:Subject>
    <saml:NameID>Bob@SecureIdP.com</saml:NameID>
  </saml:Subject>
  <saml:Conditions>
    NotBefore="2011-08-08T14:42:00Z"
    NotOnOrAfter="2011-08-08T14:47:00Z">
    <saml:AudienceRestriction>
    </saml:AudienceRestriction>
  </saml:Conditions>
</saml:Assertion>
```
Securing SAML with XML Signature

- Two typical usages
Securing SAML with XML Signature

- Naive (typical) processing:
  1. Signature validation: **Id-based**
  2. Assertion evaluation: `/Binding/Assertion/Subject`

![Diagram of SAML assertions and signatures]

- Signature Verification: **valid**
- Assertion Evaluation: **valid**
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
XML Signature Wrapping Attack on SAML

1. Place the original Assertion including its Binding element into another element
2. Change the Id of the original element
3. The Reference now points to the original element: signature is valid
4. Insert a new Assertion
XML Signature Wrapping Attack on SAML

Diagram illustrating the attack process:
- Binding
- Signature
- SignedInfo
- Reference
- URI="#123"
- Id="123"
- Assertion
- Subject
- Bob
- Admin
- Signature Verification
- valid
- Assertion Evaluation
- Admin
XML Signature Wrapping Attack on SAML – Threat model

- Change arbitrary data in the Assertion: Subject, Timestamp ...
- Attacker: everybody who can gain a signed Assertion...
  1. Registering by the Identity Provider
  2. Message eavesdropping
  3. Google Hacking
XML Signature Wrapping Attack on SAML
XML Signature Wrapping Attack on SAML

- How about them?

<table>
<thead>
<tr>
<th>Framework / Provider</th>
<th>Binding</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Axis 2</td>
<td>SOAP</td>
<td>WSO2 Web Services</td>
</tr>
<tr>
<td>Guanxi</td>
<td>HTTP</td>
<td>Sakai Project (<a href="http://www.sakaiproject.org">www.sakaiproject.org</a>)</td>
</tr>
<tr>
<td>Higgins 1.x</td>
<td>HTTP</td>
<td>Identity project</td>
</tr>
<tr>
<td>IBM Datapower XS40</td>
<td>SOAP</td>
<td>Enterprise XML Security Gateway</td>
</tr>
<tr>
<td>JOSSO</td>
<td>HTTP</td>
<td>Motorola, NEC, Redhat</td>
</tr>
<tr>
<td>WIF</td>
<td>HTTP</td>
<td>Microsoft Sharepoint 2010</td>
</tr>
<tr>
<td>OIOSAML</td>
<td>HTTP</td>
<td>Danish eGovernment (e.g. <a href="http://www.virk.dk">www.virk.dk</a>)</td>
</tr>
<tr>
<td>OpenAM</td>
<td>HTTP</td>
<td>Enterprise-Class Open Source SSO</td>
</tr>
<tr>
<td>OneLogin</td>
<td>HTTP</td>
<td>Joomla, Wordpress, SugarCRM, Drupal</td>
</tr>
<tr>
<td>OpenAthens</td>
<td>HTTP</td>
<td>UK Federation (<a href="http://www.eduserv.org.uk">www.eduserv.org.uk</a>)</td>
</tr>
<tr>
<td>OpenSAML</td>
<td>HTTP</td>
<td>Shibboleth, SuisseID</td>
</tr>
<tr>
<td>Salesforce</td>
<td>HTTP</td>
<td>Cloud Computing and CRM</td>
</tr>
<tr>
<td>SimpleSAMLphp</td>
<td>HTTP</td>
<td>Danish e-ID Federation (<a href="http://www.wayf.dk">www.wayf.dk</a>)</td>
</tr>
<tr>
<td>WSO2</td>
<td>HTTP</td>
<td>WSO2 ESB</td>
</tr>
</tbody>
</table>
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures
2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example
3. Conclusion
XML Signature Wrapping Attack on SAML – Results

Guanxi, JOSSO

WSO2
XML Signature Wrapping Attack on SAML – Results

Higgins, Apache Axis2, IBM XS 40

OpenAM, Salesforce
Attack on OpenSAML

• Is Signature Wrapping always that easy?

• OpenSAML implemented a few countermeasures:
  1. Checked if the signed assertion has the same ID value as the processed one
  2. Validated XML Schema
     Not possible to insert two elements with the same ID values
**Attack on OpenSAML**

1. **ID values checking: Basic idea** – using two identical ID values

2. **XML Schema validation:**
   1. Put the Assertion into an extensible element (e.g. `<Extensions>`)  
   2. Two identical ID attributes (XML Xerces Parser bug)

- **Which element is verified?**
  C++ takes the first found element
Attack on OpenSAML

OpenSAML C++ references the **first** found element

OpenSAML Java references the **last** found element
Beyond Signature Wrapping: Signature Exclusion

• Lame but ...

• ...Worked against:
  • Apache Axis2
  • JOSSO
  • OpenAthens
# SAML Signature Wrapping – Summary

<table>
<thead>
<tr>
<th>Framework / Provider</th>
<th>Signature Exclusion</th>
<th>Signature Wrapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Axis 2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Guanxi</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Higgins 1.x</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>IBM Datapower XS40</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>JOSSO</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>WIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OIOSAML</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OpenAM</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>OneLogin</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>OpenAthens</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OpenSAML</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Salesforce</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SimpleSAMLphp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSO2</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

- **Enterprise Applications**
  - Joomla, Wordpress, SugarCRM, Drupal
  - Shibboleth, SwissID...

- **Danish eGovernment**
  - Danish eGovernment
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
Countermeasures

• General problem:
  • different processing modules – different views
Countermeasure 1: Strict Filtering

• Forward only signed elements
• Also called *see-only-what-is-signed*
Countermeasure 2: Data Tainting

- Signature verification generates a random number $r$
- The verified data is tainted with $r$
- $r$ is forwarded to the Assertion evaluation logic
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
Penetration Test Library

• Considered all the attack vectors:
  1. Different permutations of signed / processed Assertions
Attack Permutations

- There are many possibilities
- Dependant of its position
- Dependant of its parent
  - Hard to test manually
Penetration Test Library

• Considered all the attack vectors:
  1. Different permutations of signed / processed Assertions
  2. Id processing
Id processing

Three possibilities
1. Same Id value
2. Different Id value
3. Remove Id value

Processing depends on verification and application logic
Penetration Test Library

• Considered all the attack vectors:
  1. Different permutations of signed / processed Assertions
  2. Id processing
  3. Signature exclusion attacks
Signature Exclusion Attack

Diagram:
- Binding
- Assertion
  - Signature
    - SignedInfo
      - Reference
        - DigestValue
      - SignatureValue
    - Subject
      - Bob
  - AssertionId = "123"
  - URI = "#123"
Penetration Test Library

• Considered all the attack vectors:
  1. Different permutations of signed / processed Assertions
  2. Id processing
  3. Signature exclusion attacks
  4. XML Schema extensions
• Further attacks on Salesforce interface
• Will be included in our WS-Attacker framework
  • http://ws-attacker.sourceforge.net/
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
Concept WS-Attacker

- Modular Framework for Web Services Penetration Testing

- Goals:
  - Easy to use
  - Easy to develop attacks
WS-Attacker’s Current Attacks

• SOAPAction Spoofing

- HTTP: getServerTime
- SOAP: getAdminConf

• WS-Addressing Spoofing

- ReplyTo: Server B

• XML Signature Wrapping for SOAP
  • SAML over SOAP works fine, Browser/REST based is coming soon

• XML Denial of Service will be released in Januar

• XML Encryption Attack is currently developed
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
      3. Practical Evaluation Example

3. Conclusion
Chose Target

![File Explorer]

- WSDL Loader
- Test Request
- Plugin Config
- Attack Overview
- Log
- Expert View

WSDL:
```
http://127.0.0.1:8080/axis2/services/sample06?wsdl
```

- Interface
- Operation

- Prefix
- Uri

- Request Input Table
- Request Expert View

- Name
- Parents
- Value

[INFO] Successfully loaded 10 of 10 plugins

OWASP
Load WSDL

**File**

<table>
<thead>
<tr>
<th>WSDL Loader</th>
<th>Test Request</th>
<th>Plugin Config</th>
<th>Attack Overview</th>
<th>Log</th>
<th>Expert View</th>
</tr>
</thead>
</table>

**WSDL**

http://127.0.0.1:8080/axis2/services/sample06?wsdl

**Interface**

<table>
<thead>
<tr>
<th>sample06Soap12Binding</th>
</tr>
</thead>
</table>

**Operation**

<table>
<thead>
<tr>
<th>echo</th>
</tr>
</thead>
</table>

**Prefix**

| ds | http://www.w3.org/2000/09/xmldsig# |
| ac | http://www.w3.org/2001/10/xml-exc-c14n# |
| ns1 | http://sample06.policy.samples.rampart.apache.org |
| saml | urn:oasis:names:tc:SAML:1.0:assertion |
| samp | urn:oasis:names:tc:SAML:1.0:protocol |

**Request Input Table**

<table>
<thead>
<tr>
<th>Name</th>
<th>Parents</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameIdentifier</td>
<td>soapenv:Envelope -&gt; soapenv:Header -&gt; wsse:Security -&gt; Assertion -&gt; AuthenticationStatement -&gt; Subject</td>
<td>CN=Sample Client, OU=Rampart, C=Apache...</td>
</tr>
<tr>
<td>ConfirmationMethod</td>
<td>soapenv:Envelope -&gt; soapenv:Header -&gt; wsse:Security -&gt; Assertion -&gt; AuthenticationStatement -&gt; Subject -&gt; SubjectConfirmation</td>
<td>urn:oasis:names:tc:SAML:1.0:nameid-format=kp...</td>
</tr>
</tbody>
</table>

**INFO** Has payload? false
Send Test Request

File

WSDL Loader  Test Request  Plugin Config  Attack Overview  Log  Expert View

Request:

```xml
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">  
  <wsse:SecurityTokenReference wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-1.0-WSSE.wsd1.xsd" wsu:Id="STRandAOF2B96F0C7FDCAGF713499437827757">  
    <wsse:KeyInfo Id="KeyRandAOF2B96F0C7FDCAGF713499437827757">  
      <ds:SignatureValue>3140727f2e1589B783B892968a1d4399-a1f8-b138153</ds:SignatureValue>  
      <ds:KeyInfo Id="KeyRandAOF2B96F0C7FDCAGF713499437827757">  
        <ds:SignatureValue>3140727f2e1589B783B892968a1d4399-a1f8-b138153</ds:SignatureValue>  
        <ds:Reference>http://localhost:8080/axis2/services/sample06</ds:Reference>  
      </ds:KeyInfo>  
    </wsse:KeyInfo>  
  </wsse:SecurityTokenReference>  
</wsse:Security>
```

Response:

```xml
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">  
  <soapenv:Header xmlns:wsse="http://www.w3.org/2005/08/addressing">  
  </soapenv:Header>  
  <soapenv:Body>
    <soapenv:Fault xmlns:axis2ns4="http://www.w3.org/2003/05/soap-envelope">
      <soapenv:Code>Sender</soapenv:Code>
      <soapenv:Subcode>MessageExpired</soapenv:Subcode>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

[INFO] Has payload? false
Chose Attack
Run WS-Attacker

<table>
<thead>
<tr>
<th>Time</th>
<th>Level</th>
<th>Source</th>
<th>Content</th>
</tr>
</thead>
</table>
| 16:50:29.821 | Important | Signature Wrapping | 3 signed Elements: 
\rightarrow 3 by ID 
\rightarrow 0 by XPath 
\textquoteleft\textquoteleft; \rightarrow 0 by FastXPath 
\textquoteleft\textquoteleft; \rightarrow 0 by prefix free FastXPath (best) |
| 16:50:46.997 | Critical | Signature Wrapping | Server accepted the request with possibility 0.97 |

Attack-Vector:

```
Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsat:wrapper[1]/Assertion[1]
Payload element Assertion gets a new attribute value 
AssertionID='cK7/mdh-hsfO6vFkCeqhIPAO/T3433E0y7'
Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsat:wrapper[1]/wsu:Timestamp[1]
Payload element wsu:Timestamp gets a new attribute value wsu:Id='Luaw466Dj6'
```

[INFO] Plugin finished: 100/100
Deeper Analysis

Signature Wrapping Analysis

Successful Attack

Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsat:wrapper[1]/Assertion[1]
Payload element Assertion gets a new attribute value AssertionID="VqW7/yh71R0iyVqeyy2b4+S0B5nI3g"
Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsat:wrapper[1]/wsu:Timestamp[1]
Payload element wsu:Timestamp gets a new attribute value wsu:id="wnaDF3eW8U"
1. On Breaking SAML
   1. Motivation – Single Sign-On
   2. Securing SAML with XML Signature
   3. XML Signature Wrapping Attacks
   4. Practical Evaluation
   5. Countermeasures

2. WS-Attacker
   1. Penetration Test Library
   2. Concept: WS-Attacker
   3. Practical Evaluation Example

3. Conclusion
Conclusion

- We showed critical Signature Wrappings in SAML
  - 12 out of 14 frameworks affected!
  - All providers informed
- Huge number of XSW permutations
  - Not easy to find manually
  - Very time consuming when created manually
- WS-Attacker
  - Automatic penetration testing
  - Open Source
  - New Attacks for XML-DoS and XML Encryption under development

http://ws-attacker.sourceforge.net/