On Breaking SAML: Be Whoever You Want to Be

Juraj Somorovsky and
Christian Mainka

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

OWASP
7.11.2012

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

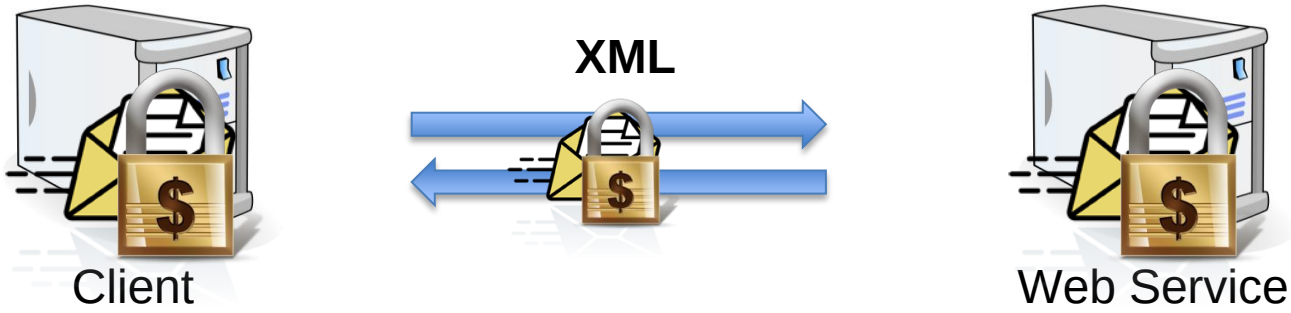
Motivation – XML Security

- 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

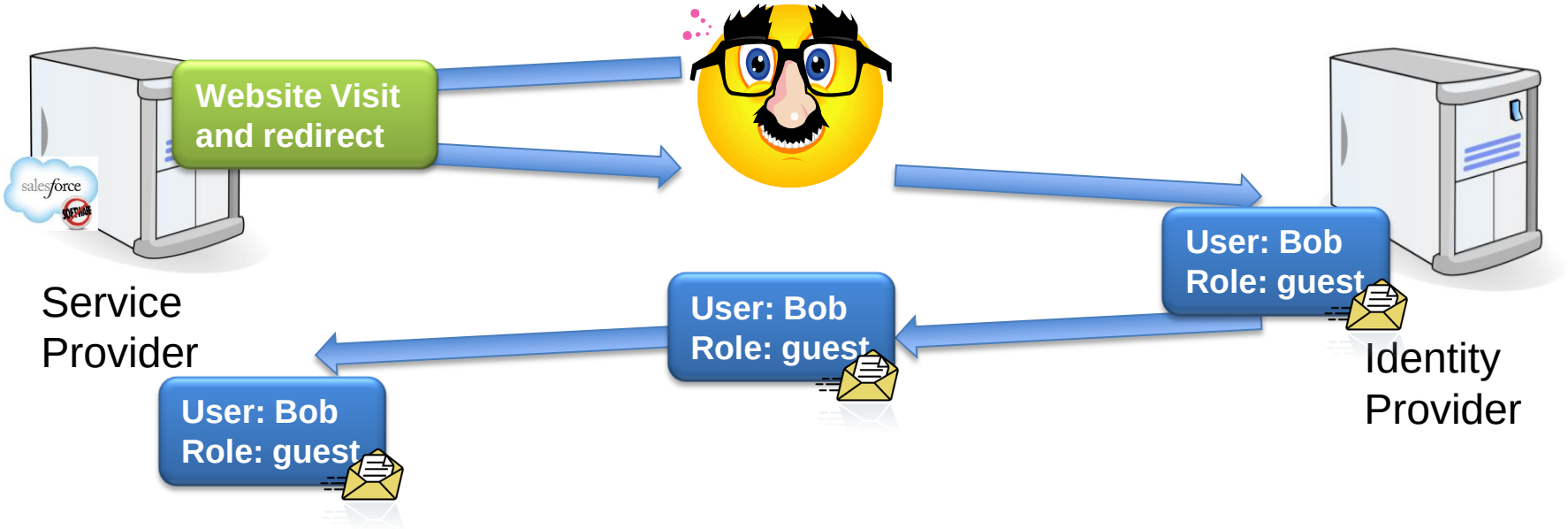
- New standards, new attacks
- Last year:
  - Signature Wrapping attacks on Amazon and Eucalyptus cloud interfaces
  - Attacks on XML Encryption
- Today:
  - Attacks on SAML-based Single Sign-On systems

    Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, Meiko Jensen: **On Breaking SAML: Be Whoever You Want to Be** - In Proceedings of USENIX Security, 2012
  - 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



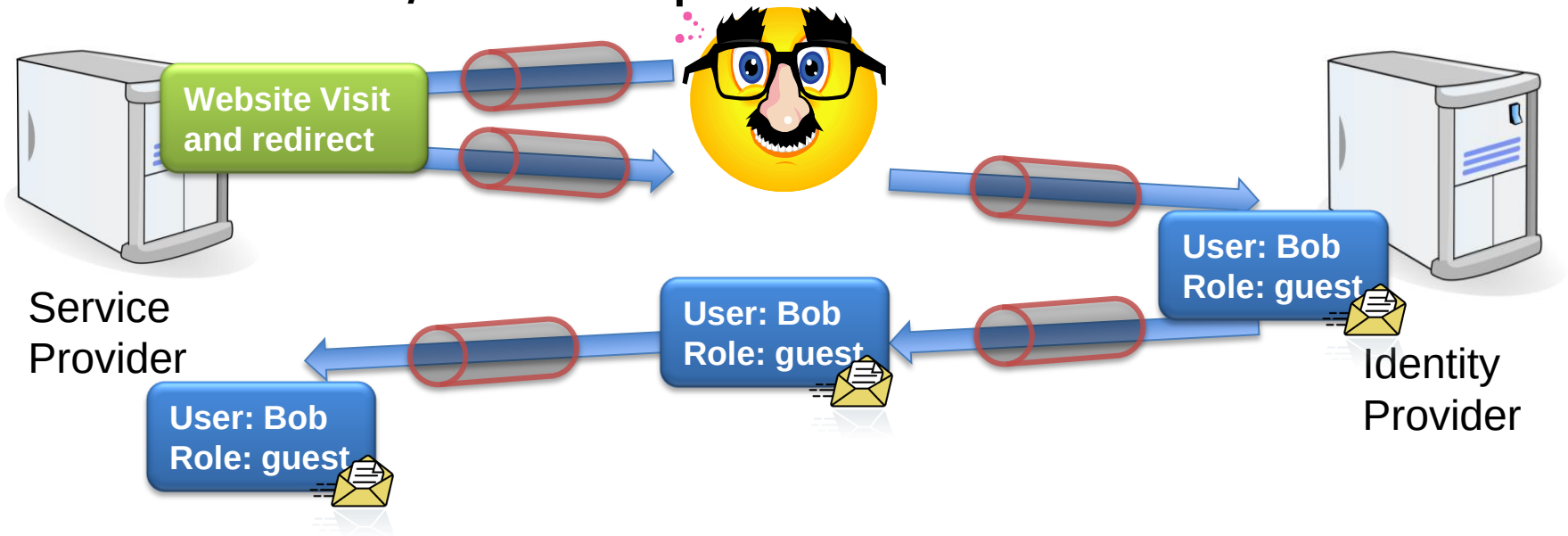
- Advantages: one password for users, no password management for Service Providers

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

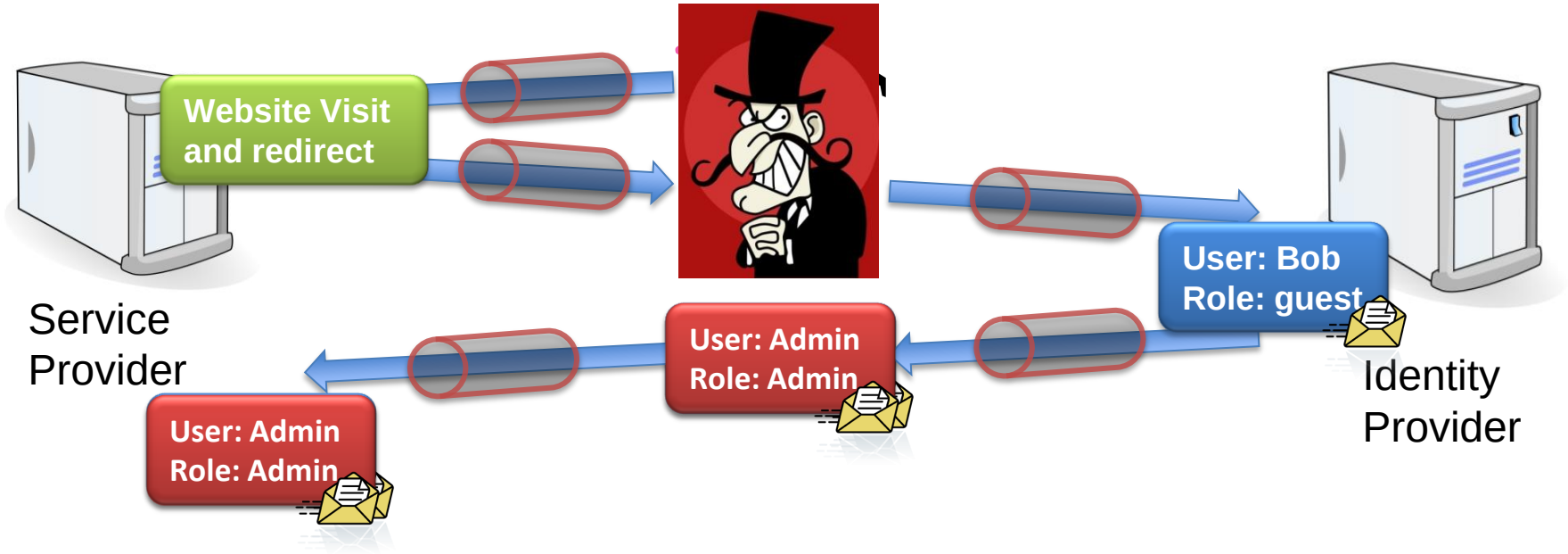
- How do we secure the messages?
- Does SSL / TLS help?



- Messages secured only during transport!

Motivation – Single Sign-On

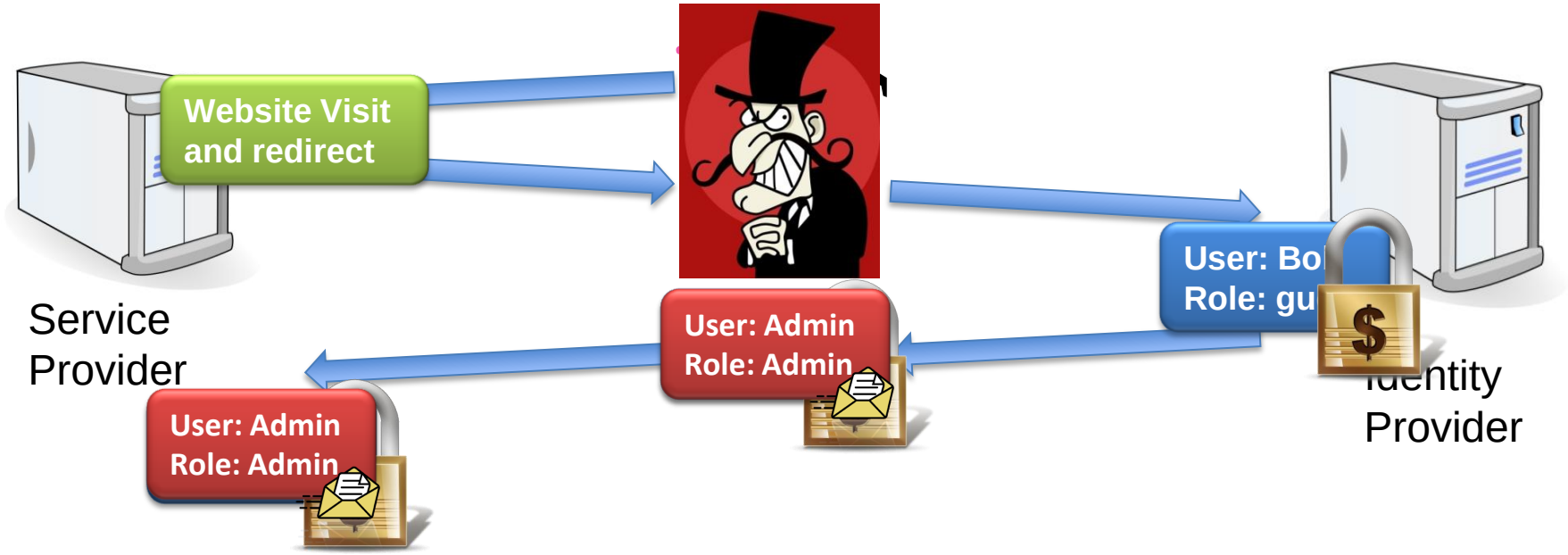
- Does SSL / TLS help?



- Need for message level security!

Motivation – Single Sign-On

- Message level security?

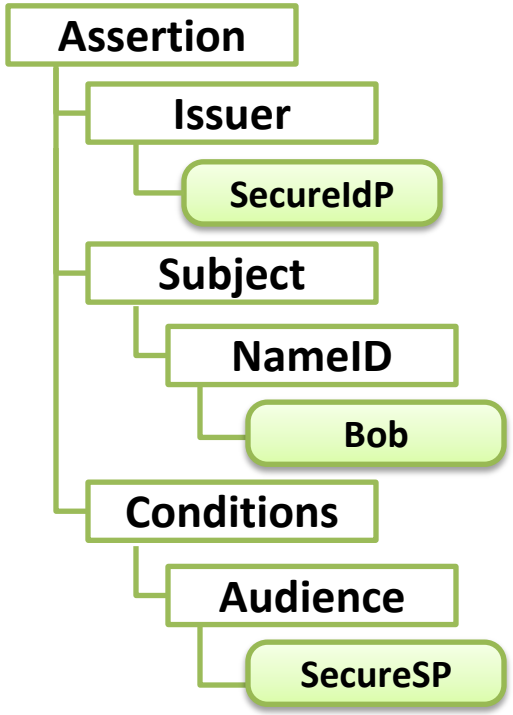


- Realized using XML Signatures
- Are we secure?

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

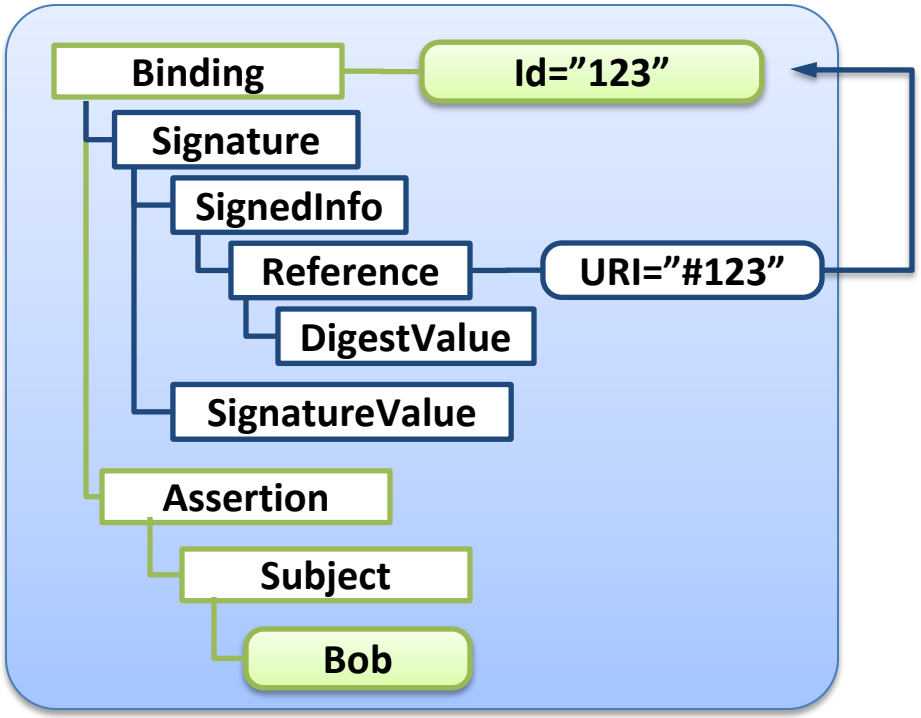
SAML Assertion

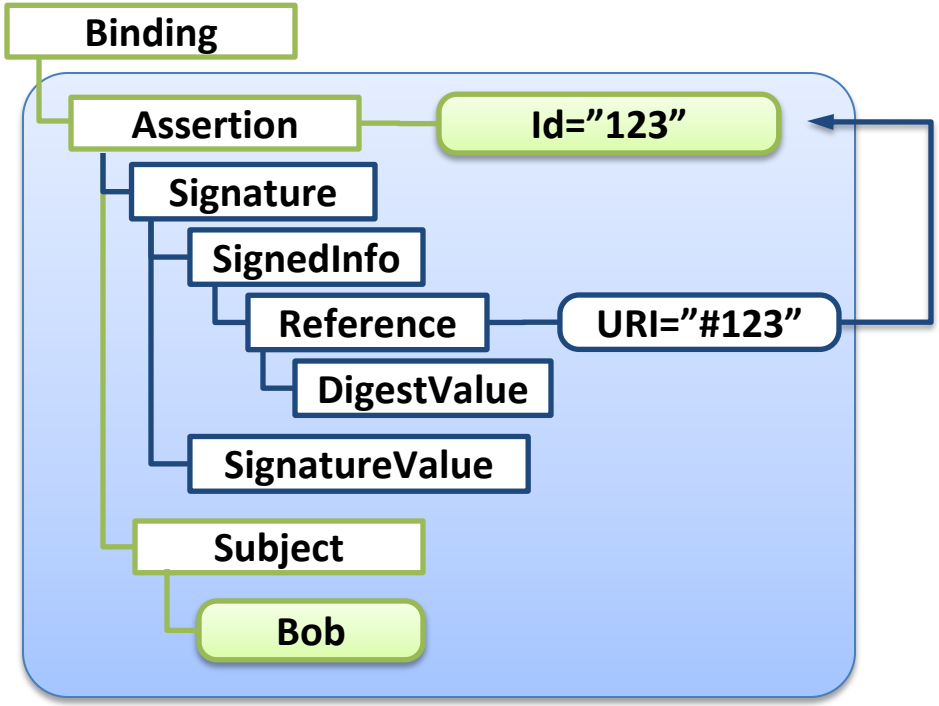
```
<saml:Assertion ID="123">
  <saml:Issuer>www.SecureIdP.com</saml:Issuer>
  <saml:Subject>
    <saml:NameID>Bob@SecureIdP.com</saml:NameID>
  </saml:Subject>
  <saml:Conditions
     NotBefore="2011-08-08T14:42:00Z"
     NotOnOrAfter="2011-08-08T14:47:00Z">
    <saml:AudienceRestriction>
      <saml:Audience>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



Securing SAML with XML Signature

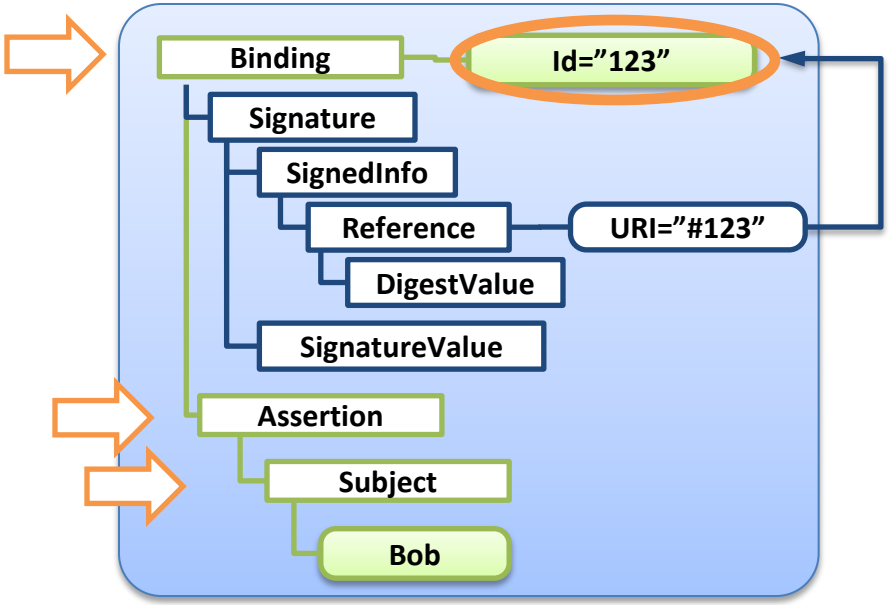
- Two typical usages

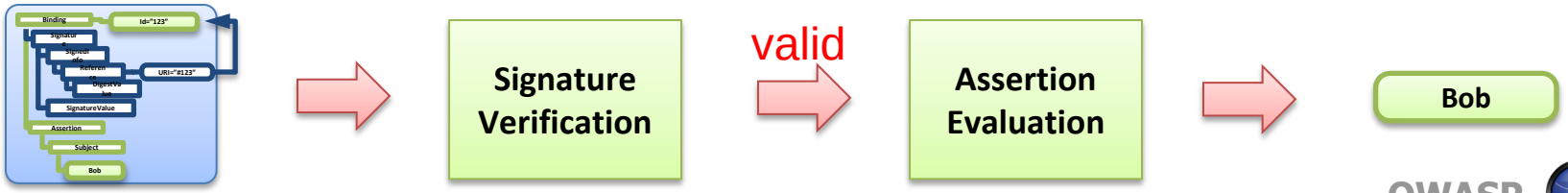




Securing SAML with XML Signature

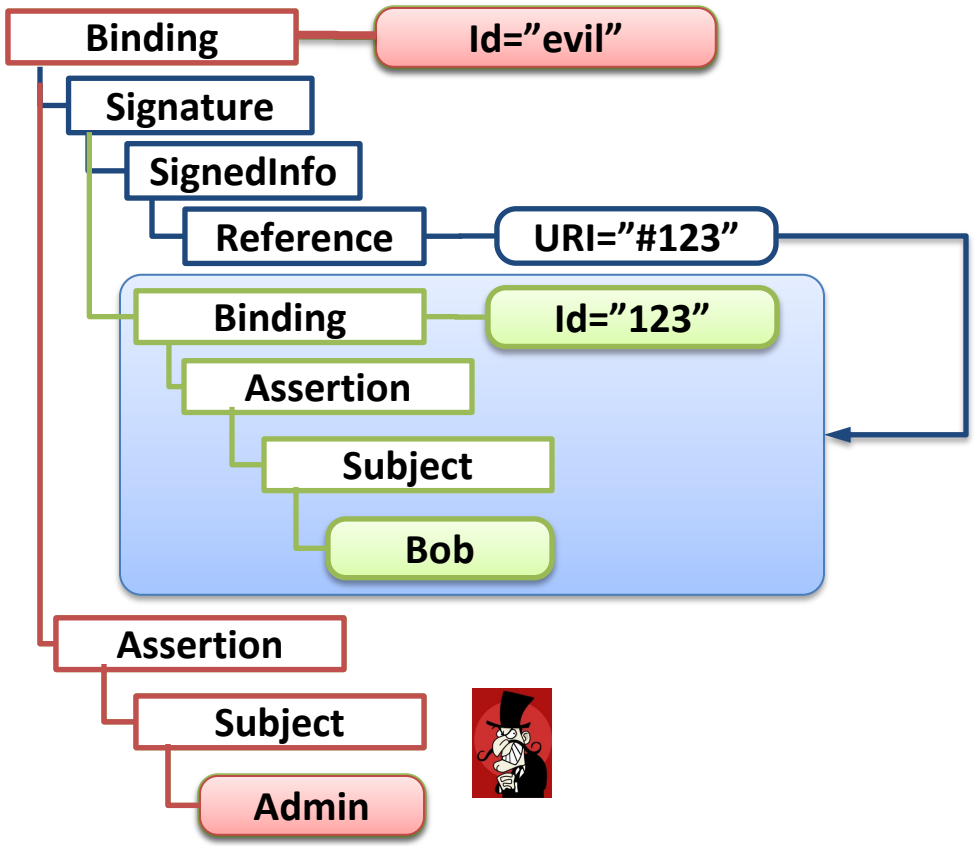
- Naive (typical) processing:
  1. Signature validation: **Id-based**
  2. Assertion evaluation: **/Binding/Assertion/Subject**





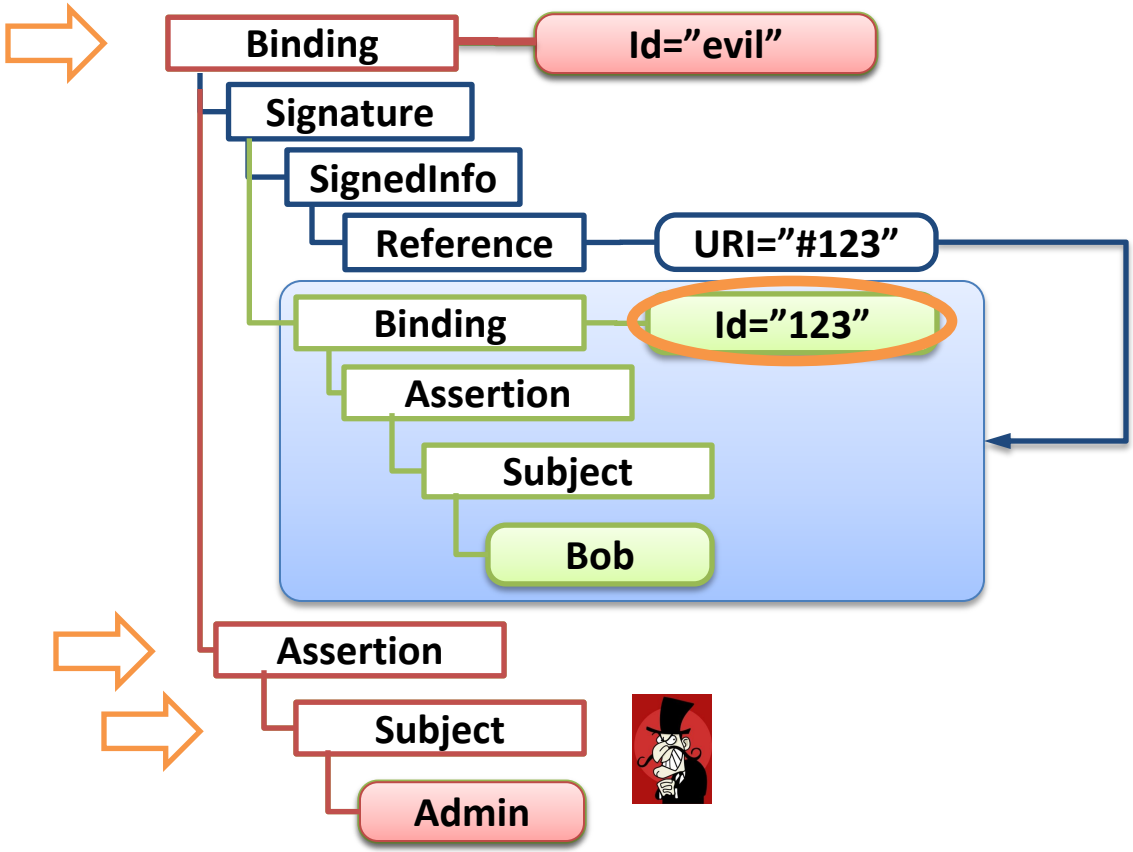
1. On Breaking SAML
    1. Motivation – Single Sign-On
    2. Securing SAML with XML Signature
    3. XML Signature Wrapping Attacks
    4. Practical Evaluation
    5. Countermeasures
2. WS-Attacker
    1. Penetration Test Library
    2. Concept: WS-Attacker
    3. Practical Evaluation Example
3. Conclusion

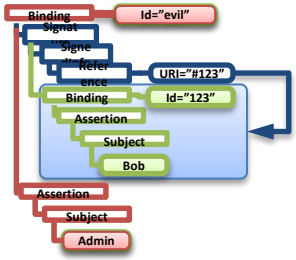
XML Signature Wrapping Attack on SAML

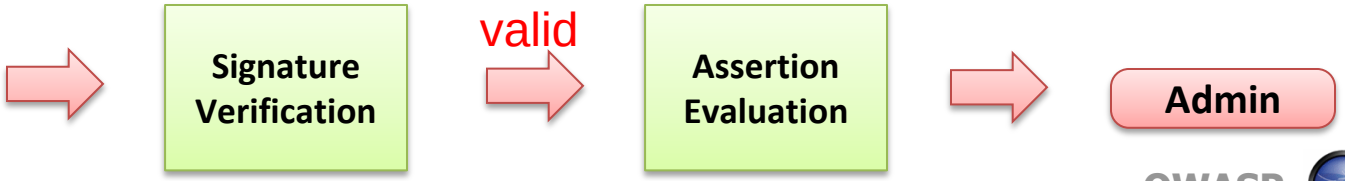


1. Place the original Assertion including its Binding element into another element
2. Change the Id of the original element
3. The Reference now points to the original element: signature is valid
4. Insert a new Assertion

XML Signature Wrapping Attack on SAML

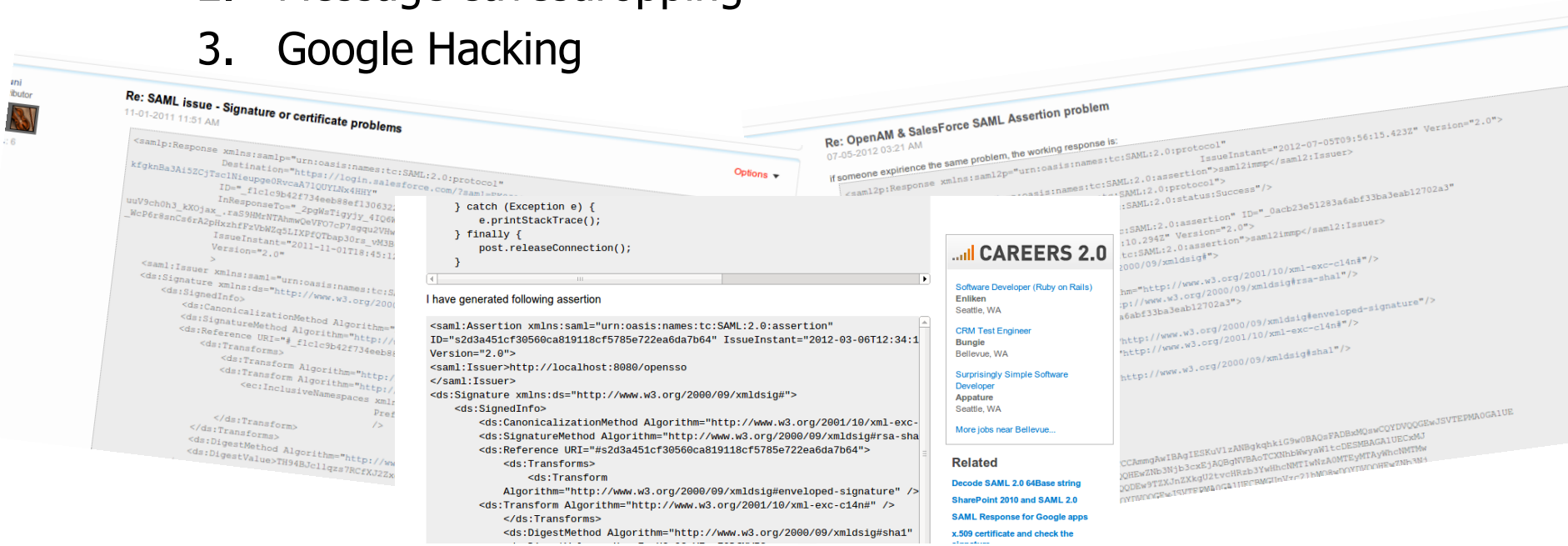




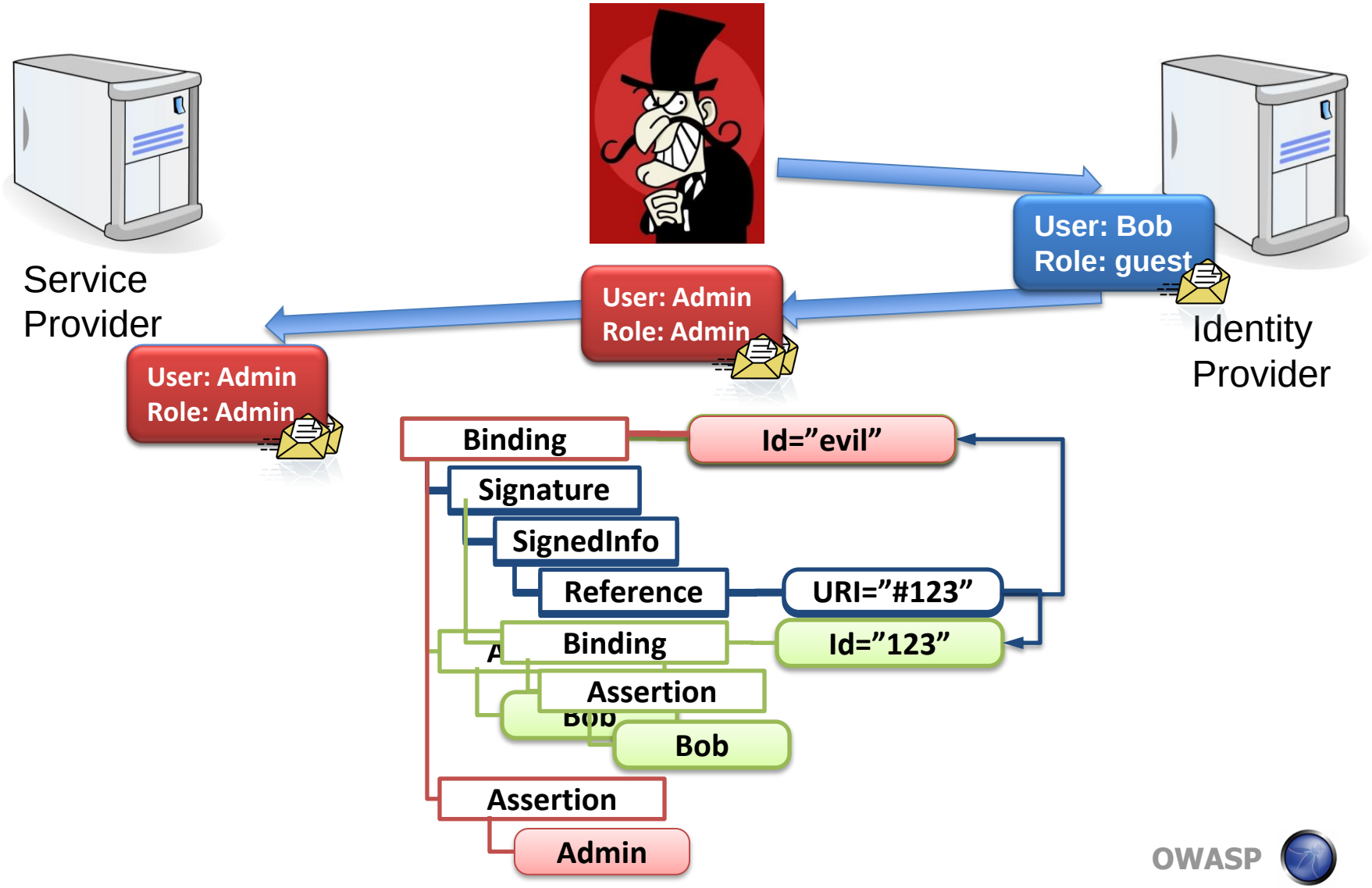


XML Signature Wrapping Attack on SAML – Threat model

- 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

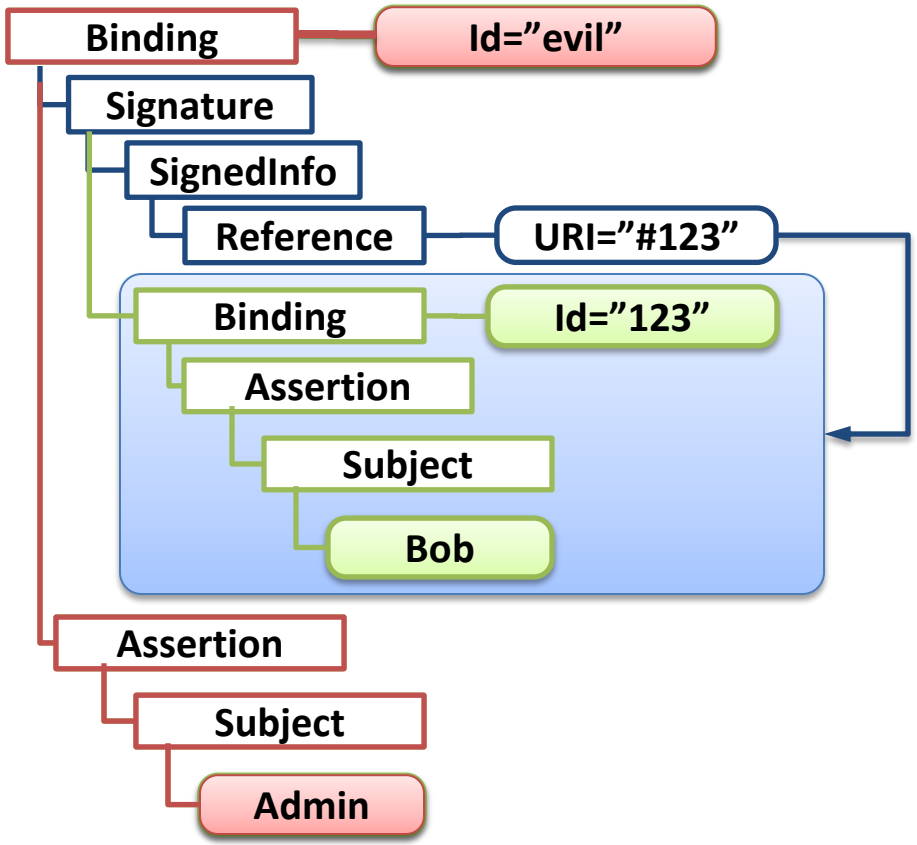
- How about them?

Framework / Provider	Binding	Application
Apache Axis 2	SOAP	WSO2 Web Services
Guanxi	HTTP	Sakai Project (www.sakaiproject.org)
Higgins 1.x	HTTP	Identity project
IBM Datapower XS40	SOAP	Enterprise XML Security Gateway
JOSSO	HTTP	Motorola, NEC, Redhat
WIF	HTTP	Microsoft Sharepoint 2010
OIOSAML	HTTP	Danish eGovernment (e.g. www.virk.dk)
OpenAM	HTTP	Enterprise-Class Open Source SSO
OneLogin	HTTP	Joomla, Wordpress, SugarCRM, Drupal
OpenAthens	HTTP	UK Federation (www.eduserv.org.uk)
OpenSAML	HTTP	Shibboleth, SuisseID
Salesforce	HTTP	Cloud Computing and CRM
SimpleSAMLphp	HTTP	Danish e-ID Federation (www.wayf.dk)
WSO2	HTTP	WSO2 ESB

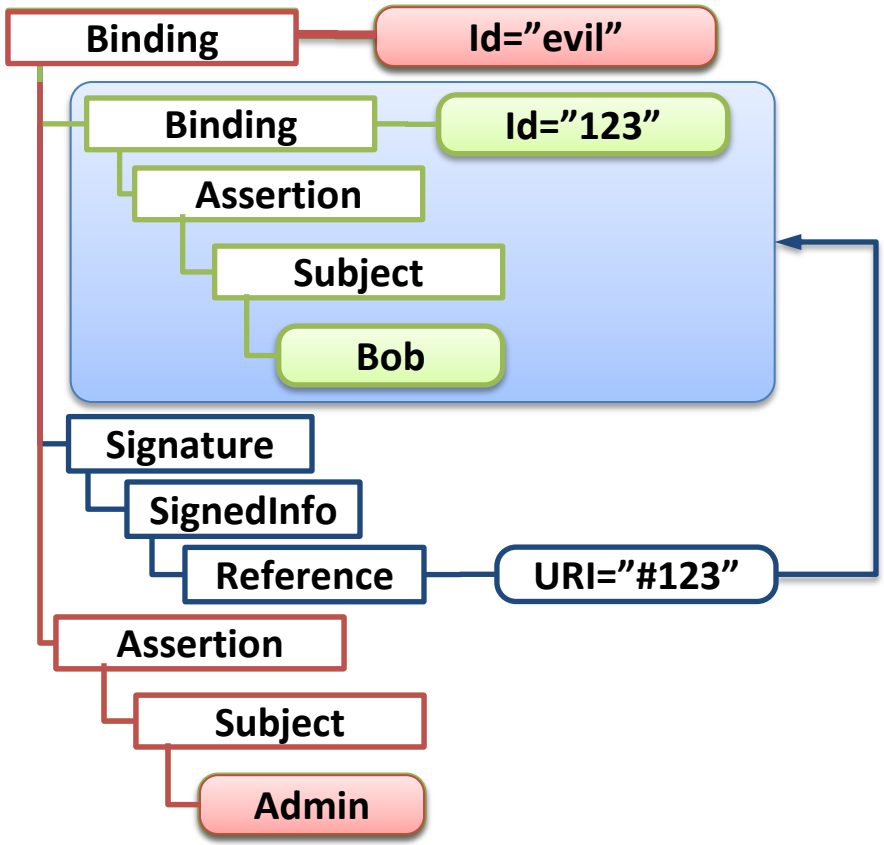
1. On Breaking SAML
    1. Motivation – Single Sign-On
    2. Securing SAML with XML Signature
    3. XML Signature Wrapping Attacks
    4. Practical Evaluation
    5. Countermeasures
2. WS-Attacker
    1. Penetration Test Library
    2. Concept: WS-Attacker
    3. Practical Evaluation Example
3. Conclusion

XML Signature Wrapping Attack on SAML – Results

Guanxi, JOSSO

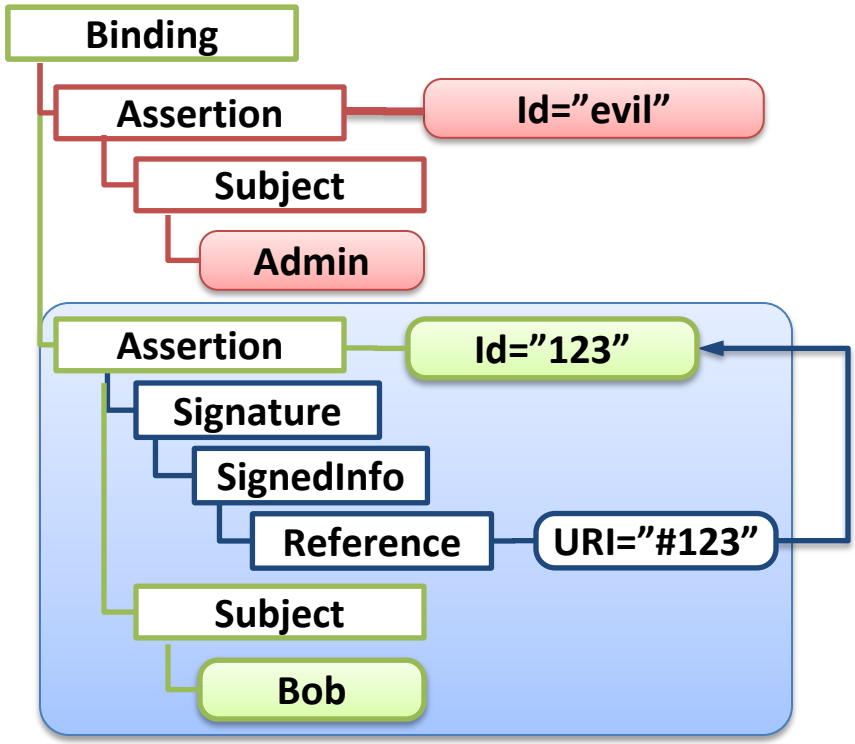


WSO2

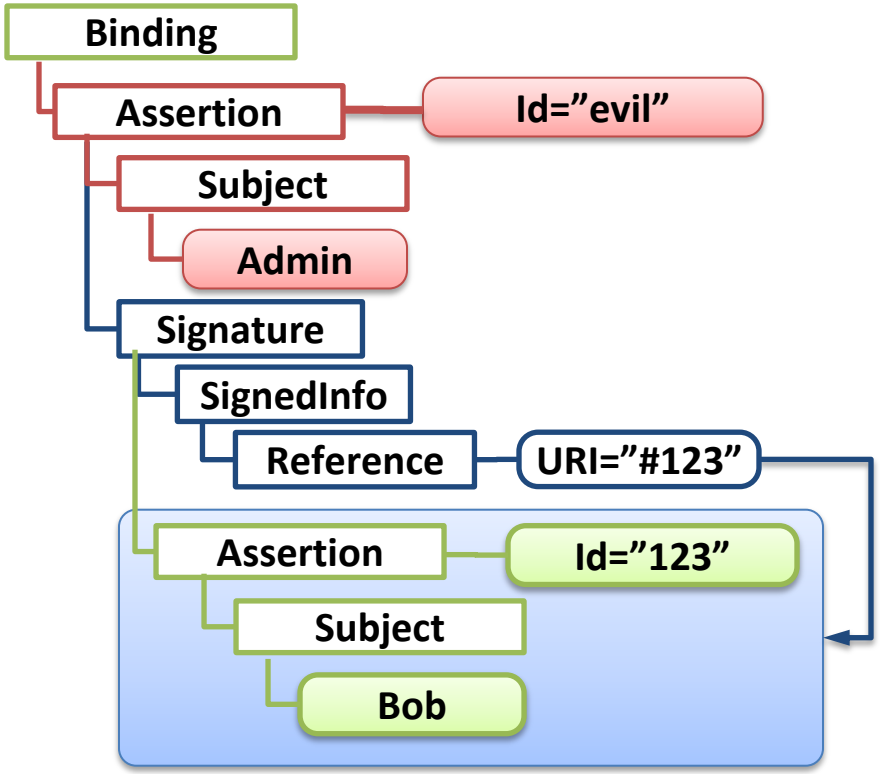


XML Signature Wrapping Attack on SAML – Results

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

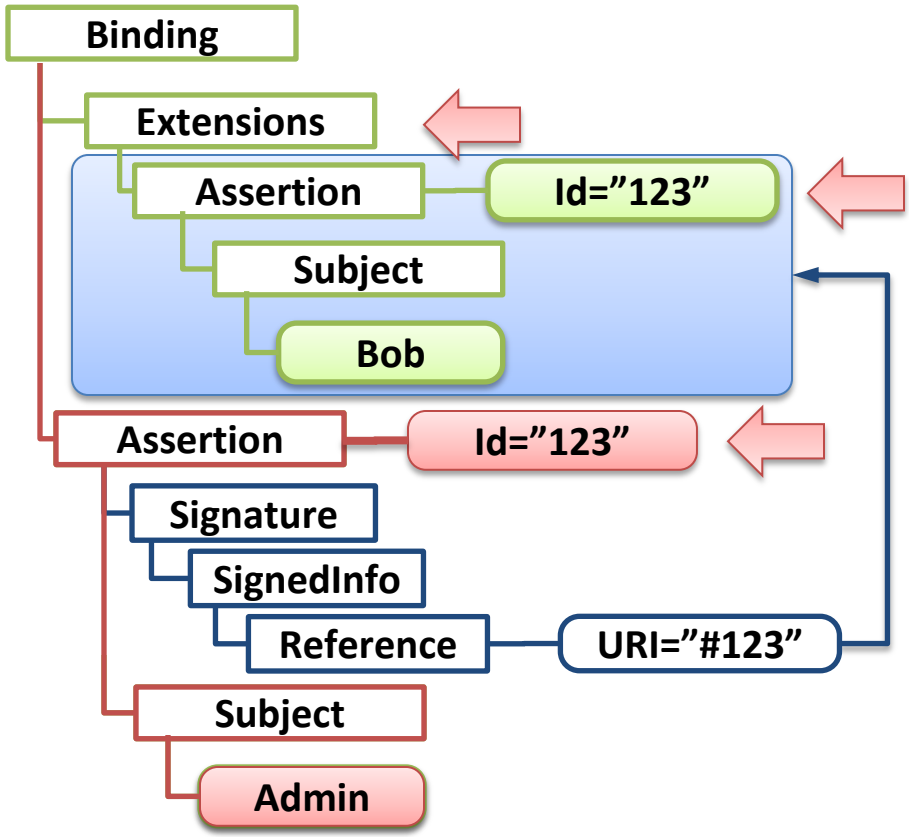
Attack on OpenSAML

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?

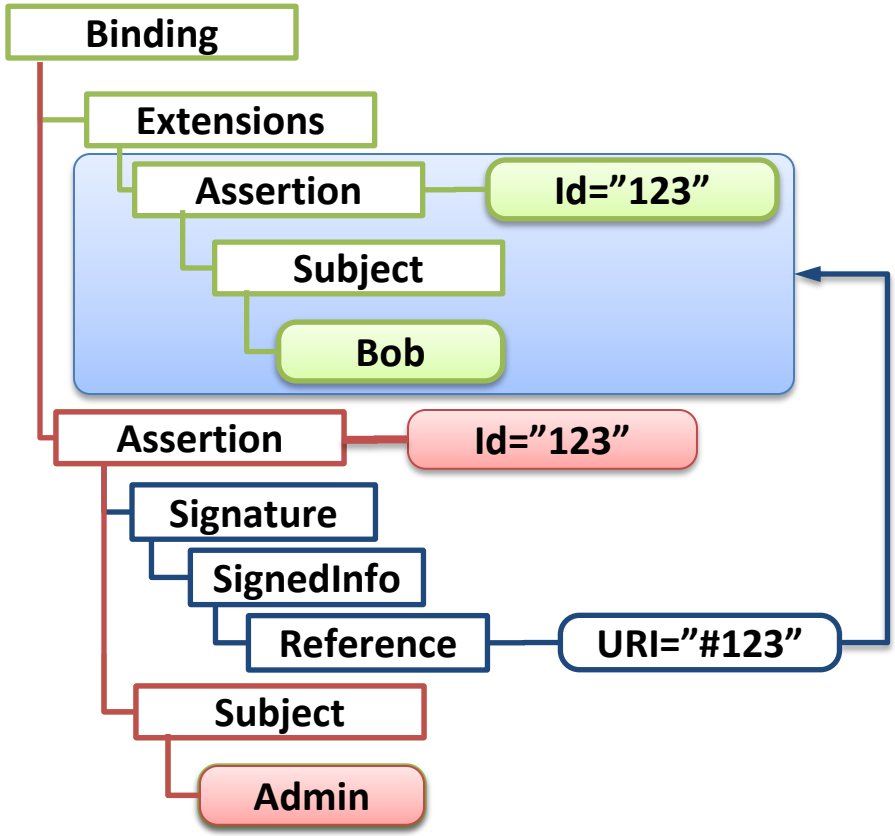
C++ takes the first found element

OpenSAML C++

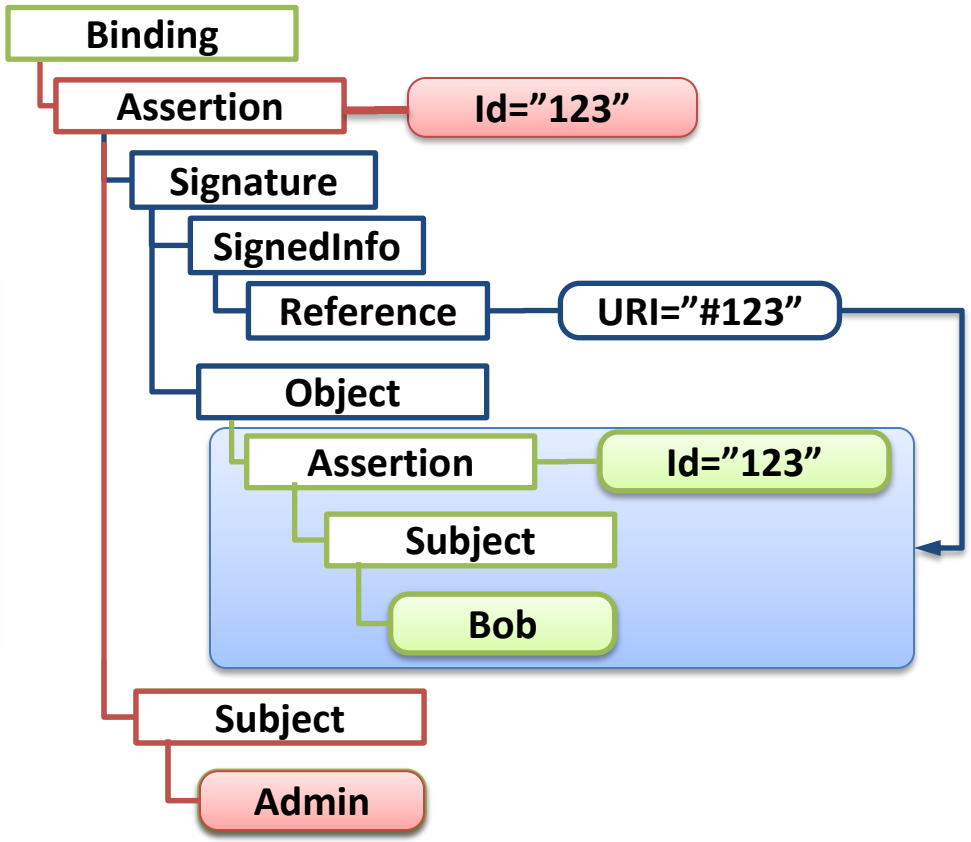


Attack on OpenSAML

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

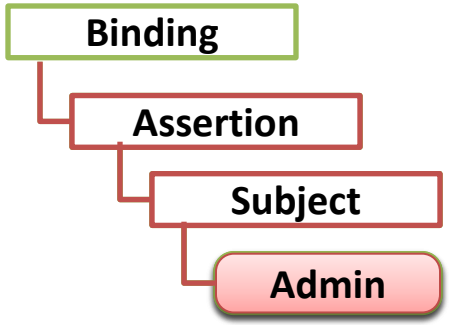


OpenSAML Java references the **last** found element



Beyond Signature Wrapping: Signature Exclusion

- Lame but …
- …Worked against:
  - Apache Axis2
  - JOSSO
  - OpenAthens



SAML Signature Wrapping – Summary

Framework / Provider	Signature Exclusion	Signature Wrapping
Apache Axis 2	X	X
Guanxi		X
Higgins 1.x		X
IBM Datapower XS40		X
JOSSO	X	X
WIF		
OIOSAML		X
OpenAM		X
OneLogin		X
OpenAthens	X	
OpenSAML		X
Salesforce		X
SimpleSAMLphp		
WSO2		X

Enterprise Applications

Danish eGovernment

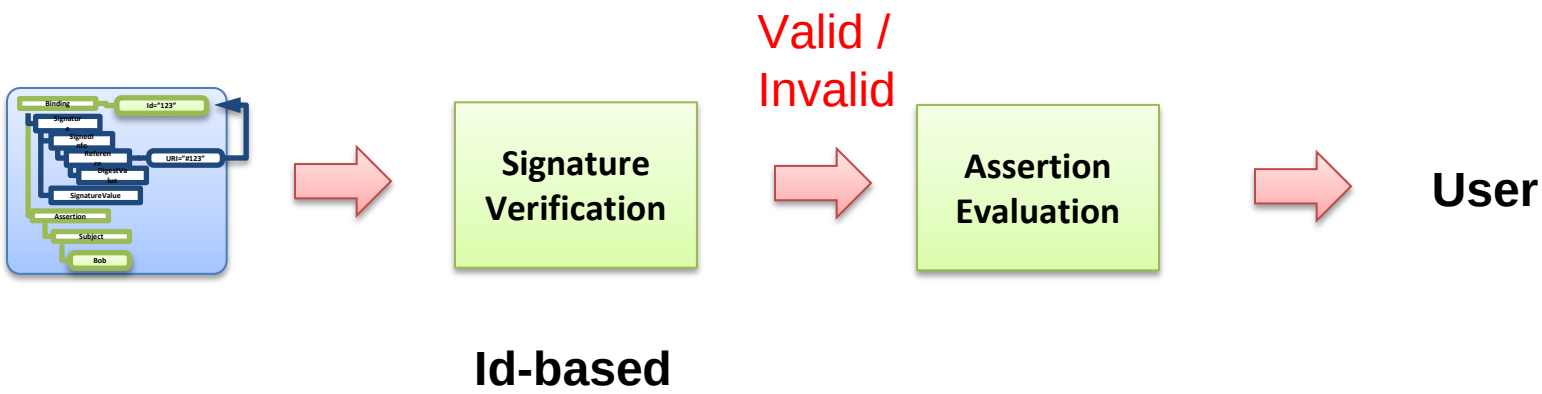
Joomla, Wordpress, SugarCRM, Drupal

Shibboleth, SwissID …

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

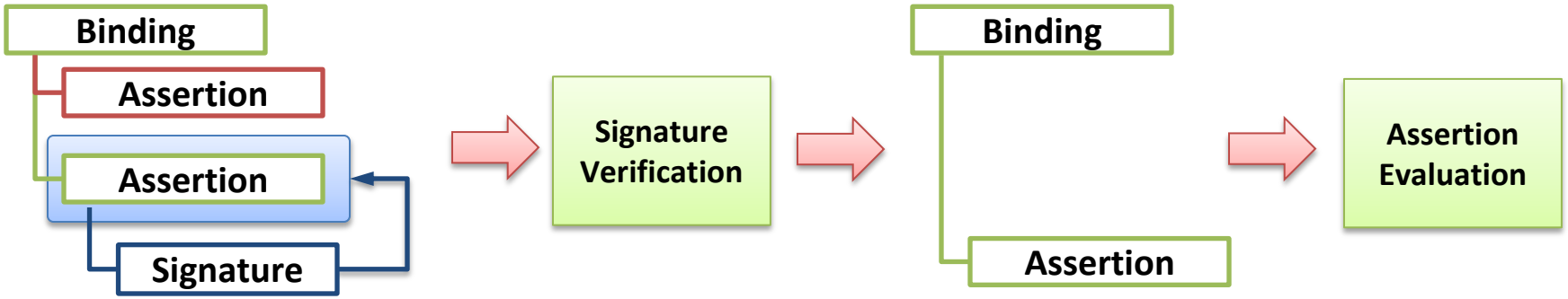


Id-based

/Binding/Assertion/Subject

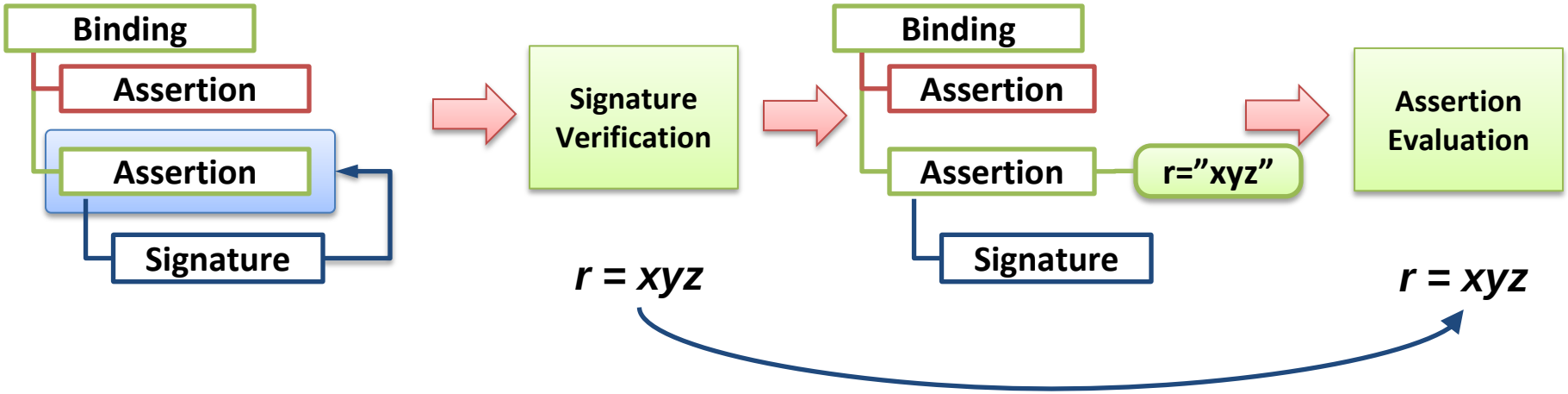
Countermeasure 1: Strict Filtering

- 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

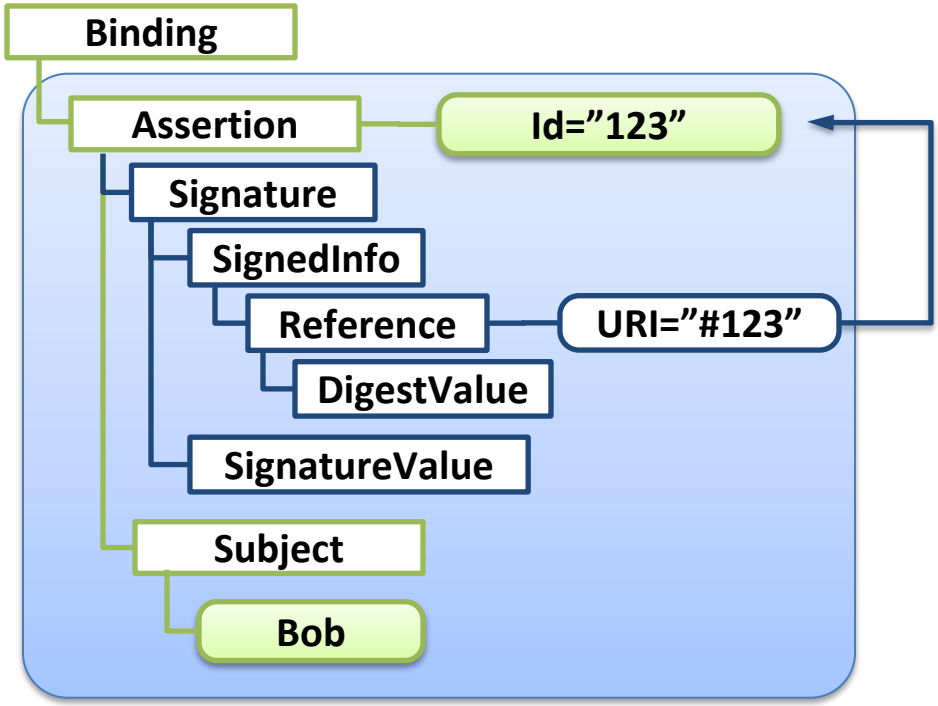


1. On Breaking SAML
    1. Motivation – Single Sign-On
    2. Securing SAML with XML Signature
    3. XML Signature Wrapping Attacks
    4. Practical Evaluation
    5. Countermeasures
2. WS-Attacker
    1. Penetration Test Library
    2. Concept: WS-Attacker
    3. Practical Evaluation Example
3. Conclusion

Penetration Test Library

- Considered all the attack vectors:
    1. Different permutations of signed / processed Assertions

Attack Permutations

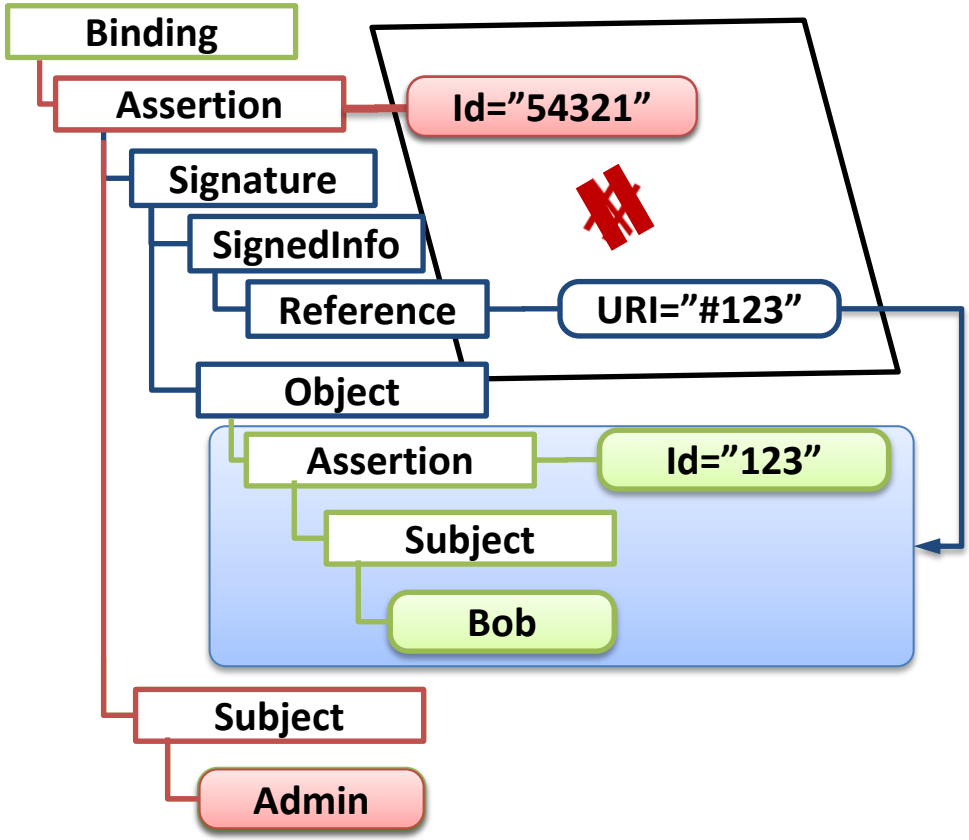


- There are many possibilities
- Dependant of its position
- Dependant of its parent
    - Hard to test manually

Penetration Test Library

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

Id processing



Three possibilities

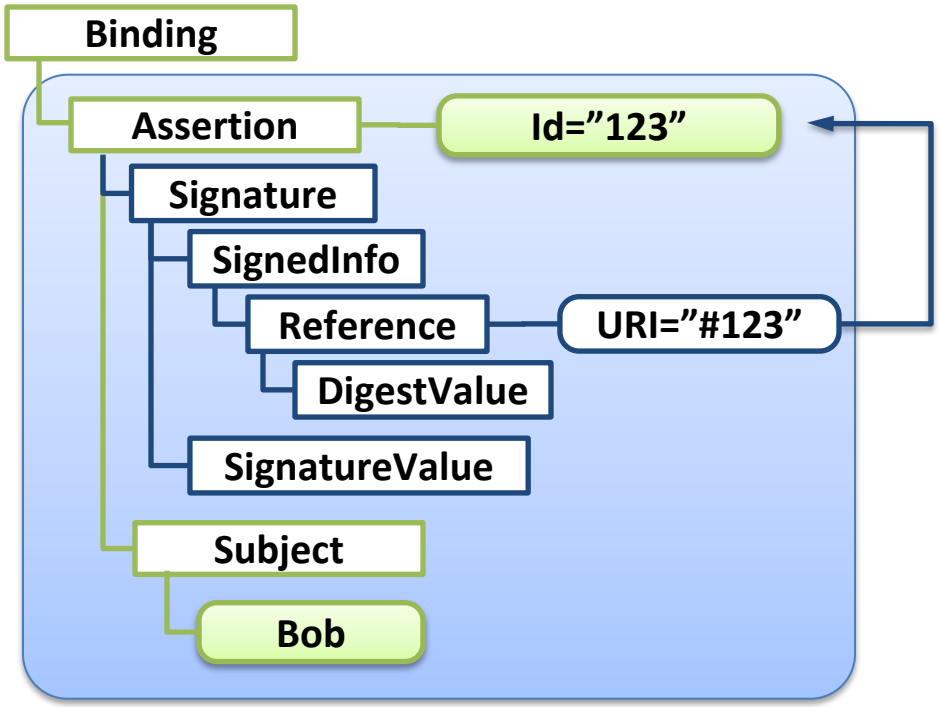
1. Same Id value
2. Different Id value
3. Remove Id value

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

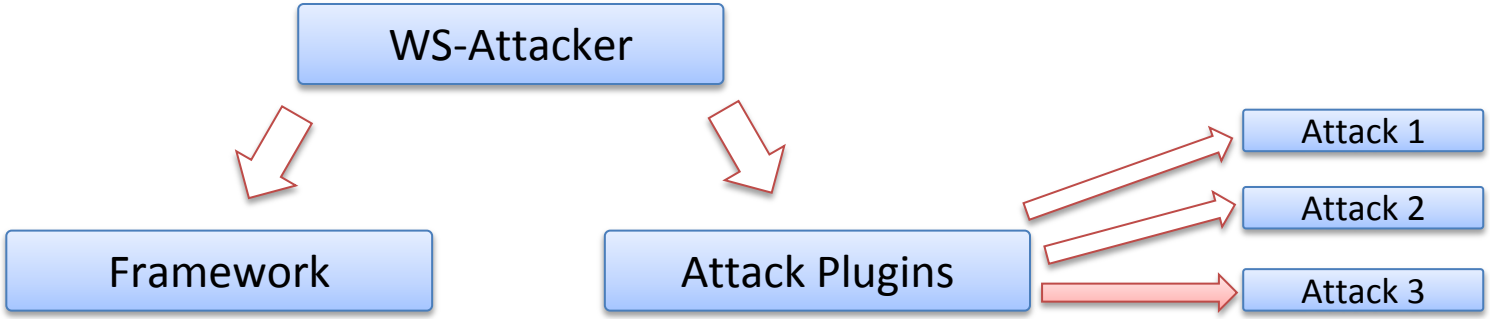


Penetration Test Library

- 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

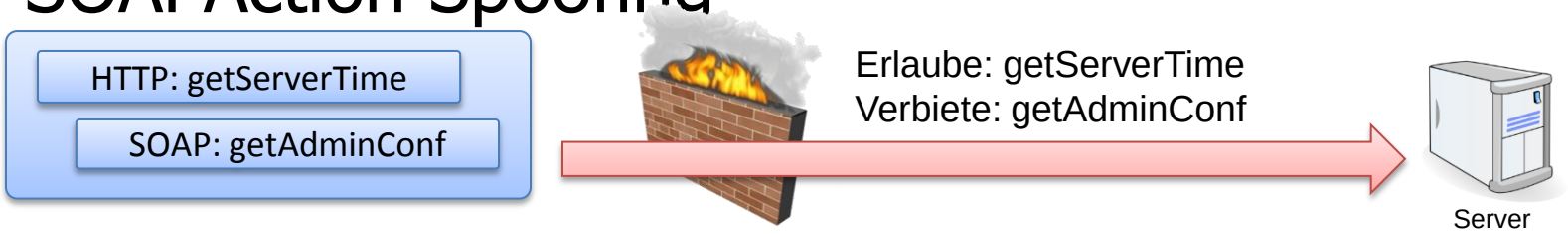
Concept WS-Attacker



- Modular Framework for Web Services Penetration Testing
- Goals:
  - Easy to use
  - Easy to develop attacks

WS-Attacker's Current Attacks

- SOAPAction Spoofing



- WS-Addressing Spoofing

- 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

OWASP

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

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

WSDL

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

Load

Interface

Operation

New

Prefix	Uri

Request Input Table | Request Expert View

Name	Parents	Value

[INFO] Successfuly loaded 10 of 10 plugins

Load WSDL

File

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

WSDL

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

Load

Interface

sample06Soap12Binding

Operation

echo

New

Prefix	Uri
ds	http://www.w3.org/2000/09/xmldsig#
ec	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
samlp	urn:oasis:names:tc:SAML:1.0:protocol

Request Input Table | Request Expert View

Name	Parents	Value
wsu:Created	soapenv:Envelope -> soapenv:Header -> wsse:Security -> wsu:Timestamp	2012-10-11T08:23:02.324Z
wsu:Expires	soapenv:Envelope -> soapenv:Header -> wsse:Security -> wsu:Timestamp	2012-10-11T08:28:02.324Z
xenc:EncryptionMethod	soapenv:Envelope -> soapenv:Header -> wsse:Security -> xenc:EncryptedKey	
wsse:KeyIdentifier	soapenv:Envelope -> soapenv:Header -> wsse:Security -> xenc:EncryptedKey -> ds:KeyInfo -> wsse:SecurityTokenReference	aqePjuZzE1lzwMMtquksvNJsbml=
xenc:CipherValue	soapenv:Envelope -> soapenv:Header -> wsse:Security -> xenc:EncryptedKey -> xenc:CipherData	U6YFjtgb2v9DgulEdegF7VUk0+QQ+Y4SH9W...
Conditions	soapenv:Envelope -> soapenv:Header -> wsse:Security -> Assertion	
NameIdentifier	soapenv:Envelope -> soapenv:Header -> wsse:Security -> Assertion -> AuthenticationStatement -> Subject	CN=Sample Client, OU=Rampart, O=Apach...
ConfirmationMethod	soapenv:Envelope -> soapenv:Header -> wsse:Security -> Assertion -> AuthenticationStatement -> Subject -> SubjectConfirmation	urn:oasis:names:tc:SAML:1.0:cm:holder-of-k...

[INFO] Has payload? false

Send Testrequest

File

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

Request: Endpoint: http://127.0.0.1:8080/axis2/services/sample06.sample06HttpSoap12Endpoint/

```
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>XRu2aDah4Jvke2S5BbbSF53hWTU=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>314Jo7vz7fwzexN8ZZJfBqS8Tn8=</ds:SignatureValue>
<ds:KeyInfo Id="KeyId-A0F2B96F8C7FDCA6F713499437827756">
<wsse:SecurityTokenReference wsu:Id="STRId-A0F2B96F8C7FDCA6F713499437827757" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wss
</ds:KeyInfo>
</ds:Signature></wsse:Security><wsa:To>http://localhost:8080/axis2/services/sample06</wsa:To><wsa:MessageID>urn:uuid:5c648296-aa1d-4399-a1f8-5138155
```

Response:

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Action>http://www.w3.org/2005/08/addressing/soap/fault</wsa:Action>
    <wsa:RelatesTo>urn:uuid:5c648296-aa1d-4399-a1f8-51381554eea6</wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>
    <soapenv:Fault xmlns:axis2ns4="http://www.w3.org/2003/05/soap-envelope">
      <soapenv:Code>
        <soapenv:Value>axis2ns4:Sender</soapenv:Value>
        <soapenv:Subcode>
          <soapenv:Value xmlns:axis2ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">axis2ns5:MessageExpired</soape
```

Start Test Request

[INFO] Has payload? false

Chose Attack

File

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

- Active Plugins (1)
- All Plugins (10)
  - Denial of Service (6)
    - Coercive Parsing (Ready)
    - Hash DOS Attack (Ready)
    - Oversized XML (Ready)
    - SOAP array attack (Ready)
    - Test DOS Attack (Ready)
    - Xml Entity Expansion (recursive) (Ready)
  - Security (1)
    - Signature (1)
      - Signature Wrapping (Ready)
  - Spoofing Attacks (2)
    - SOAPAction Spoofing (Ready)
    - WS-Addressing Spoofing (Ready)
  - Test (1)
- Alphabetical Sorted (10)

All | None | Save | Load

Signature Wrapping

Schema? — *Turn on, to not use any XML Schema.*

Used Schema files — Browse...

Set the Schema Files.
Soap11, Soap12, WSA, WSSE, WSU, DS and XPathFilter2 are included by default.

Search? — *SOAP Response must contain a specific String.*

View — View

Display the wrapping messages.

Show — Payload #1

Reference Element

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:wsa="http://
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c1
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1
<ds:Reference URI="#_13c5cf5aaaa9075df9872ed43d69f936">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-sign
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:Inc
</ds:Transforms>
```

Timestamp?

URI="#_13c5cf5aaaa9075df9872ed43d69f936"

Analyzing: //*[@AssertionID='_13c5cf5aaaa9075df9872ed43d69f936']

[INFO] Has payload? false

OWASP

Run WS-Attacker

File

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

Start | Stop | Clean | Save

Critical | Important | Info | Trace

Name	Status	Rating	Vulnerable?
Signature Wrapping	Finished	100%	true

Time	Level	Source	Content
16:50:29.821	Important	Signature Wrapping	3 signed Elements: --> 3 by ID --> 0 by XPath `--> 0 by FastXPath `--> 0 by prefix free FastXPath (best)
16:50:46.997	Critical	Signature Wrapping	Server Accepted the Request with Possibility 897
			Attack-Vector: Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsatk:wrapper[1]/Assertion[1] Payload element Assertion gets a new attribute value AssertionID='sK7/mdh+hsf8GvPkCeahHPAOT84S9E0y7' Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsatk:wrapper[1]/wsu:Timestamp[1] Payload element wsu:Timestamp gets a new attribute value wsu:Id='ufaw466D/J6' Request: <soapenv:Envelope><soapenv:Header><wsatk:wrapper><wsu:Timestamp wsu:Id="Timestamp-1"><wsu:Created>2012-10-11T08:23:02.324Z</wsu:Created><wsu:Expires>2012-10-11T08:28:02.324Z</wsu:Expires></wsu:Timestamp></wsatk:wrapper><wsatk:wrapper><Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID="_13c5cf5aaaa9075df9872ed43d69f936" IssueInstant="2012-10-11T08:23:02.704Z" Issuer="SAMPLE_STS" MajorVersion="1" MinorVersion="1"><Conditions NotBefore="2012-10-11T08:23:02.703Z" NotOnOrAfter="2012-10-11T08:28:02.703Z"/><AuthenticationStatement AuthenticationInstant="2012-10-11T08:23:02.703Z"

[INFO] Plugin finished: 100/100

OWASP

Deeper Analysis

Signature Wrapping Analysis

897

Successful Attack

Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsatk:wrapper[1]/Assertion[1]
Payload element Assertion gets a new attribute value AssertionID='VqW7/yzHf71RGyrjVqeyy2b4+SoB5mI3g'
Wrapper @ /soapenv:Envelope[1]/soapenv:Header[1]/wsatk:wrapper[1]/wsu:Timestamp[1]
Payload element wsu:Timestamp gets a new attribute value wsu:Id='wnaDF3eW6Uf'

```
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo></ds:Signature></Assertion><ds:Signature xmlns:ds
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmlds
<ds:Reference URI="#Id-691789110">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
<ds:DigestValue>jswvNhYLdb/j10XhshcA0LUYO8E=</ds:DigestValu
</ds:Reference>
<ds:Reference URI="#Timestamp-1">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
<ds:DigestValue>XRu2aDah4Jvke2S5BbbSF53hWTU=</ds:DigestVa
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>314Jo7vz7fwzexN8ZZJfBqS8Tn8=</ds:Signatur
<ds:KeyInfo Id="KeyId-A0F2B96F8C7FDCA6F713499437827756">
<wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open
</ds:KeyInfo>
</ds:Signature></wsse:Security><wsa:To>http://localhost:8080/a
```

```
</ds:Reference>
<ds:Reference URI="#SigConf-2">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig
<ds:DigestValue>eAQmJVNBa5aNKkICLjrMKday3r8=</ds:DigestVa
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>QD44vrDcOULHxjGOHoCCqHBXYek=</ds:Signatu
<ds:KeyInfo Id="KeyId-63E8BE27F0F96DB49913515258469901">
<wsse:SecurityTokenReference wsu:Id="STRId-63E8BE27F0F96DB4
<wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/w
</wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
<wsa:Action>urn:echoResponse</wsa:Action>
<wsa:RelatesTo>urn:uuid:5c648296-aa1d-4399-a1f8-51381554eea6</w
</soapenv:Header>
<soapenv:Body wsu:Id="Id-1010607471" xmlns:wsu="http://docs.oasis-ope
<ns:echoResponse xmlns:ns="http://sample06.policy.samples.rampart.a
<ns:return>Hello world1</ns:return>
</ns:echoResponse>
</soapenv:Body>
</soapenv:Envelope>
```

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/