XML Wrapping Attack Mitigation using Positional Token

Jitendra Kumar, Balaji Rajendran, Bindhumadhava BS, Dr. N Sarat Chandra Babu
Centre for Development of Advanced Computing, Bangalore
{jitendra,balaji,bindhu,sarat}@cdac.in
XML Digital Signature

- XML Signatures are a kind of digital signatures designed for catering the need of XML message exchange or XML transactions.
- XML Signature has been designed in such a way that it can sign a specific portion of a XML document rather than signing the whole document.
- XML Signature standard defined by W3C and IETF
Schematic of XML Signature

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#WithComments"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference URI=""/>
    <Transforms>
      <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>....................</DigestValue>
  </Reference>
  <SignatureValue>
    ....................
  </SignatureValue>
</SignedInfo>
```

Basic structure of an XMLDSIG

- Signed Info
  - Metadata describing the content being signed.
- Signature Value
  - Signature of the digest of the Signed Info metadata
- Key Info
  - Metadata about or the actual key used.
XML Signature generation algorithm

- Find the nodes within XML documents designated for signing.
- Digest of each node to be signed is calculated.
- For each node getting signed, a <Reference> element is added inside <SignedInfo> node.
- Before being subjected to signature generation, XML node is canonicalized using an algorithm mentioned by <Transform> node inside <Reference> node.
- Calculate digest of specified node and resultant digest value is placed inside <DigestValue> node.
- After, having calculated digest for each designated node, <SignedInfo> node, now contains a set of <Reference> elements corresponding to each node whose digest was created in previous step.
- Sign <SignedInfo> node by calculating the digest of the <SignedInfo> element and encrypting the digest and enclosing this encrypted content inside <SignatureValue> node.
XML Signature verification algorithm

• Recalculate the digest of the `<SignedInfo>` element (the algorithm to calculate digest is specified in `<SignatureMethod>` element).

• Decrypt the value inside `<SignatureValue>` using public key and match this decrypted content with the calculated digest of `<SignedInfo>`.

• If the previous step is successful, then, recalculate the digests of all references inside `<SignedInfo>` node and match the calculated digest of referenced elements with the respective digests contained inside the `<DigestValue>` inside `<SignedInfo>` node, if a perfect match happens for every references, then it results into a successful signature verification.
Web Service

- Web Service is a remote service accessible over open protocols
  - Most prominent protocol is SOAP and standards used such as WSDL and UDDI.
  - SOAP is a XML based protocol.
- As most web services performs important task involving valuable data transaction.
  - Security related issues arise.
  - Securing SOAP messages using digital signature for solving issues related to authentication, data integrity and non repudiation.
XML Wrapping Attacks

• XML signature standard defined by IETF/W3C references or identifies signed elements by their unique identities specified by “id” attribute values in the given XML document.

• Hence, signed XML elements can be shifted from one location to another location in a XML document, and still, it does not have any effect on its ability to verify its signature.

• This flexibility paves the way for an attacker to tweak original XML message without getting noticed by the receiver.
Reference

- “XML Signature Element Wrapping Attacks and Countermeasures”
  - Michael McIntosh & Paula Austel
  - IBM Research, Hawthorne, NY
  - Workshop On Secure Web Services
  - Proceedings of the 2005 Workshop on Secure Web Services
  - ACM Press

https://dl.acm.org/citation.cfm?id=1103026&jmp=cit&coll=ACM&dl=ACM#CIT
Actual SOAP request

SOAP Envelope

SOAP Header

<Header>
  <Security>
    <Signature>
      <Reference URI="#Body">
      </Reference>
    </Signature>
  </Security>
</Header>

SOAP Body

<Body id="Body">
  <ShoppingCart>
    <ItemName>Potato</ItemName>
    <ItemQty>2kg</ItemQty>
  </ShoppingCart>
</Body>
Forged SOAP request

```
<SOAP Envelope>
  <Header>
    <Security/>
    <Signature/>
    <Reference URI="#Body"/>
  </Reference>
  <Signature/>
  <Reference/>
  <DummyTag mustUnderstand="0">
    <Body id="Body">
      <ShoppingCart>
        <ItemName>Potato</ItemName>
        <ItemQty>2kg</ItemQty>
      </ShoppingCart>
      <Body>
    </Body>
  </DummyTag>
</Header>

<SOAP Body>
  <Body id="Forged">
    <ShoppingCart>
      <ItemName>Tomato</ItemName>
      <ItemQty>5kg</ItemQty>
    </ShoppingCart>
    <Body>
  </Body>
</SOAP Body>
```
Actual SOAP request

```xml
<SOAP Envelope>
  <SOAP Header>
    <Header>
      <Security>
        <Signature>
          <Reference URI="#id1"/>
          <Reference URI="#id2"/>
        </Signature>
      </Security>
    </Header>
  </SOAP Header>
  <SOAP Body>
    <ShoppingCart>
      <ItemName>Potato</ItemName>
      <ItemQty Id="id1">2kg</ItemQty>
      <ItemName>Tomato</ItemName>
      <ItemQty Id="id2">5kg</ItemQty>
    </ShoppingCart>
  </SOAP Body>
</SOAP Envelope>
```
Elements jumbled wrapping attack

```
<SOAP Envelope>
  <SOAP Header>
    <Header/>
    <Security/>
    <Signature>...
    <SignedInfo>
      <Reference URI="#id1">...</Reference>
      <Reference URI="#id2">...</Reference>
    </SignedInfo>
    <SignatureValue>...</SignatureValue>
    <KeyInfo>...</KeyInfo>
    <Signature/>
    <Security/>
    <Header>
  </SOAP Header>
  <SOAP Body>
    <Body>
      <ShoppingCart>
        <ItemName>Potato</ItemName>
        <ItemQty Id="#id2">5kg</ItemQty>
        <ItemName>Tomato</ItemName>
        <ItemQty Id="#id1">2kg</ItemQty>
      </ShoppingCart>
    </Body>
  </SOAP Body>
</SOAP Envelope>
```
Proposed XML signature generation algorithm.

For each element subjected to be signed represented by its “Id” attribute values{
ABSXPath= “Absolute XPath” of element to be signed as identified with its “id” attribute values
ProtectTree=SOAP request node identified by ABSXPath
MixedElement=AppendSyntacticToken(ProtectTree,  ABSXPath)
/*append a Positional Token as an attribute, “PosToken= ABSXPath” to the ProtectTree */
H=Hash(MixedElement)
Add ABSXPath to <Reference> node as “URI” attribute value
Enclose H to <DigestValue> node inside the <Reference> node, as defined in XML Signature standard [9].
}
SignedInfoHash=calculate hash of <SignedInfo> element
/* calculate the digest of the <SignedInfo> element */
SignedSOAP=Encrypt(SignedInfoHash , PrivateKey)
Proposed XML Signature Verification algorithm

SignInfoDigest=Calculate digest of the <SignedInfo> element
SignatureValueContent= content inside <SignatureValue> node
Flag=VerifySignature(Public Key, SignatureValueContent, SignInInfoDigest)

If(Flag){

Ids=All URI’s in <Reference> nodes inside the <SignedInfo> node

For each  Id from Ids{

ABSXpath=Get the content of Id
Subtree=Get the sub tree identified by ABSXpath
MixedElement =AppendSyntacticTokenSubTree(Subtree, ABSXpath)

H=Hash(MixedElement) /* generate hash value of signed elements. */

Digest=Get digest value under the  <Reference> node and inside <DigestValue> node, whose “URI” is equal to Id

If(H!=Digest){
return false
}
}
Wrapping attack mitigation, scenario-1

SOAP Envelope

SOAP Header

<Header>
<Security>
<Signature>
<Reference URI="/Envelope/Header/Body[@Id="Body"">

</Reference>
</Signature>
</Security>
<DummyTag mustUnderstand="0">
<Body id="Body">
<ShoppingCart>
<ItemName>Potato</ItemName>
<ItemQty>2kg</ItemQty>
</ShoppingCart>
</Body>
</DummyTag>
</Header>

SOAP Body

<Body id="Forged">
<ShoppingCart>
<ItemName>Tomato</ItemName>
<ItemQty>5kg</ItemQty>
</ShoppingCart>
</Body>
Wrapping attack mitigation, scenario-2

SOAP Envelope

SOAP Header

```xml
<Header>
  <Security>
    <Signature>
      <Reference URI="/Envelope/Header/DummyTag/Body[@id="Body"]/"
      </Reference>
      </Signature>
      </Security>
      </DummyTag mustUnderstand="0">
      <Body id="Body">
      <ShoppingCart>
      </Body>
      </DummyTag>
      </Header>

SOAP Body

```xml

<Body id="Forged">
  <ShoppingCart>
  <ItemName>Potato</ItemName>
  <ItemQty>2kg</ItemQty>
  </ShoppingCart>
  </Body>
```
Mitigation of jumbled elements attack

SOAP Envelope

```xml
<Header>
    <Security>
    <Signature>...
    <SignedInfo>
        <Reference
            URI="/Envelope/Header/Body/ShoppingCart/ItemQty[1][@id="id1"]"...>
    </Reference>
    <Reference
        URI="/Envelope/Header/Body/ShoppingCart/ItemQty[2][@id="id2"]"...>
    </Reference>
    <SignedInfo>
        <SignatureValue>...</SignatureValue>
        <KeyInfo>...</KeyInfo>
    </Signature>
    </SignedInfo>
</Header>

<Body>
    <ShoppingCart>
        <ItemName>Potato</ItemName>
        <ItemQty Id="id1">2kg</ItemQty>
        <ItemName>Tomato</ItemName>
        <ItemQty Id="id2">5kg</ItemQty>
    </ShoppingCart>
</Body>
```
Thank You