IETF
Internet-Draft
Intended status: Best Current Practice
Expires: August 8, 2019

Jitendra Kumar
Balaji Rajendran
Bindhumadhava BS
C-DAC Bangalore
February 04, 2019

Enhanced XML Digital Signature Algorithm to Mitigate Wrapping Attacks
draft-enhanced-xml-digital-signature-algorithm-01

Abstract

XML signature standard [RFC3275]identifies signed elements by their unique identities in the XML document. However this allows shifting of XML elements from one location to another within the same XML document, without affecting the ability to verify the signature. This flexibility paves the way for an attacker to tweak the original XML message without getting noticed by the receiver, leading to XML Signature wrapping or rewriting attacks. This document proposes to use absolute XPath as a "Positional Token" and modifies the existing XML Digital Signature algorithm to overcome this attack.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 8, 2019.

Table of Contents

1. Introduction

McIntosh and Austel have illustrated that a SOAP message with XML Digital Signature (described in wrapping_attack [wrapping_attack]) can be forged without invalidating the signature and they have further illustrated that a SOAP message content, protected by an XML Digital Signature, as specified in WS-Security(refer, WS-Security [WS-Security]) can be forged without invalidating the signature. This attack is possible because the XML Digital Signature refers to a signed element in XML document in a way without giving significance to the position within the XML document. An attacker may inject additional nodes replacing the signed nodes while still preserving the signed nodes inside the document at different levels in the hierarchy of the XML tree, such that it results in successful signature verification thereby resulting in XML Re-Writing or Wrapping attack.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. XML Digital Signature structure

XML Signatures (described in RFC3275 [RFC3275]) are applied to arbitrary digital content (data objects).Data objects are digested, the resulting value is placed in an element (with other information) and that element is then digested and cryptographically signed.XML digital signatures are represented by the Signature element which has the following structure (where "?" denotes zero or one occurrence; "+" denotes one or more occurrences; and "*" denotes zero or more occurrences):

```
<Signature ID?>
 <SignedInfo>
   <CanonicalizationMethod/>
   <SignatureMethod/>
   (<Reference URI? >
     (<Transforms>)?
     <DigestMethod>
     <DigestValue>
   </Reference>)+
 </SignedInfo>
 <SignatureValue>
(<KeyInfo>)?
(<Object ID?>)*
</Signature>
```

Signatures are related to data objects via URIs [URI]. Within an XML document, signatures are related to local data objects via fragment identifiers.

3. Suggested Modified Algorithm

As XML requests are prone to XML Signature wrapping attacks and these vulnerabilities stems from the usage of ID (Identity) to identify the signed XML subtree. There are many solutions proposed to mitigate such attacks but still,such attacks can't be fully eliminated. In this document, we have proposed the addition of XPath as a doping to the XML element being signed to mitigate XML Signature wrapping attacks. We propose to use "Absolute XPath" instead of ID in <Reference> node's "URI" attribute to refer to the signed element. Absolute XPath can be used as "Positional Token", as this token

exactly points to the position of the element being signed. During the signing process, this "Positional Token" gets added as an attribute e.g. PosToken= "Absolute XPath") to the element that is subjected to be signed.This absolute XPath as a "Positional Token" would identify the signed element in XML Signature and addition of this "Positional Token" as an attribute to the element being signed would eliminate the chances of XML Signature Wrapping attacks wherein the calculated digest of the signed element in forged XML document will not match with the respective digest value in <DigestValue> node during signature validation process. We propose a modified XML signature algorithm which suggests usage of absolute XPath as a "Positional Token" and it will be used during signing as well as during signature validation process. The algorithms proposed are as follows:

3.1. Algorithm for XML Signature

1. KS=Load(Keystore.JKS) //Load certificates and keys
2. For each element subjected to be signed(represented by its "id" attribute value) {
3. ABSXpath= "Absolute XPath" of element to be signed as identified with its "Id" attribute value
4. ProtectTree=Node as identified by ABSXpath
5. MixedElement=AppendSyntacticToken(ProtectTree, ABSXpath)
   /*Append a Positional Token as an attribute, "PosToken= ABSXpath" to the ProtectTree */
6. H=Hash(MixedElement)
7. Add ABSXpath to <Reference> node's "URI" attribute value
8. Enclose H to <DigestValue> node inside the <Reference> node,
   as defined in XML Signature standard.
9. }
10. SignedInfoHash=calculate hash of <SignedInfo> element
    /* Calculate the digest of the <SignedInfo> element */
11. SignedXML=Encrypt(SignedInfoHash , KS.PrivateKey)
    /*Signing that digest and enclosing the signature value in a <SignatureValue> element */

3.2. Algorithm for verification of Signature

```
1.  SignInfoDigest=Calculate digest of the <SignedInfo>
element
2.  SignatureValueContent= content inside <SignatureValue>
node
3.  Flag=VerifySignature(Public Key, SignatureValueContent,
SignInInfoDigest)
4.  If(Flag){
5.  Ids=All  URI's in <Reference> nodes inside the
<SignedInfo> node
6.  For each  Id from Ids){
7.  ABSXpath=Get the content of Id
8.  Subtree=Get the sub tree identified by ABSXpath
9.  MixedElement =AppendSyntacticTokenSubTree(Subtree,
ABSXpath)
    /* Append a Positional Token as an attribute,
           "PosToken= ABSXpath" to the Subtree  */
10. H=Hash (MixedElement)/* generate hash value of signed
elements. */
11. Digest=Get digest value under the  <Reference>
    node and inside <DigestValue> node, whose "URI" is
equal to Id
12. If(H!=Digest){
13. return "Signature Validation Failed"
14. }else{
15. return "Signature Validation Successful"
16. }
17. } //For loop
18. else
19. return "Signature Validation Failed"
20. }
```

3.2.1. Verifying SignedInfo Element Digest with Decrypted Digest from SignatureValue element

```
    1.  VerifySignature(PublicKey,
SignatureValueContent, SignInInfoDigest){
    2.  DecryptedDigest=Decrypt SignatureValueContent
with PublicKey
    3.  If(DecryptedDigest!=SignInInfoDigest){
    4.  return False
    5.  }
    6.  else{
    7.  return True
    8.  }
    9.  }
```

4. Simple Example

The <Signature> Lets consider an XML document as an example:

```
<?xml version="1.0"?>
<PatientRecord>
        <Visit date="10pm March 2018">
                <Account id="id1">1234</Account>
                <Name>ABC</Name>
                <Diagnosis>Kidney Function Test</Diagnosis>
        </Visit>
        <Visit date="12pm May 2018">
                <Account id="id2">1235</Account>
                <Name>DEF</Name>
                <Diagnosis>Liver Function Test</Diagnosis>
        </Visit>
</PatientRecord>
```

Figure 1

Existing XML Signature algorithm would produce a <Signature> element for the XML document mentioned in Figure 1, as follows:

```
<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="#id1">
                        <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>
                <Reference URI="#id2">
                        <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>
        </SignedInfo>
        <SignatureValue>
                ..........
        </SignatureValue>
        <KeyInfo>
                <X509Data>
                        <X509Certificate>
                        ............................
                        </X509Certificate>
                </X509Data>
        </KeyInfo>
</Signature>
```

The proposed XML Signature algorithm would produce a <Signature> element for the XML document mentioned in Figure 1, which is described in Figure 2. The "Positional Token" as an attribute e.g.(PosToken= "Absolute XPath") is used according to the proposed algorithm Section 3.1. Now, <DigestValue> elements inside <Signature> element will also contain the trace of "Positional Token", hence the relative position of signed elements in the given XML document:

```
<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="/PatientRecord/Visit[1]/Account[@id='id1']">
                        <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>
                <Reference URI="/PatientRecord/Visit[2]/Account[@id='id2']">
                        <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>
        </SignedInfo>
        <SignatureValue>
                .............
        </SignatureValue>
        <KeyInfo>
                <X509Data>
                        <X509Certificate>
                        ............................
                        </X509Certificate>
                </X509Data>
```

```
        </KeyInfo>
</Signature>
```

Figure 2

5. Algorithm Validation

In this section, we evaluate how the suggested algorithm can mitigate the various scenarios of XML wrapping attacks.

5.1. Mitigation of XML Signature wrapping attacks

XML Signature Wrapping attacks are possible because of the inherent flaw in the signature verification algorithm that identifies the position of signed element using ID. This makes it possible to move the signed element anywhere easily within the document and still, the document would retains its ability to verify its signature.So, in our proposed algorithm, we have suggested the use of absolute XPath in place of ID for identifying the position of signed elements.Absolute XPath has two-fold advantages as it can easily identify the position of the signed element within the XML document and it fixes both the vertical and horizontal axis of the signed element exactly. The absolute XPath expression to identify the signed element will not be same in a forged document. The signature validation will fail at step-8, of algorithm in Section 3.2, as there is no such node, Further, if the attacker modifies the URI attribute and tries to perform XML Signature wrapping attack, the digest of <SignedInfo> will not match and signature validation will fail at step-4 of the algorithm in Section 3.2.

5.2. Mitigation of XML elements jumbling type of wrapping attacks

This type of XML Signature wrapping attacks are possible as the attacker jumbles the position of signed elements within the document exploiting the existing XML Signing algorithm that takes ID into consideration for referencing the elements being signed.The proposed algorithm suggests using "Absolute XPath" for referencing the signed elements as well as doping the elements subjected to be signed with it. Hence, the digest of the signed element inside <DigestValue> node has a trace of the position of element; refer step-6 of algorithm in Section 3.1. Hence, any changes in the position of signed elements by the attackers will invalidate the signature; refer step-12 of algorithm in Section 3.2,as the calculated digest during signature validation will not match with the digest contained in <DigestValue> the forged XML document.

6. Conclusion

XML Signature wrapping attacks try to inject forged elements into the XML document structure in such a way that the valid signature covers the unmodified elements, while forged elements are processed by the application logic. This results in a scenario, where an attacker can perform arbitrary web service requests, while authenticating as a

legitimate user. The proposed algorithm takes help of the absolute XPath as a "Positional Token" for identifying the signed elements and adding this to the elements being signed as an attribute before the canonicalization process has a trace of both content of signed element and its position in the XML document as well. Hence, the proposed algorithm can solve the issue of XML signature wrapping attacks elegantly without much change in the current standard.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This draft proposes a modification to the existing algorithm of XML signature to counter XML Signature wrapping attacks. However other forms of attack may be possible that could not be mitigated.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2807] Reagle, J., "XML Signature Requirements", RFC 2807, DOI 10.17487/RFC2807, July 2000, <https://www.rfc-editor.org/info/rfc2807>.

[RFC3275] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible Markup Language) XML-Signature Syntax and Processing", RFC 3275, DOI 10.17487/RFC3275, March 2002, <https://www.rfc-editor.org/info/rfc3275>.

9.2. Informative References

[I-D.narten-iana-considerations-rfc2434bis] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <https://www.rfc-editor.org/info/rfc2629>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[wrapping_attack] McIntosh, Michael. and Paula. Austel, "XML signature element wrapping attacks and countermeasures", 2005, <https://dl.acm.org/citation.cfm?id=1103026>.

[WS-Security] OASIS., "OASIS Web Services Security (WSS) TC", 2006, <https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss>.

Authors' Addresses

Jitendra Kumar
C-DAC Bangalore
#68, Electronics City Hosur Road
Bangalore 560100
India

Email: jitendra@cdac.in

Balaji Rajendran
C-DAC Bangalore
#68, Electronics City Hosur Road
Bangalore 560100
India

Email: balaji@cdac.in

Bindhumadhava BS
C-DAC Bangalore
Old Madras Road, Opposite Hal Aero Engine Division
Bangalore 560038
India

Email: bindhu@cdac.in