Exploiting the use of canonicalization as content extractor

Pitfalls using exclusive canonicalization for context-free extraction of XMLDSig artefacts in the DSS protocol
DSS canonicalization use case: drop protocol context

• An XML fragment wrapped inside the DSS / SOAP transport wrapper may cause interference
• Namespace definitions leak the XML payload
• Exclusive canonicalization (xml-exc-c14n) offers a way to extract a XML fragment without any inherited namespace definitions
Exclusive canonicalization

• The document is rearranged into a unique representation

• Many aspects affected:
  • Line breaks
  • Attribute ordering
  • Whitespaces
  • Character encoding
  • Namespace declarations

• Let’s focus on namespace declarations ...
Applying xml-exc-c14n:

```xml
<upc1:Root xmlns:upc1="http://www.ac.upc.edu/namespaces/ns1" xmlns:upc2="http://www.ac.upc.edu/namespaces/ns2">
  <upc1:Child1 xml:lang="EN">child1 content</upc1:Child1>
  <upc2:Child2>
    <upc1:Child21>child21 content</upc1:Child21>
    <upc1:Child22>child22 content</upc1:Child22>
  </upc2:Child2>
</upc1:Root>
```

```xml
<upc1:Root xmlns:upc1="http://www.ac.upc.edu/namespaces/ns1">
  <upc1:Child1 xmlns:upc2="http://www.ac.upc.edu/namespaces/ns2" xml:lang="EN">child1 content</upc1:Child1>
  <upc2:Child2>
    <upc1:Child21>child21 content</upc1:Child21>
    <upc1:Child22>child22 content</upc1:Child22>
  </upc2:Child2>
</upc1:Root>
```
Canonicalization of a XML signature

• The XMLDSig defines the use of canonicalization for reliable verification
• Applying exclusive canonicalization twice does not change the outcome!
• XMLDSig allows ‘user defined’ transformations, like XPath, XPointer, XSLT ...
• These transformation can be applied **between** an extraction step and the pre-digesting canonicalization!
• The transformation may rely on elements affected by the extraction step!
Proof of problem

• A tailored XMLDSig can be delivered to a DSS endpoint
• Tailored transformations can detect the extraction step!
• Transformation output (and reference digest) differs depending on the applied extraction
• The outcome of the verification depends on the protocol variant
  • inline with xml-exc-c14n
  • base64 encoded
• A simple XPath transformation (like `/*/*/namespace::*`, `filter=,subtract`) can do the job. It drops the original namespace declarations
After XPath transformation

Without xml-exc-c14n extraction applied

```xml
<upc1:Root>
  <upc1:Child xml:lang="EN">child1 content</upc1:Child1>
  <upc2:Child2>
    <upc1:Child21>child21 content</upc1:Child21>
    <upc1:Child22>child22 content</upc1:Child22>
  </upc2:Child2>
</upc1:Root>
```

digest: DMCN87SXxG7GbdEvyREsoWT5/F0zoUO2pmg1FjjGnTY=

With xml-exc-c14n extraction applied

```xml
<upc1:Root xmlns:upc1="http://www.ac.upc.edu/namespaces/ns1">
  <upc1:Child1 xmlns:upc2="http://www.ac.upc.edu/namespaces/ns2" xml:lang="EN">child1 content</upc1:Child1>
  <upc2:Child2>
    <upc1:Child21>child21 content</upc1:Child21>
    <upc1:Child22>child22 content</upc1:Child22>
  </upc2:Child2>
</upc1:Root>
```

digest: mmhguB350i2jVMIoT+NNc6mYGKCJyI2ENbcF/3+Pccs=
Exploiting it..

• Produce a suitable document (e.g. an order)
• Request an enveloping signature with a tailor-made transformation
• Place the order
• Request a verification report at a DSS endpoint using inline SignatureObject
• Receive a 'failure' report
• Deny the authorship of the order

• 'Non-repudiation' property of the signature circumvented!
Mitigations

• Prohibit transformations
  XMLDSig are an integral building block of the standard, XPath support is required -> not an option.

• Inspect transformations
  The ability to detect applied xml-exc-c14n transformation depends on the signed document and the transformation type. Presumably no way to detect.

• Don’t embed XML fragments!