This document includes execution instructions, results, and analysis for homework 6. Source code is included as HW6\_dwilso.zip. Additionally, a compiled jar is provided, ‘java\_security\_hw6.jar’. Execution directions in this readme depend on this jar as well as a folder containing modified class files.

# Part 1: C14N Demo

The C14N demo implemented works on a specified XML file. The demo prints the document, canonicalizes the XML, and prints the new document. The demo uses the Apache Santaurio xmlsec library and specifically the <http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments> definition for canonicalization.

## Execution:

java -cp java\_security\_hw6.jar edu.jhu.dwilso95.C14NDemo src/main/resources/directory.xml

## Code:

All code is in a single class due to the the limited functionality required. The class is located in ‘/src/main/java/edu/jhu/dwilso95/C14NDemo’.

## Demo Output:

C:\Users\dwilson\workspace\java-security-hw6>java -cp java\_security\_hw6.jar edu.jhu.dwilso95.C14NDemo src/main/resources/directory.xml

<---------- Input XML ---------->

<?xml version="1.0" encoding="UTF-8"?>

<schema xmlns='http://www.w3.org/2000/10/XMLSchema'

targetNamespace='http://www.jhu.edu/directory' elementFormDefault='qualified'>

<directory name="project">

<file lastModified="2017-05-19" extension="txt" created="2017-05-17"> readme

</file>

<directory name="source">

<file lastModified="2017-04-22" extension="cpp" created="2017-04-19">HelloWorld

</file>

</directory>

<directory name="resources&amp;stuff" />

</directory>

</schema>

<------ Canonicalized XML ------>

<schema xmlns="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified" targetNamespace="http://www.jhu.edu/directory">

<directory name="project">

<file created="2017-05-17" extension="txt" lastModified="2017-05-19"> readme

</file>

<directory name="source">

<file created="2017-04-19" extension="cpp" lastModified="2017-04-22">HelloWorld

</file>

</directory>

<directory name="resources&amp;stuff"></directory>

</directory>

</schema>

## Security Discussion:

Standardized canonicalization enables security features in XML. Specifically called out in the W3C specification is in the use of XML signatures. Signatures can be generated based on the canonical form of the document. By normalizing the document before signing, it is more likely that signatures will be applied and verified correctly.

Furthermore with the use of Exclusive XML Canonicalization, which is not implemented in this assignment, one could sign subdocuments of an XML document. When that subdocument is removed from the original document and/or inserted into a different context it does not necessarily break validation.

# Part 2: Signing and Verifying XML Documents

## Code:

Code is distributed across a handful of classes in ‘/src/main/java/edu/jhu/dwilso95/’.

## Execution:

java -cp target\java-security-hw6.jar edu.jhu.dwilso95.OrderSignerVerifier sv src/main/resources/order.xml orderSigned.xml

## Demo Output:

C:\Users\dwilson\workspace\java-security-hw6>java -cp java\_security\_hw6.jar edu.jhu.dwilso95.OrderSignerVerifier sv src/main/resources/order.xml orderSigned.xml

SignerVerifier.addKeyInfoAndSign(): entered

--- Here is what we signed ---

<Order id="order1">

<Item dom="12Nov2002" id="item1">F-16 Jet Fighter</Item>

<Color></Color>

<Quantity>1</Quantity>

<Amount>8200000</Amount>

</Order>

<Commission id="commission1">10000</Commission>

Signature verification successful

## Security Discussion:

Most established examples for signing XML in Java, including those mentioned in the course, no longer work. The examples relied on Java searching for all “id” attributes in the document when adding references for signatures. This search behavior was vulnerable to XML Signature wrapping attacks, see Java bug\_id 8017171 for more information.

In order add references easily, an XSD had to be defined to specifically call out what field is the correct id for adding references. This XSD had to define the id attribute as the ID type defined by <http://www.w3.org/2001/XMLSchema>.

# Software Design:

All code for this assignment is in the base package “edu.jhu.dwilso95”. There are two entry points, C14NDemo and OriginSignerVerifier, responsibly for executing part 1 and 2 respectively,

Part 1, the C14NDemo, is simple so the functionality was written completely in one class. The main method takes a single argument, the path to an XML file to canonicalize. There is a method for handling input file location, one for reading the file, and one for getting a canonicalizer. The main method executes these three methods in order and then runs the canonicalizer against the input file.

The design for part 2, order signing and verifying, included multiple classes. The entry point for this functionality is included in OrderSignerVerifier. This supports signing orders, verifying signed orders, and executing both. Signing is handled by the Signer class and verification is handled by Verifier. Each utilizes utility classes for reading XML files into Document objects and handling the included Java key store.

## Class diagram:

