How I fought with XJC and won!

Stuart Maclean

Applied Physics Laboratory
University of Washington

June 2014

Outline

Structured Threat Information Expression (stix.mitre.org)

STIX^{TM} is a collaborative community-driven effort to define and develop a standardized language to represent structured cyber threat information. The STIX Language intends to convey the full range of potential cyber threat information and strives to be fully expressive, flexible, extensible, automatable, and as human-readable as possible. All interested parties are welcome to participate in evolving STIX as part of its open, collaborative community.

The MITRE Corporation is a not-for-profit company that operates multiple federally funded research and development centers.

STIX Uses XML Schema

- STIX is a vocabulary for use in the cybersecurity domain.
- This vocabulary is described using XML schema.
- I want to process STIX documents using Java, in order to share cybersecurity information with others:
  - Author/send documents
  - Receive documents
- I know next to nothing about the STIX schema authoring process, and use JAXB just often enough to forget everything I had learned last time!

STIX XML Schema Set Is Non-Trivial

Latest STIX download at
`http://stix.mitre.org/language/version1.1.1/stix_v1.1.1.zip`.

Contains 25 .xsd files, split into *core* (11) and *extensions* (14) sets.

```
% find . -name *.xsd -exec grep "<xs:complex" {} \;|wc = 187

% find . -name *.xsd -exec grep "<xs:element" {} \;|wc = 402
```

So we are looking at nearly 600 types, each mapping to a Java class.
Imagine the variation of valid STIX documents!

Sample STIX Documents

- Also available from Mitre are some sample STIX documents.
- Mandiant's Advanced Persistent Threat (APT1) Report (2013). About 3MB of XML! Mostly IP addresses, domain names, file hashes.
- Fireeye Poison Ivy Report (2013). About 2MB. Content per APT1.
- Currently, STIX documents are mostly *observables* and *indicators*.
- Short-term goal is to ingest these samples into my Java program. Only then can I "speak STIX".

Java XML Binding (JAXB)

- JAXB is a Java technology for transforming XML schema documents into Java classes.
- We then manipulate object graphs rather than dealing with XML directly.
- JAXB API provides classes for marshalling and unmarshalling.
- Central tool is `xjc`, bundled with JDK. Also available standalone. It is `xjc` that produces Java classes from XML schemas.

XML to Java

Reading XML documents into Java is called *unmarshalling*:

```
JAXBContext jc = JAXBContext.newInstance("foo.org:acme.com");
Unmarshaller um = jc.createUnmarshaller();
um.setEventHandler( new DefaultValidationEventHandler() );
FileInputStream fis = new FileInputStream( "stixDoc.xml" );
JAXBElement<STIXType> e = (JAXBElement<STIXType>)
  um.unmarshal( fis );
```

Similarly for *marshalling*, the reverse operation.

XJC – The JAXB Brains

Given this

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="a" elementFormDefault="qualified"
   attributeFormDefault="unqualified" version="1.0">
  <xs:complexType name="A">
  </xs:complexType>
</xs:schema>
```

we do this

```
% xjc a.xsd
```

and get this

```
a/A.java
a/ObjectFactory.java
```

XML Schema Composition Via Imports

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="b" version="1.0">
  < xs:import namespace="a" schemaLocation="a.xsd"/ >
  <xs:complexType name="B"/>
</xs:schema>
```

produces this

```
b/B.java
b/ObjectFactory.java
a/A.java
a/ObjectFactory.java
```

We'll call the set of .xsd documents reachable from all xjc inputs and all reachable imports the *universe*.

XJC Issue 1 — Already Defined Error

```
% xjc a.xsd
OK

% xjc a.xsd a.xsd
OK

% cp a.xsd a2.xsd
% xjc a.xsd a2.xsd
[ERROR] 'A' is already defined
```

Easy to spot this error, but in a larger set? With many imports? Authored by others?

XJC Issue 2 — Not Part of Compilation Error

To combat name clashes, we use *bindings files* to alter name generation schemes. But

```
$ cat x.xjb
<jxb:bindings version="2.0"
  <jxb:bindings schemaLocation="x.xsd" node="/xs:schema">
  </jxb:bindings>
</jxb:bindings>
```

```
% xjc a.xsd -b x.xjb
[ERROR] x.xsd is not part of this compilation...
```

STIX and JAXB

In an ideal world, this would be all we need

```
% jar xvf stix_v1.1.1.zip

./campaign.xsd
./stix_core.xsd
./stix_common.xsd
./indicator.xsd
./threat_actor.xsd
./extensions/vulnerability/cvrf_1.1_vulnerability.xsd
./extensions/address/ciq_3.0_address.xsd
./extensions/identity/ciq_3.0_identity.xsd
./and/some/more/...

% xjc *.xsd extensions    // NB: files, dirs
```

If only! Dozens of errors, described next.

STIX Schema Issues

As expected, STIX schema files use imports. But, the imports are *remote* and clash with *local* files included in the zip distro.

```
% ls
stix_core.xsd
stix_common.xsd

% cat stix_core.xsd

<xs:import namespace="http://stix.mitre.org/common-1"
    schemaLocation="http://stix.mitre.org/XMLSchema/
                    common/1.1.1/stix_common.xsd"/>

% cat stix_common.xsd

<xs:schema targetNamespace="http://stix.mitre.org/common-1">
```

STIX Schema Issues

```
% xjc *.xsd
[ERROR] 'MarkingType' is already defined
  line 26 of file:/path/to/stix_v1.1.1/data_marking.xsd

[ERROR] (related to above error) the first definition appears
  line 14 of http://stix.mitre.org/XMLSchema/data_marking/
  1.1.1/data_marking.xsd
```

STIX Schema Issues and Java

- STIX authors expertise mostly in Python.
- Github contributions to the STIX effort mostly Python (`https://github.com/STIXProject`).
- Questions to Mitre/community concerning binding STIX schemas to Java then mostly met with blank responses, or worse, "works for me with Python".

Binding STIX to Java

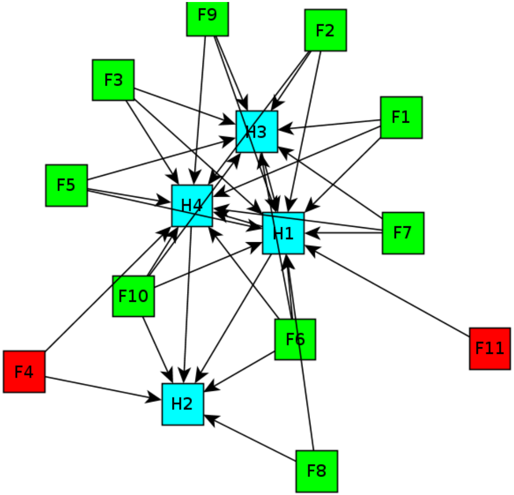
- Flailed around editing .xsd files, and tried all sorts of (random!) file combinations as input to `xjc`.
- As one error was vanquished, another would appear.
- With minimal sets, `xjc` compilation would be OK, but STIX sample documents would not unmarshal (required classes not available).
- Needed a method!

XSDWalker - A Java-Based Tool

- To solve the name clash problems, *we* must walk the xsd set, following all imports, building up a graph. Essentially mimic `xjc`.
- Graph nodes are xsd documents, edges are import relationships.
- Then identify just leaf nodes L (easy, they have in-edge count 0).
- Then prune L of nodes whose target namespace appears in any remote node, producing result R.
- Write the resulting node set out as an *uber xsd*, consisting solely of imports of all nodes in R.
- Offer just the uber xsd to xjc!

Visualizing XSDWalker Output

Legend on next slide. . .



Visualizing XSDWalker Output — Legend

Legend for previous graphic. . .

- Start with 11 core STIX .xsd files from the downloaded zip. Green nodes, local urls so labelled with F.
- XSDWalker resolves all imports (recursively) of all nodes in F. Result is cyan nodes, remote/http nodes so labelled H.
- Black edges show import dependency of one xsd document on another.
- Identify all nodes in the green set with a target namespace which is shared with some cyan node. Transform these from green to red.
- Eliminate the red nodes. Supply just the remaining green nodes as the input to `xjc`.

Recall the full .xsd set in the STIX v1.1.1 distro is 25 xsd files. That graph thus more complex, but algorithm remains the same.

XSDWalker in Action

```
% xsdwalker -u stixcore *.xsd

% cat stixcore.uber.xsd

<xs:schema targetNamespace="stixcore.uber.xsd">
 <xs:import namespace="http://stix.mitre.org/stix-1"
  schemaLocation="file:/path/to/stix_v1.1.1/stix_core.xsd"/>
 <xs:import namespace="http://stix.mitre.org/CourseOfAction-1'
  schemaLocation="file:/path/to/stix_v1.1.1/course_of_action.x
 <xs:import namespace="http://stix.mitre.org/TTP-1"
  schemaLocation="file:/path/to/stix_v1.1.1/ttp.xsd"/>
</xs:schema>

% xjc stixcore.uber.xsd     [-b some.xjb ?]
```

XSDWalker in Action II

Also includes some useful options, notably to exclude xsd files we *know* are broken, and a flag for writing out graph structure.

```
% xsdwalker -u stixcore *.xsd -e extensions/vulnerability
   -e extensions/test_mechanism/oval.xsd -g
```

Command line processing courtesy of `commons-cli`. Build via Maven.

Talk is cheap

Show me the code!

Code Organization

- Command line processing: files, directories, urls.
- Main document visit loop, builds graph.
- XSD parsing via XPath (has to be namespace aware).
- Uber output creation, optional graph output for later visualization.

Summary

- Core Java classes sufficient to write a tool to walk an xsd document set, containing both local (`file://`) and remote (`http://`) elements.
- Applied to a specific xsd set (STIX) but applicable in general.
- xjc is a beast, you cannot defeat it, so be creative.
- `https://github.com/UW-APL-EIS/xsdwalker.git`