An Incremental Learner for Language-Based Anomaly Detection in XML

Harald Lampesberger

Department of Secure Information Systems
University of Applied Sciences Upper Austria
harald.lampesberger@fh-hagenberg.at

LangSec Workshop, 26. May 2016
Motivation

Extensible Markup Language (XML)
- Data serialization format for many protocols
- SOAP/WS-*, XMPP, SAML, XHTML, RSS, Atom, ...

Schema validation is a first-line defense
- A schema specifies types of elements and production rules
- Validation rejects unacceptable inputs

Two language-theoretic flaws
1. XML Schema (XSD) extension points are wildcards
2. References raise expressiveness beyond context free
XSD Extension Points

From http://schemas.xmlsoap.org/soap/envelope/

...<xs:element name="Header" type="tns:Header"/>
<xs:complexType name="Header">
  <xs:sequence>
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
...
Signature Wrapping Attack

Digitally signed part ≠ processed part
- Used in WS-Security and SAML single sign-on
- Somorovsky et al. (2012): 11/14 SAML implementations vulnerable

```
soap:Envelope
  soap:Header
    wsse:Security
      ds:Signature
        ds:SignedInfo
          ds:Reference
            @URI
  soap:Body
    @wsu:Id
      MonitorInstances
#123
123
verified,
processed
```
Signature Wrapping Attack

Digitally signed part $ \neq $ processed part

- Used in WS-Security and SAML single sign-on
- Somorovsky et al. (2012): 11/14 SAML implementations vulnerable

Jensen et al. (2011): removing extension points is hard
Language-Based Anomaly Detection

Approach: learn the acceptable language
Language-Based Anomaly Detection

Approach: learn the acceptable language

1. Datatyped XML Visibly Pushdown Automaton (dXVPA)
   - Mixed-content XML streaming
   - Datatypes generalize character data
   - Character-data XVPA (cXVPA) for stream validation
Language-Based Anomaly Detection

Approach: learn the acceptable language

1. Datatyped XML Visibly Pushdown Automaton (dXVPA)
   - Mixed-content XML streaming
   - Datatypes generalize character data
   - Character-data XVPA (cXVPA) for stream validation

2. Incremental learner for grammatical inference
   - Constructs a dXVPA from examples
   - Unlearning and sanitization against poisoning attacks
Language-Based Anomaly Detection

Approach: learn the acceptable language

1. Datatyped XML Visibly Pushdown Automaton (dXVPA)
   - Mixed-content XML streaming
   - Datatypes generalize character data
   - Character-data XVPA (cXVPA) for stream validation

2. Incremental learner for grammatical inference
   - Constructs a dXVPA from examples
   - Unlearning and sanitization against poisoning attacks

3. Experiments
   - Train and test
   - Two synthetic scenarios from ToXgene
   - Two realistic scenarios from Axis2 web service
dXVPAs

Event stream alphabets
- $\Sigma_{\text{call}} \ldots$ startElement
- $\Sigma_{\text{ret}} \ldots$ endElement
- $\Sigma_{\text{int}} \ldots$ datatypes

Stack alphabet = states

States partitioned into modules (schema types)

Transitions in and between modules

cXVPA representation
- Unified text checks
- Fast validation

Example XML:

```
<ord>
  <itm>Product A</itm>
  <itm>8877955335</itm>
</ord>
```
Incremental Learning Step

Learner computes an updated dXVPA

- Datatyped event stream
- \( A_i \ldots \) incrementally updateable automaton
- \( \omega_i \ldots \) frequencies of states and transitions

Validator checks acceptance

\[
\begin{align*}
& \text{Training doc}_i \\
\downarrow & \quad \text{incWeightedVPA} \\
A_{i-1}, \omega_{i-1} & \quad \text{Learner} \\
\downarrow & \quad \text{trim} \\
A_i, \omega_i & \quad \text{genXVPA} \\
\downarrow & \quad \text{...} \\
dXVPA_i & \quad \text{Validator} \\
\downarrow & \quad \text{cXVPA}_i \\
\downarrow & \quad \text{accept} \\
\text{Document} & \quad \text{yes} \\
& \quad \text{no}
\end{align*}
\]
How Learning Works

Every event stream prefix gets a unique state

- A named state is a pair \((u, v)\)
- \(u\) . . . typing-context string
- \(v\) . . . left-sibling string

Merge two states if they are \(k/l\)-locally the same
How Learning Works

Every event stream prefix gets a unique state

- A named state is a pair \((u, v)\)
- \(u\) ... typing-context string
- \(v\) ... left-sibling string

Merge two states if they are \(k\)-/\(l\)-locally the same

```
dealer
  usedcars
    ad
    ad
  newcars
    ad
    ad
    ad
  ad
  model
  year
    VW
    2014
  model
    Tesla
```
How Learning Works

Every event stream prefix gets a unique state

- A named state is a pair \((u, v)\)
- \(u\) . . . typing-context string
- \(v\) . . . left-sibling string

Merge two states if they are \(k/l\)-locally the same

(latex diagram)

\((\text{dealer} \# \text{usedcars} \cdot \text{newcars}, \text{ad} \cdot \text{ad})\)
How Learning Works

Every event stream prefix gets a unique state

- A named state is a pair \((u, v)\)
- \(u\) ... typing-context string
- \(v\) ... left-sibling string

Merge two states if they are \(k/l\)-locally the same

Harald Lampesberger
An Incremental Learner for Language-Based Anomaly Detection in XML
Poisoning Attacks

$\omega_i \ldots$ frequencies of states and transitions from learning

Unlearning

- An already learned attack is later identified
- Remove specific knowledge by decrementing $\omega_i$
- Trim zero-weight states and transitions

Sanitization

- Hidden poisoning attacks
- Assumption: only few of those
- Decrement $\omega_i$ and trim zero-weight states and transitions
Experiments

Two synthetic and two realistic datasets

Learning progress
• Train and test, binary classification, mind changes (MC)

Catalog, $k = 1, l = 2$

VulnShopAuthOrder, $k = 1, l = 2$
Conclusions

Learner outperformed schema validation
- All signature wrapping attacks were detected (schema validation: 0)
- No false positives
- False negatives resulted from coarse XSD datatypes
- Fast convergence

Contributions in the paper
- dXVPA and cXVPA language representations
- Lexical datatype system for datatype inference from text
- Algorithms for the incremental learner
- Details on experiments

Use cases
- Security mechanism for any XML-based interaction
- Especially for systems using composed schemas
- XML firewall
Appendix
Lexical Datatype System

Learner needs a datatyped event stream

- Lexically distinct XSD datatypes instead of character data

1. Lexical subsumption
   - Minimally required datatypes
   - Strict subsumption is ambiguous
   - false $\mapsto \{\text{lang.}, \text{bool.}, \text{NCName}\}$

2. Preference heuristic
   - Datatypes partitioned into kinds
   - “Preferred” relation
   - e.g. boolean $<$ language
   - false $\mapsto \{\text{boolean}\}$

Example

- $\{1, 0, \text{true}, 33\} \mapsto \{\text{boolean, unsignedByte}\}$