Chapter 9

A Formal Language for XML Authorisations Based on Answer Set Programming and Temporal Interval Logic Constraints

Sean Policarpio
University of Western Sydney, Australia

Yan Zhang
University of Western Sydney, Australia

ABSTRACT

The Extensible Markup Language is susceptible to security breaches because it does not incorporate methods to protect the information it encodes. This work focuses on the development of a formal language that can provide role-based access control to information stored in XML formatted documents. This language has the capacity to reason whether access to an XML document should be allowed. The language, $A_{xml}(T)$, allows for the specification of authorisations on XML documents and distinguishes itself from other research with the inclusion of temporal interval reasoning and the XPath query language.

INTRODUCTION

The Extensible Markup Language (XML) (WWW Consortium, 2008) has steadily become a common encoding format for software applications. It is a popular and reliable formatting structure for the storage, presentation, and communication of data over the Internet. Many applications use XML to encode important, and in many cases, private information. Because XML does not have an inherent security model as part of its specification there is a necessity for methods in which access to XML documents can be controlled (WWW Consortium, 2008).

In this paper, we present the development of a formal language that will provide access control
to XML documents. A_{xml}(T) is used to define a
security policy base capable of specifying all the
access rights that subjects in the scope of an XML
environment should have or be denied.

The formal language has particular aspects that
differ from most other implementations. First, it
incorporates the XML query language, XPath, into
it for the purpose of defining which documents
(or elements within a document) we would like
to restrict access to (WWW Consortium, 1999).
An XPath is a string representation of traversing
through an XML document to return an element
within the document. For example, the following
is an XPath that follows the tree-like structure of
a document to return the element author:

/library/books/book/author

XPath also includes other interesting features.
These include, but are not limited to, XPath predic-
tates and wildcards which allow for broader and
much more expressive XPath queries (WWW
Consortium, 1999). As opposed to static XPath’s
which are only meant to return specific nodes
within XML documents, we can use these features
to write dynamic paths that can represent zero to
many elements within the database of documents.

Secondly, the formal language uses the Role-
based Access Control model (Ferraiolo et al.,
1995) as a basis for the structure of authorisations
to subjects. This primarily means rather than
applying authorisations directly to subjects, we
create roles that can have one or more specified
authorisations. This gives us better control over
which subjects have what authorisations and is the
foremost reason this model is chosen over others
(i.e., Discretionary and Mandatory Access Control
models; Ferraiolo et al., 1995). Consequently, it
also allows us to easily incorporate the principles
of separation of duty and conflict resolution di-
rectly into the language (Ferraiolo et al., 1995).

Finally, we incorporate temporal interval logic
reasoning into the formal language. Temporal
intervals are representative of specific sections of
quantitative time. Temporal interval logic is the
study of relating these various points and sections
of time with each other. We use temporal intervals
in our formal language for the purpose of specify-
ing when authorisations to XML documents should
be applied. We also use temporal logic to reason
upon relationships that authorisations could have
with each other with respect to time.

Temporal logic is a well studied field and
many models or methods have been proposed
in the last decades. For our purposes, we choose
to use Allen’s Temporal Interval Relationship
algebra (Allen, 1984). Allen’s temporal relations-
ships cover all possible ways in which intervals
can relate to one another (such as before, meets,
equal, etc.) and are incorporated into the syntax
of our formal language. However, it should be
noted that what makes Allen’s temporal interval
logic differ from others, and what makes it ap-
pealing for our work, is that it forgoes relating
intervals with specific quantities of time. Simply,
Allen’s logic relates intervals without the need to
specify or know exactly when an interval takes
place. This is possible due to the fact that when
a temporal interval takes place is implied by its
relationship(s) with all other intervals. Therefore,
for an interval to exist and be relevant, it only need
have at least one of Allen’s relationships with at
least one other interval.

The semantics of our formal language is pro-
vided through its translation into a logic program.
Answer Set Programming (ASP) is a relatively
new form of programming in the field of knowl-
dge representation and reasoning. It is a form
of declarative programming for search problems
involving non-monotonic reasoning and is based
on Gelfond’s and Lifschitz’s (1988) stable model
semantics of logic programming (Gelfond &
Lifschitz, 1988; Baral, 2003; Lifschitz, 2008).

ASP is used to represent known information
which can be reasoned upon to produce further
knowledge or answers based on the validity of
said information. This is possible because the
initial information can be non-deterministically
written with variableness so that different outputs
can be computed from it. Simply, we can describe
a scenario with an understanding that various
conclusions or answer sets are achievable within
Related Content

A Social Ontology for Integrating Security and Software Engineering
www.igi-global.com/chapter/social-ontology-integrating-security-software/29420?camid=4v1a

Estimation of Factor Scores of Impressions of Question and Answer Statements
www.igi-global.com/article/estimation-of-factor-scores-of-impressions-of-question-and-answer-statements/89775?camid=4v1a

Test-Case Mutation
www.igi-global.com/chapter/test-case-mutation/30751?camid=4v1a

Scheduling Large and Complex IT Projects Using Sliding-Frame Approach
www.igi-global.com/chapter/scheduling-large-complex-projects-using/77769?camid=4v1a