Performance assessment of XACML authorizations for Supply Chain Traceability Web Services

Citation

As Published
http://dx.doi.org/10.1109/CASoN.2012.6412432

Publisher
Institute of Electrical and Electronics Engineers (IEEE)

Version
Author's final manuscript

Accessed
Wed Jan 06 21:00:12 EST 2016

Citable Link
http://hdl.handle.net/1721.1/87637

Terms of Use
Creative Commons Attribution-Noncommercial-Share Alike

Detailed Terms
http://creativecommons.org/licenses/by-nc-sa/4.0/

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
Performance Assessment of XACML Authorizations for Supply Chain Traceability Web Services

Miguel L. Pardal†, Mark Harrison‡, Sanjay Sarma§, José Alves Marques†

†Department of Computer Science and Engineering
Instituto Superior Técnico, Technical University of Lisbon, Portugal

‡Auto-ID Labs, Institute for Manufacturing, University of Cambridge, UK

§Auto-ID Labs, Massachusetts Institute of Technology, USA

Email: miguel.pardal@ist.utl.pt, mark.harrison@cantab.net, sesarma@mit.edu, jose.marques@link.pt

Abstract—Service-Oriented Architecture (SOA) and Web Services (WS) offer advanced flexibility and interoperability capabilities. However, they imply significant performance overheads that need to be carefully considered.

Supply Chain Management (SCM) and Traceability systems are an interesting domain for the use of WS technologies that are usually deemed to be too complex and unnecessary in practical applications, especially regarding security.

This paper presents an externalized security architecture that uses the eXtensible Access Control Markup Language (XACML) authorization standard to enforce visibility restrictions on traceability data in a supply chain where multiple companies collaborate; the performance overheads are assessed by comparing ‘raw’ authorization implementations - Access Control Lists, Tokens, and RDF Assertions - with their XACML-equivalents.

Keywords—Web Services; Authorization; XACML; Performance; Supply Chain Traceability

I. INTRODUCTION

Service-Oriented Architecture (SOA) [1] and Web Services (WS) [2] were envisioned to cope with constant software change and to simplify the interoperation of heterogeneous systems. WS-related standards address non-functional concerns such as security, transactions, and reliable messaging. However, the complexity of these standards, alone and combined, raises significant performance concerns.

The main criticism against the use of WS technologies is that they are slow and difficult to use. The performance overheads have been measured in practical implementations. Juric at al. [3] compared the performance of SOAP with platform-specific technologies, namely, with Java Remote Method Invocation (RMI). They reported that a typical SOAP message is, on average, 4.3 times larger than an RMI message, and that the WS response time is 9 times slower than RMI.

If WS-Security [4] technologies are used, then the overheads are even more significant: messages are 27 times larger, and response times are 15 times slower.

The main reason for these overheads was found to be the use of verbose XML data encoding for the message formats. If WS-Security [4] technologies are used, then the overheads are even more significant: messages are 27 times larger, and response times are 15 times slower.

The main reason for these overheads was found to be the use of verbose XML data encoding for the message formats. The data access mode is also relevant. Bayardo et. al [5] found in their study that stream-based data access gives significantly better performance than document-based access.

In this paper, we investigate the performance of using WS authorizations in the supply chain traceability domain. Dynamic authorization is necessary because there are multiple companies involved, they are distributed across the world, and many of them do not have prior knowledge of the others.

A. Supply Chain Traceability

The world economy depends on countless supply chains that provide goods from producer to consumer. Supply Chain Management (SCM) solutions focus on planning and execution, integrated with companies’ Enterprise Resources Planning (ERP) systems. Their purpose is to optimize the physical object flows so that goods travel in the least amount of time and at the lowest cost [6]. To achieve this goal, it is necessary to keep track of what is moving along the supply chain.

Radio Frequency Identification (RFID) technology can significantly improve supply chain traceability, by allowing automatic data capture by RFID readers of identifiers stored inside RFID tags attached to items and/or pallets. Captured data is stored by each company at dispersed data silos.

The Electronic Product Code (EPC) Network architecture defines standard data capture and query interface for repositories of RFID data, called the EPC Information Services (EPC IS) [9]. EPC Discovery Services (EPC DS) [10] [11] can index these scattered repositories and allow traceability queries [12], such as Track (Where is the item?) and Trace (Where has the item been?) to be efficiently answered.

Each supply chain participant can benefit from exchanging data with other companies, but information such as the levels...
of demand, inventory, and supplier identities should be kept private in a restricted circle of trust. Data access control is crucial if companies will be willing to share data [13].

B. Overview

This paper compares the performance of visibility restriction approaches for supply chains using an authorization component of an externalized security architecture that relies on SOA principles and WS-related technologies; ‘raw’ implementations are compared with equivalent policies in a standard authorization language to measure the overheads.

In the next section, the traceability system architecture for the supply chain domain is described. Then the externalized security architecture is presented with specific focus on authorization. Next we describe the supply chain authorization implementations along with their conversion to a standard authorization language. The paper ends with the performance assessment, conclusions and future work prospects.

II. Traceability System

There are several architecture proposals to build traceability systems. Pardal and Alves Marques [14] summarized them using the criteria of centralization and data integration. For this work the Meta-Data Integration (MDI) architecture was chosen as the reference because it is aligned with the EPC Network architecture [8], defining two system layers:

- EPC IS servers collect the detailed event data;
- EPC DS servers store data links to EPC IS servers.

Using this approach, traceability data sharing policies can be authored by the data owner and then used both at EPC DS and IS layers, as depicted in Figure 1.

III. Externalized Security

The goal of the externalized security architecture is to unify the security management across applications so that business rules can be changed dynamically. For this reason, it is especially suited to virtualized cloud deployments [15].

Externalized security encompasses user management, authentication, authorization, logging and auditing. The security properties to be preserved from attacks are depicted in Figure 2. Companies have to be authenticated and data access must be authorized. The authentication can be achieved with identity providers [16] and the exchange of SAML assertions [17]. The authorization can be achieved with XACML [18] that is an XML vocabulary to represent authorization policies and requests that avoids hard-coded rules and allows improved consistency of policy enforcement. Hebig et al. [19] demonstrated how the different technologies needed for externalized security can work together.

A. eXtensible Access Control Markup Language

XACML [18] allows fine-grained access control and combination of (possibly conflicting) policies in an externalized security architecture. In general terms, an authorization is the verification of a subject’s right to execute an action on a resource. The standard defines a policy format, follows a processing model and requires actual implementation.

1) Policy: Figure 3 presents a simplified XACML Policy structure with target and rules.

The target defines a simplified set of conditions that determine if the policy is relevant for the request. Rules are conditions that evaluate to ‘Permit’, ‘Deny’, ‘NotApplicable’ (when no target matches), or ‘Indeterminate’ (when internal errors occur). XACML has a set of predefined functions that are used to transform and compare data attributes, to arrive at an outcome.
Policies are declarative and do not allow function definitions to keep algorithmic complexity low [20]. Additional functions can be provided by the library implementations, but this limits the compatibility of policy interpretation.

2) Execution: The authorization architecture that XACML assumes is defined by RFC 2753 [21] and 2904 [22], and defines several structural elements represented in Figure 4. The Policy Administration Point (PAP) is used to author and manage policies. The Policy Enforcement Point (PEP) is an application-specific component that intercepts requests to access a resource, and can also perform ‘before’ or ‘after’ actions, called obligations. The PEP checks with the Policy Decision Point (PDP), a generic component that takes any request along with a set of policies, and evaluates the request with respect to the applicable policies. The Policy Information Point (PIP) retrieves additional attribute values for the PDP not contained in the request, if required.

\[\begin{align*}
\text{PIP} & \rightarrow \text{PEP} & 
\text{PAP} & \rightarrow \text{PDP} & 
\text{Obligation service} & \rightarrow \\
1: \text{access attempt} & \rightarrow & 5: \text{obligations} & \rightarrow & 6: \text{permit/deny} & \\
2: \text{request} & \rightarrow & 3: \text{attribute values} & \rightarrow & 4: \text{response} & \\
0: \text{policies} & \rightarrow & & & & \\
\end{align*}\]

Fig. 4. XACML request processing.

3) Implementations: There are several available XACML libraries and tools, both open-source and commercial. The open-source implementations were surveyed and the findings are presented in Table I.

<table>
<thead>
<tr>
<th>Impl.</th>
<th>Version</th>
<th>Last update</th>
<th>Sponsors</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunxacml</td>
<td>1.2</td>
<td>Jul 2004</td>
<td>Sun Microsystems (currently Oracle)</td>
</tr>
<tr>
<td>Heras-AF</td>
<td>1.0.0-M2</td>
<td>Sep 2010</td>
<td>U. Applied Sciences Rapperswil, Switzerland</td>
</tr>
<tr>
<td>enterprise</td>
<td>r258</td>
<td>Jan 2009</td>
<td>Zinan Wang</td>
</tr>
<tr>
<td>java-xacml</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PicketBox</td>
<td>3.0.0-Final</td>
<td>Feb 2011</td>
<td>JBoss, Red Hat</td>
</tr>
<tr>
<td>xEngine</td>
<td>beta 0.2</td>
<td>Aug 2010</td>
<td>Michigan State U., North Carolina State U.</td>
</tr>
</tbody>
</table>

TABLE I
OPEN-SOURCE XACML IMPLEMENTATIONS.

Access control for supply chains is different from traditional authorization because there is a lack of prior knowledge about who should be authorized due to the way that each physical object path emerges [26].

The Supply Chain Authorization (SC-Az) Application Programming Interface (API) has been proposed recently [27] to allow companies participating in a supply chain to express their authorizations using business concepts such as item, company, etc.; and then to enforce the resulting policies using alternative underlying mechanisms.

The previous SC-Az version had two implementations: Enumerated Access Control (EAC) and Chain-of-Communication Tokens (CCT). EAC uses an Access Control List (ACL) implementation available on the Java virtual machine\(^3\). CCT represents access rights within object references, called tokens, implemented with custom code. For this research, we added a new implementation, Chain-of-Trust Assertions (CTA), based on the Apache Jena\(^4\) Semantic Web [28] library.

A. Chain-of-Trust Assertions

CTA is different from EAC and CCT because its semantics can be extended. Access rights are expressed using logical statements, called assertions, issued by the multiple parties. Figure 5 shows the assertion-related operations.

\[
\begin{align*}
\text{<<interface>> CTA} & \\
+\text{requestAssertion}(user, action, resource) : \text{Assertion} & \\
+\text{addAssertion}(assertion) & \\
+\text{checkAssertions}(user, action, resource) : \text{Decision} & \\
\end{align*}\]

Fig. 5. CTA interface operations.

Access is granted if there is an explicit unbroken chain of trust assertions leading back to the owner of the data. Figures 6 and 7 show a simple CTA policy stated in Resource Description Framework (RDF) classes and properties, expressed as subject-predicate-object tuples.

In the example, policy0 created by company0 (data owner) grants read access to record0 about item0 to company0 and company1. The extensibility can be achieved by adding new properties e.g. cta:grantsWrite to grant write access.

\(^3\)Package sun.security.acl
\(^4\)http://jena.apache.org/

\(^2\)Holistic Enterprise-Ready Application Security Architecture Framework.
Fig. 6. CTA Policy in RDF Turtle format.

Fig. 7. CTA Policy graph.

B. Conversion to XACML

To convert a CTA policy to a XACML policy the RDF statements are navigated in the following fashion: first the objects of “pol protects id” statements are found. For each item identifier, a XACML policy is created. The policy target matches the item identifier. A permit rule is created to grant each access right – e.g. read – to the objects from “pol grantsRead org” statements. A final catch rule is created so that all other requests regarding the item are also denied. The rule combining algorithm is ‘first-applicable’ so the outcome of the matched first rule is the access decision.

This conversion approach is based on the policy conversions by Karjoth et al. [29], and it allows SC-Az policy instances to be represented in the standard XACML format.

C. Deployment

For the authorizations in the traceability system, XACML is used as the canonical format for representing data access policies, both at the EPC IS and DS levels, as depicted in Figure 1. The policy master copy is maintained at the EPC DS, but a local copy is maintained in each IS to also protect its records locally.

D. Related work

WS technologies are already widely used in RFID system implementations. The most prominent example is Fosstrak [30] that provides a SOAP endpoint to the EPC IS event repository. Also, Guinard [15] developed alternative REST-based interfaces and explored cloud-based deployment.

Shi et al. [31] implemented a secure DS and used an extended attribute-based access control to implement fine-grained access policies in detail. Their custom policy engine is specialized but does not realize the benefits of using a standard policy language, making auditing and policy validation harder. In any case, their engine can be extended to recognize XACML as an input format.

Kerschbaum and Chaves [32] propose an encryption scheme that allows different levels of fine-grained access control to be enforced on each item, also relying on the central role played by a DS.

V. PERFORMANCE ASSESSMENT

The aim of the performance assessment was to evaluate if the performance of the solution is suitable for potentially very large and complex supply chains.

The SC-Az assessment tool [27] was used to perform test runs to measure both from the ‘raw’ EAC, CCT and CTA mechanisms and from their XACML-equivalent forms.

The test machine was a Quad-core CPU at 2.50 GHz, with 3.25 GB of usable RAM, and 1 TiB hard disk; running 32-bit Windows 7 (version 6.1.7601), and Java 1.7.0_04.

The policies were defined using the common SC-Az API to allow the same business needs to be represented internally by each implementation. The policies were then converted to XACML format and tested using the HERAS-AF implementation to assess the correctness and the performance.

A. Evaluation

The data sharing policies were correctly translated and enforced. The performance results are the average of repeated runs of the same experiments, to achieve statistical confidence in the data.

The number of items considered in the experiments were based on information collected by Ilic et al. [33]. We start with small number of items (100) and go up to medium number (10^4).

1) Raw: Figure 8 presents a plot of the ‘request evaluation’ time for increasing number of policies protecting items. We can see that the performance of CCT implemented with custom code is clearly the worse. EAC using Java’s ACLs and CTA using Apache Jena’s RDF handle the loads much better with results below 0.1 ms. The reason for this difference could be that the CCT implementation is stateful whereas EAC and CTA are stateless. CCT needs to perform (unoptimized) searches in token collections to find the right token for each request while the other two receive all the values as arguments.

\[\text{Intel Core 2 Quad Central Processing Unit Q8300}\]
The performance of CCT and CTA are the best, EAC is worse, but on the same order of magnitude.

3) Raw vs. XACML: Comparing the y-axis of Figures 8 and 9 it is visible that the XACML performance overheads are very significant. Table II presents each XACML time divided by the corresponding ‘raw’ time for chains handling increasing numbers of items protected by policies. We observed a 400-fold overhead, on average. Also, the performance overhead increases with the number of deployed policies.

<table>
<thead>
<tr>
<th>Nr. Policies</th>
<th>EAC</th>
<th>CCT</th>
<th>CTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01 · 10^4</td>
<td>49.9</td>
<td>58.9</td>
<td>22.7</td>
</tr>
<tr>
<td>0.05 · 10^4</td>
<td>240.0</td>
<td>63.9</td>
<td>103.5</td>
</tr>
<tr>
<td>0.1 · 10^4</td>
<td>406.7</td>
<td>57.5</td>
<td>191.4</td>
</tr>
<tr>
<td>0.5 · 10^4</td>
<td>1670.1</td>
<td>17.3</td>
<td>752.9</td>
</tr>
<tr>
<td>1 · 10^4</td>
<td>3684.4</td>
<td>10.1</td>
<td>1429.2</td>
</tr>
</tbody>
</table>

TABLE II
XACML OVERHEAD WITH INCREASING NUMBER OF ITEM POLICIES.

VI. Conclusion

The first contribution of this paper is the performance assessment of the enforcement of supply chain data visibility policies, in the context of an externalized security architecture. First, we verified that the policies could be translated and enforced using a standard XACML infrastructure. Then we compared the performance of the ‘raw’ implementations with the XACML-converted policies for increasing number of items being protected. The XACML overheads were found to be very significant: 400 times on average, and over 1000 times in the worst cases.

Our results agree with previous research [3], [5] that shows that WS technologies overheads are usually very significant. There is a clear need to simplify and streamline the standards and there is room for performance improvements in the implementations. However, despite these drawbacks, WS technologies are being widely used today, meaning that their advantages - interoperability, flexibility, tooling support - add value for developers and make up for the additional cost. That is also the case for using XACML to protect supply chain data. XACML allows the policies to be exchanged in a standard format that can be properly interpreted in all the policy enforcement points and allows the use of other tools that comply with the standard (e.g. auditing tools).

The second contribution of this work is the extension of the Supply Chain Authorization (SC-Az) API with the Chain-of-Trust Assertions (CTA) implementation based on Semantic Web technologies. CTA has extensible semantics while EAC and CCT have predefined semantics and, because of this, CTA was expected to be slower. However, for the SC-Az baseline when all three mechanisms express exactly the same visibility restrictions there are no significant performance differences for evaluation of requests, and CTA is even better than EAC. Considering that the performance is similar and that CTA is extensible, moving forward CTA should be the default choice for expressing supply chain authorizations.

A. Future work

There are indications that the performance overheads reported using the HERAS-AF library could be significantly lowered with more optimized alternatives [34], [25]. This proposition needs to be measured in practice before more general conclusions can be drawn.

The evaluation job execution will be enhanced to measure the performance of XACML with a large number of items (more than 10^4 items [33]). The performance impact of representing object groupings – batches – and company sets – groups – will also be assessed.

The suitable performance of CTA opens possibilities for more expressive traceability data sharing policies. Trust can be transitive in some cases, as tested already with good preliminary results. Also, instead of issuing plain trust assertions, parties can issue conditional assertions, like reciprocal trust (“I trust you if you trust me”). Additionally, special predicates can be designed to express dynamic chain upstream/downstream activities.
conditions, allowing data sharing (or at least, initial data discovery) between parties that did not have previous interactions. It can also support the delegation of administrative rights from one organization to another. All of these can be particularly useful in scenarios like recalls [35].

The authorization challenges of externalized security in the supply chain traceability system illustrated how WS can assist in complex and dynamic business environments involving multiple organizations. Many more WS research can be done in this domain as traceability systems requirements are a worthy match for Web Services’ advanced capabilities.

ACKNOWLEDGMENT

Miguel L. Pardal is supported by a PhD fellowship from the Portuguese Foundation for Science and Technology FCT (SFRH/BD/45289/2008).

REFERENCES