Decision-cache Based XACML Authorisation and Anonymisation for XML Documents.

Nils Ulltveit Moe [*], Vladimir Oleshchuk

University of Agder, servicebox 509, N 4898 Grimstad, Norway

Abstract

This paper suggests a fine-grained authorisation model based on the eXtensible Access Control Markup Language (XACML) for XML based messages and documents, that is extended to support privacy-enhanced anonymisation of XML elements containing sensitive information. The solution implements a decision cache for XACML decisions and anonymisation policies. The decision cache is implemented as an XACML obligations service, where a specification of the XML elements to be authorised and anonymised is sent to the decision cache in the Policy Enforcement Point (PEP) during initial authorisation. Further authorisation of individual XML elements according to the authorisation specification is then performed on all matching XML resources, and decisions are stored in the decision cache. This makes it possible to cache fine-grained XACML authorisation and anonymisation decisions, which reduces the authorisation load on the Policy Decision Point (PDP). The theoretical solution is related to a practical case study consisting of a privacy-enhanced intrusion detection system that needs to perform anonymisation of Intrusion Detection Message Exchange Format (IDMEF) XML messages before they are sent to a security operations centre that operates in privacy-preserving mode. The solution increases the scalability of XACML based authorisation significantly, and may be instrumental in implementing federated authorisation and anonymisation based on XACML in several areas, including intrusion detection systems, web services, content management systems and GRID based authentication and authorisation.

Keywords: Privacy Policy, Authorisation, Anonymisation, Caching, XML, IDS, XACML

1. Introduction

The eXtensible Access Control Markup Language (XACML) is an access control policy language that is gaining popularity [1]. It can for example be used together with the Security Assertion Markup Language (SAML) for authorisation of web services. It can also be used for authorisation in federated environments like Shibboleth[1], or for giving users control over their data [2]. Our objective is to use XACML for fine-grained authorisation and anonymisation of IDMEF XML messages from Intrusion Detection Systems (IDS), in order to control what information that can be disseminated to who from an IDS service.

There are also other authorisation languages that could have been considered. For example the Enterprise Privacy Authorisation Language (EPAL) [3] or the Platform for Privacy Preferences (P3P) [4]. P3P is more end-user oriented focusing mainly on web based authorisation. It seems to lack the rich functionality and extensibility of XACML and is perhaps more a supplement than a replacement to XACML for web-based authorisations. XACML can in many respects be considered a superset of EPAL [5]. However EPAL supports obligations, so a similar framework for cache control and anonymisation of XML data may be possible to implement also for this policy language. The main reason for choosing XACML, is that it is a mature OASIS standard [1], that fits well into a Service Oriented Architecture (SOA). Furthermore, XACML has quite broad vendor support compared to EPAL.

However, a limitation with XACML is that the current implementations do not scale well [6]. It is a for example a risk that the central rule processing engine in the Policy Decision Point (PDP) may be a bottleneck for a potentially large amount of authorisation requests from individual XML elements. Another challenge, that has not been solved as far as we are aware of, is how to do fine-grained anonymisation or pseudonymisation of XML documents or messages by using XACML. We propose how this can be mitigated by adding a decision cache as an XACML obligations service that can store decisions based on unique key values.

Our solution is not limited to the domain of IDS services. Fine-grained access control and anonymisation of XML documents backed up by a client-side decision cache may also be useful for GRID services to provide a more scalable authorisation that effectively can delegate simple decisions to a distributed set of decision caches. It can be useful for authorisation and anonymisation of web services, middleware like for example JBoss or even content management systems, in order to ensure that some information deemed sensitive is not distributed via the service.

[*]Corresponding author. Phone: +47 91876897, fax: 37233001 .
Email addresses: nils.ulltveit-moe@uia.no (Nils Ulltveit Moe [*]), vladimir.oleshchuk@uia.no (Vladimir Oleshchuk)
[1]See http://shibboleth.internet2.edu

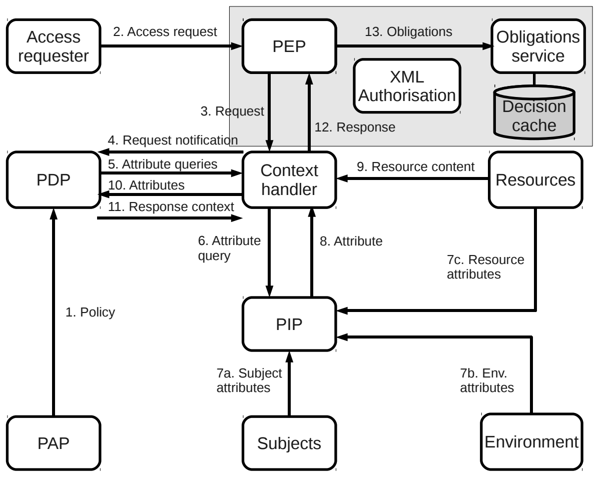

Figure 1: XACML architecture with decision cache.

In that respect, the solution can also be regarded as a simple XACML controlled application level firewall for XML documents.

This paper is organised as follows: The next section gives an introduction to XACML and an overview of the proposed solution. i Section 3 describes the architecture and Section 4 covers the technical solution in more detail. Section 5 shows an example authorisation of XML resources based on the proposed XACML solution including initial authorisation, individual element authorisation request and response and decision cache handling. Section 6 describes the efficiency of the proposed solution. Related work is subsequently discussed in Section 7 and Section 8 concludes the paper and gives some suggestions for further research.

2. Overview of the Proposed Solution

XACML is an access control policy language based on policies written in XML. It uses a model for access control that clearly separates policy decisions in the Policy Decision Point (PDP) from policy enforcement the Policy Enforcement Point (PEP) as shown in Figure 1. The Context Handler and Policy Information Point (PIP) ensure that subjects, resources and other environment attributes can be made available to the PDP when policies are being evaluated. Subjects, resources and environmental attributes can also be passed in via the XACML Request message. We use this approach, since the anonymisation and authorisation service basically is an extension of the PEP.

Our solution implements an XML authorisation service that is integrated with both the PEP and the obligations service. The obligations service furthermore manages the decision cache.

From an architectural and system management perspective, it is preferable to be able to reuse XACML as far as possible for fine-grained authorisation and anonymisation of XML documents and messages. This is viable under the assumption that access control decisions for authorisation or anonymisations can be regarded as final and do not change within a defined time span. This means that an access control decision to publish sensitive material will not be undone or reconsidered under normal circumstances.

Rules for access control policies will in many cases be static, meaning that they are based on some stable conditions. For example rules using fixed strings or rule patterns identifying IP addresses, e-mail addresses or URLs accessed. For static rules, it will be possible to have decision cache entries with infinite expiry time, that only will be ejected from the cache if the cache is invalidated, for example due to an updated authorisation policy. In other cases it may be useful to only grant access for a limited time period before authorisation needs to be renegotiated.

Utilising a decision caching authorisation system also means that cache entries and rules can be made much simpler than the original XACML expressions, however at the expense of using more memory. It can however be expected that the cache has a minimum working set of active authorisations, which means that the decision cache will need at least a certain amount of memory for cache entries in order to operate efficiently. However, if the working set of cached decisions fit into memory, then the load on the XACML rule engine is expected to be tolerable. These assumptions make it viable to use a caching strategy for access control decisions.

3. Architecture

Figure 2 illustrates how the XACML-based anonymising proxy for IDMEF XML reports is implemented. Initially, the Managed Security Service (MSS) providers will be authorised towards the PEP. In this example, two MSS providers are shown: an outsourced first line service that only is allowed to see anonymised IDS alerts and a second line service, possibly run in-house, that can see non-anonymised IDMEF alerts. This initial authorisation opens a secure connection from the anonymiser thread and to the alert database of the MSS provider.

Then the IDS sensors are authorised towards the PEP in order to open a connection from the IDS to a dedicated *Producer* thread in the PEP for each IDS. The *Producer* thread is responsible for copying IDMEF messages to all input queues of authorised anonymisers/proxies. Each *Anonymiser/proxy* thread will then read IDMEF messages and anonymise them according to the XACML policy.

Policy decisions are cached in the decision cache to improve the overall efficiency, so that cached decisions that have not timed out will be reused to save the overhead on XACML requests. Different authorised sessions can then have different anonymisation policies based on security level. For example so that a first line outsourced IDS

Symbol	Description
i	XACML policy number.
j	Decision number.
k	scope parameter number for XACML identifiers.
$a_{i,j}$	The XACML authorisation decision number j by resource policy number i.
$b_{i,j}$	The block marker or pattern used to anonymise the data (optional).
$d_{i,j}$	Decision number j performed by the XACML resource policy number i.
$K_{i,j}$	Unique dictionary key for decision j and policy i.
$l_{i,j}$	Last time this decision cache entry was used.
$p_{i,j}$	Anonymisation policy to perform on the content of r_i for decision j.
R	All resource XPath expressions for XML elements/attributes that need authorisation.
r_i	Resource number i that needs authorisation.
$s_{i,k}$	XPath scope expression that extracts required parameter values for the the XACML policy i.
$t_{i,j}$	The absolute time (UTC) when the cached authorisation decision times out.
$v_{i,j,k}$	Parameter values identified by $s_{i,k}$that are required by the XACML policy i in order to perform decision number j.

Table 1: List of notations

Parameter	Decision cache XACML AttributeId
$b_{i,j}$	$b_{i,j}$ is stored in an *AttributeAssignment* with ID urn:prile:org:resource:i:policy:$function$
r_i	urn:prile:org:resource:i:id
p_i	urn:prile:org:resource:i:policy:$function$ where $function$ =[replace-with|pad-with|...]
$s_{i,k}$	urn:prile:org:resource:i:assertion:k:scope
Δt_i	urn:prile:org:resource:i:cache-timeout (PEP calculates t_i from the current time plus Δt_i)
$v_{i,j,k}$	urn:prile:org:resource:i:assertion:k:value for decision j

Table 2: Mapping of XACML response parameters.

service, that handles the bulk of the alerts, operates with anonymised data; and a second line service, that operates in-house, can have access to the full alerts. This limits the amount of sensitive information that is visible to the outsourced first-line service.

4. Technical Solution

This section performs a more formal analysis of the technical solution. Figure 3 shows an example IDMEF report that matches the XPath expressions used in the case study and Tables 1 and 2 show the formal notations used. The proposed solution uses the initial XACML authorisation request from the data consumer to return an obligation with a list of $n \geq 0$ XPath expressions identifying XML resources $R = \{r_1, r_2, ..., r_n\}$ that require further authorisation. This is shown in Figure 4. The figure shows a successful XACML Response that permits access to the PEP, but with a cache specification sent as an XACML obligation to authorise any XML elements referenced by the XPath expression
/Alert/AdditionalData[@meaning='payload']
with a requirement to also request the element referred to by the XPath expression */Alert/Classification/@ident* from the XML document, and send the requested value as a resource attribute in the XACML request. The PDP can based on this information perform a decision on whether

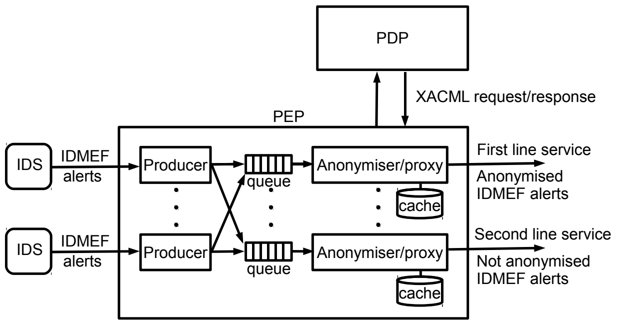

Figure 2: XACML-based IDMEF anonymiser/proxy with decision cache.

the payload for a given type of IDS alert is considered privacy violating or not.

The other resource requires authorisation of all XML elements below the XPath expression
*/Alert/Source/Node/**. The cache specification also requires that */Alert/Source/Node/Address/address* is retrieved from the XML document and passed to the XACML policy for evaluation. Later, this value is also used as part of the cache key for a given document.

The XACML obligations service in the anonymiser/proxy will subsequently perform XACML authorisation requests the first time a new (uncached) decision for a resource element is identified. The XACML response contains an access control decision from the PDP that will be cached for a retention time period as defined in the obligations of the access control decision.

Caching access control decisions require some knowledge about the authorisation policy being used, since checking for a cache hit requires that all relevant parameter values that the access control decision is based upon are known. These parameter values are together with the resource id used as keys when checking whether a cache entry matches the relevant set of parameters in the XML document being checked.

The decision process for XACML authorisation and anonymisation can be considered as a mapping from a resource and a set of parameter values that are required by a given XACML resource policy and to a decision. If this decision is positive, then the decision may have additional obligations, like an obligation to anonymise data or an obligation that expresses authorisation timeout. The parameters required by the system in order to *make a decision* are defined more formally below:

- r_i identifies the set of one or more XML resource(s) to be authorised by the XACML resource policy i, expressed as an XPath expression on the current XML document, for example:
  r_1 =*/Alert/AdditionalData[@meaning='payload']*
  r_2=*/Alert/Source/Node/** (applies to any elements below *node*);

- $s_{i,k}$ are the XPath expressions used to extract required parameters for the the XACML policy i and parameter number k.

- $v_{i,j,k}$ are the parameter values extracted from the XML document by applying the XPath search expression $s_{i,k}$. These parameter values are required by the XACML policy i in order to evaluate decision number j. Sent as XACML resource context parameter number k.

The *decision related* parameters are explained below:

- $a_{i,j}$ is the XACML authorisation, which can be either *Permit* or *Deny*.

- $b_{i,j}$ is the block marker or pattern used to anonymise the data. This parameter is optional, and the default block marker is an empty string if it is not specified.

- $p_{i,j}$ specifies the anonymisation policy to perform on the content matching resource r_i for decision j, which can be one of a set of P predefined anonymisation policies, for example to anonymise by removing or replacing content, anonymise by padding content using a block marker instead of the content (leaves the length of content intact), modify content using regular expression or perform a pseudonymisation policy, for example prefix-preserving pseudonymisation of IP addresses or use an encryption policy.

- $t_{i,j}$ is the absolute time (UTC) when the authorisation decision times out. Different timeout values may be applicable for different authorisations. It is for example natural that authorisations that are based on dynamic variables may need a relatively short timeout period. On the other hand, decisions based on static parameters, like IP address ranges, may not need any timeout value, so the timeout value can be set very large or even infinite. It is then sufficient to have a notification service that can invalidate the policy cache in case the PDP reloads a new policy from the PAP. After $t_{i,j}$ times out, then the cached decision will be discarded the next time the cache entry is used, and a new XACML authorisation will be performed;

- $l_{i,j}$ shows the last time this decision cache entry was used. (Useful for debugging and optimising the Least Recently Used cache.)

With these definitions a decision, denoted by $d_{i,j}$, is represented as a tuple $d_{i,j} = (a_{i,j}, t_{i,j}, l_{i,j}, p_{i,j}, b_{i,j})$ which reflects the j^{th} decision performed by the XACML resource policy number i. The decision cache is implemented as a dictionary where the key $K_{i,j}$ consists of the resource policy number and all n values concatenated i.e.
$i||v_{i,j,1}||v_{i,j,2}||...||v_{i,j,n}$, so that the dictionary indexed on the key returns the cached access decision. The resource policy number i needs to be part of the key to avoid ambiguities between the values, for example that source IP address and destination IP address are being confused for different resource policies.

5. XACML Policy Example

This section provides an example of how the envisaged IDS XACML profile can be used. It does not focus on the authentication part, which is expected to be very similar to existing federated access control solutions using SAML to convey XACML requests [7]. We assume in the following sections that the XML schema namespace *(http://www.w3.org/2001/XMLSchema#)* is denoted by *&xs;*.

In this example, a company considers information about hosts residing on the network 10.0.2.0/24 as sensitive. The company does not want to reveal IP addresses clear-text in the IDS alerts. Furthermore, the payload is considered sensitive for certain classes of IDS alerts, as indicated by the *ident* attribute of the *Classification* element in the IDMEF report. IDMEF alerts from IDSs on this network can for example look like the simplified IDMEF excerpt in Figure 3.

```
<IDMEF-Message>
  <Alert messageid="0c18ec3c-1b2e-11e0-99b2">
    <Source spoofed="unknown"
          interface="wlan0">
      <Node category="unknown">
        <Address category="ipv4-addr">
          <address>10.0.2.2</address>
        </Address>
      </Node>
    </Source>
    <Classification ident="1:5976"
        text="SNMP AgentX/tcp request">
    </Classification>
    <AdditionalData type="byte-string"
                  meaning="payload">
      REhDUEM=
    </AdditionalData>
  </Alert>
</IDMEF-Message>
```

Figure 3: Simplified excerpt of IDMEF message used in the case study.

5.1. Initial Authorisation

The initial XACML request is an ordinary XACML authorisation request to get read access to the Anonymiser/proxy in the PEP, similar to the one described in [8], and is not shown in this article for space reasons. However, the XACML response is shown, to illustrate how the PEP is being made aware of the cache parameter specification necessary manage the decision cache in the form of XACML obligations. The mapping between the notation used in this article and XACML identifiers is shown in Table 2.

The initial authorisation shown in Figure 4 returns a set of XML resource identifiers r_i that consists of XPath expressions that cover authorisation of one or more XML elements in the document. Each XACML response also contains k XPath expressions $s_{i,k}$, that uniquely define the parameters required by the XACML policy to authorise the resources defined by r_i and that will be sent in subsequent XACML resource authorisation requests as resource attributes.

Since an XPath expression may return more than one element, it is then up to the XACML policy to define the attributes so that the cache is kept consistent. The simplest way to do this, is to require that $s_{i,k}$ is defined to return *only a single element* from the XML document instance being authorised. If an assertion XPath expression returns more than one element, and their result is different, then the evaluation of the policy would also potentially be inconsistent. One element may claim access and the other may not. If it is necessary to do conflict resolution, then all individual assertion elements must be passed in to the XACML policy, which defines how the conflict resolution should be done. All XPath expressions from the initial authorisation are precompiled and stored in a two dimensional list indexed by resource number i and scope expression k.

5.2. XML Element Authorisation Request

After the initial authorisation, the XML parser of the Anonymiser/proxy in the PEP will get XML messages (IDMEF alerts) from the queue and start parsing them. The PEP then iterates through all XPath matches for all resources in R. If there is no authorisation cached for the XML resource elements r_i refers to, then the PEP will perform XACML authorisation requests for all non-authorised resources, asking for read access to the resource elements. An example authorisation request for an XML element is shown in Figure 5. The request authorises the subject *soc1@outsourced.example.com* for access to the resource: r_1 =*/Alert/AdditionalData[@meaning='payload']*.

In addition, the XACML request contains additional resource context parameters representing the set of necessary parameters $s_{i,k}$ that are required to evaluate the given security policy by the PDP. Here, the first element of the tuple $s_{i,1}$ =*/Alert/Classification/@ident* refers to the IDMEF Alert classification of the XML message being authorised and $v_{i,1} = 1 : 5976$ refers to the unique identification of the alert class in the XML document being inspected (See Figure 3). The next section describes how the decision cache works for a cache miss. A cache hit, is subsequently described in Section 5.4.

5.3. XML Element Authorisation Response

An accepted XACML response is illustrated in Figure 6. The obligations in XACML responses are mapped as shown in Table 2.

The PEP will then collect all decision parameters $d_{i,j} = (a_{i,j}, t_{i,j}, l_{i,j}, p_{i,j}, b_{i,j})$. All of these except $l_{i,j}$ and $t_{i,j}$ are fetched from the obligations in the XACML response. Then $l_{i,j}$ is set to the current time and $t_{i,j}$ is set to the timeout value $\Delta t_{i,j}$ in the XACML response plus the current time. Subsequently, the anonymisation policy $p_{i,j}$ from the obligations in the XACML response will be applied to the content of all resources matching r_i. This can for example be to anonymise the content by padding it with the block marker *"X"* if $p_{i,j} = pad-with$ and $b_{i,j} = "X"$. The anonymisation policy will then be cached in the dictionary using the resource number and parameter values concatenated as key, i.e. $K_{i,j} = i||v_{i,j,1}||v_{i,j,2}||...||v_{i,j,n}$.

If an authorisation request is *denied*, then the XML message will be discarded, since it is not authorised to be sent to the resource consumer.

A *Deny* authorisation decision can be cached in the same way as a *Permit* decision, however this requires that the XACML response includes an obligation with the necessary parameters for the cache entry, as shown in Table 1. The anonymisation policy $p_{i,j}$ can be omitted in this case, since a *Deny* decision implies that the XML message is dropped. This sequence is not illustrated, since it will be very similar Figure 6, except that the decision is changed

```
<Response>
  <Result ResourceID="PEP">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
    <Obligations>
      <Obligation ObligationId="urn:prile:org:authorize-elements" FulfillOn="Permit">
        <AttributeAssignment AttributeId="urn:prile:org:resource:1:id"
          DataType="&xs;string">/Alert/AdditionalData[@meaning='payload']
        </AttributeAssignment>
        <AttributeAssignment AttributeId="urn:prile:org:resource:1:assertion:1:scope"
          DataType="&xs;string">/Alert/Classification/@ident
        </AttributeAssignment>
        <AttributeAssignment AttributeId="urn:prile:org:resource:2:id"
          DataType="&xs;string">/Alert/Source/Node/*
        </AttributeAssignment>
        <AttributeAssignment AttributeId="urn:prile:org:resource:2:assertion:1:scope"
          DataType="&xs;string">/Alert/Source/Node/Address/address
        </AttributeAssignment>
      </Obligation>
    </Obligations>
  </Result>
</Response>
```

Figure 4: XACML reply to initial authorisation of the IDS-PEP.

from *Permit* to *Deny*, and there will typically only be a cache timeout value as parameter.

5.4. XML Element Authorisation for Cache Hit

Checking for cache hits is performed for all resources matching the pattern r_i after the necessary scope values $v_{i,k}$ have been extracted from the XML document. A cache hit means that there exists a cached decision $d_{i,j}$ for a key $K_{i,j}$ in the decision cache. If the cache has timed out, then entry $d_{i,j}$ is deleted, and a full XACML resource authentication is performed.

Finally, the anonymisation policy $p_{i,j}$ is enforced and the anonymised XML document is sent to the authorised data consumer.

6. Efficiency of the Proposed Solution

The XACML decision cache is implemented in Jython running on Sun Java 6. The Jython interpreter gives a performance overhead, so a native Java implementation can be expected to be somewhat faster, however testing this is left to future work. The implementation uses XimpleWare's Java based Virtual Token Descriptor XML parser (VTD-XML)[2] which has a small memory footprint compared to traditional DOM implementations (1.3-1.5 times the size of the XML document) and has also got a very fast XPath 1.0 implementation.

[2]VTD-XML can be found at http://vtd-xml.sourceforge.net

The experiments are performed using Jython 2.2.1 on a 64 bit machine running Ubuntu with 8 Gb ram and 2.53 GHz Intel Core 2 Duo CPU. The decision cache was limited to 3000 entries, using a Least Recently Used policy for pruning the cache when it runs full. The cache was tested with between one and thirty relatively simple anonymisation policies that performed simple regular expression match for any content.

The LRU class was implemented in Jython based on the *LinkedHashMap* Java class by overriding the *removeEldestEntry()* method. LRU functionality was then achieved by first retrieving and removing the referenced cached entry and then reinserting it at the tail of the linked hash structure. The oldest entry was then automatically removed from the head of the data structure by *LinkedHashMap* when the cache capacity was exceeded.

The experiment consisted of first identifying a set of resources with corresponding scope values that needs to be cached. 30 resources were selected that it would be reasonable to consider anonymising or that it would be reasonable to consider using as a scope variable for that resource. With the exception of payload, which uses the IDS rule classification as scope (as discussed in this paper), the rest of the simple rules tested the same parameter they anonymised, amongst others: *source IP address, destination IP address, source port, destination port* etc. We attempted to stress the cache by including scope variables that referred to the TCP sequence and acknowledgment numbers.

We used simple XACML policies in order to see the

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context:schema:os"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context:schema:os
         http://docs.oasis-open.org/xacml/access_control-xacml-1.0-context-schema-os.xsd">
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
               DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue>soc1@outsourced.example.com</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="&xs;string">
      <AttributeValue>urn:prile:org:resource:1:id</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:prile:org:resource:1:assertion:1:scope"
          DataType="&xs;string">
      <AttributeValue>/alert/classification</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:prile:org:resource:1:assertion:1:value"
          DataType="&xs;string">
      <AttributeValue>1:5976</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="&xs;string">
      <AttributeValue>read</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

Figure 5: XACML request for XML element authorisation.

```
<Response>
  <Result ResourceID="urn:prile:org:resource:1:id">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
    <Obligations>
      <Obligation ObligationId="urn:prile:org:element-restrictions" FulfillOn="Permit">
        <AttributeAssignment AttributeId="urn:prile:org:resource:1:cache-timeout"
          DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration">P1D
        </AttributeAssignment>
        <AttributeAssignment AttributeId="urn:prile:org:resource:10:policy:pad-with"
          DataType="&xs;string">X</AttributeAssignment>
      </Obligation>
    </Obligations>
  </Result>
</Response>
```

Figure 6: XACML reply to successful XML element authorisation.

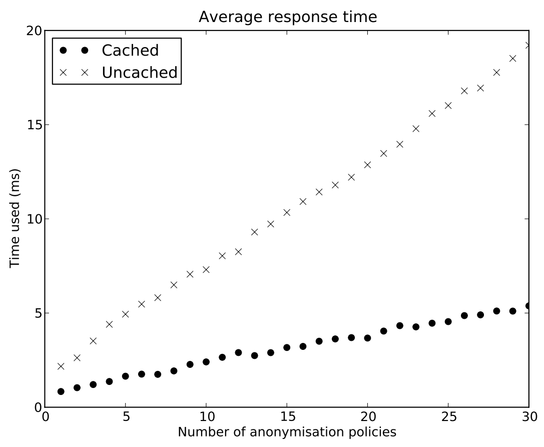

Figure 7: Average response time of decision cache as a function of number of anonymisation policies.

worst case performance of the decision cache compared to not caching decisions.

A simple XACML policy generator was then used to perform a random selection of n out of these 30 resources, and then test the decision cache on 5000 alerts generated by Snort 2.8 using the standard VRT rule set. Traffic was generated by replaying the 1999 KDD cup data set[3]. A problem with this data set, is that it does not give a representative picture of the diversity of attack vectors today and also not the diversity of data seen by a large MSS provider. The cache hit rate (97% for 30 enabled rules with 3000 cache entries in the LRU cache) is therefore probably unrealistically high compared to what can be expected with real data. The experiments still give a representative picture of the cache performance, given that the cache hit rate is high.

Each result presented in Figure 7 is the average of 20 experiments, each anonymising 5000 alerts for a given number of resources n. The experiment was then repeated for $n = (1, 2, ..., 30)$. Using an ensemble of 20 experiments limits the effect of random selection of rules with varying cache hit rates. This makes it possible to better see the underlying trends. Only IDMEF Alert messages was sent to the cache. Heartbeat messages was not processed, since they are not relevant for the anonymisation policy.

Figure 7 shows the average response time of the decision cache as a function of number of anonymisation policies (i.e. number of XML elements being anonymised). There seems to be a linear relationship between the number of anonymisation policies and the time used, as can be expected. Also, the relative cache efficiency (fraction of uncached to cached time used) increases with increasing number of anonymisation policies, from a speedup factor of 2.6-3.0 for less than 5 policies to around 3.5 for 25-30 policies. This shows that the cached solution both performs better in terms of efficiency and scales somewhat better than the non-cached solution with increasing number of anonymisation policies. The speedup factor can be expected to be even larger for more complex XACML policies, as long as the cache hit rate is kept sufficiently high.

30 anonymisation policies is probably sufficient for the IDMEF use case. Most of the remaining IDMEF elements and attributes were either constant or varied between a few values, which means they would fit into the cache without causing any significant additional load on the cache. For 30 anonymisation rules, the decision cache will be able to process up to 185 IDS alerts/s (vs max 52 IDS alerts/s for the non-cached solution). If this is not sufficient, then the architecture can easily be parallellised, for example by adding individual anonymising PEPs for each IDS sensor or even splitting traffic from single IDS sensors.

Memory usage is not a problem for the given experiment since the cache had a hit rate of 97% with only 3000 cache entries. The JVM heap size went down to 130 Mb between each garbage collection, and memory increased slowly after garbage collection, which is another indication that memory usage was not problematic when using the VTD-XML parser[4]. However, more realistic data (for example from a MSS provider) are needed to verify that memory usage is not a problem.

The decision cache is in other words useful for increasing both the performance and scalability of XACML authorisations. This means that it should be viable to perform fine-grained access control of XML elements and attributes in IDMEF alerts from IDS by using an anonymising decision cache.

7. Related Work

This paper extends the simple XACML policy for anonymisation proposed in [8]. The previous paper presented the idea of anonymisation based on an XACML obligations service for course-grained access control of IDMEF messages. This paper extends the solution to provide fine-grained access control of XML messages in general with decision caching support and support for several different anonymisation policies.

There is as far as we are aware of no other similar solutions. However, some other systems cover part of the same functionality. A solution for controlling access to XML documents is proposed in [9]. However, this solution is not based on XACML and it does not support anonymisation policies. An XACML-based privacy-centred access control system is proposed in [2, 10]. This system focuses on credential management to provide users with control over their data. Our solution is different, since it proposes an XACML caching solution with fine-grained access control and anonymisation of data.

[3]KDD Cup 1999 data (DARPA IDS test set) http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[4]This picture was however different for Javas standard DOM implementation, which showed heavy memory allocation/free patterns.

An extension of XACML to improve the performance of decision making processes when dealing with stable conditions is explained in [11]. This solution aims at reducing the time that the Policy Information Point (PIP) uses for accessing remote services like SNMP agents and also the decision making time. Our solution is different, since it aims at performing access control of individual elements and attributes in XML documents using a decision cache based solution.

The BRO IDS [12] supports a way to anonymise the payload of a packet instead of removing the entire payload [13, 12]. There also exists some earlier work on privacy-enhanced host-based IDS systems that pseudonymises audit data and performs analysis on the pseudonymised audit records [14, 15, 16, 17, 18]. However neither of these solutions are based on XACML or provide native authorisation and anonymisation of XML document instances.

8. Conclusions and Future Work

The paper proposes a viable solution for fine-grained XACML authorisation and anonymisation of elements and attributes in XML documents. This allows for central management of authorisation and anonymisation policies for XML documents instead of using a hybrid solution with several different access control solutions or languages.

The decision caching protocol can easily be adapted to other authorisation schemes by choosing a different cache key generation scheme that reflects the authorisation scenario. Caching can then be enabled by adding the timeout parameter as an obligation in order to manage the cached decisions. This opens up a possibility to significantly improve the efficiency and scalability of other XACML based authorisation schemes.

A potential critique of the proposed solution, is that fine-grained access control decisions are delegated from the PDP to the PEP via XACML obligations. This violates the clear interface between policy authorisation and policy enforcement.

Future work involves adding more functionality and if necessary moving time critical parts to Java. It would also be interesting to support the Multiple Resources Profile of XACML in order to process several resources simultaneously by XACML. Last but not least, the anonymising decision cache should be tested under realistic conditions at a MSS provider.

Acknowledgments

This work is funded in part by Telenor Research & Innovation under the contract DR-2009-1.

Bibliography

[1] T. Moses (ed), OASIS eXtensible Access Control Markup Language (XACML) Version 2.0, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf (2005).

[2] C. A. Ardagna, S. D. C. di Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, An XACML-based privacy-centered access control system, in: Proceedings of the first ACM workshop on Information security governance, ACM, Chicago, Illinois, USA, 2009, pp. 49–58.

[3] C. Powers, M. Schunter (ed), Enterprise privacy authorization language (epal 1.2), http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/index.html (2003).

[4] M. Marchiori (ed), The platform for privacy preferences 1.0 specification, http://www.w3.org/TR/P3P (2002).

[5] A. H. Anderson, A comparison of two privacy policy languages, in: Proceedings of the 3rd ACM workshop on Secure web services - SWS '06, Alexandria, Virginia, USA, 2006, p. 53.

[6] A. X. Liu, F. Chen, J. Hwang, T. Xie, T.: XEngine: a fast and scalable XACML policy evaluation engine, Conference on Measurement and Modeling of Computer Systems.

[7] P. Periorellis, Securing Web Services, Idi Global, 2007.

[8] N. Ulltveit-Moe, V. Oleshchuk, Two tiered privacy enhanced intrusion detection system architecture, in: 2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Rende, Italy, 2009, pp. 8–14.

[9] E. Damiani, P. Samarati, S. De Capitani di Vimercati, S. Paraboschi, Controlling access to xml documents, IEEE Internet Computing 5 (2001) 18–28.

[10] C. Ardagna, M. Cremonini, S. D. C. di Vimercati, P. Samarati, A privacy-aware access control system, Journal of Computer Security 16 (4) (2008) 369–397.

[11] R. Laborde, T. Desprats, An extension of xacml to improve the performance of decision making processes when dealing with stable conditions, in: L. Boursas, M. Carlson, W. Hommel, M. Sibilla, K. Wold (Eds.), Systems and Virtualization Management. Standards and New Technologies, Vol. 18 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2008, pp. 13–24.

[12] Lawrence Berkeley National Laboratory, Bro intrusion detection system, http://bro-ids.org.

[13] R. Pang, V. Paxson, A high-level programming environment for packet trace anonymization and transformation, in: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, ACM, Karlsruhe, Germany, 2003, pp. 339–351.

[14] T. Holz, An efficient distributed intrusion detection scheme, in: COMPSAC Workshops, 2004, pp. 39–40.

[15] M. Sobirey, S. Fischer-Hübner, K. Rannenberg, Pseudonymous audit for privacy enhanced intrusion detection, in: Proceedings of the IFIP TC11 13th International Conference on Information Security (SEC'97), 1997, pp. 151–163.

[16] S. Fischer-Hübner, IDA - An Intrusion Detection and Avoidance System (in German), Aachen, Shaker, 2007.

[17] M. Sobirey, B. Richter, H. König, The intrusion detection system AID - architecture and experiences in automated autid trail analysis, in: Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security, 1996, pp. 278–290.

[18] R. Büschkes, D. Kesdogan, Privacy enhanced intrusion detection, in: G. Müller, K. Rannenberg (Eds.), Multilateral Security in Communications, Information Security, Addison Wesley, 1999, pp. 187–204.