Gestion de conditions stables dans XACML : intérêt d'une approche par notification

Romain Laborde, Thierry Desprats

IRIT UMR 5505, Université Paul Sabatier
118 route de Narbonne, 31062 Toulouse cedex 9, France
{laborde, desprats}@irit.fr

RÉSUMÉ. *XACML (eXtensible Access Control Markup Language) est un langage standardisé par OASIS basé sur XML et dédié au contrôle d'accès. Dans ce langage, toute entité concernée par le contrôle d'accès (i.e. sujets, ressources, actions et environnement) est spécifiée par un ensemble d'attributs. Le standard inclut également la description d'une architecture qui explique comment un point de décision de politique (PDP) obtient les attributs nécessaires lorsqu'il évalue la politique pour prendre sa décision d'autorisation.*
Dans cet article nous montrons que cette approche d'acquisition n'est pas adaptée pour des attributs dont le processus de récupération de la valeur est long et dont le changement de cette valeur n'impacte que rarement la politique. Ainsi, nous proposons une amélioration de l'architecture XACML afin d'accélérer le processus de prise de décision dans le cas où le PDP doit traiter ce genre d'attributs.
ABSTRACT. *XACML (eXtensible Access Control Markup Language) is an XML-based language for access control that has been standardized in OASIS. In this language, any entities involved in access control (i.e. users, resources, actions and environment) are specified by a set of attributes. This specification also includes the description of an architecture that explains how the policy decision point (PDP) retrieves the needed attributes values when it evaluates the policy to take its authorization decision.*
In this paper, we show this approach for getting the attributes values is a bottleneck to the performance of the authorization decision-making-process for attributes whose process for retrieving the value is long and the changing of its value doesn't impact the policy frequently. Thus, we propose an improvement of the XACML architecture in order to accelerate the decision-making-process when PDP has to treat such kind of attributes.

.

MOTS-CLÉS : *Politique de contrôle d'accès, XACML, modèle ABAC, SNMP*
KEYWORDS: *Access control Policy, XACML, ABAC model, SNMP*

1. Introduction

Les ressources et les services sont de plus en plus complexes et hétérogènes. Les utilisateurs de ceux-ci sont également variables et appartiennent parfois à plusieurs organisations. Pour considérer cette réalité, les modèles et les solutions de contrôle d'accès ont également évolué. Les modèles de contrôle d'accès traditionnels basés sur l'identité (Harrison *et al.,* 1976) s'avèrent limités en flexibilité. Ils sont remplacés par de nouveaux modèles tel que RBAC (Role-Based Access Control) (Ferrariolo *et al.,* 2001) et ses dérivés ou plus récemment par ABAC (Attribute Based Access Control) (Wang *et al.,* 2004). Contrairement à IBAC et RBAC, le modèle ABAC permet de définir des permissions en ne se basant que sur des attributs. Un attribut est une caractéristique, pertinente en termes de sécurité, associée à un sujet, à une action, à une ressource ou encore à l'environnement. Cette approche dote les solutions de contrôle d'accès de davantage de flexibilité et d'extensibilité. Elle est d'autant plus adéquate aux besoins des systèmes ouverts, distribués où l'identité des utilisateurs n'est plus l'unique caractéristique prise en compte pour le contrôle d'accès

Standardisé par OASIS, XACML (eXtensible Access Control Markup Language) est un langage basé sur XML qui est dédié au contrôle d'accès (Oasis, 2005). Il permet l'expression de politiques selon une approche ABAC. La spécification de XACML inclut également la description d'une architecture et celle des modalités de récupération des attributs lors du processus de prise de décision. Ces modalités s'appuient sur un patron d'interaction de type requête/réponse synchrone. Cette approche peut s'avérer peu performante dans les cas où le processus de récupération de la valeur d'un attribut est long. De plus, cela est accru si le changement de valeur de cet attribut n'influence que peu fréquemment le résultat de l'évaluation de la politique. On appellera condition stable, une condition d'une politique dont la valeur change rarement.

Nous proposons d'améliorer l'architecture XACML afin d'accélérer le processus de prise de décision lors de sa phase d'évaluation des attributs. Notre approche repose sur deux points : 1) supprimer les conditions stables de la politique évaluée 2) ajouter la possibilité à XACML d'adopter un comportement guidé par l'occurrence d'évènements. L'idée est de tirer profit de mécanismes de notification asynchrones pour informer l'environnement XACML, de façon opportune, que la valeur d'un attribut impliqué dans une condition stable a changé de telle manière que la valeur de qu'une condition stable a également changé.

L'article est organisé comme suit. La section 2 présente le langage et l'architecture XACML. La section 3 décrit un scénario propice à une amélioration des performances puis s'attache à caractériser les conditions stables. Les sections 4 et 5 présentent les améliorations proposées à l'architecture XACML pour traiter les conditions stables ainsi que des éléments de mise en œuvre pour le scénario. Enfin la section 6 décrit des travaux connexes alors que la section 7 conclue le papier.

2. Le standard XACML 2.0

XACML (eXtensible Access Control Markup Language) est d'abord un langage basé sur XML dédié au contrôle d'accès. Il s'agit à la fois d'un langage de politique de contrôle d'accès basé sur les attributs et d'un langage protocolaire de type requêtes/réponses. De plus, la spécification fournit une architecture qui définit différentes entités impliquées dans le processus de prise de décision d'une autorisation d'accès.

2.1. *Le langage de politiques*

Le langage de politique XACML est utilisé pour décrire les exigences générales de contrôle d'accès en termes de contraintes sur des attributs. Un attribut peut être n'importe qu'elle caractéristiques d'un sujet, d'une action, d'une ressource ou de l'environnement dans lequel la requête d'accès est produite. Le fait de considérer les attributs rend le langage très flexible. De plus, XACML présente des points d'extension standards pour définir de nouveaux types de données, des fonctions additionnelles, des combinaisons de logiques, etc.

A la racine de toute politique XACML, il y a une politique ou un ensemble de politiques. Un ensemble de politiques est une sorte de conteneur qui peut héberger d'autres politiques ou ensembles de politiques mais également des références vers des politiques situées sur des sites distants. Une politique est ici une politique de contrôle d'accès unique exprimée par un ensemble de règles.

Puisqu'un ensemble de politiques ou une politique elle-même peut contenir de multiples politiques ou règles, chacune d'entre elles pouvant amener à différentes décisions de contrôle d'accès, XACML propose des moyens de réconciliation des différentes décisions prises. Un ensemble extensible de six algorithmes de combinaison est défini.

Afin de rendre plus efficace la recherche d'une règle qui doit s'appliquer à une requête donnée, XACML a introduit la notion de « cible » (target). Une cible est un ensemble d'exigences simples sur le sujet, la ressource et l'action qui sont à considérer dans un ensemble de politiques, une politique ou une règle. Les cibles utilisent des fonctions booléennes simples qui sont rapides à évaluer. Par exemple, une cible peut restreindre la portée d'une politique de sécurité seulement au service FTP. Derrière ce concept, l'idée est de trouver très rapidement la règle de politique à évaluer dans le cas de politiques ou d'ensembles de politiques complexes.

Une fois que la requête correspond aux cibles d'un ensemble de politiques ainsi qu'aux cibles d'une des règles de l'une de ses politiques, la règle est évaluée. La section « condition » d'une règle de politique en constitue le cœur. Comme pour les cibles, il s'agit d'expressions booléennes. Cependant, les conditions peuvent inclure

des expressions plus complexes sur les attributs respectifs des sujets, des ressources, des actions et de l'environnement.

Finalement, la décision retournée peut être : 1) *non applicable* si aucun ensemble de politique ou politique ne s'applique 2) *indéterminée* dans le cas d'une erreur ou d'un attribut manquant 3) *permis* ou *interdit* tel que spécifié dans la règle appliquée. Au-delà de la décision d'autorisation, la réponse peut également contenir une section d'obligations.

2.2. *L'architecture de gestion*

La spécification XACML propose également une architecture de gestion qui est décrite au moyen d'un modèle de flots de données. Une version simplifiée de ce modèle est représentée dans la Figure 1.

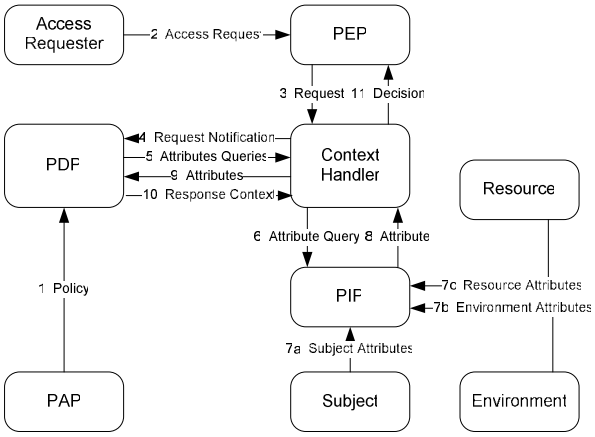

Figure 1. *Modèle simplifié du flot de données XACML*

Ce modèle opère selon les étapes suivantes.

1. les points d'administration des politiques (PAP – Policy Administration Point) rédigent les politiques et les rendent disponibles au point de décision de politiques (PDP – Policy Decision Point). Ces politiques ou ensemble de politiques représentent une politique de contrôle d'accès complète pour une cible spécifique.
2. Les quémandeurs d'accès envoient leur requête d'accès au point de mise en exécution des politiques (PEP - Policy Enforcement Point).
3. Le PEP envoie la requête dans le format natif de l'application permettant d'accéder à la ressource au gestionnaire de contextes (CH – Context Handler) en y incluant optionnellement les attributs des sujets, des ressources, des actions et de l'environnement.

4. Le CH construit alors une requête au format du standard XACML et l'envoie au PDP.
5. Celui-ci demande si nécessaire au CH des attributs supplémentaires relatifs aux sujets, aux ressources, aux actions et/ou à l'environnement.
6. A son tour, le CH demande les attributs au point d'information de politiques (PIP – Policy Information Point).
7. Le PIP obtient les valeurs des attributs.
8. Il retourne les attributs demandés au CH.
9. Celui-ci les retourne à son tour au PDP. Le PDP procède alors à l'évaluation de la politique.
10. Le PDP retourne la réponse au format XACML au CH (y compris la décision d'autorisation).
11. Le CH traduit la réponse au format natif utilisé par le PEP avant de l'envoyer à ce dernier.

La manière dont le PDP obtient les valeurs des attributs lorsqu'il évalue la politique peut diminuer sérieusement la performance du processus de prise de décision. Nous examinons dans la suite le cas des conditions stables.

3. Conditions stables

Une *condition stable* est une expression dont l'évaluation retourne toujours le même résultat durant une période donnée qui est considérée comme étant longue.

3.1. *Illustration*

Dans cet exemple illustré par la Figure 2, nous considérons un serveur FTP dont l'accès est régulé selon une solution basée sur XACML.

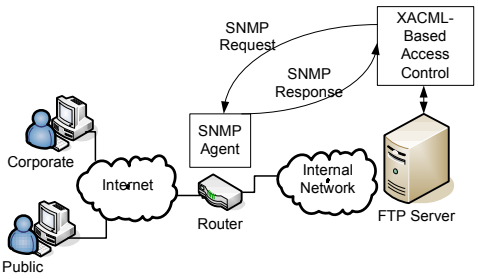

Figure 2. *Architecture de l'exemple*

Le serveur FTP maintient deux répertoires : l'un privé et l'autre public. Le répertoire privé contient des fichiers confidentiels concernant l'organisation. A l'opposé, les fichiers stockés dans le répertoire public sont accessibles à n'importe qui. La politique édicte que les utilisateurs qui ont un rôle « corporate » peuvent

accéder à tout fichier du répertoire privé. Les fichiers du répertoire public sont accessibles à n'importe qui à moins que le réseau soit trop chargé, ce qui signifie que le taux d'usage de la bande passante du routeur frontière est supérieur à 60%. Cette dernière mesure permet de favoriser la protection du réseau d'une situation de congestion. Elle vise à donner l'assurance aux personnes de l'organisation d'un accès permanent aux fichiers du répertoire privé. La spécification XACML de cette politique est définie dans la Figure 3.

```
<Policy PolicyId="ExamplePolicy"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Target>
   <Subjects> <AnySubject/> </Subjects>
   <Resources> <AnyResource/> </Resources>
   <Actions> <AnyAction/> </Actions>
  </Target>
  <Rule RuleId="CorporateAccess" Effect="Permit">
   <Description>
        This rule allows users with role corporate to access the directory "private".
   </Description>
   <Target>
     <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                        corporate
                </AttributeValue>
                <SubjectAttributeDesignator
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        AttributeId="role"/>
          </SubjectMatch>
        </Subject>
     </Subjects>
     <Resources>
      <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                ftp://ftp.example.com/private/.*
        </AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
     </Resource>
     </Resources>
     <Actions> <AnyAction/> </Actions>
   </Target>
  <Rule RuleId="PublicAccess" Effect="Permit">
   <Description>
        This rule allows any users corporate to access the directory "public" unless the bandwidth
        usage rate of the edge router is greater than 60%.
   </Description>
   <Target>
     <Subjects>
        <Subjects> <AnySubject/> </Subjects>
     <Resources>
      <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                ftp://ftp.example.com/public/.*
        </AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
     </Resource>
     </Resources>
     <Actions> <AnyAction/> </Actions>
   </Target>
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-less-than">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
     <EnvironmentAttributeDesignator
                DataType="http://www.w3.org/2001/XMLSchema#double"
                AttributeId="http://www.irit.fr/recherches/SIERA/xacml/attribute#usedBW"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">60.0</AttributeValue>
  </Condition>
  </Rule>
  <Rule RuleId="DefaultRule" Effect="Deny"/>
</Policy>
```

Figure 3. *Politique XACML pour l'exemple*

Le routeur frontière héberge un agent SNMP qui est capable de fournir le taux de bande passante. Dans un souci de simplification de notre exemple, nous considérons qu'il existe un objet de gestion capable de contenir cette valeur. Actuellement cela n'est toujours le cas et les MIBs (Management Information Bases) standards contraignent un gestionnaire SNMP à envoyer plusieurs requêtes afin d'obtenir cette valeur (Cisco, 2005).

En considérant principalement les étapes 5 à 10 du modèle de flot de données de XACML, il est notoire qu'à chaque fois qu'un utilisateur veut accéder à un des fichiers du répertoire public, le CH doit solliciter le PIP pour obtenir la valeur du taux d'utilisation de la bande passante du routeur. Le PIP qui joue le rôle d'un gestionnaire SNMP récupère cette valeur par envoi d'un message SNMP GET REQUEST. Lorsque l'agent SNMP reçoit cette requête, il examine sa MIB et retourne la valeur via un message SNMP RESPONSE. Ensuite, le PIP peut répondre au CH qui peut alors fournir l'attribut au PDP. Celui-ci peut désormais évaluer la politique.

Ce processus est consommateur à la fois de temps et de bande passante réseau. De plus, le fait que le réseau soit chargé doit normalement revêtir un caractère exceptionnel. Ainsi, l'évaluation systématique du taux d'utilisation de la bande passante apparaît inutile. La section suivante analyse les caractéristiques de ce genre d'attributs.

3.2. *Caractérisation d'une condition stable*

Comme l'a souligné l'exemple précédent, ce n'est pas nécessaire d'évaluer systématiquement que le taux d'utilisation de la bande passante est inférieur à 60% car cette situation est exceptionnelle. On appelle *conditions stables* ce type de contraintes.

Dans notre exemple, la condition « le taux d'utilisation de la bande passante est inférieur à 60% » est presque toujours vraie. Un autre exemple, davantage commun, pourrait être « l'heure actuelle doit être comprise entre 7:00 et 19:00 ». Cet exemple est moins démonstratif car le processus de collecte de la valeur est plus rapide : la valeur de l'heure actuelle étant récupérée grâce à un processus s'exécutant sur la même machine que le système XACML. Mais de façon analogue au cas de la bande passante, cette expression retourne toujours vrai entre 7:00 et 19:00 et toujours faux entre 19:00 et 7:00.

Comment identifier une condition stable ? Il est difficile de déterminer si une condition est stable ou pas. En effet, la stabilité d'une condition ne dépend pas seulement des attributs. Dans l'exemple précédent, la bande passante varie en permanence : la propriété de stabilité est due à la variance de l'attribut, à l'opérateur d'infériorité stricte et à la valeur de la constante "60%". La stabilité ne peut être déterminée que d'après l'expérience des administrateurs ou par analyse de l'historique des évaluations. On appelle attributs de conditions stables, des attributs qui sont en arguments de conditions stables.

4. Evolution proposée

L'évaluation des conditions stables s'avérant inutile, nous proposons d'enrichir le cadre XACML de façon à ce que les conditions stables n'y soient considérées que seulement si nécessaire. Toutefois, nous voulons aussi que le système XACML enrichi demeure compatible avec la spécification originelle de XACML.

Notre idée est de : 1) supprimer les conditions stables des politiques évaluées par le PDP 2) modifier cette politique selon les changements des valeurs des conditions stables..

Soit *R* une règle de politique qui comprend une condition stable *C*. Soit *R-True* (resp. *R-False*), la règle *R* sans *C* lorsque *C* retourne Vrai (resp. Faux). Par exemple, la règle PublicAccess de la Figure 3 sera dédoublée en la règle PublicAccess-True (Figure 4-a) lorsque le réseau est opérationnel et la règle PublicAccess-False (Figure 4-b) lorsque le réseau est congestionné.

```
<Rule RuleIc='PublicAccess-True' Effect='Permit'>
 <Target>
  <Subjects> <AnySubject/> </Subjects>
  <Resources>
   <Resource>
   <ResourceMatch MatchId='urn oasis names tc xacml ´ C function regexp-string-match'>
    <AttributeValue DataType='http://www w3 org/2001/XMLSchema#string'>
                ftp://ftp example com/public/.*
    </AttributeValue>
    <ResourceAttributeDesignator DataType='http://www w3 org/2001/XMLSchema#string'
        AttributeId='urn oasis names tc xacml ´ C resource resource-ic"/>
   </ResourceMatch>
   </Resource>
  </Resources>
  <Actions> <AnyAction/> </Actions>
 </Target>
</Rule>
```

Figure 4-a. *Règle PublicAccess-True*

```
<Rule RuleIc='PublicAccess-False' Effect='Deny'>
 <Target>
  <Subjects> <AnySubject/> </Subjects>
  <Resources>
   <Resource>
   <ResourceMatch MatchId='urn oasis names tc xacml ´ C function regexp-string-match'>
    <AttributeValue DataType='http://www w3 org/2001/XMLSchema#string'>
                ftp://ftp example com/public/.*
    </AttributeValue>
    <ResourceAttributeDesignator DataType='http://www w3 org/2001/XMLSchema#string'
        AttributeId='urn oasis names tc xacml ´ C resource resource-ic"/>
   </ResourceMatch>
   </Resource>
  </Resources>
  <Actions> <AnyAction/> </Actions>
 </Target>
</Rule>
```

Figure 4-b. *Règle PublicAccess-False*

Pour permettre au PDP de commuter de *R-True* vers *R-False* et vice-versa, nous avons doté le système XACML de la capacité à adopter un comportement piloté par des évènements. L'idée est de tirer profit des mécanismes de notifications asynchrones pour traiter efficacement la gestion des conditions stables.

D'abord, nous avons donné au PIP la possibilité d'être notifié par des fournisseurs de valeurs d'attributs d'environnement (Figure 5). Il peut ainsi recevoir des messages l'informant que la valeur d'un attribut d'une condition stable a changé de telle sorte que la valeur de la condition stable associée a également changée.

En réaction à cet évènement, le PIP va répercuter le changement de la condition stable à l'intérieur du cadre XACML en alertant le PAP, entité responsable de l'administration des politiques.

```
<xs element name="NotificationMessage' type="NotificationMessageType"/>
<xs complexType name=" NotificationMessageType ">
<xs attribute name="StableConditionID' type="xs anyURI' use="required"/>
<xs attribute name="Value' type="xs string' use="required"/>
</xs complexType>
```

Figure 5. *Spécification d'un message de notification*

Le PAP doit être capable de modifier dynamiquement la(es) règle(s) concernée(s) par ce changement (*R-True* vers *R-False* ou vice-versa). Pour cela, le PAP stocke à la fois *R-True* et *R-False*. Il peut aussi maintenir un fichier de configuration qui lui indique, pour chaque message de notification, quelle(s) est(sont) la(es) règle(s) à changer.

Cette nouvelle architecture permet au PDP d'utiliser soit *R-True* soit *R-False* au moment de l'évaluation de la politique sans avoir à évaluer *C*, et a fortiori sans avoir à collecter les valeurs des attributs dans *C* comme il aurait été indispensable de le faire dans le cas de l'évaluation classique de *R*.

La Figure 6 montre que les impacts de notre proposition dans le cadre standard XACML sont limités. Les comportements des entités classiques ainsi que les interactions entre celles-ci ne sont pas modifiés. Les comportements du PAP et du PIP sont enrichis sans toutefois empêcher leurs activités classiques standards. Les rôles de gestion des entités tels qu'énoncés par le standard OASIS sont conservés. Le PIP agit toujours en tant que collecteur d'informations et le PAP en tant qu'administrateur de politiques.

Initialement le PAP adresse au PDP une politique qui inclut soit *R-True* soit *R-False* selon ce qu'est la valeur lorsque l'état du système est opérationnel. Le PIP a enregistré auprès d'un fournisseur d'attribut(s) d'environnement (EAP – Environment Attribute Provider) de confiance son intérêt pour être notifié lorsque la valeur de *C* change. Le PIP est mis à l'écoute de messages de notification relatifs provenant de l'EAP. Lorsqu'il reçoit une telle notification, le PIP construit un message spécifique qui mentionne la condition *C* ainsi que sa valeur actuelle (Figure 5). Ce message est ensuite envoyé au PAP. Ce dernier reconnaît la valeur de la condition. Si celle-ci est à *Faux*, il change la politique courante en utilisant la règle *R-False*. Si c'est *Vrai*, c'est la règle *R-True* qui est chargée. On notera que pendant ce processus, aucune interaction asynchrone ne s'est produite entre le système

XACML et l'EAP concerné par la condition stable. Le processus d'évaluation qui concerne des conditions non stables s'opère de manière standard.

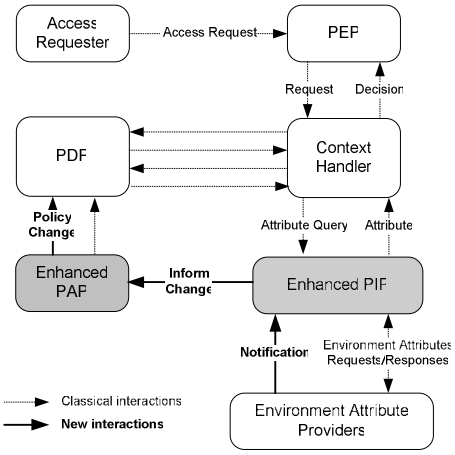

Figure 6. *Architecture proposée*

5. Exemple de mise en œuvre

Nous présentons (Figure 7) un exemple d'implémentation que nous développons comme preuve de concept. Cette implémentation se limite au scénario décrit dans la section 3. Nous utilisons NET-SNMP (Net-SNMP) pour manipuler les notifications asynchrones SNMP et l'implémentation de XACML proposée par Sun (Sun-XACML).

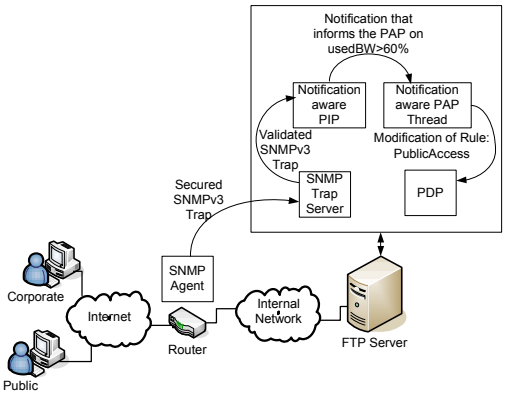

Figure 7. *Exemple de mise en oeuvre*

L'agent SNMP, qui s'exécute sur le routeur en bordure, envoie des traps SNMPv3 au processus démon snmptrapd (Net-SNMP) qui opère sur le serveur FTP.

Nous avons choisi d'utiliser SNMPv3 car cette version du protocole fournit des mécanismes d'authentification, d'intégrité, de confidentialité et d'anti-rejeu (Blumenthal *et al.,* 2002). L'agent SNMP envoie alternativement des notifications qui indiquent que le taux d'utilisation de la bande passante est supérieur à 60% (OID: .iso.org.dod.internet.experimental.siera.bw.greater60) et d'autres qui signalent que ce taux est inférieur à 60% (OID: .iso.org.dod.internet.experimental .siera.bw.less60). Lorsque ces notifications sont reçues par le démon snmptrapd, elles sont d'abord validées (intégrité et confidentialité) pour être ensuite relayées au PIP. Le PIP opère ensuite une traduction. A partir de l'OID de la condition stable qui change, il construit le message de notification qui utilise les URNs comme système de nommage. Enfin, le PAP modifie la règle d'accès au répertoire public. Lorsque la notification reçue est <NotificationMessage StableConditionID=urn:irit:siera:notification:name:bwlessthan60 value="false"/>, la nouvelle règle chargée interdit l'accès au répertoire public, c'est-à-dire, celle définie par la Figure 4-b. Lorsque la notification reçue est <NotificationMessage StableConditionID=urn:irit:siera:notification:name:bwlessthan60 value="true"/>, la règle qui garantit l'accès est chargée (Figure 4-a).

6. Travaux relatifs

Les solutions d'infrastructure de gestion de privilèges (PMI – Privilege Management Infrastructure) proposent deux manières pour collecter des attributs (par exemple (Chadwick *et al.,* 2006) pour PERMIS et (Thompson *et al.,* 2003) pour Akenti) : les modèles push et pull. Leurs architectures diffèrent légèrement de celle de XACML car il n'y a ni CH ni PIP explicitement mentionnés. Ainsi, le PEP dialogue directement avec le PDP. Dans le modèle push, le PEP fournit au PDP tous les attributs. Par exemple, le PEP envoie un certificat d'attributs avec le nom (DN – Distinguished Name) de l'utilisateur et de ses rôles. Dans le modèle pull, le PEP ne fournit que certains attributs. Le PDP doit compléter la liste des attributs nécessaires. Dans ce cas, le PEP envoie seulement le DN de l'utilisateur et le PDP obtient à partir de ce DN le certificat d'attributs dans un répertoire basé sur LDAP. Les attributs d'environnement se limitent à la date et à l'heure.

D'importants travaux de recherche ont été entrepris pour considérer les exigences spécifiques des environnements sensibles au contexte (Context-sensitive) comme l'informatique mobile et pervasive, les systèmes d'intelligence ambiante ou les « smart spaces ». Ces travaux se concentrent sur les propositions d'architectures et/ou de canevas dont la caractéristique principale est d'être « conscients du contexte ». Cela signifie que c'est le contexte qui pilote le comportement du système.

Inspirés par le modèle push des PMIs, (Al Muthadi *et al.,* 2003) ont proposé un « Context Provider » qui est capable d'obtenir de l'information contextuelle à partir de capteurs ou d'autres sources de données. Un moteur d'inférence constitue le cœur de leur système de sécurité conscient du contexte. Celui-ci peut soit interroger un « Context Provider » pour obtenir des valeurs du contexte, soit lui demander d'être notifié lorsqu'une condition change. Cependant, les auteurs n'explicitent ni quand ni

pourquoi utiliser des messages requêtes/réponses ou des messages de notification. D'autre part, aucun mécanisme n'améliore le processus de prise de décision. La même approche est adoptée par (Wullens *et al.,* 2004) dans laquelle un « Dynamic Context Service » constitue une entité de confiance utilisée pour acquérir de l'information contextuelle soit directement, soit par l'intermédiaire d'une tierce partie. Une politique pilote la fréquence à laquelle le contexte est acquis et mis à jour. Elle définit également des seuils servant au déclenchement de notification vers le cœur du système. Par contre, quand une condition change, toute la politique est re-évaluée ce qui peut constituer un goulot d'étranglement pour les environnements traitant beaucoup de requêtes.

(Covington *et al.,* 2002) ont proposé dans leurs premiers travaux un modèle analogue au modèle pull des PMIs appelé « Generalized RBAC ». Ce modèle rajoute des rôles d'environnement qui peuvent être activés ou pas. Des techniques de cache de valeurs sont utilisées et sont combinées à d'autres mécanismes visant à évaluer leur fraîcheur. Cette approche améliore les performances du processus de prise de décision puisque les rôles d'environnement désactivés ne sont pas considérés lors de l'évaluation. Cependant, les techniques de cache ne peuvent pas être utilisées dans le cas des conditions stables puisque dès que celles-ci changent, le système doit en être immédiatement informé. Dans (Covington *et al.,* 2006), les auteurs ont proposé le modèle Contextual ABAC (CABAC) qui adapte les principes de GRBAC au modèle ABAC. Selon la même approche, les attributs d'environnement peuvent être activés ou désactivés. Cette proposition se limite au modèle conceptuel et ne traite pas des problèmes d'implémentation. Bien que ces auteurs soulignent le fait que certains attributs nécessitent d'être évalués à chaque fois alors que d'autres sont davantage sujets à des conditions qui ne changent pas durant une même session, aucune argumentation claire n'est développée sur ce sujet.

7. Conclusion

Nous avons montré dans cet article que les tous les attributs ne devraient pas être obtenus de la même façon. Nous avons défini ce qu'étaient des conditions stables et les raisons pour lesquelles elles ne devraient pas être traitées par les systèmes XACML comme les conditions usuelles. Nous avons proposé une architecture pour améliorer les performances du processus de prise de décision lorsque des conditions stables y sont traitées. Cette architecture reste conforme et compatible à la spécification XACML existante. Enfin, nous avons présenté une mise en œuvre qui s'appuie sur l'implémentation XACML de Sun et sur Net-SNMP.

Malgré tout, notre solution présente encore un inconvénient : l'administrateur doit connaître quelles sont parmi toutes les conditions celles qui peuvent être qualifiées de stables ; il doit également configurer l'EAP, le PIP et le PAP. Cela rend plus difficile la spécification d'une politique.

Notre futur travail va se focaliser sur l'automatisation de ce processus. L'administrateur spécifiera une politique sans tenir compte de l'existence de

conditions stables. Le processus de prise de décision observera les résultats de chaque évaluation des conditions exprimées dans les règles. Il pourra ainsi détecter automatiquement la stabilité de certaines conditions. Dès qu'une condition stable sera décelée, un processus configurera l'EAP, le PIP et le PAP afin d'optimiser à l'avenir le traitement de cette condition. Ces travaux devraient permettre à un cadre XACML d'endosser un comportement d'auto-optimisation, une des propriétés de l'informatique autonomique telle que définie dans (Kephart *et al.,* 2003).

8. Bibliographie

Harrison M.A., Ruzzo W.L., Ullman J.D. "Protection in Operating Systems", *Communication of the ACM*, 1976

Ferraiolo D.F, Sandhu R., Gavrila S., Kuhn D.R., Chandramouli R. "Proposed NIST Standard for Role-Based Access Control", *ACM TISSEC*, 4(3):222--274, 2001

Wang L., Wijesekera D., Jajodia S. "A Logic-based Framework for Attribute based Access Control", *2nd ACM Workshop on FMSE*, 2004.

OASIS "eXtensible Access Control Markup Language (XACML) version 2.0", February 2005. URL: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.

Cisco "How To Calculate Bandwidth Utilization Using SNMP", CISCO TN 8141, 2005

Net-SNMP web site, URL: http://net-snmp.sourceforge.net/

Sun's XACML implementation web site, URL: http://sunxacml.sourceforge.net/

Blumenthal U., Wijnen B. "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", *IETF RFC 3414*, 2002.

Chadwick D., Zhao G., Otenko S., Laborde R., Su L., Nguyen T-A. "Building a Modular Authorisation Infrastructure", *The UK e-Science All Hands Meeting*, 2006.

Thompson M. R., Essiari A., Mudumbai S., "Certificate-based Authorization Policy in a PKI Environment", *ACM TISSEC*, Vol. 6, N° 4, 2003.

Al-Muhtadi J., Ranganathan A., Campbell R., Mickunas M. D. "Cerberus: A Context-Aware Scheme for Smart Spaces", *1st IEEE PerCom'03*.

Wullens C., Looi M., Clark A. "Towards Context-aawre Security : An Authorization for Intranet Environments", *2nd IEEE PerComW'04*

Convington M.J., Fogla P., Zhan Z., Ahamad M. "A Context-aware Security Architecture for Engineering Applications", *18th* ACSAC, 2002

Convington M.J., Sastry M.R. "A Contextual Attribute –Based Access Control Model", *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, LNCS 4278

Kephart J. O. ,Chess D. M., The Vision of Autonomic Computing, *Computer*, v.36 n.1, p.41-50, 2003