XML文書のアクセス制御

岩井原　瑞穂

XML文書による情報共有では，課金処理や機密保護，個人情報保護など多様な目的に応じて，文書やその一部を許可された利用者にのみ読み書きを許すというアクセス制御が重要である．文書の細粒度単位制御を基本とする種々のアクセス制御モデルや標準案，実装技術ならびにアクセス制御を用いた応用について解説する．

キーワード：XML，アクセス制御，データベース，情報セキュリティ，アルゴリズム

1. はじめに

XML文書はネットワークにおける情報交換のための共通言語としての地位を確立しつつあり，電子商取引や電子政府，企業内情報システムなど，個人のプライバシを含む情報や企業秘密がXML文書に格納されることも多い．情報を保護するためには，アクセス権限を付与された人のみによるアクセスを行うというアクセス制御が重要になる．XMLデータベースでは，XML文書の持つ構造の多様性や，情報交換のための言語という特質を考慮したアクセス制御が必要になる．またXMLが持つネットワークとの親和性を生かして，分散環境でアクセス制御や認証などの制御情報を交換するための言語としても，XMLは用いられている．

XMLデータベースにおけるアクセス制御の特徴として，XMLの構造的な柔軟性への対応が挙げられる．XML文書を一つのテキストファイルとして扱う場合には，OSのファイルシステムが提供するパーミッションなどの機能により単純なアクセス制御を行うことができる．しかし，例えば，顧客リストにおいて，年収は顧客対応係員には見せないようにするといった，XML文書の細かい意味構造によるアクセス制御はファイル単位では行えない．このため，XML文書の要素，属性，部分木といった**細粒度**のアクセス制御が必要になる．関係データベースにおいては表の行や列，値単位でアクセス制御を行うことに相当する．XMLではさらに文書構造の柔軟性に対応する必要がある．本稿では，**ノード**という用語でXML文書の要素と属性の両方を指すものとする．XML文書のアクセス制御ルールでは，制御対象のノードあるいは部分木を指定するためにXPathを用いることが多い．アクセス制御ルールが多数存在する状況では，ルールの間で衝突が起きるため，権限判定の最終的な決定を行う方式（衝突解消方式）をあらかじめ与えておくことが必要である．

本稿で用いるXMLに関する基本的な概念については，本特集号の文献[1]を，またXPathについては文献[16]を参照されたい．またXPathのパス式を単にパス式と呼ぶことにする．

本稿では，XMLのアクセス制御ルールの基本モデルおよび衝突解消方式について節2で述べる．また，アクセス制御の標準としてOASISにより承認されているXACML[17]を節3で述べる．XML文書のアクセス制御を効率良く処理するためには，大量のルールの冗長性を除いたり，XMLデータベースへの問い合わせにおいて，アクセス不許可の部分を問い合わせ結果から効率良く除去する技術，さらにはルールを容易に定義できるビューなどの技術が必要であり，XMLデータベースの他の機能とアクセス制御機能を統合してゆくことも必要である．節4ではこのようなXMLのアクセス制御に関連した研究動向を紹介する．

2. 基本的なXMLアクセス制御モデル

本節では，これまでに提案されているXMLアクセス制御モデルについて，ほぼ共通している基本的構造を述べる[2, 7, 11, 17]．

2.1 アクセス制御ルール

XML文書のアクセス制御の基本的モデルについて述べる．アクセス制御ルール（以降，ルール）とは，次の組である．

(*subject*, *object*, *action*, *decision*, *options*)

いわいはら　みずほ
京都大学 大学院情報学研究科
〒606-8501 京都市左京区吉田本町

アクセス制御ルールをアクセス制御ポリシと呼ぶことがあり，またアクセス制御ルールの集合をアクセス制御ポリシと呼ぶ文献もあるが，本稿では後者に従うようにする．

subject とは，アクセスする主体のことであり，アクセス権限を与える対象である．subject の指定方法にはいくつかあり，(1)ユーザ名，(2)ユーザグループ名，(3)ロール名，(4)サービス名などを用いることが考えられる．(3)のロール名とは，role-based access control [15]に基づいており，ロール（役割）ごとに権限を与え，ロールとユーザの対応関係を別に管理することにより，ユーザに柔軟に権限を付与する仕組みである．

object は，アクセス制御対象を定める部分であり，URI (Uniform Resource Identifier)，パス式，または両者の組み合わせを指定できる．URI によりインターネットにおける文書の格納場所が指定される．パス式を用いると，パス式を満たす XML 文書の部分木がアクセス制御対象となり，細粒度のアクセス制御が可能になる．アクセス要求に対する権限判定を効率的に行なう観点から，ルールのパス式を効率的に処理できるサブセットに限定することが多い．典型的なサブセットは/x, /*, //x, //*，および述語のみからなるパス式である．

上記のフォーマットとは異なり，XML 文書の制御対象の要素にその属性値としてアクセス制御ルールを持たせるという（この場合 *object* は省略される），埋め込み式のルールもあり，医療情報の XML 文書スキーマ HL 7[9]で用いられている．また文書木のノードごとに適用されるルール集合をノードのラベルとして表現する方法もある[7]．

action は，アクセス方法の種類であり，通常は read または write を指定する．*decision* は，grant または deny のいずれかであり，grant のときはアクセスを許可するルール（許可ルールまたは正ルールという），deny のときはアクセスを拒否するルール（拒否ルールまたは負ルールという）であることを示す．

options は，ルールの適用範囲や適用方法についてのオプションであり，次のものからなる．

- cascade：*object* で指定されたノードについて，その子孫と自身全体にルールを適用させるとき，cascade を指定する．これを**ルールを伝播させる**ともいう．cascade がないときは，伝播は行われず *object* で指定された対象にのみにルールが適用される．
- schema：これはルールがある DTD に対応付けられており，その DTD に従うすべての文書インスタンスにそのルールが適用されるという，スキーマレベルのルールであることを意味する．schema が指定されなければ，ルールがある文書インスタンスに対応付けられており，ルールはそれに対してのみ適用されるという，インスタンスレベルのルールであることを意味する．

この他にも，必須処理（obligation）をルールに持たせるモデルがある[11]．必須処理とは，アクセスを許可する際に実行しなければならない操作であり，例えば許可の際にログに記録したり，書き込みの際に暗号化方式を指定するなどである．必須処理は節 3 で述べる XACML にも取り込まれている．

図 1 に XML 文書の例，図 2 にこの文書に対するアクセス制御ルールの例を示す．例えば，ロール名 customer_desk では，customer_id, name が読めることをルール r_1 および r_2 で設定し，同ロールでは/customer/address の内容全体の read/write が可能であることを cascade オプションを持つ r_3 と r_4 で設定する一方，/customer/profile の内容を読むことを禁止するのを cascade オプションを持つ r_5 で指定している．

```
<customer>
  <customer_id>azXXXXX </customer_id>
  <name>John</name>
  <address>
    <postal>
      <zip>8A3 72B </zip>
      <street>000 City St.</street>
      <city>Tucson</city>
      <state>Arizona</state>
    </postal>
    <work_phone>(712)585-XXXX</work_phone>
    <home_phone>(712)585-YYYY</home_phone>
    <mobile_phone>(903)7065-ZZZZ
    </mobile_phone>
  </address>
  <profile>
    <occupation_code> GOV </occupation_code>
    <annual_income>10K</annual_income>
    <credit_status>Good</credit_status>
  </profile>
</customer>
```

図 1　XML 文書の例

r_1: (customer_desk,/customer/customer_id,read,grant,)
r_2: (customer_desk,/customer/name,read,grant,)
r_3: (customer_desk,/customer/address,read,grant,cascade)
r_4: (customer_desk,/customer/address,write,grant,cascade)
r_5: (customer_desk,/customer/profile,read,deny,cascade)
r_6: (sales_agent,/customer,read,grant,cascade)
r_7: (sales_agent,/customer/address,read,deny,cascade)
r_8:
(sales_agent,/customer/address/home_phone,read,grant,)

図 2　アクセス制御ルールの例

2.2 衝突解消方式

XML文書のあるノードに対しアクセス要求を行った場合，そのノードに適用されるルールが複数存在することが起こり得る．それらのルールを評価した結果が許可と拒否に分れた場合に，最終的な権限判定をどうするかという衝突解消（conflict resolution）を行う必要がある．また，ノードに適用できるルールが存在しなかった場合のデフォルトの権限判定も考えておく必要がある．様々な衝突解消方式が存在するが，ルールを作成する者（例えばセキュリティ管理者）にとって結果が容易に把握できるものでなければならない．

XMLでは階層構造に従った衝突解消方式が特徴的である．

デフォルトの権限判定は2種類考えられ，**デフォルト拒否**（default-denial）は，適用できるルールがなかったときに判定結果を拒否とするものであり，**デフォルト許可**（default-grant）は逆に許可をデフォルトの判定結果とするものである．

衝突解消方式における**詳細性優先**（most specific rule takes precedence）とは，ルールの適用される範囲が狭いものの権限判定が優先されるというものであり，衝突解消方式の原則的な考え方として有用である．詳細性優先の考え方に基づいて，より具体的な衝突解消方式が導ける．例えば，スキーマレベルのルールとインスタンスレベルのルールがある文書に適用されるとき，詳細性優先のもとではインスタンスレベルのルールが優先される．またある文書木において，二つのルールAとBが対立しており，ルールAはあるノードとその子孫を対象とし，ルールBはその子孫のうちのノードの一つを対象としている場合は，詳細性優先のもとではルールBが優先される．しかしパス式を用いた指定方法では，二つのパス式の評価結果に包含関係はないが交差するような場合があり，このときは詳細性優先では衝突解消は行えない．**インスタンス優先**（instance takes precedence）は詳細性優先の考え方をスキーマレベルのルールとインスタンスレベルのルールの衝突に適用し，インスタンスレベルのルールを優先させるものである．

詳細性優先とは異なり，権限判定値に基づいた衝突解消方式もある．**拒否優先**（denial-takes-precedence）は，あるノードに適用されるルールのうちで，一つでも拒否の判定を行うルールがあれば，衝突解消の結果も拒否とするものである．**許可優先**（grant takes precedence）は，逆に許可を行うルールを優先するものである．**先行優先**（first-applicable）では，ルールの集合に番号を振ったり，リストに並べるなど線形順序を与えておき，ルールの評価をその順序に従って行う．そして最初に得られた判定結果が衝突解消の結果となるものである．

図2のアクセス制御ルールの例では，ロール名sales_agentについて，cascadeオプションを持つルールr_6により/customerとその子孫を読むこと許可しているが，その一方でルールr_7では，同ロールが/customer/addressを読むことを禁じ，さらにルールr_8では/customer/address/home_phoneを読むことを許可している．ここでr_6とr_7は/customer/addressの部分について衝突しているし，r_7とr_8では/customer/address/home_phoneの部分で衝突している．衝突解消方式として拒否優先を採用したならば，r_7がr_6およびr_8より優先され，/customer/addressとその子孫のreadは禁止，それ以外の部分は許可となる．一方，許可優先を採用したならば，r_7は無視されr_6によりsales_agentは/customerとその子孫全体を読むことができる．さらに，先行優先として，ルール間の優先順位としてr_8, r_7, r_6を指定した場合，sales_agentは，/customer/addressの子孫のうちhome_phoneのみを読むことができ，他のaddressの子孫は読めず，一方address以外のcustomerの子供は読むことができる．詳細性優先を採用していると，この先行優先と同じ結果が得られる．

上記以外にも衝突解消方式がいくつか提案されている．**最近祖先優先**（nearest ancestor takes precedence）は，ノードに対し，最も近い祖先（あるいはノード自身）に定義されているルールを適用するものであり[6]，詳細性優先の考え方に基づいている．例えば，パス式/a/bに伝播型の拒否，パス式/a/b/cに許可のルールが設定されているとき，ノードcへのアクセスは最近祖先優先では許可され，拒否優先では拒否される．

また文献[7]では，hardおよびsoftというオプションを用意しており，hardは打ち消されることなく，例外なく適用されるべきルールであることを示し，softは打ち消されることを許すルールであることを示す．例えば，インスタンス優先が設定している場合でも，スキーマレベルのルールAがhardと指定されると，ルールAが優先されるようになり，またインスタンスレベルのルールBがsoftと指定されると，スキーマレベルのルールが優先されるようになる．

以上の衝突解消方式やデフォルト設定方式は，ルール/ポリシを設定する状況に応じて適切なものを選択すればよい．例えば，ほとんどすべてのノードへのアクセスが許可され，一部ノードが拒否される場合は，拒否を指定するルールを列挙し，デフォルト許可と拒否優先を採用した方が，デフォルト拒否と許可優先を採用するよりも，ポリシのルール数を小さくすることができる．

あるノードのアクセスが許可されるには，そのノードのすべての祖先ノードについて許可が必要であるとする方式と，そうではなく親が不許可でも子は許可するという方式が有る．前者はノードのアクセス権限には，子供へのリンクをたどる権限も含まれており，あるノードのアクセスには根からそのノードまでのリンクすべてのアクセス許可が必要であるという立場である．これに対し後者では，ノードのアクセス権限とはノードの値に対する権限であり，リンクのアクセス権限とは別と考える立場である．

3. XACML

アクセス制御を記述するための標準言語として，Extensible Access Control Markup Language (XACML) が策定されており，2005年3月2日にVersion 2.0がOASIS標準として承認されている[17]．XACMLはアクセス制御ポリシをXMLで記述するための拡張性のあるフレームワークであり，組織間や企業間でアクセス制御ポリシをやり取りする相互運用を意識しており，特定の動作環境に依存しない仕様となっている．XPathやXML Schemaなどの標準フレームワークを活用し，基本的なアクセス制御ルール/ポリシをXMLで記述することに加え，Obligationも指定できる．また衝突解消方式として，節2.2で述べたものの他に，利用者が独自の方式を指定することができる．他にも多様なアクセス制御モデルに対応できるように拡張性を備えている．

図3にXACMLのポリシに基づいたアクセス制御方式のデータフローを示す．中心的なコンポーネントはPDP (Policy Decision Point) であり，アクセス制御のRequestとしてSubject Action/Environment/の3種類の属性値をXMLでPDPに送り，結果はResponseとしてPermit/Deny/NotApplicable/IndeterminateのいずれかがXMLで返される．ポリシ評価においては，権限判定結果が常に明確に定まるように配慮されている．XACML 2.0では，目的に応

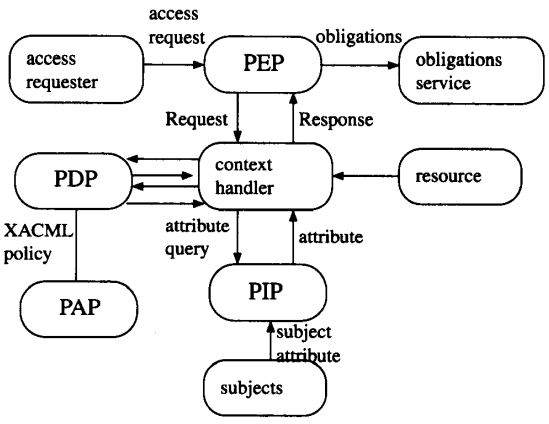

図3 XACMLのデータフロー

じてXACMLを使用するためのプロファイルが用意されており，例えばXMLでない階層型のリソースをXMLで表現してアクセス制御を行うプロファイル，プライバシポリシのためのプロファイルなどが用意されている．

4. XMLアクセス制御に関する研究動向

本節では，XMLのアクセス制御ポリシの処理技術についていくつか紹介する．XMLデータベースの検索においては，アクセス制御ポリシを適用し，検索の発行者に許可されている部分を検索結果に残し，不許可の部分を除く処理が必要である．様々な方式が考えられる．

4.1 ラベルの圧縮

節2で述べたように，XML文書のノードに対し適用されるルール集合をラベル付けしておいて，ノードへのアクセス要求時に権限判定を行う方式について考えてみる．アクセス主体であるユーザのIDやロールの数が多い場合など，ルール数が非常に大きくなるので，ラベルを圧縮することが考えられる．文献[18]では，Compressed Accessibility Map (CAM) というデータ構造を提案しており，cascadeの有無と許可/拒否の組からなるラベル集合において，冗長なラベルを除去することによりユーザIDごとに最小のCAMを文書サイズの線形時間で構築するアルゴリズムを示している．またCAMを用いた権限判定は，ノードの深さ（根からの距離）とCAMサイズの対数の積に比例する時間で行うことができる．

4.2 問い合わせの静的解析と書き換え

XMLデータベースへの問い合わせにおいて，問い

合わせの結果にアクセス不許可の情報が含まれないかを，問い合わせの実行前に判定する手法として静的解析[13]がある．アクセス制御ポリシと問い合わせを判定するオートマトンをそれぞれ構築し，オートマトンの包含関係を検査する．問い合わせオートマトンがポリシオートマトンに包含されるならば，問い合わせは常に許可できることが分かる．逆に問い合わせオートマトンとポリシオートマトンが互いに素ならば問い合わせは常に拒否されるべきである．その中間の場合は静的解析では不明という結果になり，XMLデータベースで問い合わせを実行することにより最終的な権限判定を行う．

QFilter[6]は，XMLデータベースで問い合わせを実行する前の段階で，問い合わせをポリシに適合したものに変換する方式を取る．/x, /*, //x, //*，および述語からなるパス式によるルールの集合を，線形時間で非決定性オートマトン（NFA）に変換する．このNFAと問い合わせのパス式のマッチングを行い，アクセス不許可の部分はアクセスしないように問い合わせを書き換える．QFilterは，XMLデータベースを変更せずにアクセス制御を導入できる特徴がある．NFAの構築と問い合わせの書き換えにおいて文献[13]の性能を改良している．

4.3 セキュリティビュー

DTDをユーザに見せると，ユーザには許可されていない要素名が見えてしまうし，DTDの情報を利用することにより本来ユーザに許可されている問い合わせの結果の差分を取るなどにより情報が漏洩してしまうことがある．ポリシのパス式と文書のDTDを分析し，ロールごとに安全なDTDをセキュリティビューとして導出する[8]．セキュリティビューに対して，問い合わせをパス式で与えると，問い合わせはもとのDTDに従った問い合わせに変換され実行される．この枠組みでは上述のような漏洩を防ぐことができる．ただし，その目的のためにポリシの細粒度性がある程度犠牲になっている．

4.4 文書流通におけるポリシの変換

XML文書による情報流通の特色として，多数の組織間で流通が行われるが，個々の組織間では異なるXML文書のスキーマを採用していることである．情報流通ではスキーマの変換が行われるが，その前後でも等価なアクセス制御が行われる必要がある．そのためにはポリシも変換されなければならない．インスタンスレベルのポリシでのこのような等価変換手法や，スキーマレベルのポリシで等価性が保てるためのパス式の条件がいくつか示されている[4, 5]．

4.5 アクセス制御とバージョン管理の統合モデル

XML文書のバージョン管理とは，XML文書に対してテキストの変更やノードの削除，移動，コピー等の更新操作を行ったときに，どのノードがどのノードから派生したかや，同じ内容であるという関係をシステムで維持し，派生関係による検索を可能にするものである．時間とともに蓄積されるXML文書のアーカイブにおいて，文書の派生関係や変更履歴を調べるのにバージョン管理は有用である．会社組織内の情報やニュースサイト，ブログ等，日々更新追加が行われるXMLのアーカイブにおいては，派生関係に基づいたアクセス制御を定義することができる．インスタンスレベルのポリシでは煩雑になりすぎ，かつスキーマレベルのポリシでは制御の単位が大きすぎる場合でも，派生関係で連結された部分グラフを単位としてアクセス制御を行うことができる．これは例えば掲示板のスレッドを単位として制御を行うことに相当する．文献[10]では，XPathの軸にあらたにバージョンの派生関係の軸を追加した経路問い合わせ言語を定義し，これを用いたポリシの定義方法を示している．

5. おわりに

本稿では，XML文書におけるアクセス制御について，基本的手法と標準化動向，研究動向について概説した．XMLによる情報流通が盛んになると同時に情報保護は益々重要になると考えられ，個人情報保護やプライバシなど実社会からの要求に答えるかたちで，今後もアクセス制御技術に関して新たな発展があると思われる．

参考文献

[1] 天笠俊之，吉川正俊，“XMLデータベース技術概説，”オペレーションズ・リサーチ，vol. 50, no. 6, pp. 365-372, 2005.

[2] E. Bertino, S. Castano, E. Ferrari, M. Mesiti, “Specifying and Enforcing Access Control Policies for XML Document Sources,” WWW Journal, vol. 3, no. 3, 2000.

[3] E. Bertino, E. Ferrari, “Secure and selective dissemination of XML documents,” ACM Trans. Inf.Syst. Secur. 5 (3), pp. 290-331, 2002.

[4] S. Chatvichienchai, M. Iwaihara, and Y. Kambayashi,“Translating Content-Based Authorizations of

XML Documents," Proc. 4th Int.Conf. Web Information Systems Engineering (WISE 2003), pp. 103-112, Dec. 2003.

[5] S. Chatvichienchai, M. Iwaihara, and Y. Kambayashi, "Authorization Translation for XML Document Transformation," World Wide Web: Inernet and Web Information Systems, Kluwer, vol. 7, No. 1, pp. 111-138, Mar 2004.

[6] S. Cho, S. Amer-Yahia, L. V. S. Lakshmanan, and D. Srivastava, "Optimizing the Secure Evaluation of Twig Queries," Proc. VLDB, pp. 490-501, 2002.

[7] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "A Fine-Grained Access Control System for XML Documents," ACM TISSEC, vol. 5, no. 2, 2002.

[8] W. Fan, C.-Y. Chan, M. N. Garofalakis, "Secure XML Querying with Security Views," ACM SIGMOD Conference, pp. 587-598, June 2004.

[9] HL 7 Standards, http://www.hl7.org/library/standards_non 1.htm

[10] M. Iwaihara, S. Chatvichienchai, C. Anutariya,and Vilas Wuwongse, "Relevancy Based Access Control of Versioned XML Documents," Proc. ACM Symp. Access Control Models and Technologies (SACMAT), June 2005 (to appear).

[11] M. Kudo and S. Hada, "XML Document Security based on Provisional Authorization," Proc. 7th ACM Conf. Computer and Communications Security, pp. 87-96, 2000.

[12] B. Luo, D. Lee, W.-C. Lee, P. Liu, "QFilter: Fine-Grained Run-Time XML Access Control via NFAbased Query Rewriting," Proc. 13th ACM CIKM, pp. 543-552, 2004.

[13] M. Murata, A. Tozawa, M. Kudo, S. Hada, "XML Access Control Using Static Analysis," Proc. ACM Conf. Computer and Communications Security, pp. 73-84, 2003.

[14] OASIS Security Assertion Markup Language (SAML), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[15] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models," IEEE Computer, 29(2), pp. 38-47, 1996.

[16] 田島敬史, "XML 用木パターン言語 XPath 解説," オペレーションズ・リサーチ, vol. 50, no. 6, pp. 373-378, 2005.

[17] OASIS XACML Technical Committee, "eXtensible Access Control Markup Language (XACML) Version 2.0," http://www.oasis-open.org/committees/download.php/10578/XACML-2.0-CD-NORMATIVE.zip.

[18] T. Yu, D. Srivastava, L. V. S. Lakshmanan, H. V. Jagadish, "Compressed Accessibility Map: Ef-ficient Access Control for XML," Proc. 28th VLDB, pp. 478-489, 2002.