XML 문서를 위한 효율적인 미세 접근 제어 시스템

이승현° 이헌길 강정모
강원대학교 컴퓨터정보통신공학과
nakayami@hotmail.com°, hglee@kangwon.ac.kr , boo95002@hanmail.net

An Efficient Fine-Grained Access Control System for XML Documents

Seunghyun Lee° Heonguil Lee Jungmo Kang
Dept. of Computer and Communication, Kangwon National University

요 약

본 논문에서는 XML 문서의 보다 효율적인 접근 제어를 위하여 XML 문서 전체 단위의 접근이 아닌 문서 내 노드 등과 같이 세밀한 부분에 접근 제어를 할 수 있는 미세 접근 제어 기법을 사용하였다. 현재 제안되어 사용되고 있는 묵시적 권한 기법은 상위 구성요소에 한번의 권한 부여로 하위 노드에 권한을 부여하는 효과를 가지는 장점을 가지고 있지만, 각각의 구성요소간의 권한을 체크하는 시간의 오버헤드를 가지는 단점이 있다. 이러한 권한을 체크하는 시간을 감소시키기 위하여 본 논문에서는 저장 공간의 오버헤드가 있지만, 권한체크의 시간을 감소시켜서 사용자에게 보다 빠른 view를 제공할 수 있는 XML 문서를 위한 명시적 트리 모델과 view 생성 알고리즘을 제안하였다.

1. 서 론

현재 전자상거래나 병원관리 등과 같은 여러 응용분야에서 문서의 논리적인 구조를 표현하는데 XML이 사용되고 있다. 이와 같은 응용분야에서의 XML 문서는 공개되어도 상관없는 부분과 개인 정보와 같은 타인에게 공개되지 않아야 하는 부분으로 나뉘게 되고, 이로 인하여 문서 내 서로 다른 권한을 가지는 각 구성요소에 대한 액세스 제어를 위한 여러 기법들이 제안되었다.

접근 제어 기법 중 미세 접근 제어 기법(Fine-Grained Access Control)[4,6]은 문서 내 서로 다른 권한을 가지는 각 구성요소에 대한 세밀한 액세스 제어를 하기 위한 기법으로써 XML 문서와 같이 큰 단위가 아닌 XML 문서 내의 documentElement, node 등과 같이 작은 단위로 계층적인 항목에 대하여 세밀한 접근 제어를 할 수 있다[4]. 이러한 미세 접근 제어를 위한 권한 기법은 명시적 권한 기법과 묵시적 권한 기법이 있다. 명시적 권한 기법은 각각의 구성요소 마다 권한의 정보를 저장하는 기법으로 묵시적 권한 기법에 비하여 저장 공간의 오버헤드가 있다. 그러나 각 구성 요소가 권한의 정보를 가지고 있으므로, 즉각적인 접근 제어를 할 수 있고, 묵시적 권한 기법보다 권한 체크의 시간이 작으므로, 권한 체크가 자주 발생하는 응용분야에 적용하면 권한 체크 시간의 오버헤드를 감소시킬 수 있다. 묵시적 권한 기법은 상위 구성요소에 대하여 한번의 권한 부여로 하위 구성 요소에 권한을 부여하는 효과를 갖는 권한 기법이다[8,10]. 묵시적 권한 기법은 모든 구성 요소가 권한에 대한 정보를 가지고 있을 필요가 없으므로 명시적 기법보다 저장 공간의 오버헤드가 적게 드는 장점을 가진다[7]. 하지만, 구성요소에 권한을 부여하거나 수정 할 경우, 권한 충돌 판별을 위하여 권한이 부여된 상위 구성 요소까지 탐색해야 하는 시간의 오버헤드를 지닌다[8].

Damiani 모델[4,6]은 Tokyo IBM Research Lab[11]의 방식 중 묵시적 권한 기법에 중점을 두어 설계하였으며, XML의 구조와 묵시적 권한 기법의 특성에 따라 권한을 8가지의 타입으로 분류하였으며, 권한의 레벨에 따라 schema/instance로, 권한이 명시적인지 묵시적인지에 따라 Local/Recursive로, 권한의 강도에 따라 Hard/Soft로 분류하였다. 제안된 Damiani 모델은 XML 문서의 응용 분야 중 read에 대한 ratio가 write에 대한 ratio보다 현저하게 많은 분야를 적용대상으로 하였으며[3], 이에 따라 사용자가 문서를 요청하면 권한을 체크하여, 권한에 해당 되는 문서만을 제공하는 Computing view 알고리즘을 제안하였다[5].

본 논문은 권한을 체크하는 시간을 감소시키기 위하여 XML문서를 위한 명시적 권한기법을 제안하고, 이를 이용하여 노드의 권한에 대한 모든 정보를 가지고 있는 접근 권한 트리 모델을 제안하였다.

본 논문의 구성은 다음과 같다. 2장에서는 XML 문서를 위한 명시적 권한 부여 기법을 제안하고, 이를 이용하여 문서의 모든 구성요소 내에 구성요소가 가질 수 있는 모든 권한의 정보를 저장하는 접근 권한 트리 모델을 제안한다. 또한, 트리 모델을 생성하는 알고리즘과 생성된 트리 모델을 이용하여 사용자가 문서를 요청했을 때 view를 제공하는 view 생성 알고리즘에 대하여 기술하였다. 3장에서는 제안된 기법을 시간과 공간에 대한 오버헤드 측면에서 분석하였다. 마지막으로 4장에서 결론을 제시한다.

2. 제안된 권한 기법

2.1 XML 문서를 위한 명시적 권한 기법

XML 문서를 위한 명시적 권한 기법은 XML 문서를 위한 효율적인 미세 접근 제어와 묵시적 권한 기법을 사용한 Damiani 모델의 단점인 권한 계산의 오버헤드를 감소시켜 사용자에게 빠른 view를 제공하기 위하여 제안되었다. 또한, 본 논문은 문서를 관리하는 권한 정책과 문서 내 각 노드가 가지고 있는 모든 정보를 계산하여 명시적 접근 권한 트리에 저장하는 방식을 제안하고 있다. 접근 권한 트리는 이후에 제안될 view 생성 알고리즘을 수행하는데 사용되는데, 문서에 대한 권한 정책이 자주 수정되지 않고, read에 대한 ratio가 write에 대한 ratio보다 현저하게 많은 응용분야에 적용하면 사용자에게 보다 빠른 view를 제공할 수 있다.

XML 문서를 사용하는 각각의 응용 분야에서는 그 분야에 대한 정책들을 가지고 있다. 이 정책들은 권한 정책 Table에 저장하여 유지 관리한다[3]. 권한 정책 Table은 view를 만들 때마다 매번 권한의 정책을 파싱하여 사용하는 Damiani 모델과 달리 본 논문에서 제안된 권한 기법은 권한의 정보를 Table에 유지한다.

권한 정책 Table은 subject, object, action, sign로 구성되어있다. subject는 user-group, ip-address, symbolic name의 3가지 구성요소로 이루어져 있다. 예로서 subject (public, 210.115.48.*, hospital.com)는 210.115.48.*의 ip주소로 hospital. com을 통하여 접속하는 모든 사람을 의미한다.

object는 권한이 표현되는 각 구성요소를 나타내며, XML 문서 내에서는 각 노드에 해당한다. object의 표현은 XPATH를 이용한다. action은 read라는 하나의 행위이다. 그리고 sign은 +, -로 표현되는데, 권한의 긍정과 부정을 나타낸다.

2.2 명시적 접근 권한 트리 생성 알고리즘

명시적 접근 권한 트리 생성 알고리즘은 권한의 정책과 XML 문서를 이용하여 XML 문서 내의 각 구성요소에 권한 정보를 표현하고 권한 트리를 생성하는 알고리즘이다.[그림1] 제안한 알고리즘은 크게 세 단계로 이루어져있다. 권한 정책 Table에 명시된 명시적 권한을 일단 트리 내에 저장하는 단계, 명시되지 않은 구성요소에 대하여 명시적으로 권한을 저장시키는 단계, 그리고 권한 충돌을 방지하기 위한 단계이다. 권한이 없는 구성 요소에 대하여 권한을 부여하는 단계는 다음과 같다. 하위의 트리로 탐색하다가 하위 구성 요소에 상위에서 부여되는 권한과 같은 subject에 대하여 명시적 권한이 부여 되어있을 경우, 권한 부여를 중단하고, subject가 다르면 권한의 부여가 계속 이루어지게 된다.

```
•P : 권한 정책(Policy)
•D : XML document
•N : 임의의 노드, n의 상위 노드
•n : N의 하위노드
•R : root
•A : Access Authorization of node n (subject, sign)
•subject : Authorization Subject
•sign : + , -
•Q : 노드들의 집합(queue)
•S : 노드들의 집합(stack)
•i, j : 임의의 정수 값

struct Policy {                    struct Access_Authorization {
        subject ;                          subject ;
        object ;                           sign ;
        sign ;                     } A ;
} P ;

Input : 권한 정책 P[] and XML document
Output : 명시적 Access Authorization Tree
Algorithm 명시적 Access Authorization Tree 생성
begin
N = R ;
// 정책 테이블에 명시된 정책을 문서내의 노드에 설정한다.
1. 권한 정책 P의 모든 P[i]에 대하여 P[i].subject와 P[i].sign을 D의 P[i].object에 추가 ;
2. N의 A가 설정되지 않았다면, (public,*,*, +)으로 설정 ;
// 각 구성요소에 명시적 권한을 부여 한다
while (Q is not empty) {
    3. N = Delete(Q) ;
        4. N의 A[j]와 n의 A[k]에 대하여 다른 subject를 비교 ;
            3.1 subject가 다르다면 N의 A[j]를 n의 A에 추가 ;
            // subject에 대한 권한 정책이 이미 존재한다면, 그것을 유지한다
          3.2 subject가 동일하다면 기존의 권한을 유지 ;
        5. Add(Q, n) ;
        6. push(S, N) ;
}
// 충돌 탐지
while (S is not empty) {
    7. n = pop(S) ;
    8. n의 A[j]와 N의 A[k] subject가 동일하고 A[j]의 sign이 '+'이고, A[k]의 sign이 '-'
            라면, A[k]의 sign을 '+'로 설정 ;
}
end
```

그림 1 명시적 접근 권한 트리 생성 알고리즘 및 자료구조

명시적 권한 기법은 묵시적 권한 기법과 달리 구성요소마다 권한을 명시적으로 저장하고 있으므로, 권한을 계산하는 단계에서 발생하는 권한 충돌이 거의 일어나지 않는다. 하지만 상위 구성 요소의 권한이 -이고, 하위 구성 요소의 권한이 +이면 트리에 모순이 발생하게 되어 권한 충돌이 일어나게 된다. 이를 방지하기 위하여 큐를 이용하여 권한을 부여하는 방식의 역순으로 즉, 상위의 구성 요소부터 하위의 구성 요소 까지가 아닌, 하위 구성 요소부터 거슬러 올라가며 권한을 비교하여, 충돌이 있는지 여부를 판정한다.

생성된 명시적 접근 권한 트리는 사용자가 요청하는 문서와 사용자의 권한, 그리고 권한의 정보를 가지고 있는 권한 정책 Table을 이용하여, 문서 내의 각 노드가 가지고 있는 모든 권한 정보를 트리의 형태로 저장하고 있는 것을 말한다. 이후, 사용자가 view를 요청할 때 접근 권한 트리와 view 생성 알고리즘을 이용하여 사용자에게 view를 제공하게 된다.

접근 권한 트리 내의 권한은 권한의 주체인 subject와 권한의 sign의 두 가지의 구성 요소로 표현된다. subject는 user-group, ip-address, symbolic-address의 집합으로 이루어져 있으며, sign은 +/-로 표현되며, 권한의 긍정과 부정을 나타낸다.

2.3 제안된 view 생성 알고리즘

제안된 view 생성 알고리즘은 [그림2]와 같다. Requester rq가 XML 문서를 요청하면 문서의 rootnode의 권한 subject와 sign을 체크하여 subject가 일치하고, sign +일 때, 하위의 트리를 탐색하여 권한이 일치하고 sign이 +인 노드들을 view에 추가 시킨다. 위와 같은 조건을 만족하지 못 할 경우 트리 탐색을 하지 않고, view를 만들 수 있는 최하위 노드까지 탐색하여 추가된 노드를 최종적인 view에 추가 하면 사용자에 만들어진 view를 전송한다.

```
•T : rootnode of Document's 명시적 Access Authorization Tree
•N : 임의의 노드 (권한이 표시되는 대상)
•n : N의 하위노드
•A : Access_Authorization of node n (subject, sign)
•subject : Authorization subject (권한을 가지고 있는 주체)
•sign : + / - (긍정과 부정)
•V : Result of view
•v : view에 추가되는 Node들의 queue

struct Access_Authorization          struct Requester
 {    subject ;                      {    subject ;
      sign ;                              document URL ;
} A ;                                } rq ;

Input : A requester rq and XML document URI
Output : The document to be returned to rq
Algorithm Generate view
begin
1. V is empty, N = T , v is empty ;
2. 노드 N의 A의 subject가 rq의 subject와 일치하고, sign이 +이면 Add(v, N) ;
while (v is not empty) {
     3. N = Delete(v), Add(V, N) ;
      4. 문서 내 node N의 하위노드 n에 대한 A의 subject가 rq의 subject와 일치하
   고, sign이 +이면, Add(v,n) ;
}
6. V를 rq에게 전송;
end
```

그림 2 view 생성 알고리즘 및 자료구조

제안된 명시적 접근 권한 트리와 view 생성 알고리즘을 이용하여 사용자에게 최종적인 view를 제공하는 단계를 흐름도로 표현하면 [그림3]과 같다.

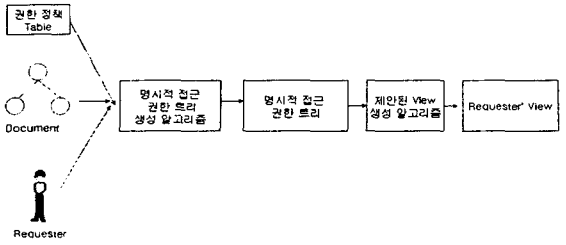

그림 3 view 생성 과정

3. 제안된 기법 분석

이 장에서는 제안한 기법에 대해 view를 생성하는데 걸리는 시간과 저장 공간의 오버헤드에 대해 분석하기로 한다. 먼저, n을 XML 문서 내의 권한의 정보를 가질 수 있는 전체 노드의 수라고 가정하면 제안된 기법의 시간 오버헤드는 다음과 같다.

제안된 기법은 권한에 대한 정보를 미리 계산하여 저장하여 둔 명시적 접근 권한 트리를 이용하므로 view를 생성하는 시간에 대한 오버헤드는 best case가 1, worst case가 n이 된다. 이와 같은 결과가 나오는 이유는 최상위 노드가 부정으로 지정되는 경우(best case)와 전체 노드의 권한이 긍정으로 되는 경우(worst case)를 가지기 때문이다. 그리고 제안된 기법은 접근 권한 트리를 생성하는 시간이 추가로 걸리는데 이 시간은 2n이다. 이유는 모든 권한의 정보를 이용하여 만들어 내는 접근 권한 트리의 생성 알고리즘은 권한을 명시적으로 부여하는 시간과 부여한 권한간의 충돌 유무를 판별하는데 걸리는 시간까지 총 두 번의 전체적인 트리 탐색 시간이 소요되기 때문이다.

위의 수치만으로 볼 때 Damiani 모델과 제안된 기법의 view 생성에 걸리는 시간의 오버헤드가 비슷하다고 생각 할 수 있으나, 응용분야의 실제적인 적용에 있어서 제안된 기법은 Damiani 모델보다 현저한 시간 오버헤드의 감소를 가져온다. 이유는 권한에 관한 권한 정책은 사용자가 view를 요청할 때마다 매번 바뀌는 것이 아니기 때문이다. 한번 정해진 정책은 자주 바뀌지 않으므로 권한의 정보를 미리 계산한 접근 권한 트리를 사용하였을 경우 사용자가 view를 요청했을 때, Damiani 모델은 평균 5n/2 만큼의 시간의 오버헤드가 걸리고, 제안된 기법은 평균 n/2 만큼의 시간의 오버 헤드가 걸리게 된다. 이상에서 볼 수 있듯이, 제안된 기법은 사용자의 요청에 대해 Damiani모델에 비하여 빠른 view를 제공한다.

다음으로 제안된 기법에 대한 저장 공간의 오버헤드는 다음과 같다. k를 권한의 정보를 가지고 있는 노드를 표현하는데 필요한 XML DTD 문서의 저장 공간이라고 가정한다.

제안된 기법에서는 권한의 정보를 표현하는데 필요한 XML DTD의 수가 최소한 XML 문서를 구성하고 있는 전체 노드의 수 이상 존재하게 된다. 제안된 기법에서는 모든 구성요소에 권한의 정보를 가지기 때문이다. 그러므로 제안된 기법의 공간 오버헤드는 O(k)가 된다. 이상에서 볼 수 있듯이 저장 공간의 오버헤드가 O(1)인 Damiani 모델에 비하여 제안된 기법은 더 많은 저장 공간의 오버헤드를 갖는다.

덧붙여, 제안된 기법은 instance를 포함한 XML 문서의 모든 구성요소에 권한의 정보를 저장하는 것이 아니고, 권한을 정의하고 있는 XML DTD에 대한 권한의 정보를 각 노드가 저장 하고 있으므로, 모든 문서 내의 구성요소에 권한을 저장하는 명시적 권한 기법방식에 비하여 저장 공간의 오버헤드를 감소시킨다.

4. 결론

본 논문에서는 XML 문서를 위한 명시적 권한 기법을 제안하고 이 기법을 사용한 XML 문서를 위한 미세 접근 제어 시스템을 제안하였다. 본 시스템의 세밀한 접근 제어를 위하여 XML 문서 전체의 큰 단위가 아닌 XML 문서 내의 document element, node 등과 같은 작은 단위로 접근 제어를 하는 방식인 미세 접근 제어 기법을 사용하였다. 현재 XML의 권한 접근 기법 중 많은 연구가 이루어지고 있는 묵시적 권한 기법은 상위 구성요소에 한 번의 권한 부여로 하위 구성요소에 권한을 부여하는 효과를 가지는 장점이 있다. 하지만, 권한을 가지는 구성요소에 접근 할 경우나 권한을 추가로 부여하는 경우 상위 구성요소와의 충돌의 유무를 판별하는 권한 체크시간의 오버헤드가 크다는 단점이 있다. 반면, 명시적 권한 기법은 각각의 구성요소마다 권한의 정보를 저장하는 기법으로 묵시적 권한 기법에 비하여 저장 공간의 오버헤드가 있지만, 각 구성 요소가 권한의 정보를 가지고 있으므로, 즉각적인 접근 제어를 할 수 있고, 묵시적 권한 기법보다 권한 체크의 시간이 작으므로, 권한 체크가 자주 발생하는 응용분야에 적용하면 권한 체크 시간의 오버헤드를 감소시킬 수 있다.

즉, 문서에 대한 권한 정책이 자주 수정되지 않고, read에 대한 ratio가 write에 대한 ratio보다 현저하게 많은 응용분야에 제안된 기법을 사용하면, 문서를 요청한 사용자에게 Damiani모델에 비하여 빠른 view를 제공한다.

참 고 문 헌

[1] S. Jajodia et al. "Flexible support for Multiple Access Control Policies", to be published in ACM Trans. Database Systems, 2001.

[2] C. Nlioudis, G. Pangalos and A. Vakal-i. "Security Model or XML data", Proc. 2th International Conference on Internet Computing, 2001.

[3] E. Damiani et al. "Securing XML Documents", Proc. 2000 Internet Conf, Extending Database Technology(EDBT2000), pp.121-135, 2000.

[4] E. Damiani, S. D. C di Vimercati, S. Paraboschi, P. Samarati. "A fine grained access control system for XML documents", ACM Transaction on Information and System Security, Vol.5 No.2, pp.169-202, 2002.

[5] E. Damiani, S. D. C di Vimercati, S. Paraboschi, P. Samarati. "Controlling access to XML documents", IEEE Internet Computing, pp.18-28, 2001.

[6] E.Bertino, S.Castano, and E.Ferriari. "Specifying and enforcing access control policies for XML document sources, World Wide Web J.3,3, 2000.

[7] Bertino, E., "Data Hiding and Security in Object-Oriented Database," In Proc. Int'l Conf. on Data Engineering, Tempe, Arizona, pp.338-347, Feb. 1992.

[8] Bertino, E., Samarati, P., and Jajodia, S., "An Extended Authorization Model for Relational Databases," IEEE Trans. on Knowledge and Data Engineering, Vol.9, No.1, pp.85-101, 1997.

[9] Fernandez, E. B., Larrondo-Perie, M. M., and Gudes, E., "A Method-Based Authorization Model for Object-Oriented Databases," In Proc. OOPSLA93 Conf. Workshop on Security for Object-Oriented System, Washington D.C., pp.135-150, 1993.

[10] Rabitti, F. *et al.*, "A Model of Authorization for Next-Generation Database Systems," *ACM Trans. on Database Systems*, Vol.16, No.1, pp.88-131, 1991.

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