산업기술연구(강원대학교 산업기술연구소 논문집), 제21권 A호, 2001.
Journal of Industrial Technology, Kangwon Nat'l Univ.,Korea, No. 21 A, 2001.

XML 문서를 위한 권한 부여 기법

An Authorization Technique for an XML Document

강 정 모* 이 헌 길**
Kang, Jung-Mo Lee, Heon-Gil

Abstract

An XML is an markup language which has been focused on the next generation Web programming language. It easily represents the complex structure of a document, and it is possible to provide the access control over each component of an XML document. An implicit authorization technique means that granting an authorization to a node has effect on granting the same implicit authorization to its all descendants. Therefore, it enhances the time for the authorization grant and reduces the memory required for the authorization information. An authorization technique using an intention type and a authorization replacement solves a redundancy problem and decides whether the access is possible or the authorization conflict occurs at the first attempt.

키워드 : *XML, 묵시적 권한 부여 기법, 의도형 권한부여 기법, 권한 대체 기법*
Keywords: *XML, implicit authorization, intention authorization, authorization replacement*

1. 서론

정보화 사회로 발전해감에 따라 시스템 및 응용에 독립적인 문서정보가 요구되고, 문서 정보 교환, 검색 등의 처리를 위한 표준이 필요하게 되었다. 이에 웹 문서 양식인 XML이 등장하게 되었는데, XML(eXtensible Markup Language)은 SGML의 부분 집합으로서W3C (World- Wide Web Consortium)에 의해 제안된 확장성 마크업 언어이다 [7,9]. '확장성'이라는 말에서 알 수 있듯이 문서의 내용에 관련된 태그를 사용자가 직접 정의할 수 있으며, 그 태그를 다른 사람들이 사용할 수도 있다. 이런 이유로 XML을 구조적인 자료로 구성된 문서를 위한 마크업 언어라고도 한다.

현재 XML은 인터넷 상의 EDI나 전자상거래

* 강원대학교 대학원 컴퓨터 정보통신공학과 석사과정
** 강원대학교 컴퓨터 정보통신공학과 교수, 공학박사

등의 응용들로 적용이 확대되고 있는데 [7,8,9], XML 문서가 타인에 의해 쉽게 도청이나 조작되거나 오용된다면 문서에 대한 신뢰성이 떨어져 그 이용이 제한될 것이다. 이에 따라, XML 문서를 보호하기 위한 여러 보안 기법들이 제안되고 있다 [8]. 본 논문에서는 저장된 XML 문서의 액세스 제어를 위한 묵시적 권한부여 기법과 의도형 권한 부여 기법을 제안한다. 명시적 권한 부여 기법은 모든 구성요소에 대해 규칙들을 명시적으로 저장하는 것으로서 구현이 단순한 반면 상당히 비효율적이다 [2,5]. 하지만, 묵시적 권한부여 기법은 명시적으로 저장된 권한으로부터 유도되는 권한이므로 메모리를 감소시킬 수 있다 [2,5]. 묵시적 권한부여 기법을 사용하면 상위 구성 요소에 대한 한번의 권한 부여로 하위 구성 요소들에 동일한 권한 부여 효과를 얻을 수 있어 권한 부여 시간을 줄일 수 있다.

하지만 묵시적 권한 부여기법[3,5]은 명시적으로 저장된 권한으로부터 유도되는 권한으로서 상위 구성요소에 한번의 권한 부여로 하위 구성요소에 동일한 권한 부여 효과를 얻을 수 있는 반면 액세스 제어나 권한 부여 시 계산의 오버헤드를 수반한다. 제안된 의도형 권한 부여 기법[6]은 묵시적 권한 부여 기법과 달리 상위 노드에 하위의 노드에 대한 정보는 부여하여 액세스 제어나 권한 부여 시에 권한을 부여하고자 하는 노드에서 바로 판별이 가능하므로 계산의 오버헤드가 없다.

앞으로 2장에서는 객체지향형 데이터베이스를 위해 제시된 묵시적 권한 부여 기법과 의도형 권한 부여기법에 대하여 소개하고, 3장에서는 XML 문서를 위한 묵시적 권한 부여 기법과 대체 의도형 권한 부여 기법에 대하여 설명 할 것이며, 4장에서는 분석 및 평가를 제시한다. 마지막으로 5장에서는 결론을 제시하여 본 논문의 끝을 맺는다.

2. 권한 부여 기법 소개

권한부여는 명시적 권한부여와 묵시적 권한부여, 그리고 의도형 권한 부여로 나눌 수 있다. 명시적 권한부여 기법은 모든 구성 요소들에 권한을 명시적으로 저장하는 것으로서 구현이 단순한 반면에 상당히 비효율적이다. 묵시적 권한 부여 기법은 명시적으로 저장된 권한으로부터 유도되는 권한으로 유도를 결정하기 위한 계산 오버헤드를 수반하지만 메모리의 절약과 권한 부여 시간을 단축할 수 있다 [6]. 묵시적 권한부여와 의도형 권한 부여는 긍정적 권한부여와 부정적 권한부여로 구별할 수 있다 [1,3,4]. 묵시적 권한 부여 기법에서는 명시적으로 권한이 부여되기 전에는 한 주체는 어느 객체에 대해서도 권한을 갖지 못함을 가정한다. 한 객체에 대해 권한을 가지지 못한다는 것은 주체가 그 객체에 접근할 수 없다는 의미이다. 부정적 권한 부여의 개념은 긍정적 권한 부여를 보장한다[1,3,4]. 즉, 주체가 객체에 대해 권한을 가지고 있지 않거나 부정적 권한을 가지고 있으면 객체에 대한 그 주체의 접근 시도는 거부된다. 실제로 현재 존재하는 권한부여 모델에서는 권한의 부재가 부정적 권한부여를 대신하고 있다. 권한부여는 강성과 약성 권한부여로도 구별할 수 있는데, 강성 권한 부여는 그 권한과 그 권한에 유도되는 모든 권한들에 대해 덮어쓰기를 허용하지 않는다. 반면에, 약성 권한 부여에 의해 유도된 권한들은 다른 권한부여에 의해 덮어쓰기를 허용한다 [1,3,4].

묵시적 권한 부여 기법은 구성요소에 일일이 권한을 부여하는 오버헤드를 줄이기 위하여 제안되었다.[3,5] 묵시적 권한은 사용자 측면에서는 각 구성요소에 반복적 권한 요청 대신에 상위 구성요소에 한번의 권한 요청으로 동일한 권한 부여 효과를 얻을 수 있다는 장점이 있다.[6] 그러나 묵시적 권한 부여 방식에서는 상위 노드에 정의된 권한으로부터 유도되는 묵시적 권한과 그 하위 노드에 대한 권한 사이에 충돌이 발생할 수 있으므로 서로 다른 노드간의 효율적인 충돌 탐지 기법이 필요하다. 충돌을 효율적으로 탐지하기 위하여 의도형 권한 부여 기법이 제시되었다.[6] 의도형 권한 부여 기법은 권한을 부여할 노드의 모든 조상 노드에 대하여 의도형 권한을 부여함으로써 하위 노드에 대한 권한과 그 상위 노드에 대한 권한으로부터 유도되는 묵시적 권한간에 서로 충돌이 발생하는 것을 그 노드에서 바로 탐지할 수 있는 것이다. 기존의 권한 부여 모델[1,3,4]에서는 권한 충돌 탐지 시 최하위 노드에 대한 권한들까지 모두 비교해야 하는 어려움이 있었다.

묵시적 권한 부여와 의도형 권한 부여 기법에서 충돌의 탐지는 권한 호환성 행렬(authorization compatibility matrix)을 사용하여 이루어진다. 권한 호환성 행렬에 대해서는 다음 장에서 소개한다.

3. XML 문서를 위한 권한 부여 기법

3.1 DOM과 권한 부여 기법

DOM(Document Object Model)은 XML로 표현된 문서의 객체 모델로서 트리 구조로 XML 문서를 표현한다 [10]. 그림1은 우리가 흔히 볼 수 있는 구매 주문서를 XML의 형식으로 표현한 것이다. 구매 주문서에서 보듯이 구매 주문서 내 신용카드

정보와 같은 중요 정보는 노출되면 개인에게 피해를 줄 수 있으므로 그에 대한 권한은 특정인만이 가져야 한다. 따라서, 특정 구성 요소 각각에 대해 권한 부여를 할 수 있는 방법이 필요하다

```
<?xml version="1.0" encoding="ECT-KR"?>
<주문서 SOURCE="web" 고객타입="소비자" 화폐단위="천원">
  <고객>
    <성명> 홍길동 </성명>
    <주소> 강원도 춘천시 효자동 </주소>
  </고객>
  <주문품목록>
    <품목1>
      <부품명> 메인보드 </부품명>
      <수량> 1 </수량>
      <단가> 200 </단가>
    </품목1>
  </주문품목록>
  <결재방법>
    <신용카드>
      <소유자> 홍길동 </소유자>
      <카드번호> 4128240012301234 </카드번호>
      <유효기간> 2002 1/12 </유효기간>
    </신용카드>
  </결재방법>
</주문서>
```

그림 1. 구매주문서의 XML 표현

그림2는 그림1의 XML로 표현된 구매주문서를 DOM 모델로 표현한 것으로 각 노드들의 권한부여 예를 보여준다. 그림 2에서 보듯이 XML 문서를 DOM으로 나타내면 세분화된 계층구조로 표현되어 각 구성 요소 단위로 액세스 제어가 가능하다. 본 논문에서는 DOM으로 표현된 XML 문서에 대해 묵시적 권한부여 기법과 의도형 권한 부여 기법을 제안한다. 본 논문에서 제시한 묵시적 권한 부여 기법과 의도형 권한 부여 기법은 Rabitti이 제안한 권한부여 원칙을 적용한다 [4]. 첫째는 **일관성**으로 권한 강도가 같은 권한 집합 내에 각각 적용되는 성질로서, 어느 한 명시적 혹은 묵시적 권한의 타입이 다른 명시적 혹은 묵시적 권한의 타입의 부정이 될 수 없다. 둘째는 **비중복성**으로 강성 권한 집합 내에만 적용되는 성질로서, 임의의 명시적 강성 권한과 그에 의해 유도되는 묵시적 강성 권한이 권한 집합 내에 같이 존재할 수 없다.

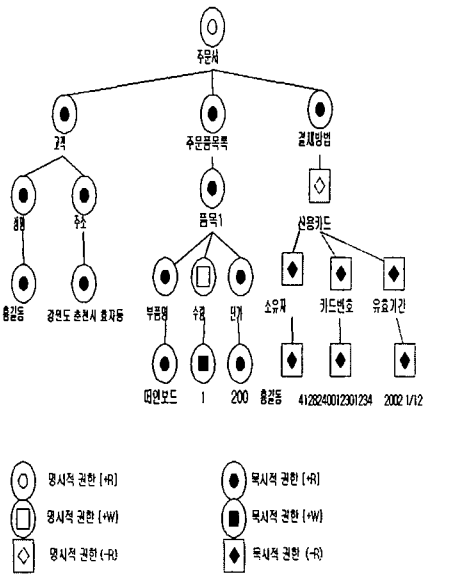

그림 2. 구매주문서에 대한 DOM의 트리구조

마지막은 **공존성**으로 권한 강도가 다른 권한 집합간의 관계에 적용되는 성질로서, 임의의 한 노드에 명시적 약성 권한이 부여되어 있는 상태에서 그 노드에 강성 권한이 명시적 혹은 묵시적으로 부여될 때, 어느 한 권한의 타입이 다른 권한의 타입의 부정이 아닌 경우에 한하여 두 권한의 타입은 같이 존재할 수 있다. 이 세 가지 원칙을 이용하여 권한 부여 시 권한 충돌 여부를 판정할 수 있는 호환성 행렬을 표 1,2,3,4와 같이 구할 수 있다.

표 1. 강성-강성 권한 호환성 행렬

추가 / 기존	(IR)	(IW)	(I-R)	(I-W)	(R)	(W)	(-R)	(-W)
(IR)	T	T	T	T	T	F	F	F
(IW)	T	T	T	T	F	T	F	F
(I-R)	T	T	T	T	F	F	T	F
(I-W)	T	T	T	T	F	F	F	T
(R)	F	F	F	F	F	T	F	T
(W)	F	F	F	F	F	F	F	F
(-R)	F	F	F	F	F	F	F	F
(-W)	F	F	F	F	T	F	T	F

표 1의 강성-강성 권한 호환성 행렬은 기존 권한이 강성이고 추가 권한도 강성인 경우의 두 권한 유형간의 충돌여부를 나타낸다. 의도형-명시적 권한인 경우 (표 1을 동일한 크기로 사 등분했을 경우 우상면에 해당함) 기존 권한과 추가 권한이 동일한 권한 유형인 경우에만 추가 권한이 허가된다. 강성 권한의 정의를 따르면 추가 강성 권한이 기존 강성 권한을 덮어쓰는 것을 허용치 않기 때문이다. 단, 기존 의도형 강성 권한과 추가 명시적 강성 권한이 동일한 권한 유형일 경우에는 권한 상승(authorization escalation)이 되는 경우이므로 추가 권한은 허가된다. 권한 상승이란 임의의 노드에 기존 강성 권한이 부여된 상태에서 그 상위 노드에 동일 권한 유형의 추가 강성 권한이 부여되는 현상이다. 권한 대체는 이런 권한 상승을 허용하는 개념이다. 기존의 의도형 권한에서는 비중복성을 제한적으로 이용한 반면, 권한 대체를 이용하면 Rabbit의 비중복성 원칙을 유지할 수 있어, 메모리의 오버헤드를 줄일 수 있다. 본 모델에서는 강성 권한간의 권한 상승을 허용하는 권한 대체를 적용한다.

명시적-명시적 권한인 경우(표 1의 우하면) 두 권한이 상존 할 수 있거나 추가 권한이 기존 권한을 변환시키는 경우에만 추가 권한이 허가된다. 권한의 불변성에 따르면 동일한 노드에서 두 명시적 강성 권한간에 일관성이 없거나 중복이 발생하거나 포함되는 경우에는 충돌이 발생하기 때문이다.

명시적=의도형 권한인 경우(표 1의 좌하면) 추가 권한은 모두 거절된다. 강성 권한의 정의에 따르면 추가하는 권한은 기존 강성 권한에 의해 유도되는 묵시적 강성 권한을 덮어쓸 수 없기 때문이다.

표 2. 약성-강성 권한 호환성 행렬

추가 / 기존	(IR)	(IW)	(I-R)	(I-W)	(R)	(W)	(-R)	(-W)
[IR]	T	T	T	T	T	T	F	T
[IW]	T	T	T	T	T	T	F	F
[I-R]	T	T	T	T	F	F	T	T
[I-W]	T	T	T	T	T	F	T	T
[R]	T	T	T	T	T	T	F	T
[W]	T	T	T	T	T	T	F	F
[-R]	T	T	T	T	F	F	T	T
[-W]	T	T	T	T	T	F	T	T

의도형-의도형 권한인 경우(표 1의 좌상면) 충돌여부는 하위 단계에서 판단되므로, 현 단계에서는 충돌이 없는 것으로 판정된다. 그 이유는 두 권한이 모두 의도형인 경우 단위 계층의 서로 다른 브랜치(branch)에 부여된 권한일 수 있으므로, 상위 노드에서는 실제적인 충돌 여부를 정확히 알 수 없기 때문이다. 실제로 강성-강성 권한간의 충돌 여부 판정은 명시적 강성 권한이 있는 두 개의 노드 중 상위에 위치한 노드에서 결정된다.

표 2의 약성-강성 권한 호환성 행렬은 기존 권한이 약성이고 추가 권한이 강성인 경우의 두 권한 유형간의 충돌 여부를 나타낸다. 의도형-의도형 권한인 경우의 충돌 여부는 강성-강성 호환성 행렬의 의도형-의도형 권한이 경우와 동일한 결과를 갖는다.

의도형-명시적 권한일 경우 두 권한간에 일관성이 없으면 추가 권한은 거절되고, 그 외의 경우에 추가 권한은 허가된다. 즉, 기존 약성 권한이 추가 강성 권한의 묵시적 권한에 포함되므로 덮어쓰기를 허용치 않는 강성권한의 정의에 따라 추가 권한이 거절되어야 하지만 공존성의 정의에 의해 약성 권한은 강성 권한에 따라야(conformable)하기 때문이다.[1,3,4]

명시적-명시적 권한인 경우 두 권한간에 일관성이 있으면 추가 권한은 허가된다. 두 권한이 서로 공존하고, 추가 권한이 기존 권한보다 확대되는 경우 추가 권한은 허가된다.

표 3. 강성-약성 권한 호환성 행렬

추가 / 기존	[IR]	([W]	[I-R]	[I-W]	[R]	[W]	[-R]	[-W]
(IR)	T	T	T	T	T	T	T	T
(IW)	T	T	T	T	T	T	T	T
(I-R)	T	T	T	T	T	T	T	T
(I-W)	T	T	T	T	T	T	T	T
(R)	F	F	F	F	F	T	F	T
(W)	F	F	F	F	F	F	F	F
(-R)	F	F	F	F	F	F	F	F
(-W)	F	F	F	F	T	F	T	F

또한, 강성 / 약성 권한의 정의에 따라 추가 권한이 기존 권한으로부터 유도되는 경우라도 추가

권한이 강성 권한인 경우 기존 권한인 약성 권한보다 우선(priority)되는 권한이기 때문에 두 권한간에 일관성만 있으면 추가 권한은 허가된다. 즉, 강성-강성 권한 호환성 행렬의 명시적-명시적 권한인 경우와 다른 결과를 갖는 것이다.

명시적-의도형 강성 권한일 경우 추가 권한은 모두 허가된다. 약성 권한의 정의에 따르면 기존 약성 권한에 의해 유도되는 묵시적 권한에 추가되는 권한은 기존의 묵시적 권한을 덮어쓸 수 있기 때문이다.

표 3의 강성-약성 권한 호환성 행렬은 기존 권한이 강성이고 추가 권한이 약성인 경우의 두 권한 유형간의 충돌 여부를 나타낸다. 의도형-명시적 권한인 경우의 충돌여부는 약성-강성 호환성 행렬의 명시적-의도형 권한인 경우와 동일한 결과를 갖는다. 또한, 의도형-의도형 권한인 경우의 충돌 여부는 강성-강성 호환성 행렬의 의도형-의도형 권한인 경우와 동일한 결과를 갖는다.

명시적-명시적 권한인 경우 두 권한이 서로 공존하거나 추가 권한이 기존 권한을 확대하는 경우에 추가 권한이 허가된다. 이 경우는 강성-강성 호환성 행렬의 명시적-명시적 권한인 경우와 동일한 결과를 갖는다. 명시적-의도형 권한인 경우 추가 권한은 모두 거절된다. 이 경우는 강성-강성 호환성 행렬의 명시적-의도형 권한인 경우와 동일한 결과를 갖는다.

표 4. 약성-약성 권한 호환성 행렬

추가 / 기존	[IR]	[IW]	[I-R]	[I-W]	[R]	[W]	[-R]	[-W]
[IR]	T	T	T	T	T	T	T	T
[IW]	T	T	T	T	T	T	T	T
[I-R]	T	T	T	T	T	T	T	T
[I-W]	T	T	T	T	T	T	T	T
[R]	T	T	T	T	F	T	F	T
[W]	T	T	T	T	F	F	F	F
[-R]	T	T	T	T	F	F	F	F
[-W]	T	T	T	T	T	F	T	F

표 4의 약성-약성 권한 호환성 행렬은 기존 권한이 약성이고 추가 권한도 약성인 경우의 두 권한 유형간의 충돌 여부를 나타낸다. 명시적-명시적 권한인 경우의 충돌 여부는 강성-강성 호환성 행렬의 명시적-명시적 권한인 경우와 동일한 결과를 갖는다. 또한, 명시적-의도형 권한인 경우의 충돌 여부는 약성-강성 호환성 행렬의 명시적-의도형 권한인 경우의 충돌 여부는 약성-강성 호환성 행렬의 명시적-의도형 권한인 경우와 동일한 결과를 갖는다. 그리고, 의도형-의도형 권한인 경우의 충돌 여부는 강성-강성 호환성 행렬의 의도형-의도형 권한인 경우와 동일한 결과를 갖는다.

의도형-명시적 약성 권한일 경우 추가 권한은 모두 허가된다. 이 경우는 강성-약성 호환성 행렬의 의도형-명시적 권한인 경우와 동일한 결과를 갖는다.

표1,2,3,4에서 제안된 권한 호환성 행렬은 약성 권한부여 개념을 필요로 하며, 약성 권한 부여 개념은 또한 부정적 권한부여 개념을 필요로 한다. 이 개념은 명시적인 권한 부여가 없는 주체는 객체에 대한 어떠한 권한도 없다는 가정에 기초하고 있다. 표 1,2,3,4,5에서 사용된 기호의 의미는 다음과 같다. ( ) 는 강성 권한을 [ ]는 약성 권한을 의미하며, - 표시는 부정 권한을 의미한다. R은 읽기 권한이며, W는 쓰기 권한, -R은 읽기 부정 권한이며, -W는 쓰기 부정 권한이다. T(True)는 권한 충돌이 없음을 F(False)는 권한 충돌이 있음을 나타낸다. 그리고, 의도형 권한은 I를 사용하여 나타낸다.

표 5. 의도형 권한

용 어	의 미
(IR)	의도 강성 긍정 읽기 권한
(IW)	의도 강성 긍정 쓰기 권한
(I-R)	의도 강성 부정 읽기 권한
(I-W)	의도 강성 부정 쓰기 권한
[IR]	의도 약성 긍정 읽기 권한
[IW]	의도 약성 긍정 쓰기 권한
[I-R]	의도 약성 부정 읽기 권한
[I-W]	의도 약성 부정 쓰기 권한

3.2 권한 부여 및 충돌 알고리즘

기존의 권한에 새로운 권한을 부여할 때 권한의

충돌 여부를 판정해야 한다. 묵시적 권한 부여 시 충돌 판정을 위해서는 권한을 부여하고자 하는 노드로부터 역으로 상위 노드로 거슬러 올라가는 방법으로 충돌을 판정한다. 그림3과 4는 묵시적 권한 부여에 대한 알고리즘이며, 그림5와 6은 의도형 권한 부여 기법에 대한 알고리즘이다. 그림3은 묵시적 권한 부여하의 권한 충돌을 판정하는 알고리즘을 기술한 것이고, 그림4는 DOM의 특정 노드를 액세스할 때 액세스가 가능한가를 판별하는 알고리즘이다. 또한, 그림5는 의도형 권한 부여 알고리즘이고, 그림6은 의도형 권한 충돌 알고리즘이다. 시스템에서의 권한은 기본적으로 3-tuple (s, o, a)로 정의한다. 여기서, s, o, a는 다음과 같다.

- s: 권한에 대한 주체(authorization subject)의 집합인 S의 한 원소 (사용자 혹은 사용자 그룹 등)
- o: 권한의 객체(authorization object)의 집합인 O의 한 원소 (DOM의 노드, 즉 텍스트, 엘리먼트, 엔터티 등)
- a: 권한의 타입(authorization type)의 집합인 A의 한 원소 (읽기, 쓰기 등)

그림3과 그림4의 알고리즘에서 사용한 기호는 다음의 의미를 갖는다.

- n : 권한을 부여하거나 액세스하고자 하는 노드에 대한 포인터
- (s_n, n, a_n) : 노드 n에 부여하거나 액세스할 권한
- p : 부모 노드에 대한 포인터
- (s_p, p, a_p) : 노드 p의 명시적 권한
- parent(n) : 노드 n의 부모 노드를 반환하는 함수

```
algorithm Conflict (n, (sn, n, an))
begin
1. p = parent(n)
2. 명시적 권한이 부여된 노드를 찾거나 최상위 노드
   까지 다음을 수행한다.
   2.1 p = parent(p)
3. 만약 p가 최상위 노드이고 명시적 권한이 없으면 p
   의 권한 (sp, p, ap)를 모든 경우 부정으로 설정한
   다.
4. 표 1,2,3,4에 의해 p의 권한 (sp, p, ap)와 n에 부
   여할 권한 (sn, n, an)의 충돌 여부를 판정한다.
end
```

그림 3. 권한 충돌 알고리즘(묵시적 권한부여)

```
algorithm AccessControl(n, (sn, n, an))
begin
1. p = n
2. 명시적 권한이 부여된 노드를 찾거나 최상위 노드 까지
   다음을 수행한다.
   2.1 p = parent(p)
3. 만약 p가 최상위 노드이고 명시적 권한이 없으면 p의
   권한 (sp, p, ap)를 모든 경우 부정으로 설정한다.
4. p의 권한 (sp, p, ap)와 액세스하고자 하는
   권한 (sn, n, an)를 비교한 후 노드 n에 대
   한 액세스 가능 여부를 결정한다.
end
```

그림 4. 액세스 제어 알고리즘(묵시적 권한부여)

그림5와 그림6의 알고리즘에서 사용한 기호는 다음의 의미를 갖는다.

- n : 추가로 권한을 부여하고자 하는 노드에 대한 포인터
- (s^n, n, a^n) : 노드 n에 부여할 명시적 권한
- (s', n, a') : 의도형 권한

```
algorithm Conflict (n, (sn, n, an))
begin
// 추가될 권한이 기존 권한과 충돌되지 않은 경우,
// 권한 부여를 허가하기 위하여 다음을 수행한다.
1. 노드 n의 모든 조상 노드들에 대하여 의도형
   권한 (s', n, a')을 삽입한다.
2. 노드 n에 명시적 권한 (s^n, n, a^n)을 삽입
   한다.
end
```

그림 5. 의도형 권한부여 알고리즘

```
algorithm Conflict (n, (sn, n, an))
begin
// 기존의 권한과 추가되는 권한과의 충돌을
// 판정하기 위하여 다음을 수행한다.
1. 표1,2,3,4에 의해 노드 n의 의도형 권한 (s', n, a')
   과 노드 n에 부여할 명시적 권한(s^n, n, a^n)과
   의 충돌 여부를 판정한다.
2. 충돌이 없으면 참, 있으면 거짓을 반환한다.
end
```

그림 6. 의도형 권한충돌 알고리즘

4. 분석 및 평가

본 논문에서 제시한 묵시적 권한 부여 기법을 사용하면 XML 문서의 액세스 제어를 문서를 구성하는 각 구성 요소 레벨에서 표현할 수 있다. 제안된 묵시적 권한 부여 기법은 모든 구성 요소에 대해 규칙을 기술하는 명시적 권한 부여 기법에 비해 메모리를 절약한다. 또한, 상위 구성 요소에 대한 권한 부여가 하위 구성 요소들에 동일한 권한을 자동적으로 부여함으로써 권한 부여 시간을 감소시킬 수 있다. 그러나, 권한 부여 시 권한 충돌을 탐지하는 계산과 특정 구성 요소에 대한 액세스 가능 여부를 계산해야 하는 오버헤드가 있다. 그림 3과 그림 4의 알고리즘의 시간 복잡도 (time complexity)는 DOM 트리의 노드 수가 n일 때 최악의 경우 $n-1$이지만 보통 DOM 트리가 균형을 이룬다고 가정하면 평균적으로 $\log n$이 된다. 따라서, 계산 시 고려해야 할 내부 노드의 개수는 전체 노드에 비해 상대적으로 적으므로 계산 시간의 오버헤드가 그렇게 심각하지는 않다.

대체 의도형 권한 부여 기법은 이와 같은 묵시적 권한 부여 기법의 충돌 판정 시 시간의 오버헤드를 줄이기 위한 방법으로 제안되었다. 제안된 대체 의도형 권한 부여 기법은 하위 노드를 탐색하여 권한의 정보를 얻는 묵시적 권한 부여 기법과는 달리 권한을 추가하려는 노드의 권한과 추가 권한의 권한 비교로 충돌 여부를 바로 판정할 수 있다. 하지만 충돌 여부판정 시간이 빠른 반면, 각 구성요소의 메모리가 자신의 권한에 대한 정보를 가지고 있어야 하므로, 메모리 사용이 증가되는 단점을 가지고 있다.

5. 결론

본 논문에서는 XML 문서의 액세스 제어를 위한 묵시적 권한 부여 기법과 대체 의도형 권한 부여 기법을 제안하였다. 제안된 기법은 XML 문서의 각 구성 요소 레벨에서 액세스 제어를 가능하게 한다. 묵시적 권한 부여 기법은 명시적 권한 부여 기법에 비해 메모리의 절약과 권한 부여 시간을 감소시킨다, 하지만, 권한 부여 시 권한 충돌 여부를 판정하고, 구성 요소 액세스 시 액세스 가능 여부를 검사하는 계산 오버헤드가 있다. 제안된 대체 의도형 권한부여기법은 비 중복성 원칙을 유지하면서, XML 문서의 각 구성 요소 레벨에서 액세스 제어를 가능하게 한다. 그리고, 묵시적 권한 부여 기법에 비해 권한 충돌 판정 시 시간을 감소시킨다. 하지만, 권한에 대한 정보를 가지고 있어야 하므로 메모리의 사용이 증가된다. 향후 묵시적 권한 부여 기법과 대체 의도형 권한 부여 기법을 적절히 활용한 XML문서에 가장 적합한 권한 부여 기법에 대해 연구할 예정이다.

감사의 글

본 논문은 정보통신부의 정보통신우수대학원지원 사업의 일부 지원으로 이루어졌음.

참 고 문 헌

[1] 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.

[2] Fernandez, E. B., Gudes, E., and Song, H., "A Model for Evaluation and Administration of Security in Object-Oriented Database," *IEEE Trans. on Knowledge and Data Engineering*, Vol.6, No.2, pp.275-292, 1994.

[3] 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.

[4] 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.

[5] 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, Sep. 1993.

[6] 손태종, 조완섭, 황규영, "객체지향 데이터베이스 시스템에서 계승을 통한 묵시적 권한부여 기법의 특성 분석," *한국정보과학회 가을 학술발표논문집*, vol.23, no.1, pp.131-134, 1996.

[7] David Hunter, *XML*, Information Publishing Group, Oct. 2000.

[8] IETF W3C, *XML-Signature Core Syntax and Processing*, Jan. 2000.

[9] W3C, *Extensible Markup Language (XML) Version 1.0*, http://www.w3.org/TR/REC-xml/, Feb. 1998.

[10] W3C, *Document Object Model Level1*, http://www.w3.org/TR/REC-DOM-Level-1/, 1998.