RBAC 기법의 성능을 고려한 XML 문서 접근제어 보안모델에 관한연구

김문석*, 이희조**
*고려대학교 컴퓨터 정보통신 대학원
ibkop@hitel.net

A Study on Security Models Access Control XML Document for Performance RBAC

Moon-Seok Kim*, Hee-Jo Lee**
*Dept of Graduate of School Computer and Information Technology,
College of Information and Communication, Korea University

요 약

XML(eXtensible Markup Language)은 문서구조나 내용, 스타일의 자유로운 표현, 웹상에서 구조화된 문서의 전송이 가능하도록 설계된 표준 마크업 언어로, HTML에 비해 인터넷상에서의 정보 및 문서교환, 정보 검색 등 이 편리한 장점을 갖는다. 그러나 XML 데이터가 방대해지고 복잡해짐에 따라 접근제어 정책을 규정하고 수행하기 위한 모델과 메커니즘의 필요에 따라 RBAC를 기반으로 하는 모델과 메커니즘에 대한 연구가 이루어지고 있다. 그러나 기존 연구들은 개념적 모델들 위주로 연구의 초점이 되어 있어 실제 모델 구현 시 시간적 공간적 오버헤드가 발생하는 단점이 있다. 따라서 본 연구에서는 XML 문서의 RBAC 기법을 기반으로 하는 노드의 권한 정보를 가지고 있는 접근 권한 트리 모델을 제안하였다. 이를 이용하여 노드권한에 모든 정보를 저장하고 권한을 가지는 구성요소에 접근할 경우나 권한을 추가로 부여하는 경우에 XML 엘리먼트, 속성, 링크 등의 세부요소에 대한 권한 체크 시간을 감소시켜 사용자에게 빠른 뷰를 제공할 수 있는 성능이 우수한 모델을 제시 하고자 한다.

1. 서론

XML(eXtensible Markup Language)은 문서구조나 내용, 스타일의 자유로운 표현, 웹상에서 구조화된 문서의 전송이 가능하도록 설계된 언어로, HTML에 비해 인터넷상에서의 정보 및 문서교환, 정보 검색 등이 편리한 장점을 갖는다. 이와 같은 장점 때문에 W3C 에서는 XML을 웹 데이터 표준으로 제정 했으며[1], SAML(Security Assertion Markup Language), XACML(XML Access Control Markup Language), XrML(eXtensible Rights Markup Language) 등과 같은 다양한 XML과 관련된 보안 기술들이 OASIS, IBM, Apache등과 같은 여러 기관 및 단체에 의해 연구가 진행되고 있다[2][3]. 그러나 기존 진행된 연구에서는 XML 문서에 포함되어 있는 각각의 요소에 대한 민감성의 수준을 정의하고 정의된 민감성 수준에 따라 문서의 각 요소에 대해 차별화된 보안정책을 적용하는 방식에 대해 초점이 집중되었을 뿐, 각 구성요소 간의 충돌 유무를 판별하는 권한체크 시간과 같은 사용자 중심의 성능을 고려하여 보안 정책을 적용하는 방식에 대해서는 아직도 기본적인 수준에서 연구가 진행되고 있다[4][5]. 따라서 본 연구에서는 RBAC 기존모델에의 공간 복잡도와 시간 복잡도 를 고려한 명시적 접근기법을 사용한 제안모델을 제시하여 기존 모델보다 시간과 공간 복잡도 에서 우수한 모델을 제시하고자 한다. 본 논문의 2장은 기존에 연구된 역할기반 접근제어 모델과 XML 문서를 위한 미세접근제어 기법을 통하여 XML 문서에서의 보안 요구사항 등에 대해 분석한다. 3장에서는 XML 문서를 위한 역할기반 접근제어 모델 제안으로 권한 부여 및 연산 등 각각의 특징과 XML 문서의 구성 요소가 가질 수 있는 권한정보를 저장하는 접근 권한 알고리즘을 제시한다. 4장에서는 제안된 모델의 성능을 분석하여 기존 모델과 비교 및 평가하여 정리한다. 5장에서는 결론과 향후에 진행되어야할 연구방향에 대해 정리한다.

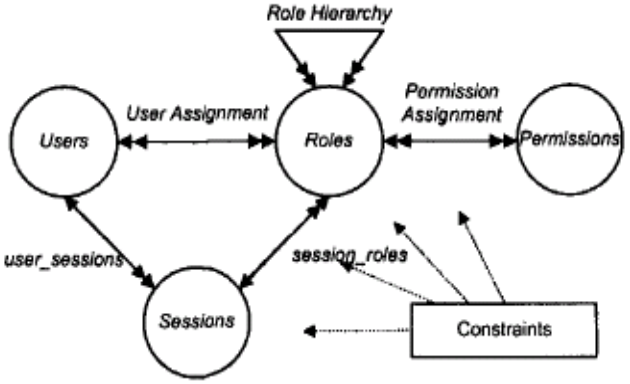
2. 관련 연구

본 장에서는 본 논문에서 제안된 접근제어 정책 모델인 RBAC의 기본 개념과 뷰 생성 시 발생한 시간 복잡도를 설명하였고, 2.2장에서는 제안방식의 기본 모델이 되는 Author-X의 XML을 위한 접근제어 수행방법과 한계점을 언급한다.

2.1 역할기반 접근제어

Ravi S.등에 의해 제안된 역할기반 접근제어의 중심적인 개념은 사용자가 기업이나 조직의 정보 자원을 임의로

접근할 수 없도록 하는 것이다. 즉, 접근권한이 역할에 부여되고 사용자는 적절한 역할에 소속됨으로써 역할의 수행에 필요한 최소 자원만을 접근할 수 있도록 한다. 이러한 개념은 권한 관리를 매우 단순화 시켜주고 기업의 특정한 보안정책 구현에서 유연성을 제공하는 장점을 가진다. 그림 1에서 RBAC 기본 모델을 보여준다. 기본모델은 사용자, 역할, 인가권한, 세션 으로 구성된다[6]. 기본적인 RBAC 모델에 역할계층구조, 제약조건이라는 특성이 추가될 수 있는데 역할계층(Role Hierarchy)은 권한과 책임에 대한 조직 내의 순서를 나타낼 수 있는 가장 일반적인 방법으로 트리구조로 나타내며 상위역할의 권한은 하위역할에 상속될 수 있다. 제약조건(Constraints)은 접근제어 정책이 실제 시스템 환경에 적용 가능하도록 하는 사전 규약들로 사용자 할당, 권한 할당, 그리고 접근제어 세션 에 적용된다.



(그림 1) 역할기반 접근제어 96

역할기반 접근제어의 경우 묵시적 권한기법에 중점을 두어 설계한 Damiani 모델을 기반으로 설계되었으므로 Damiani 모델의 뷰 생성 알고리즘을 사용하였을 경우를 가정하였을 때 시간 복잡도 는 다음과 같다. 사용자가 뷰를 요구할 때 마다 매번 뷰 생성 알고리즘의 set label 과 getfinal label 프로시저에서 두 번의 전체적인 트리 탐색(2n)과 최종적으로 권한이 부여되어진 트리만을 찾아가는 과정(n)이 이루어져야 하므로, 뷰를 만드는데 걸리는 시간은 최선의 경우 2n이고, 최악의 경우 3n이다. 최선의 경우는 권한을 가지는 최 상위 노드(root/node)가 부정("-")으로 지정되어 이후 트리 탐색이 이루어지지 못하는 경우이며, 최악의 경우는 모든 노드의 권한이 긍정으로 지정되어 뷰를 생성하는데 모든 노드를 탐색해야 하는 경우이다.

$$\frac{2n^2+(1+2+...+n)}{n}=\frac{2n^2+\frac{n(n+1)}{2}}{n}=2n+\frac{n+1}{2}=\frac{5}{2}n+1$$

$$O(\frac{5}{2}n+1)=O(\frac{5}{2}n)\ \ \therefore Complexity=\frac{5}{2}n$$

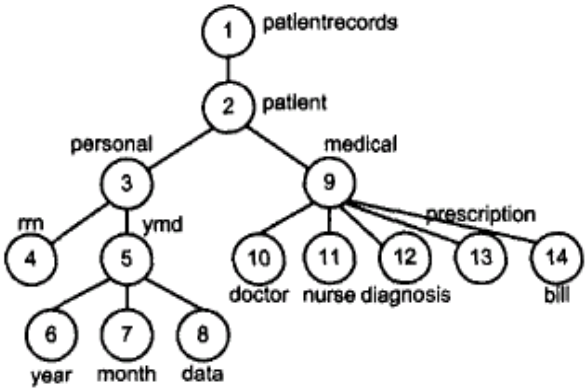
결과적으로 평균 $\frac{5n}{2}$ 만큼의 시간복잡도 를 갖고 XML 문서를 요청할 때 마다 매번 뷰 생성 알고리즘을 수행하여 뷰를 계산해야 한다.

2.2 Author-X

Author-X[7]는 사용자의 권한 등급에 따른 암호화된 키를 분배하여 미세 접근 제어를 시도한 시스템으로서, Bertino 에 의하여 만들어 졌으며, 기본적인 접근제어 방법은 해당 문서를 DOM 트리로 변환하여 각 노드에 대한 접근 권한을 명시하여 사용자가 요청한 권한과 비교 후, 접근이 승인된 노드 집합을 재구성하여 반환함으로서 해당 문서에 대한 적절한 접근제어를 수행하는 방법이다. Securing XML Documents에서 제안하는 접근제어 모델은 DOM 트리를 이용하여 엘리먼트에 접근권한을 설정하고, 설정된 접근권한 정보에 의해 사용자의 XML 문서의 접근을 제어한다[8][9][10].

```
<? XML Version="1.0" encoding="euc-kr"?>
<PatientRecords>
  <Patient Name="Alice">
    <Personal>
      <RRN> 750305-1234567 </RRN>
      <YMD>
          <Year>1975</Year>
          <Month>03</Month>
          <Date>05</Date>
      </YMD>
    </Personal>
   <Medical>
      <Doctor>KimJH</Doctor>
      <Nurse> BaeKM</Nurse>
      <Diagnosis> cold </Diagnosis>
      <Prescription> Chemo medicine </Prescription>
      <Bill> 100,000 </Bill>
   </Medical>
  </Patient>
</PatientRecords>
```

(그림 2) 객체 정보를 포함하는 XML 문서의 예



(그림 3) 그림 2에 대한 계층적 표현

그러나 이러한 방법들은 사용자의 요구에 대해 DOM 트리를 생성하므로 다수의 사용자가 동시에 동일한 데이터에 접근할 때 사용자가 요구하는 데이터에 대해 매번 DOM 트리를 생성해야 하는 단점이 있다.

3. XML 문서의 성능을 고려한 RBAC 모델

본 장에서는 본 논문에서 제안된 접근제어 모델인 RBAC를 기반으로 XML 문서에 접근제어를 적용하기 위한 접근제어 메커니즘을 기술한다. 권한 트리를 생성하기 위한 접근제어 메커니즘은 사용자 서비스 요청 과정, 사용

자 할당 과정, 허가할당 과정, 권한부여 과정, 권한트리 생성과정의 총 5단계로 구성된다.

3.1 사용자(User) 서비스 요청과정

사용자는 사전 등록 절차를 통해 ID와 Password를 부여 받았으며, 서비스 요청에 앞서 부여받은 ID와 Password로 인증(Authentication)된 상태라고 가정한다. 인증된 사용자는 XML 객체에 대한 서비스를 요청할 수 있는데, 그림 4와 같은 형태의 정보를 권한 부여 서버에 전송한다.

```
Input: {ID, Req-Target, Req-Operation}
```

(그림 4) 사용자 요청값

3.2 사용자 할당(User Assignment) 과정

역할기반 접근제어(Role-Base Access Control)정책에서 사용자는 하나 이상의 역할을 할당 받는다. 그림 5는 사용자 요청 정보(User-Req)의 ID값과 일치하는 사용자 정보 저장소의 역할 값을 읽어보는 과정을 보여준다.

```
①Input: {ID, Req-Target, Req-Operation}

②output: Role{ R1, R2 … , Rn }
```

(그림 5) 사용자 할당(User Assignment)

3.3 허가 할당(Permission Assignment) 과정

사용자는 사용자 할당 과정을 통해 하나 이상의 역할을 부여 받는다. 그림 6은 해당 역할에 접근권한부여 과정을 보여준다. ③은 사용자 할당 단계에서 사용자가 부여받은 역할집합,④,⑤,⑥은 특정 Rn에 할당된 권한 부여 객체와 연산에 대한 목록을 보여준다.

```
③Input: {ID, Req-Target, Req-Operation}

output:
④ PA(R1)={document(1)-root/element-1, read}
⑤ PA(R2)={document(2)-root/element-2, write}
 …
⑥ PA(Rn)={document(n)-root/element-n, delete}
```

(그림 6) 허가 할당(Permission Assignment)

3.4 권한부여(Authorization) 과정

그림 7은 사용자의 요청에 대한 권한 부여 과정을 나타낸다. ⑦은 첫 번째 입력 값으로 사용자 서비스 요청 과정에서 부여받은 값으로 요청 객체와 요청 연산 타입을 나타내고 ⑧,⑨,⑩,은 허가 할당 과정의 결과이고, ⑪은 첫 번째 입력 값과 두 번째 입력 값을 비교하여 동일 값이 있으면 권한부여를 허가하는 “+” 기호를 나타낸다. ⑬,⑭는 첫 번째 입력 값과 두 번째 입력 값을 비교하여 동일 값이 없으면 권한부여를 거부하는 “-” 기호를 나타낸다. ⑫,⑭는 부여된 명시적 권한을 권한정책 테이블에 저장한다. 테이블의 구성요소는 subject, object, action, sign 으로 구성되어 있다.

```
⑦ Input(1): {ID, Req-Target, Req-Operation}
⑧ Input(2): {document(1)-root/element-1, read}
⑨           {document(2)-root/element-2, write}
            …
⑩           {document(n)-root/element-n, delete}
⑪ output: if input(1) ∩input(2) ≠ ∅then
⑫ Table ⇐ {Req-Target, Req-Operation, +}
⑬         if input(1) ∩input(2)= ∅ then
⑭ Table ⇐ {Req-Target, Req-Operation, -}
```

(그림 7) 요청에 대한 권한부여(Authorization)

3.5 명시적 접근권한 트리 생성 과정

그림 8은 효율적으로 사용자에게 뷰를 제공하기 위한 접근 권한 트리 생성 과정을 나타낸다. ⑮는 권한정책 테이블로 subject는 user-group, ip-address, symbolic name로 이루어져 있다[1]. object는 권한이 표현되는 각 구성요소를 나타내며, action은 읽기라는 하나의 행위이고, sign은 +, -로 표현된다. ⑯,⑰은 권한정책을 대상으로 파싱하기 위한 XML문서이고, ⑱,⑲는 권한 정책 테이블에 명시된 명시적 권한을 접근권한 트리에 저장한다. ⑳은 접근권한이 설정되지 않은 경우 {public**,+}으로 설정하여 subject로 접속하는 모든 사람에게 공유한다.

```
⑮ Input(1): Table{ T1, T2 … Tn }
⑯ Input(2):{document(1)-root/element-1, read,+}
           {document(2)-root/element-2, write,-}
            …
⑰          {document(n)-root/element-n, delete,+}
⑱ output: if input(1) ∩input(2)= ∅then
⑲ Tree ⇐ {subject, object, action, sign}
         if input(1) ∩input(2) ≠ ∅ then
⑳ Tree ⇐ {public**.+}
```

(그림 8) 명시적 접근권한 트리생성(Tree Generation)

4. 제안된 기법분석

n을 XML 문서 내의 권한 정보를 가질 수 있는 전체 노드의 수라고 가정하면 제안된 기법의 시간 복잡도 는 다음과 같다. 제안된 기법은 권한에 대한 정보를 미리 계산하여 저장하여 둔 명시적 접근 권한 트리를 이용하므로 뷰를 생성하는 시간 복잡도 는 최선의 경우 1, 최악의 경우가 n이 된다. 이와 같은 결과가 나오는 이유는 최 상위

노드가 부정으로 지정되는 경우(최선의 경우 "-")와 전체 노드의 권한이 긍정으로 되는 경우(최악의 경우"+")를 가지기 때문이다. 그리고 제안된 기법은 접근 권한 트리를 생성하는 시간이 추가로 걸리는데 이 시간은 2n이다. 이유는 모든 권한 정보를 이용하여 만들어내는 접근권한 트리의 생성 알고리즘은 권한을 명시적으로 부여하는 시간과 부여한 권한간의 충돌 유무를 판별하는데 걸리는 시간 등 두 번의 전체 트리 탐색 시간이 소요되기 때문이다. 위의 수치만으로 볼 때 관련 연구에서 제시한 기존모델과 제안된 기법의 뷰 생성에 걸리는 시간 복잡도 는 비슷하다고 생각 할 수 있으나, 응용분야에 실제적인 적용에 있어서 제안된 기법은 기존 모델보다 뷰 계산을 현저히 감소시킬 수 있다. 이유는 권한에 관한 정책은 뷰를 요청할 때 마다 매번 바뀌는 것이 아니기 때문이다. 한번 정해진 정책은 자주 바뀌지 않으므로 권한 정보를 미리 계산하여 저장한 접근 권한 트리를 사용 하였을 경우 사용자가 뷰를 요청했을 때, 기존 모델은 평균 $\frac{5n}{2}$ 만큼의 시간이 걸리지만, 제안된 기법은 평균 $\frac{n}{2}$ 만큼 시간이 걸린다.

$$\frac{(1+2+\ldots+n)}{n}=\frac{\frac{n(n+1)}{2}}{n}=\frac{n+1}{2}=\frac{1}{2}n+1$$

$$O(\frac{1}{2}n+1)=O(\frac{1}{2}n)\therefore Complexity\frac{n}{2}$$

이상에서 볼 수 있듯이, 제안된 기법은 사용자의 요청에 대해 기존모델에 비하여 빠른 뷰를 제공한다.

5 결론

본 논문에서는 XML 문서가 웹 환경에서 안전하게 보호될 수 있도록 하는 접근제어 모델과 메커니즘의 권한 접근 성능을 고려하여 접근 권한 트리 모델을 제안 하였다. 기존 역할기반 접근제어 기법은 문서에 접근하고자 하는 사용자의 접근권한을 미리 설정된 보안정책에 따라 판단하여 해당 사용자가 가진 권한에 따라 해당문서에 대한 접근 범위를 결정한다. 이를 통해 전체 문서에 대한 접근 허용 또는 거부뿐만 아니라, 문서를 구성하는 각각요소에 대한 수준별 접근제어가 가능한 장점을 갖고 있다. 각 노드의 권한부여 하기 위해 사용되는 묵시적 권한 기법은 한 번의 권한 부여로 하위 구성 요소에 권한을 부여하는 효과를 가진다. 그러나 권한을 가지는 구성요소에 접근할 경우나 권한을 추가로 부여하는 경우 상위 구성요소와의 충돌의 유무를 판별하는 권한 체크 시간의 오버헤드가 크다는 단점을 가지고 있다. 본 논문에 제안된 명시적 접근 권한 트리 기법은 각 구성요소 마다 권한 정보를 가지고, 즉각적인 접근 제어를 할 수 있다. 따라서 뷰를 생성할 때 마다 매번 권한 정책을 파싱 하여 사용하는 기존 모델과는 달리 제안된 기법은 권한 정보를 미리 한번 파싱 하여 권한 정책 테이블에 저장하고 뷰를 계산할 때 사용 하므로 기존모델 보다 권한 체크 시간이 짧다. 그러므로 권한 체크가 자주 발생하는 전자상거래나 병원관리 등의 응용분야와 같이 XML문서에 대한 권한 정책이 자주 수정되지 않고, 읽기에 대한 비율이 쓰기에 대한 비율보다 현저하게 많은 응용분야에 제안된 기법을 적용가능 하며 기존 기법에 비해 더 높은 성능을 기대할 수 있다. 향후 추가적으로 연구되어야 할 부분은 XML 문서에 대한 접근제어 수행 시 발생할 수 있는 예외처리 및 제약조건에 대한 정의와 규정에 대한 연구와 관계형 데이터베이스를 이용하여 실제적인 구현을 통하여 제안된 모델의 효율성과 가용성에 대해 평가 하는 작업이 이루어질 예정이다.

참고문헌

[1] www.w3c.org, "eXtensible Markup Language(XML) 1.0", W3C Recommendation, 04 Febutuary 2004.
[2] www.w3c.org, "XML-Signature Syntax and Processing," W3C Recommendation, 12 February 2002.
[3] OASIS, "eXtensible Access Control Markup Language Version 1.1," 24 July 2003.
[4] Bertio, E. B, Gudes, E. and Song. H, "An Extended Authorization Model for Relational Database." IEEE Trans. on Knowledge and Data Engineering, Vol..9, No 1, pp85-101. 1997.
[5] Rabitti, F et al.,"A Model of Authorization for Next-Geneqldbfn Database Systems." ACM Trans on Database Systems.16, No 1, pp.88-131.1991.
[6] Ravi S. Sandhu, E,J.Coyne, and H.L.Feinstein, "Role-base Access Control Modesl," IEEE Computer, Vol. 29, No. 2, pp. 33-47, 1996.
[7] E.Bertino, S.Castano, and E.Ferriari. "Securing XML Documents with Author-X', IEEE Internet Computing, Vol5, No.3, pp.21-31, 2001.
[8] C. G. Pollmann, "XML Poll Encryption," XMLSEC02, USA, pp.1-9, 22, Nov, 2002.
[9] E.Damiani, C. Vimercati, S. Paraboshi, P. Samarati, "Securing XML Documents," EDBT 2000, Germany, pp. 27-31, June, 2000.
[10] 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 No2, pp.169-202. 2002.