XACML 정책 충돌 탐지 알고리즘

김재진, Scott Uk-Jin Lee
한양대학교 컴퓨터공학과
jaejinkim@hanyang.ac.kr, scottlee@hanyang.ac.kr

Conflict Detection Algorithm for XACML Policies

JaeJin Kim, Scott Uk-Jin Lee
Department of Computer Science & Engineering, Hanyang University

요 약

XACML은 OASIS에서 공표한 접근제어 정책언어의 표준이며 뛰어난 확장성으로 여러 환경에서 효과적인 접근제어를 가능하게 한다. 특히, 자원의 양이 방대하고 사용자의 증감이 빈번하여 세밀한 접근제어가 필수적인 여러 컴퓨팅 환경에서 효과적인 접근제어 솔루션을 제공할 수 있다. 하지만 XACML은 잘못된 설계로 인한 정책 충돌의 발생 위험성 또한 가지고 있으며, 이는 의도하지 않은 권한부여를 가능하게 한다. 정책충돌은 XACML을 접근제어에 효과적으로 활용하기 위하여 반드시 풀어야 할 문제이다. 따라서 본 연구는 XACML의 정책 충돌에 대하여 분석하고, 정책 충돌 탐지를 위한 알고리즘을 제안한다.

1. 서 론

접근 제어 (Access Control)는 클라우드 컴퓨팅 (Cloud Computing), 빅 데이터 (Big Data) 등과 같이 방대한 양과 다양한 종류의 자원을 활용하는 IT 분야에서 매우 중요한 기술이다. 특히, 클라우드 컴퓨팅과 같이 모든 사용자가 개개인의 자원에 대한 권한설정이 가능한 환경은 세밀하고 유동적인 접근제어가 필수적이다. 이를 가능하게 하기 위하여 OASIS 국제표준위원회는 확장 가능한 접근제어 정책 언어인 XACML (eXtensible Access Control Markup Language)을 표준으로 채택하였다 [1]. XACML은 XML기반 접근제어 정책 언어로써 급격하게 변화하는 다양한 IT 환경에서 유동적이며 세밀한 접근제어가 가능하도록 개발되었다. 이러한 장점을 가진 XACML은 현재 대부분의 대규모 분산시스템의 정책 언어로 사용되고 있는 실정이다 [2].

XACML은 다양한 환경에서의 활용도를 높이기 위하여 자유로운 정책의 생성을 허용하는 반면, 이에 따른 정책 충돌의 문제점 역시 가지고 있다. XACML로 정책을 생성하는 과정에서 신중한 설계와 철저한 분석 없이는 충돌이 발생할 수 있으며, 이는 XACML의 정책-조합알고리즘 (Policy-Combining Algorithms)과 규칙-조합알고리즘 (Rule-Combining Algorithms)의 순차적인 정책 계산 과정으로 인하여 사용자가 의도하지 않은 권한부여가 가능하게 되는 문제를 초래한다. XACML이 다양한 환경에서 좀 더 세밀하고 유동적인 접근제어를 제공하기 위해서는 이러한 충돌의 탐지와 해결은 필수적이지만 현재까지 관련 연구는 매우 미흡한 편이다.

2010년 Florian Huonder[3]는 사용자의 요청에 해당하는 정책의 모든 요소들을 N-차원 공간에 사각형으로 매핑하여 서로 겹치는 정도에 따라 충돌을 탐지하는 방법과 이와 같은 오버랩을 제거하여 충돌을 해결하는 알고리즘을 제안하였다. 하지만 위 연구에서 고안한 방법으로는 정책을 나타내는 사각형이 일부 오버랩되는 경우까지 모두 고려하여 충돌이 잘못 탐지되는 결과가 나타날 수 있다.

2010년 Kamalbir Singh외 1명[4]은 정책의 Target 요소에 속한 Subject, Resource, Action, Environment 요소들의 모든 값을 순차 탐색으로 비교하는 접근법을 제안하였다. 하지만 순차 탐색을 사용한 단순한 값의 비교는 방대한 양의 정책에 적용하기에는 계산 속도가 너무 느리며, 정책이 가지고 있는 요소의 모든 값이 중복되지 않고도 일어날 수 있는 정책의 충돌은 탐지하지 못하는 단점이 있다.

2011년 Stefan Oberholzer[5]는 Florian Huonder의 연구를 확장하여 부분 정책 충돌 탐지 알고리즘과 정책의 Target 요소의 최적화 알고리즘을 제안하였다. 하지만 이 방법 또한 N-차원 공간에 정책의 오버랩 정도를 기준으로 충돌 여부를 결정하기 때문에 잘못된 충돌의 탐지를 허용한다.

관련연구에서 볼 수 있듯이 XACML의 정책 충돌 탐지 방법을 제안하는 연구는 다수 진행되었다. 하지만 정책의 모든 요소들의 값을 비교하는 방법과 정책의 모든 요소의 매치함수, 속성, 데이터타입, 값들을 N-차원 공간에 매핑하는 방법은 정책 충돌을 정확하게 탐지하기에는 아직 미흡하며, 다양한 컴퓨팅 환경에 적용하기에 접합하지 않다. 따라서 본 연구에서는 XACML을 사용하여 접근제어가 구현된 경우 발생할 수 있는 접근제어 정책의 모든 충돌 패턴을 분석하고, 이를 기반한 정책 충돌 탐지 알고리즘을 고안하였다.

2. XACML 3.0 정책 아키텍처

그림 1. XACML Policy 구조

XACML Policy 는 접근제어 정책이 기술되는 요소이며 위 그림 1에서 보여지듯 접근제어 상황을 나타내는 Target 과 권한부여를 결정하는 Rule 로 구성되어 있다. Target 은 PolicySet, Policy, Rule 내에 포함되어 있으며, 사용자의 접근권한을 결정하기 위하여 필요한 주체 (Subject), 자원 (Resource), 작업 (Action), 환경 (Environment)의 요

소를 가질 수 있다. 그리고 Condition 요소는 논리 연산을 지원하여 더욱 세밀한 접근제어를 구현하기 위하여 존재한다.

```
<Target>                                  <!– 1 –>
  <AnyOf>                                 <!– 0 | 1 –>
    <AllOf>                               <!– 1..N –>
      <Match>                             <!– 1..N –>
        <AttributeValue />                <!– 1 –>
        <AttributeDesignator />           <!– 1 –>
      </Match>
    </AllOf>
  </AnyOf>
</Target>
```

그림 2. XACML Target 구조

Target 은 위의 그림 2와 같이 AnyOf, AllOf, Match, AttributeValue, AttributeDesignator 요소를 포함하고 있으며, 이 중에서 자원에 대한 사용자의 접근요구 (Request)와 정책을 비교할 때 실제 사용되는 요소는 Match 이다. Target 의 결과값은 Match 의 내용에 따라 달라지게 되는데, Target 은 1개에서 N 개까지의 Match 를 포함할 수 있으며 AttributeDesignator 와 AttributeValue 를 이용하여 Subject, Resource, Action, Environment 의 카테고리와 값을 비교한다.

3. XACML 정책 충돌

XACML은 모든 경우에서 접근제어가 가능하도록 하기 위해 아무런 제약 없이 정책을 생성할 수 있도록 한다. 그러므로 XACML 정책을 생성할 때, Target의 4가지 요소를 이용하여 무수한 패턴의 정책 생성이 가능하다. 또한 XACML은 확장성을 높이기 위하여 정책의 중복 생성도 허용하며, 다수의 규칙 또는 정책을 빠르게 연산하는 조합알고리즘을 제공함으로써 사용자의 요구사항에 해당하는 정책이 중복되는 경우 하나의 승인결과를 반환한다.

이처럼 정책 생성에 제약이 없고 중복된 정책의 생성을 허용하는 XACML에서는 접근제어 정책의 설계 시 주의를 기울이지 않으면 정책들 사이에서 쉽게 충돌이 일어날 수 있고 이를 탐지하는 것은 매우 어렵다. 하지만 모든 충돌 가능성을 탐지하는 알고리즘 개발은 현재까지 미흡한 편이며, 의도하지 않는 권한부여가 일어나게 되어 심각한 문제를 일으킬 수 있는 위험이 있다. 이러한 문제를 해결하기 위해서 본 연구에서는 XACML 정책 충돌의 패턴을 먼저 분석 한다.

기본적인 XACML 정책의 충돌은 정책에 기술된 4개의 요소 중 Subject, Resource, Action가 오버랩이 되며 서로 다른 승인결과 (Permit/Deny)를 가지는 정책이 존재할 경우 발생한다. 이는 XACML 정책의 충돌이 일어날 수 있는 모든 경우를 나타내며, 이를 확장하여 ABAC, RBAC, 그리고 Hybrid 접근제어 모델에서 정책 충돌이 일어날 수 있는 모든 경우를 나타낼 수 있다. XACML 정책의 충돌은 아래와 같이 정의 할 수 있다.

3-Element 충돌: 2개의 정책의 Subject, Resource, Action 요소가 모두 오버랩 되며, 승인결과가 다른 경우

ABAC 정책 충돌: 3-Element 충돌에 추가적으로 Environment 요소가 오버랩 되어 충돌이 발생하는 경우

RBAC 정책 충돌: 한 사용자가 2가지 이상의 역할을 소유한 상황에서 각 역할의 접근권한 정책들 사이에 3-Element 충돌이 발생하는 경우

Hybrid 정책 충돌: RBAC 정책 충돌의 경우에 ABAC 정책 충돌의 경우가 더해져서 충돌이 발생하는 경우

다음으로 위의 패턴들이 어떻게 ABAC, RBAC, 그리고 Hybrid 접근제어 모델에서 정책의 충돌을 나타내는지 예를 들어 자세히 설명한다. 아래 예제에 해당하는 실제 XACML Policy 파일은 다음 URL 에서 찾아볼 수 있다. http://selab.hanyang.ac.kr/xacml

```
PolicySet {PolicyCombiningAlg=Deny.-Override
  Policy {RuleCombiningAlg=Deny-Override, Target {Resource=Course.pdf}
    Rule A [Effect=Permit] {Subject=User A, B, C, Action=Download}
    Rule B [Effect=Deny] {Action=Download, Environment=18:00~24:00}}}
```

그림 3. ABAC 접근제어 모델에서 정책 충돌의 예

그림 3은 클라우드 스토리지 서비스에서 사용자의 파일 접근에 관한 XACML 정책을 축약하여 나타낸 것으로, 정책이 포함하는 2가지의 규칙의 충돌을 표현한 것이다. Rule A 는 사용자 A, B, C 가 Course.pdf 파일을 다운로드 하는 것을 허가, Rule B 는 같은 파일을 모든 사용자가 18시부터 24시 사이 다운로드 하는 것을 거부한다. 위 정책은 Rule A 와 B 의 Subject, Resource, Action, Environment 요소가 오버랩 되며 승인결과가 다른 경우를 나타내는 ABAC 기반 정책 충돌에 해당된다. 여기서 Rule B 의 Subject 는 명세 되어 있지 않지만 XACML 은 이러한 경우 Any 의 의미를 부여하기 때문에 AnySubject (모든 사용자)를 나타내며 Rule A 의 Subject 와 오버랩이 된다. XACML 은 Action, Resource, Environment 등 다른 요소에 대해서도 마찬가지로 직접적인 값이 주어지지 않은 경우 Any 의 의미를 부여한다. 그러므로 Rule A 와 B 의 Environment 요소 역시 오버랩이 된다. 위 정책의 규칙-조합 알고리즘은 다수의 규칙에서 Deny 승인결과를 우선으로 반환하는 Deny-Override 이므로 Rule A 와 B 모두에 해당되는 접근요구에는 항상 Rule B 가 매치되어 접근이 거부된다. 만약 Rule A 가 B 보다 더 높은 우선권을 가지고 있다면 이는 의도하지 않은 권한의 부여이다.

```
PolicySet {PolicyCombiningAlg=Deny-Override
  PolicySet (UserRole_PhD) {PolicyCombiningAlg=Deny-Override
    Target {Subject=Tom} PolicyIdReference=Role_TA, Role_Student)
  PolicySet (Role_TA) {PolicyCombiningAlg=Permit-Override
    Permission_TA (Policy) {RuleCombiningAlg=Permit-Override
    Rule [Effect=Permit] {Resource=MidTermGrade.xlsx, Action=Edit}}}
  PolicySet (Role_Student) {PolicyCombiningAlg=Deny-Override
    Permission_Student (Policy) {RuleCombiningAlg=Deny-Override
    Rule [Effect=Deny] {Resource=MidTermGrade.xlsx, Action=Edit}}}
```

그림 4. RBAC 접근제어 모델에서 정책 충돌의 예

그림 4는 박사과정 학생인 Tom 이 조교와 학생의 역할을 동시에 소유하고 있어 두 역할이 같은 자원에 대하여 가지는 접근권한의 충돌을 나타낸 것이다. 위의 예에서, 첫 번째 정책집합은 사용자의 역할을 기술하며 나머지 두 정책집합은 역할별 접근권한을 기술한다. Role_TA 정책집합은 조교의 MidTermGrade.xlsx 파일 수정을 허가하는 반면 Role_Student 정책집합은 학생의 파일 수정을 거부한다. 위 정책집합 역시 Deny-Override 정책-조합알고리즘을 사용하며, Tom 의 접근요구에는 UserRole_PhD 정책집합의 PolicySetIdReference 요소 중 항상 Role_Student 정책집합이 매치되어 접근이 거부된다. 만약 학생의 역할보다 조교의 역할에 우선권이 있다면 위의 예제 역시 의도하지 않은 권한부여를 하게 된다. 또한 위와 같은 BRAC 접근제어 모델에 Environment 요소가 추가되면 이를 Hybrid 접근제어 모델이라 하며 여기서 일어나는 충돌 역시 의도하지 않은 권한부여를 가능하게 한다.

앞서 설명한 것과 같이, 사용자가 새로이 생성한 정책이 이미 존재하는 다른 정책과 충돌을 일으킨다면 의도하지 않는 권한부여를 가능하게 하는 심각한 문제를 초래한다. 하지만 사용자가 XACML 의 구문으로 표현된 여러 정책을 비교하여 충돌을 탐지 하는 것은 매우 어렵다. 따라서 올바른 접근제어를 제공하기 위해서는 정책의 생성시 앞서 설명한 패턴에 따른 충돌의 효과적인 탐지는 필수적이다.

4. 충돌 탐지 알고리즘

이번 장에서는 앞서 정의한 XACML 정책의 충돌 패턴들을 탐지하는 알고리즘을 의사코드로 제안하고 이를 단계별로 설명한다.

```
FOR each policy in policyList
   CALL Match_TargetSrc with policyTarget, src RETURNING result
   IF result = TRUE THEN
      FOR each rule in ruleList
         IF srcEffect != ruleEffect THEN
            CALL Match_TargetSrc with ruleTarget, src RETURNING result
         IF result = TRUE THEN RETURN CONFLICT ELSE RETURN NOCONFLICT
FUNCTION Match_TargetSrc
   result := FALSE
   FOR each src in srcList
      FOR each target in targetList
         IF srcCategory = targetCategory AND srcValue = targetValue THEN
            result := TRUE
         ELSE IF srcCategory != targetCategory AND targetCategory = null THEN
            result := TRUE
         ELSE result:= FALSE
   RETURN result
```

알고리즘 1. 3-Element 충돌 탐지 알고리즘

위 알고리즘은 다음과 같이 3-Element 충돌을 탐지한다.

1. 먼저 policyLiist 의 각 Policy 마다 Match_TargetSrc 함수를 호출하여 정책 생성에 필요한 요소를 나타내는 src 와 Policy Target 의 요소를 나타내는 target 을 서로 비교한다.
2. 비교연산은 src 와 target 이 완전히 같으면 TRUE 를 반환하고, 다르면 다시 target 을 검사하여 null 이면 TRUE 를 반환한다.
3. 2번 과정의 결과가 TRUE 일 경우, ruleEffect 와 srcEffect 를 비교하여 값이 다르면, Match_TargetSrc 함수를 호출한다. 그리고 함수가 반환하는 결과가 TRUE 일 경우 CONFLICT, 아니면 NOCONFLICT 를 반환한다.

위 알고리즘은 충돌 탐지 속도를 향상시키기 위하여 충돌을 결정하는 비교연산을 최소화하고 충돌을 결정하는 특징을 우선적으로 검사하기 때문에 정책 생성시 빠른 자동 충돌 탐지를 가능하게 한다. 그리고 Environment 요소를 더불어 비교하면 ABAC 접근제어 모델에서의 정책 충돌 역시 쉽게 탐지 가능하다.

```
FOR each policySet in policySetList
   IF targetCategory = "Subject" AND targetValue = src THEN
      FOR each childNode in childNodeList
         IF childNodeType = "PolicySetIdReference" THEN
            policySetIdList.add(childNodeValue)
result := FALSE
IF policySetList.length >= 2 THEN
   FOR each foundPolicySetId in policySetIdList
      FOR each policySet in policySetList
         IF policySetId = foundPolicySetId THEN
            foundPolicySetList,add(policySet)
   CALL Match_PolicySet with foundPolicySetList RETURNING result
IF result = TRUE THEN RETURN CONFLICT ELSE RETURN NOCONFLICT
```

알고리즘 2. RBAC 모델의 정책 충돌 탐지 알고리즘

RBAC 정책 충돌 탐지는 사용자가 2개 이상의 역할을 가지는 지 먼저 검사하여 실제 충돌을 일으키는 정책들을 추출하고 앞서 설명한 3-Element 충돌 탐지 알고리즘을 이용하여 다음과 같이 충돌을 탐지한다.

1. 먼저 policySetList 를 탐색하여 각 PolicySet 의 targetCategory 가 Subject 이고 target 과 src 의 값이 같으면, 해당 PolicySet 의 모든 자식노드를 탐색한다.
2. 자식노드가 PolicySetIdReference 요소라면 그 값인 PolicySetId 를 policySetIdList 에 저장한다. 이렇게 policySetIdList 에 저장된 PolicySetId 가 2개 이상이면, policySetList 를 다시 탐색하여 저장된 PolicySetId 에 해당하는 PolicySet 을 모두 찾은 후 foundPolicySetList 에 저장한다.
3. 저장된 PolicySet 들의 충돌을 검사하기 위여 Match_PolicySet 함수를 호출하고 반환된 결과가 TRUE 이면 CONFLICT, 아니면 NOCONFLICT 를 반환한다. Match_PolicySet 함수는 중첩된 PolicySet, Policy, Rule 에 포함된 모든 Target 을 비교하기 위하여 앞서 고안한 Match_TargetSrc 함수를 이용한다.

위 알고리즘은 RBAC 접근제어 모델에서 정책의 충돌을 쉽고 빠르게 탐지하며 Environment 요소만을 추가 비교연산 하면 RBAC 와 ABAC 의 Hybrid 모델에서 역시 정책 충돌의 탐지가 가능하다. 이처럼 본 연구에서 고안한 충돌 탐지 알고리즘은 기존에 모든 접근제어 모델에 적용 가능하고, 기존의 알고리즘에 비하여 탐지 속도가 개선되었고, 확장성을 고려한 설계로 더 복잡하고 다양한 접근제어 정책의 충돌 탐지가 가능하다. 따라서 XACML 을 사용하는 모든 환경 및 시스템은 본 연구에서 고안된 알고리즘을 이용하여 접근제어 정책의 모든 충돌을 탐지할 수 있다.

5. 결론 및 향후 연구 방향

클라우드 컴퓨팅과 같이 방대한 양의 자원과 빈번한 사용자의 증감이 있는 환경에서는 세밀한 접근제어를 필수로 하며 접근제어 언어의 표준인 XACML은 뛰어난 확장성으로 이러한 환경에 널리 적용되고 있다. XACML은 정책생성이 자유로운 반면 정책 충돌의 문제점 또한 가지고 있는데 이는 효율적이며 올바른 접근제어를 구현하기 위하여 반드시 해결되어야 할 문제이다. 하지만 기존의 관련 연구 대부분이 일부분의 정책 충돌 가능성만 탐지하거나 잘못된 충돌까지 탐지하는 등 아직까지 미흡한 편이다. 따라서 본 논문은 XACML의 정책 충돌의 문제를 분석하여 충돌 패턴을 확립하였으며 이를 해결하기 위하여 효율적인 충돌 탐지 알고리즘을 제안하였다.

향후 연구로는 본 논문에서 제안한 충돌 탐지 알고리즘을 구현하여 접근제어 정책을 생성할 때 충돌의 가능성을 알려주고 해결방안을 제시할 수 있는 반 자동 충돌 탐지 솔루션을 개발하려고 한다. 또한 여러 실제 사례에 알고리즘을 적용하는 실험을 통하여 충돌 탐색 속도 및 정확성을 분석하고 고안한 알고리즘을 최적화 하고자 한다.

6. 참고 문헌

[1] Bo Lang, Nan Zhao, Kun Ge and Kai Chen, "An XACML Policy Generating Method Based on Policy View", In Proceedings of the 3rd International Conference on Pervasive Computing and Applications, pp.295-301, 2008

[2] Abd EL-Aziz Ahmed Abd EL-Aziz and A, Kannan, "Access Control for Healthcare Data using Extended XACML-SRBAC Model", In Proceedings of the 3rd International Conference on Computer Communication and Informatics, pp.1-4, 2012

[3] Florian Huonder, "Conflict Detection and Resolution of XACML Policies", Master's Thesis, University of Applied Sciences Rapperswil, 2010

[4] Kamalbir Singh and Sarbjeet Singh, "Design and Evaluation of XACML Conflict Policies Detection Mechanism", International Journal of Computer Science & Information Technology, Vol.2, No.5, pp.65-74 2010

[5] Stefan Oberholzer, "Optimizing XACML Policies", Master's Thesis, University of Applied Sciences Rapperswil, 2011