XACML 정책 작성시 요청에 따른 효과적인 정책 평가 요인 수집 방법

오용택 ○ 이욱진

한양대학교 컴퓨터공학과

ka123ak1@gmail.com, scottlee@hanyang.ac.kr

How to collect effective policy evaluation factors upon request when creating XACML policies

YongTaek Oh ○ Scott Uk-Jin Lee

Department of Computer Science and Engineering, Hanyang University

요 약

IoT 환경이 확대됨에 따라 접근 제어에 대한 이슈도 계속해서 떠오르고 있다. 접근 제어 분야에서 충돌은 반드시 해결되어야 하는 이슈이며 다양한 해결 방법들이 연구되고 있다. 하지만 현재로썬 정책 충돌은 정책 작성자가 해결해야하는 영역이며 정책 충돌을 해결하기 위해서 정책 평가 결정에 영향을 주는 정책 및 조건에 대한 정보를 효과적으로 수집해야 할 필요가 있다. 본 논문에서는 요청에 따라 평가에 영향을 주는 정책 및 조건에 대한 정보를 파악하기 위한 효과적인 방안을 제시한다.

1. 서 론

최근 인공지능 스피커 출시 등으로 인해 IoT 환경이 매우 급격하게 확장되고 있다. IoT 디바이스는 스마트 TV, 스마트 냉장고, 스마트 에어컨 등 계속해서 늘어나고 있으며 IoT 플랫폼 역시 여러 기업에서 디바이스가 플랫폼에 종속되지 않도록 하는 oneM2M과 플랫폼이 등장하고 있다. 삼성전자 LG 등의 국내 기업들 또한 참여하고 있으며 SKT는 oneM2M 기반의 IoT 플랫폼을 개발하여 서비스를 제공하고 있다.

한편, IoT 플랫폼과 디바이스가 급격하게 증가함에 따라 보안에 대한 이슈가 증가하고 있는데 그 중 IoT 기기가 증가함에 따라 접근 제어 정책이 뚫리게 되는 사건이 증가하고 있다[1]. 접근 제어는 정책을 통해 자원을 관리하는 것인데 IoT 환경에서는 디바이스 및 자원의 수가 매우 많아 관리하기가 어려워지기 때문에 충돌이 더욱 많이 생기게 된다. 이러한 접근 제어를 위한 정책 언어로 XACML(eXtensible Access Control Markup Language)이 사용되는데 XACML은 ABAC 정책을 작성하기 위한 언어, 아키텍처 및 접근 요청을 평가하는 방법을 정의하고 있다. 하지만 XACML은 정책을 작성할 때 정책 간의 충돌(논리적 충돌, 의미적 충돌)을 고려하여 작성하게 되어있지 않기 때문에 충돌이 빈번하게 발생하게 된다. 그 때문에 자원의 수가 매우 많은 IoT 환경에서 접근 제어 충돌은 더욱 중요한 사항이 되고 있다.

접근 제어 충돌을 탐지하기 위한 연구로는 SAT 문제를 생성하여 해결하고자 하는 방식과 그 방식을 확장한 SMT 기반의 분석 방식이 있다[2]. 하지만 아직 실험 단계에 머무르고 있다. 다른 연구로는 요청을 평가한 다음 기계 학습을 이용하여 정책 속성을 추론함으로써 의미적인 충돌을 해결하고자 한 연구가 있다[3]. 이 경우 정책을 비슷하게 만들어냄으로써 의도에 맞지 않는 충돌을 탐지하는 데 기여했지만 RBAC에 한정된 한계를 가지고 있다. 또한 접근 제어 정책을 충돌을 고려한 IoT 플랫폼에 대한 연구가 진행되고 있으며[4] 정책 충돌을 고려하여 정책의 문맥을 전달하기 위한 XACML 조언 구문(AdviceExpression)을 자동화하여 생성하려는 노력[5]도 있었다. 하지만 수집한 정보를 표현하는 방법에 대해서는 다루지 않았고 원하지 않는 추가 정보도 노출될 수 있다. 그 때문에 아직 접근 제어 충돌을 위한 연구는 부족하며 많은 연구가 필요한 시점이다. 본 논문에서는 사용자가 정책을 작성할 때 정책 충돌을 인지하고 정책 평가 시 평가의 요인이 되는 정책 및 조건들을 효과적으로 수집할 수 있는 방법을 제안한다.

본 논문의 구조는 다음과 같다. 2장에서는 정책을 평가하고 분석하기 위해 XACML 구조를 분석한다. 3장에서는 시나리오를 바탕으로 XACML 정책 평가를 분석하고 그 결과를 바탕으로 정책의 평가 요인을 제공하는 방법을 제안한다. 4장에서는 본 논문의 결론과 향후 연구에 대해 논의한다.

2. XACML 구조 분석

본 논문에서는 다른 연구[4]에서 언급된 정책 충돌이 일어나는 상황을 보여주는 시나리오를 바탕으로 정책을 작성하고 평가 요인을 찾아내고 제공하는 방법을 제시한다. 2장에서는 평가 요인을 찾기 위해 정책의 구조를 분석한다.

그림 1은 충돌 시나리오를 바탕으로 작성한 정책을 간단하게 나타낸 것이다. 그림 1을 통해 XACML 정책의 구조를 확인할 수 있는데 XACML은 전체적으로 Target과 해당 Target에 대한 제어로 이루어져 있다. 요청은

정책(Policy) 혹은 정책 집합(PolicySet)을 통해 평가가 이루어진다. 정책 집합은 정책과 다른 정책 집합으로 이루어져 있다. 또한, 정책들의 결과를 조합하는 조합 알고리즘을 가지고 있다. 정책은 여러 개의 규칙(Rule)으로 이루어져 있고 규칙들의 결과를 조합하는 조합 알고리즘을 가지고 있다. 규칙은 조건(Condition)으로 이루어져 있는데 조건은 여러 개의 Apply로 구성되어 있다. Apply는 다시 여러 Apply로 이루어져 있으며 함수를 통해 조건을 평가한다.

Apply를 통해 규칙을 평가하게 되는데 표 1은 규칙의 Target과 조건의 결과에 따른 Truth table을 나타낸다. Truth table은 True/False의 결과를 나타내는 테이블이지만 본 논문에서는 XACML 정책의 결과인 Permit/Deny/Not Applicable/Indeterminate를 나타내는 결과와 그 결과에 영향을 미치는 Target의 Match 여부, 규칙 혹은 조건의 결과 등을 나타내는 것으로 한다. 조건의 결과는 규칙의 결과에 영향을 미치며 규칙의 결과를 바탕으로 정책의 결과를 평가하는데 정책의 결과는 규칙 조합 알고리즘에 따라 결정된다.

정책의 결과는 정책 집합 결과의 바탕이 되는데 규칙과 마찬가지로 정책 조합 알고리즘으로 조합되어 결정된다. XACML에서는 5종류의 기본 조합 알고리즘을 제공하고 있는데 그 종류는 다음과 같다.

- **Deny overrides** : 거부가 하나라도 반환될 경우 거부
- **Permit overrides** : 승인이 하나라도 반환될 경우 승인
- **First applicable** : 처음으로 반환된 결과를 반환
- **Ordered combining** : 정의된 순서를 고려하는 Deny overrides, Permit overrides.
- **Deny unless permit and Permit unless deny** : 승인 혹은 거부의 반환이 없으면 기본값으로 반환

- 표 1 – Rule truth table

Target	*Condition*	*Rule*
Match	True	Effect
Match	False	Not applicable
Match	Indeterminate	Indeterminate
No-match	True	Not applicable
No-match	False	Not applicable
No-match	Indeterminate	Not applicable

한편, 평가에 대한 추가정보를 조언 구문을 이용하여 삽입할 수 있으며 조언 구문을 이용하여 요청을 평가할 때 평가에 영향을 준 정책, 규칙의 문맥을 얻을 수 있다. 하지만 모든 요인을 수집하기 위해선 모든 정책 집합, 정책, 규칙에 조언 구문이 필요하게 되는데 이는 매우 비효율적이다.

3. 효과적인 XACML 평가 요인 제공 방법

그림 1에 나타낸 스마트카 시나리오에서 정책 충돌은 정책

```
Policy 1:
  if role contains family: return Permit
  else: return Deny

Policy 2(for son):
  if assist is not father: return Deny
  else: return Permit

Policy 3(for son):
  if hour >= 9 and hour <= 16: return Permit
  else: return Deny

Result:
  if Deny in {Policy 1, Policy 2, Policy 3}: return Deny
  else if Permit in {Policy 1, Policy 2, Policy 3}: return Permit
  else: return Not Applicable or Indeterminate
```

그림 1. 스마트카에 대한 정책 예시

표 2 – Policy truth table (deny-overrides)

Rule1 (Permit)	*Rule2 (Permit)*	*Result*
Permit	Permit	Permit
Permit	Deny	Deny
Deny	Deny	Deny
Not applicable	Permit/Deny	Permit/Deny
Permit/Deny	Not applicable	Permit/Deny
Indeterminate	Indeterminate	Indeterminate

표 3 – Policy truth table (deny-unless-permit)

Rule 1 (Permit)	*Rule 2 (Permit)*	*Result*
Permit	Permit	Permit
Permit	Not applicable	Permit
Not applicable	Not applicable	Deny
Indeterminate	Indeterminate	Deny

집합의 조합 알고리즘이 deny overrides이기 때문에 일어난다. 구체적으로, 위의 정책에서 아들이 오후 6시에 아버지와 함께 스마트카를 타고 시동 권한을 요청하게 되면 Policy 2에 의해 요청은 거부된다.

위의 정책에서 아버지가 정책의 충돌 여부 혹은 정상 작동을 확인하기 위해 정책을 평가하려고 한다고 가정하자. 기존의 방법으로는 위의 모든 집합, 규칙 각각에 조언 구문을 삽입해야 하는 노력을 거친 뒤 요청을 보내 어떤 정책 혹은 규칙이 결과에 영향을 주었는지 수집해야 한다. 하지만 평가에 영향을 주는 정책 혹은 규칙은 조합 알고리즘을 통해 알 수 있다. 만약 조합 알고리즘이 deny-overrides일 경우 반환된 결과가 Deny인 정책 혹은 규칙이 우선으로 결과에 영향을 주게 되고 Deny가 없다면 Permit이 영향을 끼치게 된다. 이러한 정책 혹은 규칙의 결과는 Truth table을 통해 간단하게 확인할 수 있다. 표 2는 조합 알고리즘이 deny-overrides일 경우, 표 3은 조합 알고리즘이 deny-unless-permit일 경우를 나타낸다.

간단한 Truth table을 통해 결과에 영향을 미치는 정책 혹은

규칙을 판별할 수 있기 때문에 요청에 대해 영향을 주는 정책을 쉽게 찾아낼 수 있다. 그림 2는 정책을 트리 형태로 나타낸 것이다. 트리에서 루트는 최종 평가 조합 알고리즘이 되며 편집자가 예상한 평가 결과와 이 결과가 다를 경우 의도와 맞지 않는 정책으로 인한 충돌로 해석할 수 있다. 즉, 최상위 노드인 Deny overrides 노드의 결과가 예상 결과와 다를 경우 하위 노드 3개의 결과를 보고 결과에 영향을 주는 노드를 찾아낼 수 있다. 같은 방법을 반복하여 최하위 노드의 결과를 알 수 있고 과정 중 의도와 맞지 않게 작동한 지점이 어느 지점인지를 알 수 있다.

정리하면 트리를 통해 정책 사용자는 다음과 같이 자신의 의도와 맞지 않는 부분을 발견할 수 있다.

1. 전체 결과가 예상 결과와 일치하는지 확인한다.
2. 결과가 예상 결과와 다를 경우 그 원인이 되는 자식 노드를 찾는다.
3. 2번을 최하위 노드까지 탐색한 후 의도에 맞지 않게 된 원인이 되는 노드를 인지한다.
4. 인지된 노드 혹은 다른 노드를 의도에 맞도록 정책을 수정한다. (필요의 경우 truth table을 참조할 수 있다)

시나리오의 경우 아버지가 동승 중이기 때문에 예상 결과는 Permit이지만 실제 결과는 Deny인 것을 보고 충돌이 있음을 알 수 있다. 조합 알고리즘이 Deny overrides인 것을 확인하고 자식 노드에서 Deny 결과를 가지는 것을 찾는다. 자식 노드 중 왼쪽 노드의 결과가 Deny임을 찾고 그 과정을 계속하면 최종적으로는 18 <= 16 조건이 False이기 때문에 Deny임을 알 수 있다.

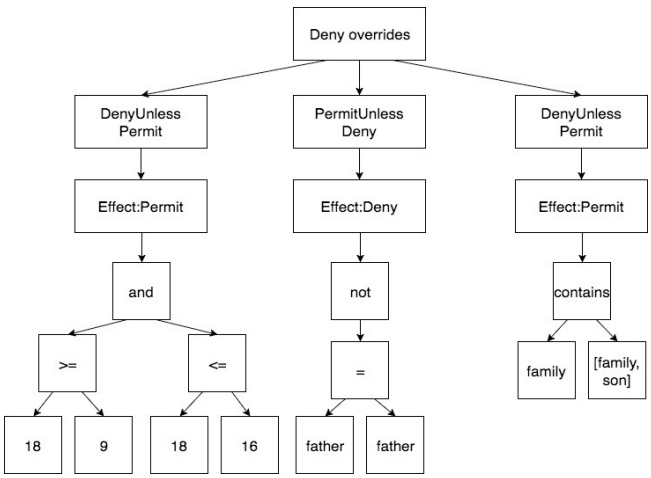

그림 2 – 정책 평가 요인 예시

이 방법을 통해 사용자는 정책에 대한 추가 정보를 효과적으로 수집할 수 있다. 정책에 대한 제약이 없고 사용자의 추가적인 노력이 필요하지 않게 된다. 또한, 작성 시 판단할 수 있는 방법이기 때문에 외부로 원하지 않는 추가 정보를 노출할 걱정이 없다.

4. 결론 및 향후 연구

IoT 환경에서는 디바이스 수가 매우 많기 때문에 접근 제어 정책을 작성하는 것은 매우 어려운 일이다. 또한, XACML 언어는 사용자에게 충돌에 대한 대안을 마련해주지 않기 때문에 IoT 환경에서는 기존 접근 제어 환경보다 더 많은 정책 충돌을 일으킬 수 있고, 이는 치명적인 보안 사고를 불러올 수 있다. 본문에서는 정책을 분석하고 요청에 대해 정책의 평가 요인을 트리를 이용하여 제공하는 방법을 제안하였다. 이 방법은 정책의 조합 알고리즘에 따른 Truth table을 기반으로 정책을 표현하는 것으로 이루어진다. 이렇게 표현된 정책과 평가 요인을 바탕으로 정책 작성자는 자신의 의도와 맞지 않는 정책 충돌에 대하여 필요한 정보를 얻을 수 있다.

하지만 의도에 부합하지 않는 조건이 어디에 있는지 찾아내는 것은 별도의 문제이다. 시나리오에서 18 <= 16 조건이 False라는 것이 의도와 다른 것이 아니다. 따라서 다시 위로 탐색하며 의도에 맞지 않는 부분을 찾으면 최종적으로 자식 노드에는 문제가 없지만, 그 결과를 조합하는 것이 Deny overrides라는 것에 문제가 있음을 알 수 있다. 이처럼 사용자는 별도로 의도에 맞지 않는 정책이 어느 부분인지 찾아야 하는 한계를 가진다.

향후 연구에서는 사용자의 요청을 바탕으로 하는 평가할 때 다른 시나리오에서도 충돌을 쉽게 확인할 수 있는지 사례연구를 하며 평가하고 충돌을 일으키는 시점이 사용자의 의도 측면에서 어느 시점인지 좀 더 명확하게 알 방안을 연구할 예정이다.

사사

이 성과는 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-22016R1C1B2008624)

참고 문헌

[1] ‘IoT’ 해킹에 미국이 뚫렸다, 김학재 <파이낸셜뉴스>, 2016/10/23 17:45, https://goo.gl/5oKD91

[2] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone, “Analysis of XACML Policies with SMT,” in Principles of Security and Trust, ser. LNCS 9036. Springer, 2015, pp. 115-134.

[3] Martin, Evan, and Tao Xie. "Inferring access-control policy properties via machine learning." Policies for Distributed Systems and Networks, 2006. Policy 2006. Seventh IEEE International Workshop on. IEEE, 2006.

[4] 이기찬, 이욱진. (2016). 접근 제어 정책의 충돌을 고려하는 IoT 플랫폼 제안. 한국정보과학회 학술발표논문집, , 300-302.

[5] 안윤근, 이기찬, 이욱진. (2017). XACML 정책 평가에 영향을 미치는 문맥적 요소 및 추가 정보의 효과적인 수집 방안. 한국정보과학회 학술발표논문집, , 549-551.