ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 24, No. 2, pp. 82-87, 2018. 2
https://doi.org/10.5626/KTCP.2018.24.2.82

XACML 정책 평가에 영향을 미치는 문맥적 요소 및 추가 정보의 효과적인 수집 방안

(Effective Methodology for Collecting Contextual Factors and Information that Affects The XACML Policy Evaluation)

안 윤 근 † 이 기 찬 † 이 욱 진 ††
(Youn-geun Ahn) (Gichan Lee) (Scott Uk-Jin Lee)

요 약 접근 제어 분야에서 정책 충돌 문제는 반드시 해결되어야 하는 이슈이며, 이를 위한 다양한 해결 방법들이 연구 및 개발되고 있다. 정책 충돌을 해결하기 위해서는 충돌 원인을 먼저 확인해야 하며, 이를 위한 최소한의 조건으로써 정책 평가 결정에 영향을 준 정책의 문맥적 요소를 탐지할 필요가 있다. XACML 정책 언어 명세에 이를 정의하는 기능이 있으나 정책 작성자가 모든 문맥적 요소마다 일일이 정의해야 하는 한계를 가진다. 또한, 정책 충돌의 원인 파악을 위해서는 문맥적 요소 이외에 다른 정책 조합 알고리즘과 같은 추가적인 정보도 알아야 할 필요가 있다. 본 논문에서는 정책 충돌의 원인이 되는 문맥적 요소 및 추가 정보를 파악하기 위한 효과적인 방안을 제시한다.

키워드: 접근 제어, XACML, 정책 충돌, XACML 조언 구문

Abstract In the field of access control, policy conflicts must be solved and various related solutions are being researched and developed. In order to resolve the policy conflict problem, it is necessary to first identify the cause of the conflict, and as a minimum condition, it is necessary to detect the contextual elements of the policy that have influenced the policy evaluation decision. Although the XACML policy language specification provides a way to define this, the policy creator currently has limitations in not clearly describing the causes of conflicts in every contextual elements. In addition, in order to identify the causes of the policy conflict, it is necessary to acquire additional information such as other policy combination algorithms, in addition to these contextual factors. In this paper, we propose an effective method to identify contextual factors, as well as to locate additional information that cause policy conflicts.

Keywords: access control, XACML, policy conflict, XACML advice expression

· 이 성과는 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-22016R1C1B2008624)
· 이 논문은 2017 한국컴퓨터종합학술대회에서 'XACML 정책 평가에 영향을 미치는 문맥적 요소 및 추가 정보의 효과적인 수집 방안'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 한양대학교 대학원 컴퓨터 공학과
frebern@hanyang.ac.kr
fantasyopy@gmail.com
†† 종신회원 : 한양대학교 ERICA 소프트웨어학부 교수
(Hanyang Univ.)
scottlee@hanyang.ac.kr
(Corresponding author임)

논문접수 : 2017년 9월 1일
(Received 1 September 2017)
논문수정 : 2017년 11월 23일
(Revised 23 November 2017)
심사완료 : 2017년 12월 7일
(Accepted 7 December 2017)

정보과학회 컴퓨팅의 실제 논문지 제24권 제2호(2018. 2)

1. 서 론

최근 미래를 주도할 주요 IT 기술로써 사물 인터넷, 클라우드 컴퓨팅, 머신 러닝 및 인공지능 분야가 주목을 받고 있다. 각 기술은 생활 속에서 사람들의 삶과 밀접하게 연관되어 다양한 서비스를 제공하고 있으며, 기업의 경영이나 정보 관리에도 사용되고 있다. 기업은 사용자의 생활 패턴이나 주소, 선호도, 종교 등의 사적인 정보들을 수집하고 클라우드나 빅데이터 기술을 사용하여 서비스에 활용한다. 기업은 이러한 중요한 정보들을 보호하고자 하는 시장의 요구에 따라 자신들의 서비스에 접근 제어(access control) 기술을 적용한다[1].

접근 제어란 어떤 자원에 접근할 때 적합한 사용자인지 여부를 검사하여 접근을 승인(permit)하거나 거부(deny)하는 과정을 말한다. 접근 제어에서 접근 요청을 평가하기 위해 정책(policy)을 사용한다. 여기서 정책이란 어떤 사용자가 권한을 부여받을 수 있는지에 대한 규칙들을 정의한 것을 말한다. 하나의 접근 요청을 평가하기 위해 복수의 정책을 적용할 수 있다. 이때, 정책들은 서로 다른 결과를 반환할 수 있으며 이러한 경우를 정책 충돌(policy conflict)이라고 할 수 있다. 정책 충돌은 정책 작성자가 의도하지 않는 접근 제어 결정을 야기할 수 있으며 이는 중요한 정보를 부적절한 사용자에게 보여주는 치명적인 보안 문제로 이어질 수 있다.

최근 접근 제어 분야에서는 정책 작성을 위해 XACML (eXtensible Access Control Markup Language) 정책 언어를 사용한다. XACML에서는 정책 충돌을 크게 5가지의 정책 및 규칙 조합 알고리즘으로 해결하지만, 이는 단순한 논리 연산에 불과하며 모든 정책 충돌 문제에 적용하기는 어렵다[2].

정책 충돌을 탐지하거나 해결하기 위한 다양한 연구[2-4]가 있으나 대부분 암묵적으로 정책 충돌에 연관된 문맥적 요소들을 이미 알고 있다고 가정한다. 그러나 정책 충돌을 탐지하거나 해결하기 위해서는 우선 어떤 상황에서 충돌이 발생할 수 있는지 알아야 하며, 이는 다른 말로 충돌에 관여하는 구체적인 속성 및 문맥 정보를 수집해야 한다는 것을 의미한다. 예를 들어, 누가 언제 어디서 어떤 자원에 접근할 때 어떤 정책들이 접근 제어에 적용되며 각 정책의 결과는 무엇인지 등의 다양한 정보를 수집해야 정책 작성자가 정책 충돌의 원인을 파악하고 해결할 수 있게 된다.

현재, XACML 정책 언어에 관련된 연구에서는 아직 정책 평가시의 추가 정보를 수집하는 방안은 이루어진 바가 없으며, 이에 따라, 본 논문에서는 정책 충돌의 탐지 및 해결을 위한 연구의 기반 연구로써 정책 평가시 문맥적 요소 및 추가 정보를 파악하기 위한 간단하고 효과적인 방안을 제시한다.

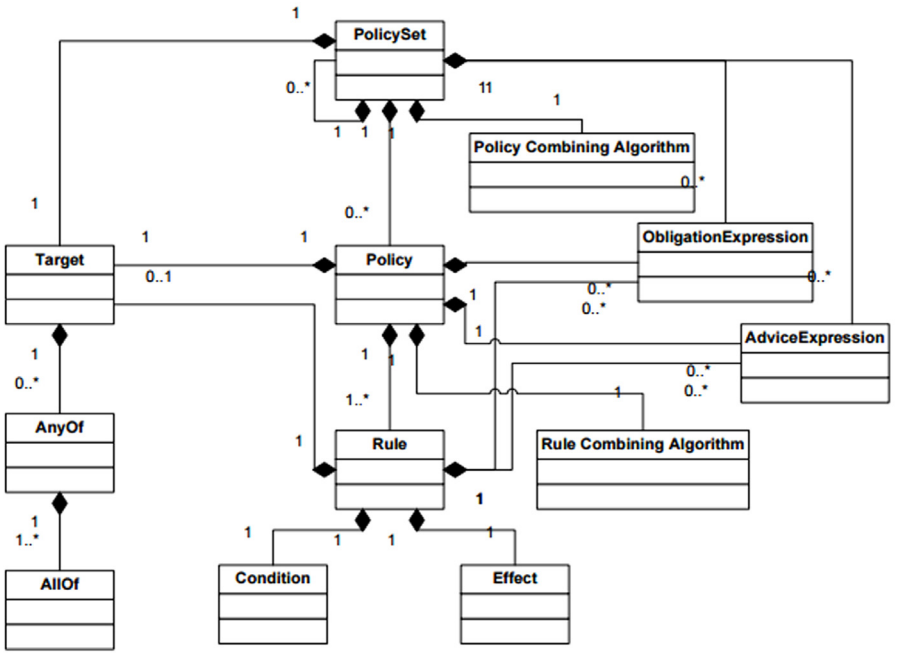

그림 1 정책 언어 모델[5]
Fig. 1 Policy Language Model

본 논문의 구성은 다음과 같다. 2장에서는 XACML 정책 언어의 구조에 대해 간단하게 살펴보고 정책 평가 관련 정보를 수집하면서 어떤 문제점들이 존재하는지 논의한다. 3장에서는 2장에서 살펴본 문제점들을 해결하기 위한 문맥적 요소 및 추가 정보를 효과적으로 수집하는 방법을 구체적으로 다루고 필요성에 대해 논의한다. 4장에서는 앞서 다룬 내용을 정리하고 본 논문에서 제안하는 방법이 접근 제어 분야에 어떻게 기여할 수 있을지와 향후 연구 방향에 대해 논한다.

2. 배경 및 문제 정의

그림 1은 XACML[5] 정책 언어의 각 구문의 포함 관계를 나타내는 모델을 클래스 다이어그램으로 표현하고 있다. 하나의 정책 집합(PolicySet)은 여러 개의 정책 집합 혹은 정책(Policy)을 포함하고, 하나의 정책은 여러 개의 규칙(Rule)을 포함한다.

각 규칙은 속성의 평가를 위한 조건절(Condition)과 조건절이 참일 때, 규칙이 반환하는 결과(승인, 거부 등)가 무엇인지 나타내는 영향(Effect)을 정의한다. 정책 평가에 따른 추가 정보를 제공하기 위한 조언 구문(AdviceExpression)은 정책 집합, 정책, 규칙 각각에 삽입될 수 있다.

XACML에서는 정책 평가에 영향을 준 정책이나 규칙의 추가 정보를 제공하기 위해 조언 구문을 명세한다. 조언 구문은 표현식을 통해 정책 집합이나 정책 혹은 규칙의 평가 결과에 대한 추가적인 정보를 PEP(Policy Enforcement Point)에 제공한다. 이를 통해 PEP에서는 정책 평가 결정이 어떤 정책 혹은 규칙에 의해서 그리고 어떤 문맥 조건하에서 이루어졌는지 파악할 수 있다. 그러나 이를 위해서는 정책 작성자가 정책 집합과 정책 그리고 규칙마다 조언 구문을 일일이 기술해야 하는 상당한 노력이 요구된다. 만약 10개의 정책 집합에 각각 10개의 정책이 있고 정책마다 10개의 규칙이 있다면 정책 작성자는 추가 정보를 제공하기 위해 1,110개의 조언 구문을 별도로 기술해야 한다. 물론 사물 인터넷이나 클라우드 플랫폼에서의 접근 제어에는 이보다 훨씬 더 많은 정책과 규칙이 필요할 것이며 이에 대한 조언 구문을 정책 작성자가 일일이 기술하는 것은 불가능에 가깝다. 모든 규칙에 대해 조언 구문을 기술해야 할 필요는 없으나 기술할 필요가 있는 규칙과 필요 없는 규칙을 구분하는 것 역시 상당한 작업량을 요구한다.

또한, PEP에서 조언 구문을 활용하기 위해 정책 작성자는 조언 구문을 일관된 형식으로 작성할 필요가 있다. 그림 2는 한 규칙에 조언 구문을 기술한 예시이다. <AdviceExpressions>부터 조언 구문의 시작이며 내용으로 추가 정보에 대한 표현식을 기술한다. 그림 2에서는 규칙의 승인 결과에 대한 추가 정보로써 “Permit: You can start car if your are the family member.”라는 일반 텍스트를 기술하고 있다.

```
<Rule RuleId="family-rule" Effect="Permit">
  <Condition>
    ...
  </Condition>
  <AdviceExpressions>
    <AdviceExpression
      AdviceId="permit-family-advice" AppliesTo="Permit">
      <AttributeAssignmentExpression AttributeId="text">
        <AttributeValue DataType="string">
Permit: You can start car if you are the family member.
        </AttributeValue>
      </AttributeAssignmentExpression>
    </AdviceExpression>
  </AdviceExpressions>
</Rule>
```

그림 2 조언 구문의 예시
Fig. 2 Advice Expression Example

이 조언 구문에는 9시부터 16시라는 조건(Condition)과 이 규칙이 이 조건을 만족할 때 승인이라는 결과(Effect)를 반환한다는 내용이 작성자 나름의 형식으로 기술되어 있다. PEP가 이 조언에서 필요한 정보를 추출하기 위해서는 문장을 자르고 자연어 처리를 통해 9~16시라는 조건을 얻어야 하는 어려운 작업을 수행해야 한다. 물론 정책 작성자가 자연어로 표현하지 않고 나름의 형식을 사용하여 조언을 작성할 수도 있다. 그러나 작성자마다, 혹은 접근 제어를 적용하고자 하는 도메인마다 기술하는 정보의 형식이 달라질 수 있으며 이는 조언 구문 형식의 불일치가 야기하게 된다. 결국, PEP에서는 일관성 없는 형식의 조언 구문을 처리해야 하는 문제를 갖게 된다.

3. 해결 방안

본 논문에서는 정책 평가 결정과 관련된 추가 정보를 조언 구문에 담아 정책 파일의 모든 정책 집합, 정책 그리고 규칙마다 자동으로 삽입하는 방법을 제시한다.

그림 3은 본 논문에서 제시하는 방법을 도식화하여 간략하게 설명한다. 정책 작성자가 정책을 정의하면,

① 정책을 분석하기 위해 해당 정책 파일에 대해 XML 구문 분석을 수행하고 DOM 객체를 생성한다.

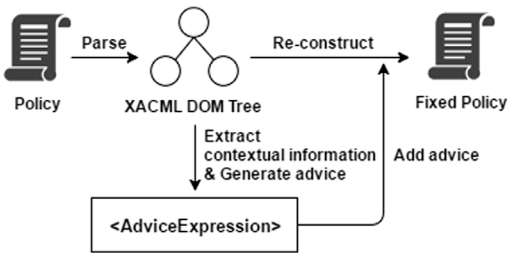

그림 3 조언 구문 삽입 과정
Fig. 3 Advice Expression Insertion Progress

② 생성한 DOM 객체를 바탕으로 정책 집합과 정책 그리고 규칙을 분류하고 관련 문맥 정보를 정리한다.

③ 각 정책 집합과 정책과 규칙 요소마다 정책 평가와 관련된 정보를 XACML 표현식으로 변환하여 조언 구문(AdviceExpression)을 생성하고 각 요소에 삽입한다.

- 정책 집합에서는 해당 정책 집합의 대상(Target)과 정책 집합의 ID와 자식 정책 혹은 정책 집합의 ID, 그리고 평가에 사용된 정책 조합 알고리즘을 평가와 관련된 정보로 취급한다.
- 정책에서는 해당 정책의 대상과 정책 ID와 자식 규칙의 ID, 그리고 규칙 조합 알고리즘을 관련 정보로 취급한다.
- 규칙은 해당 규칙의 대상과 규칙 ID, 그리고 사용된 조건식(Condition)과 영향(Effect)을 관련 정보로 취급한다.

만약 별도로 작성된 AdviceExpression 구문이 있다면 해당 조언 구문의 내용을 추가로 기술한다.

그림 3의 과정을 거치고 나면 각 PolicySet, Policy, Rule 요소들은 각각 하나의 AdviceExpression 구문을 포함하게 된다. 각 AdviceExpression 구문은 부모 요소가 검사하는 문맥적 요소(정책의 조건식(Condition), 대상(Target) 등)의 표현식을 기술하며, 필요하다면 정책 외적 요인인 추가 정보(예를 들어, 접근 요청이 들어온 IP나 지역, 접근 요청 시간 등)도 XACML의 표현식으로 기술할 수 있다. 이러한 표현식들은 PDP로 XACML 접근 요청이 들어오고 정책을 평가하는 접근 요청이 갖는 속성값에 따라 계산이 이루어진다. 위와 같이 삽입된 AdviceExpression 구문들을 통해 PEP는 접근 요청에 따른 정책 평가와 관련된 정보만을 PDP로부터 수집할 수 있게 된다.

아래에서는 실제 정책 예제를 통해 앞서 서술한 조언 구문 삽입 절차가 실제로 어떻게 진행되는지를 단계별로 설명한다.

그림 4는 앞서 설명한 과정을 거쳤을 때 단계별로 어떻게 처리되는지를 보이기 위한 XACML 정책 예시이다. 가독성을 위해 설명에 불필요한 몇몇 속성들과 각 속성값의 URI와 URN은 간소화하였다.

위 정책은 'car'라는 자원을 대상으로 적용되는 정책이며, 각 규칙의 결과는 'deny-unless-permit' 규칙 조합 알고리즘(규칙 평가 결과 중 승인이 없다면 거부를 반환)으로 조합된다. 'FamilyRule' 규칙은 역할 속성의 값 중에 'family'가 포함된 경우 승인을 반환한다. 예를 들어, 만약 'car' 자원에 역할이 'family'인 사용자가 접근을 요청한다면 'FamilyRule' 규칙이 허용을 반환하여 정책도 허용을 반환하고, 'family'가 아닌 사용자라면 어느

```
<Policy
  PolicyId="FamilyPolicy" RuleCombiningAlgId="deny-unless-permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">car</AttributeValue>
          <AttributeDesignator
            AttributeId="resource" DataType="string"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Rule RuleId="FamilyRule" Effect="Permit">
    <Condition>
      <Apply FunctionId="string-is-in">
        <AttributeValue DataType="string">family</AttributeValue>
        <AttributeDesignator AttributeId="role" DataType="string"/>
      </Apply>
    </Condition>
  </Rule>
</Policy>
```

그림 4 XACML 정책 예시
Fig. 4 XACML Policy Example

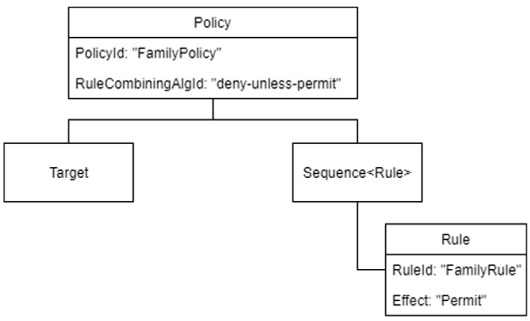

그림 5 정책 요소의 DOM 트리
Fig. 5 Policy DOM Tree

규칙도 허용을 반환하지 않으므로 규칙 조합 알고리즘에 의해 거부를 반환한다.

우선, 조언 구문 삽입의 첫 단계로 정책을 구문 분석하여 DOM 객체를 생성할 필요가 있다. 여기서는 각 요소의 의미보다는 DOM 구조에 주목할 필요가 있다. 우선 Policy 요소를 DOM Tree 형태로 표현하면 아래의 그림 5와 같다.

XACML에서 하나의 Policy는 복수의 Rule을 포함할 수 있는데 XACML 3.0 명세[5]에는 이를 Policy가 Rule의 Sequence를 가지는 것으로 정의되어있다.

Target 요소의 DOM Tree는 그림 6과 같이 표현된다. 여기서 Match 요소는 표현식을 정의하기 위한 함수(MatchId)를 속성으로 가지며 자식 요소로 해당 함수에 필요한 인자를 포함한다. XACML 3.0 명세[5]에 의하면 Match 요소는 인자로 반드시 하나의 AttributeValue와 하나의 AttributeDesignator 혹은 AttributeSelector 요소를 포함해야 한다.

그림 7은 예시 정책의 Rule 요소를 표현한다. 예시 정책의 Rule에는 Condition만 정의되어 있으나, Policy와 마찬가지로 Target을 지정할 수 있으며 그 외에도

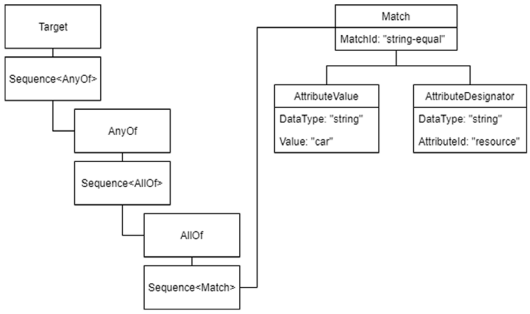

그림 6 타깃 요소의 DOM 트리
Fig. 6 Target DOM Tree

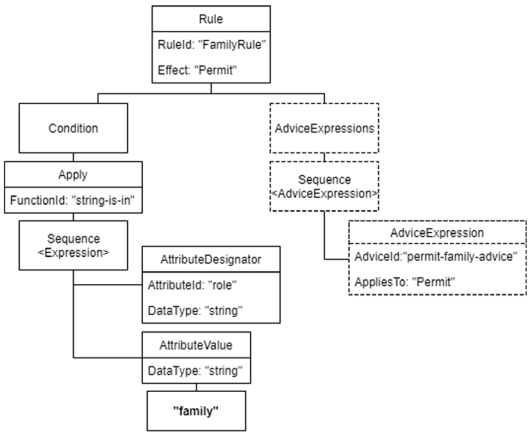

그림 7 규칙 요소의 DOM 트리
Fig. 7 Rule DOM Tree

몇몇 다른 요소들을 정의할 수도 있다. Condition 요소는 반드시 하나의 Expression 요소를 포함해야 한다. 여기서 Expression은 추상 요소로 직접 정책에 삽입될 수 없으며, 대신 이를 상속받는 요소인 Apply, AttributeValue, Function 등의 요소가 삽입될 수 있다. 그림 7의 Advice-Expressions은 다음의 문맥 정보 추출 및 조언 구문 표현식 생성 단계에서 생성되고 정책에 삽입된다.

그림 5~7과 같이 정책을 DOM 객체로 변환한 후 해야 하는 일은 생성한 DOM 객체를 정책 집합과 정책 그리고 규칙을 분류하고 각 요소에 관련된 요소를 추출하는 작업이다. 예시 정책은 설명의 편의를 위해 하나의 정책과 규칙만을 작성하였다. 앞서 조언 구문 삽입 과정에서 언급한 관련 정보를 토대로 문맥 정보를 표 1, 표 2와 같이 정리할 수 있다.

여기서 의미 분석의 편의를 위해 Target의 AnyOf와 AllOf는 각각 Or과 And로, Match는 Apply로 치환한다. Condition의 경우, XACML 요청에 따라 참 또는 거짓이 될 수 있다. 예를 들어 표 1의 Condition은 접근

표 1 정책 요소의 관련 정보
Table 1 Associated Information about Policy

Field	Value	Type
PolicyId	"FamilyPolicy"	String
RuleCombiningAlgId	"deny-unless-permit"	String
Target	Or(And(Apply(funcId:"string-equal", value: AttributeValue(type: "string", value: "car"), bag: AttributeDesignator(type: "string", id: "resource"))))	Boolean
Rules	["FamilyRule"]	Array <String>

표 2 규칙 요소의 관련 정보
Table 2 Associated Information about Rule

Field	Value	Type
RuleId	"FamilyRule"	String
Condition	Apply(funcId: "string-is-in", args: [AttributeValue(type: "string", value: "family"), AttributeDesignator(type: "string", id: "role")])	Boolean
Effect	"Permit"	String

요청의 'resource' 속성이 'car'와 같은지 비교한다.

표 1, 표 2의 정보를 XACML 조언 표현식으로 변환한 결과는 그림 8, 그림 9와 같다. PDP는 각 정책 집합과 정책 그리고 규칙이 평가 결과와 관련된다면 해당 요소의 조언 구문을 PEP로 평가 결과에 대한 추가 정보로써 반환한다. 즉, PEP는 모든 정책과 규칙에 대한 정보가 아닌 접근 요청과 관련된 문맥적 요소와 추가 정보만을 반환받는다.

이 방법은 세 가지 장점을 가진다. 첫 번째로, XACML 명세에서 제공하는 기능을 사용하므로 모든 XACML 정책에 대해 적용할 수 있다. 즉, XACML 표준 명세를 따르는 어떤 접근 제어 기술에도 적용할 수 있다. 두 번째로, 정책 작성자의 조언 구문 작성 및 관리 부담이 줄어든다. 정책 작성자가 따로 조언 구문을 작성하지 않아도

```
<AdviceExpression
  AdviceId="AutoGeneratedFamilyPolicyAdviceForPermit" AppliesTo="Permit">
  <AttributeAssignmentExpression AttributeId="PolicyId">
    <AttributeValue DataType="string">FamilyPolicy</AttributeValue>
  </AttributeAssignmentExpression>
  <AttributeAssignmentExpression AttributeId="RuleCombiningAlgId">
    <AttributeValue DataType="string">deny-unless-permit</AttributeValue>
  </AttributeAssignmentExpression>
  <AttributeAssignmentExpression AttributeId="Target">
    <Apply FunctionId="string-equal">
      <AttributeValue DataType="string">car</AttributeValue>
      <Apply FunctionId="string-one-and-only">
        <AttributeDesignator DataType="string" AttributeId="resource-id"/>
      </Apply>
    </Apply>
  </AttributeAssignmentExpression>
</AdviceExpression>
```

그림 8 자동 생성된 정책 조언 구문

Fig. 8 Auto Generated Advice Expression for Policy

```
<AdviceExpression
  AdviceId="AutoGeneratedFamilyRuleAdviceForPermit" AppliesTo="Permit">
  <AttributeAssignmentExpression AttributeId="RuleId">
    <AttributeValue DataType="string">FamilyRule</AttributeValue>
  </AttributeAssignmentExpression>
  <AttributeAssignmentExpression AttributeId="Condition">
    <Apply FunctionId="string-is-in">
      <AttributeValue DataType="string">family</AttributeValue>
      <Apply FunctionId="string-one-and-only">
        <AttributeDesignator DataType="string" AttributeId="role"/>
      </Apply>
    </Apply>
  </AttributeAssignmentExpression>
</AdviceExpression>
```

그림 9 자동 생성된 규칙 조언 구문

Fig. 9 Auto Generated Advice Expression for Rule

자동으로 평가에 관련되는 문맥적 요소를 파악하고 조언을 생성하기 때문이다. 이는 정책 작성자가 PEP에 넘겨야 할 추가 정보에 관여하지 않고 오로지 정책의 핵심 메커니즘에만 집중할 수 있도록 돕는다. 세 번째로는 PEP가 일관된 형식으로 정보를 수집할 수 있도록 돕는다. 문맥적 요소를 자동으로 수집하여 XACML 표준을 사용하여 기술하기 때문에 PEP는 누가 정책을 정의하던지 상관없이 일관된 형식의 조언을 받을 수 있다.

4. 결론 및 향후 연구

접근 제어 분야에서 정책 충돌은 정책 작성자의 의도에 맞지 않는 평가 결정을 유발하며, 이는 부적절한 사용자에게 중요한 정보를 허용하는 치명적인 보안 사고를 초래한다. 본문에서는 정책 충돌을 해결하기 위한 선행 작업으로 정책 평가 결정에 연관되는 추가 정보를 자동으로 수집하는 방법에 대해 논의하였다. 본 논문에서 제안하는 조언 구문 작성의 자동화 방안은 정책 작성자가 정책의 핵심 메커니즘 작성에 집중하고, PEP에서는 일관된 형식으로 문맥 정보를 제공받는다. 이 방법은 이러한 장점들을 바탕으로 정책 충돌 문제를 해결하는데 필요한 추가 정보와 문맥적 요소를 수집하는데 기여할 수 있다. 다만, 이 방법은 조언 구문의 삽입으로 인해 정책의 크기가 커지고 PEP가 필요하지 않은 정보를 같이 전달받을 수 있다는 단점이 존재한다. 향후 연구에서는 사례 연구를 통해 XACML 기반의 다양한 접근 제어 솔루션에 적용해보면서 정책 충돌 시 반드시 수집이 필요한 정보가 무엇인지 선별하여 불필요한 정보의 전달을 막고 정책의 크기를 줄일 계획이다.

References

[1] Axiomatics. (2017). Making a Business Case for Attribute Based Access Control (ABAC) [Online]. Available: https://goo.gl/8Qg5qU

[2] G. Lee and U.-J. Lee, "Proposition of IoT Platform Considering the Access Control Policy Collision," *Proc. of the KIISE Korea Computer Congress 2016*, pp. 300-302, 2016. (in Korean)

[3] J. Kim and U.-J. Lee, "Conflict Detection Algorithm for XACML policies," *Proc. of the KIISE Korea Computer Congress 2013*, pp. 550-552, 2013. (in Korean)

[4] R. Chadha, "A Cautionary Note About Policy Conflict Resolution," *MILCOM IEEE Military Communications conference*, pp. 1-8, 2006.

[5] OASIS Standard. (2013, Jan 22). eXtensible Access Control Markup Language (XACML) Version 3.0 [Online]. Available: https://goo.gl/VLUMZs

안 윤 근

2017년 한양대학교 ERICA 컴퓨터공학과 졸업(학사) 2017년-현재 한양대학교 컴퓨터공학과 석사과정. 관심분야는 함수형 프로그래밍, 소프트웨어 공학

이 기 찬

2016년 한양대학교 ERICA 컴퓨터공학과 졸업(학사) 2017년-현재 한양대학교 컴퓨터공학과 석박사통합과정. 관심분야는 스마트 그리드, 빅데이터 분석, 분산 에너지 자원 제어

이 욱 진

2004년 University of Auckland, Bachelor of Engineering in Software Engineering (학사). 2009년 University of Auckland, Doctor of Philosophy in Computer Science(박사). 2011년~현재 한양대학교 소프트웨어학부 교수. 관심분야는 소프트웨어 공학, 웹