클라우드 컴퓨팅 환경에 적합한 통합 접근제어 모델 제안

김재진, Scott Uk-Jin Lee, Amna Al Dhaheri

한양대학교 컴퓨터공학과
경기도 안산시 상록구 한양대학로 55 한양대학교
jaejinkim@hanyang.ac.kr, scottlee@hanyang.ac.kr, amna.aldhaheri90@gmail.com

요약: 본 연구에서는 기존에 연구되고 있는 접근제어 모델을 분석하고 역할기반 접근제어(RBAC) 모델과 속성기반 접근제어(ABAC) 모델을 통합하여 클라우드 컴퓨팅 환경에 더욱 적합한 접근제어 모델을 제안하고자 한다.

핵심어: 접근제어 모델, 클라우드 컴퓨팅, 역할기반 접근제어 모델, 속성기반 접근제어 모델

1. 서론

클라우드 컴퓨팅은 대규모 오픈 분산시스템의 하나로써 현재 급속히 개발되고 있는 컴퓨팅 환경이다[1]. 클라우드 컴퓨팅 환경에서는 인터넷 연결이 가능하면 언제 어디서나 자원을 획득할 수 있다. 이러한 환경의 특성상 사용자와 자원의 관계는 밀접하고 유동적으로 작용한다. 예를 들어, 클라우드 스토리지 서비스에서 사용자는 파일을 자신의 계정에 올려놓기만 하면 자연스레 서버 공간 어딘가에 저장이 되어 다른 단말기기에서 다시 클라우드 스토리지 서비스를 이용하면 자신이 올려놓은 파일을 접근할 수 있다.

클라우드 컴퓨팅 환경에서는 방대한 규모의 자원과 유동적인 사용자 그리고 동적인 제약들을 올바르게 처리하기 위해 적절한 접근제어가 필요하며, 이는 매우 중요한 이슈로 떠오르고 있다[2]. 접근제어는 기존의 다른 환경에서 많이 연구가 되어 현재는 역할기반 접근제어(RBAC) 모델[3], 속성기반 접근제어(ABAC) 모델[4], 그 밖에 확장 모델 등의 연구가 진행되고 있는 상황이다.

2004년 Xu Feng 외 2명[5]은 역할기반 접근제어 모델을 웹 서비스에 이용하기 위해 상황(Context)에 따라 동적으로 역할을 활성화하는 방법으로 CSRBAC라는 모델을 제안하였다. 이 모델에서는 시간과 장소 같은 상황을 인식하여 동적으로 역할을 활성화/비활성화함으로써 역할기반 접근제어의 정적인 역할을 동적으로 사용할 수 있게 하였다. 그러나 제안된 방법은 사용자 식별을 위한 시간과 장소 같은 환경의 속성을 이용하기 때문에 클라우드 컴퓨팅의 방대한 자원과 서비스에 적합하지 않다.

2012년 Zhuo Tang 외 4명[6]은 기존의 계층적인 역할기반 접근제어 모델에서 소유자 역할(Owner Role)의 개념을 도입하여 동적인 역할 추가 및 삭제가 가능한 CARBAC 모델을 제안하였다. 이 모델에서는 데이터 소유자 역할에 계층적으로 기존의 사용자 역할(User Role)을 연결하여 역할기반 접근제어의 근본적인 문제점인 동적인 변경의 어려움을 해결하였다. 하지만 클라우드 컴퓨팅 환경의 특성상 다양하고 방대한 자원, 서비스 그리고 동적 사용자를 포함하기에는 무리가 있다.

2012년 ByungRae Cha[2]는 속성기반 접근제어 모델을 이용하여 클라우드 컴퓨팅 환경에서의 접근제어 모델을 요청자(Requestor), 서비스(Service), 자원(Resource) 그리고 환경(Environment) 요소로 구성하여 설계하는 방법을 제안하였다. 이 방법은 속성기반 접근제어 모델의 개체를 확장하여 조금 더 클라우드 컴퓨팅 환경에 적합하게 설계하는 방법을 유일하게 제시하였다. 하지만 많은 정책이 생성되면 속성의 분석이 매우 어려워지는 속성기반 접근제어의 문제점은 해결되지 않았다. 정책이 많아졌을 때 속성의 분석이 어렵지 않아야 클라우드 컴퓨팅 환경에 적합한 접근제어 모델이 될 수 있다.

관련 연구에서 나타나듯이 현재 대다수의 연구들이 웹 서비스에 치우치고 있어 클라우드 컴퓨팅 환경에 적합한 접근제어 모델의 연구는 아직 미흡한 편이며 꼭 필요한 연구이다. 이에 본 연구에서는 클라우드 컴퓨팅 환경에 더욱 적합한 접근제어 모델을 제안 하고자 한다.

2. 기존 접근제어 모델 적용의 문제점

급속하게 변화하는 클라우드 컴퓨팅 환경에서 가장 적합한 접근제어를 제공하기 위해서는 먼저 아래와 같은 요구사항을 고려해보아야 한다.

- 다양하고 방대한 자원 및 사용자의 접근제어
- 유동적인 사용자의 접근제어
- 제약의 동적인 생성 및 삭제

첫 번째로 다양하고 방대한 자원 및 사용자의 접근제어를 지원하기 위해서는 예측할 수 없는 자원에 대한 접근제어 또한 모두 고려하여야 한다. 예를 들어, 클라우드 스토리지 서비스에서 공유하는 파일의 유형은 이미지, 동영상, 텍스트, 바이너리 등과 같이 아주 다양하다. 그리고 폴더 역시 업무용, 공부용, 연구용 등과 같이 사용자의 용도에 따라 여러 가지로

만들어질 수 있다. 또한 클라우드 컴퓨팅 환경의 발전에 따라 사용자도 방대하게 증가할 것이다. 이러한 이유로 클라우드 스토리지 서비스뿐만 아니라 다른 클라우드 서비스에서는 예측할 수 없는 자원과 사용자에 대한 접근 제어를 지원할 수 있어야 한다[7].

두 번째로 유동적인 사용자의 접근제어를 지원하기 위해서는 실제로 클라우드 컴퓨팅 환경을 이용하는 사용자의 정적이거나 동적인 역할(Roles) 및 정책(Policies) 생성과 삭제가 용이해야 한다. 예를 들어, 사용자가 용도에 따라 만들어진 자원 유형들을 접근할 때 사용자와 사용자간의 공유, 사용자와 그룹간의 공유 그리고 그룹과 그룹간의 공유를 통해 클라우드 스토리지 서비스를 이용하게 된다. 이때 다수의 사용자간의 자원 공유는 빈번하게 설정 및 해제될 수 있기 때문에 정책의 동적인 생성 및 삭제가 용이해야 한다. 또한 클라우드 환경에서는 다수의 사용자가 수시로 생겨나기도 하고 없어지기도 하는데, 이때 삭제된 사용자에게 할당되어 있었던 많은 정책들이 불필요하게 남아있게 되면 새로운 사용자의 정책을 생성할 때 오류 및 불필요한 공간 낭비가 생기게 된다.

세 번째로 제약의 동적인 생성 및 삭제를 지원하기 위해서는 정책의 추가 및 삭제에 따라 제약도 함께 추가 및 삭제가 되어야 한다. 예를 들어, 클라우드 스토리지 환경에서는 사용자가 관리자 역할이 되어 여러 사용자간의 자원 공유를 설정 또는 해제 할 수 있다. 이때 접근제어는 정책을 통하여 자원에 대한 권한을 동적으로 변경할 수 있어야 하며 올바른 접근권한의 제어를 위해서는 제약이 필요하게 된다. 제약은 실질적으로 제약을 포함하는 정책이 동적으로 변경됨에 따라 같이 변경될 수 있다. 따라서 제약의 동적인 생성 및 삭제가 용이해야 한다.

현재의 역할기반 접근제어 모델은 기존의 모델에 속성을 추가하거나 역할을 더 확장하여 사용자의 동적인 추가 및 삭제를 가능하도록 연구가 진행되었다[8]. 하지만 역할기반 접근제어 모델은 아직까지 다양하고 방대한 자원을 포함할 수 없다. 자원과 사용자가 증가하면 할수록 역할과 정책이 늘어나게 되어 역할기반 접근제어 모델의 고질적인 문제인 역할의 확장성(Scalability) 문제가 발생하기 때문이다[8]. 이러한 확장성 문제는 역할기반 접근제어 모델의 특성인 정적인 역할 때문에 발생하는데 이를 해결하기 위해 속성기반 접근제어 모델이 개발되었다. 이 모델은 역할중심 구조가 아닌 속성중심 구조로써 정책이 필요하면 속성을 추가하여 보다 유동적이게 접근제어를 할 수 있도록 제안된 모델이다. 하지만 속성기반 접근제어 모델을 클라우드 컴퓨팅 환경에 적용하게 되면 정책 분석이 매우 힘들어지는 문제점이 발생한다. 클라우드 컴퓨팅 환경의 방대한 자원과 사용자 그리고 제약들을 포함하기 위해서는 이에 해당하는 대량의 정책을 나타낼 수 있는 많은 수의 속성을 만들어야 한다. 그리고 속성은 하나의 정책안에 무수히 포함될 수 있기 때문에 역할기반 접근제어 모델과는 달리 정책의 분석이 매우 복잡해 진다. 따라서 이러한 문제점을 가지고 있는 현재의 접근제어 모델을 클라우드 컴퓨팅 환경에 적용하기에 아직 부적합하다고 판단된다.

최근의 연구에서 볼 수 있듯이 역할의 확장성 문제와 속성의 복잡성 문제는 클라우드 컴퓨팅 환경에 적합한 접근제어 모델을 개발하기 위해 필수적으로 해결되어야 한다. 이러한 문제점들을 해결하기 위해서는 클라우드 환경의 특성을 잘 파악하고 분류하여 클라우드 환경의 요구사항에 맞게 접근제어를 해야 하며 특히 유동적인 접근제어가 필요하다. 하지만 클라우드 환경에 존재하는 모든 개체 혹은 자원이 모두 유동적이지는 않음으로 사용자의 경력, 이력, 소속과 같이 미리 정의할 수 있는 정적인 속성들에 대해서는 굳이 유동적인 접근제어를 적용할 필요가 없다. 따라서 본 연구에서는 상대적으로 결정되는 속성을 동적과 정적으로 나누어 각각의 접근제어 정책을 생성함으로써 위에서 언급한 두 가지 문제점을 보완하고자 한다. 그리고 이를 바탕으로 기존에 연구되고 있는 역할기반 접근제어 모델과 속성기반 접근제어 모델을 통합하여 클라우드 컴퓨팅 환경에 더욱 적합한 접근제어 모델을 제안한다.

3. 통합 접근제어 모델 제안

서로 다른 접근법으로 고안된 역할기반 접근제어 모델과 속성기반 접근제어 모델을 통합하여 각 모델의 장점만을 부각시키는 것은 사실상 간단하지가 않다. 최근의 연구되고 있는 모델들은 역할기반 접근제어에 속성을 더하거나 속성기반 접근제어에 역할을 더하여 확장하는 방법으로는 오히려 두 가지의 모델이 가지고 있는 문제점들을 더 악화시킬 수도 있다. 증폭시킬 수 있다. 따라서 우리는 앞에서 언급한 확장성 문제와 정책 분석의 어려움을 해결하기 위해 정적인 속성은 역할기반 그리고 동적인 속성은 속성기반 접근제어 모델을 이용하여 역할의 양과 속성의 복잡함을 줄이는 방법을 제시한다. 클라우드 환경의 자원 및 사용자의 속성은 상대적으로 환경에 따라 성향이 달라질 수 있다. 본 연구에서 제안한 방법에서는 이를 고려하여 사용자의 경력, 이력, 주소, 소속 등은 정적인 속성으로 분류 할 수 있으며[9] 시간, 사용자의 현재위치, 공유하는 자원 등은 동적인 속성으로 분류할 수 있다. 이러한 속성의 분류를 통하여 자원 및 사용자에 대한 정책을 두 가지의 접근제어 모델로 나누게 되면 확장성의 문제와 속성의 복잡성의 문제를 해결할 수 있다. 예를 들어, 4개의 정적인 속성과 6개의 동적인 속성이 있다고 가정하자. 이때 단일 역할기반 접근제어 모델 또는 단일 속성기반 접근제어 모델을 이용하여 10개의 속성을 분류하지 않은 채 정책으로 생성하면 다음과 같이 1,024개의 정책이 만들어진다.

2^{10} = 1,024 개

접근제어의 승인은 허가 혹은 거부로 결정되므로 N개의 속성에 따른 정책은 2^N개로 계산할 수 있다. 1000개 이상의 역할이나 속성을 생성한다면 역할기반 접근제어는 확장성 문제가 발생하며 속성기반 접근제어는 속성의 복잡성 문제가 발생하게 된다. 하지만 두 가지의 모델을 서로 통합하면 다음과 같이 총 80개의 정책 (역할기반 접근제어 모델은 16개의 정책, 속성기반 접근제어 모델은 64개의 정책)을 생성하므로 매우 효과적인 정책의 감소를 볼 수 있다.

- 정적인 정책의 수: **2^4 = 16** 개
- 동적인 정책의 수: **2^6 = 64** 개

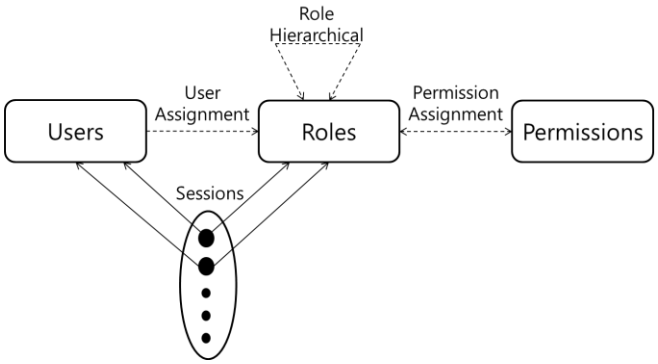

그림 1. 역할기반 접근제어 모델

그림 1은 역할기반 접근제어 모델이다. 정적인 정책에서 이 모델을 이용해서 정책을 정의한다. 기존의 역할기반 접근제어 모델의 방법대로 역할과 사용자를 할당하고, 역할에 해당하는 접근권한을 할당한다. 정적인 정책의 속성은 클라우드 환경에 따라 달라질 수 있지만 사용자의 집주소, 회사주소, 경력, 이력, 역할과 같이 이미 알고 있거나 자주 변경되지 않는 속성으로 구분할 수 있다. 따라서 이러한 정적인 속성들의 구분을 통해 정적인 정책을 정의한다.

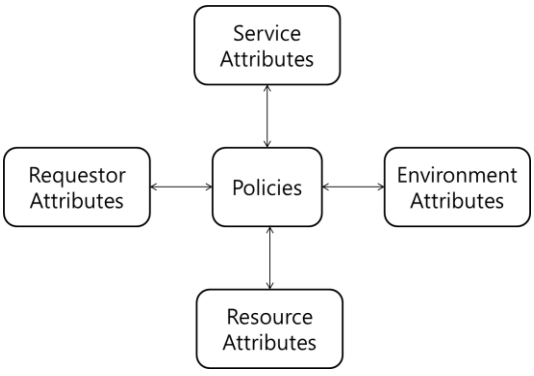

그림 2. 속성기반 접근제어 모델

그림 2는 속성기반 접근제어 모델이다. 이 모델은 역할기반 접근제어 모델과는 달리 자주 변경되는 속성을 위주로 정책을 정의한다. 동적인 속성은 요청, 자원, 서비스, 환경과 같은 컴포넌트로 구분할 수 있다. 요청은 사용하는 주체가 되고 자원은 파일, 폴더, 시스템, 데이터베이스와 같은 자원이 되며 서비스는 플랫폼 서비스(PaaS), 소프트웨어 서비스(SaaS), 인프라 서비스(IaaS) 등과 같은 서비스의 형태가 된다. 그리고 환경은 사용자의 시간, 날짜, IP주소, 현재위치 등을 식별하기 위한 보가 된다. 예를 들어, 클라우드 스토리지 서비스에서 파일, 폴더와 같은 다수의 사용자와 빈번하게 공유설정을 하거나 해제하는 속성들은 동적이므로 위에 설명한 네 가지 컴포넌트를 이용하여 동적인 정책을 정의한다.

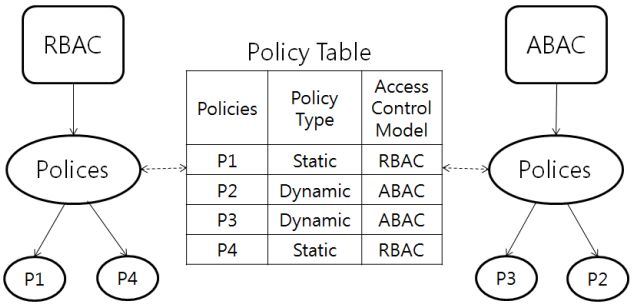

그림 3. 통합 접근제어 모델

그림 3은 역할기반 접근제어 모델과 속성기반 접근제어 모델을 통합한 접근제어 모델의 구조이다. 메커니즘이 다른 두 가지의 접근제어 모델을 사용하기 위하여 정책 테이블을 만들어서 각각의 서로 다른 정책들의 유형에 맞게 정책을 입력한다. P1과 P4는 역할기반 접근제어 모델로 정의된 정적인 정책이다. P2와 P3은 속성기반 접근제어 모델로 정의된 동적인 정책이다. 정책 테이블은 서로 다른 두 가지의 모델의 정책을 통합하기 위해 제안한 방법론이며 이를 통해 서로 다른 정책들을 색인 하여 통합할 수 있다. 또한 정책의 유형이 다르더라도 테이블은 정책의 인덱스와 정보만으로 구성되기 때문에 추상화(Abstraction)가 가능하여 서로 다른 유형의 정책도 쉽게 처리할 수 있다.

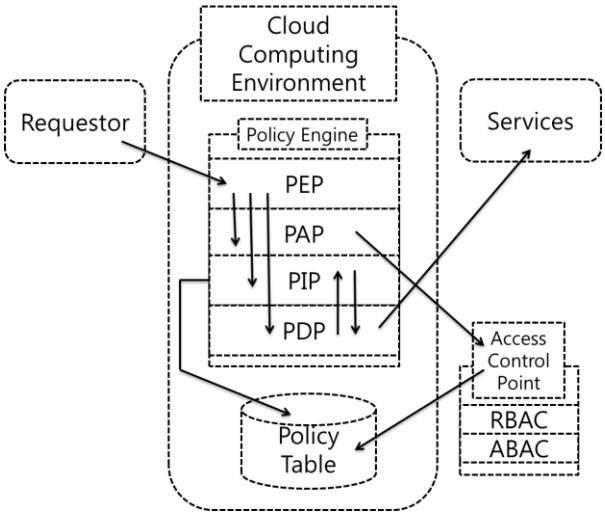

그림 4. 통합 접근제어 모델의 시나리오

그림 4는 통합 접근제어 모델의 시나리오로써 구성요소로는 요청자(Requestor), 정책 관리 엔진(Policy Engine), 접근제어 포인트(Access Control Point), 정책 테이블(Policy Table), 서비스(Services)가 있다. 아래에 구성요소에 대한 세부 내용을 설명한다.

통합 접근제어 모델의 구조 구성요소

- **요청자(Requestor):** 사용자가 요청을 한다.
- **정책 관리 엔진(Policy Engine**): 정책 집행 포인트(PEP), 정책 관리 포인트(PAP), 정책 정보 포인트(PIP), 정책 결정 포인트(PDP)로 이루어진 정책 관리 엔진이다.
- **접근제어 포인트(Access Control Point):** 역할기반 접근제어 모델과 속성기반 접근제어 모델의 정책을 생성 및 수정한다.
- **정책 테이블(Policy Table):** 역할기반 접근제어와 속성기반 접근제어의 정책 인덱스 테이블이다.
- **서비스(Services):** 플랫폼 서비스, 소프트웨어 서비스, 인프라 서비스와 같은 실제 클라우드 서비스이다.

본 연구에서 제안한 통합 접근제어 모델에서는 서비스 혹은 자원에 접근하려는 요청 그리고 정책의 생성, 수정, 삭제 등의 요청을 다음과 같이 처리한다. 우선 해당 요청은 정책 집행 포인트를 거쳐 정책 정보 포인트로 보내어져 해당 정책의 정보를 가져온다. 그 후 접근 요청의 경우에는 필요한 정보가 정책 결정 포인트로 전달되어 서비스 또는 자원에 접근이 허가(Permit)되거나 거부(Deny)된다. 그리고 정책의 생성, 수정, 삭제 등의 요청은 필요한 정보와 함께 접근제어 포인트로 보내져 수행되며 결과물은 정책 테이블에 저장된다.

4. 결론

클라우드 컴퓨팅 환경에서는 다양하고 방대한 자원 및 사용자 그리고 제약을 모두 수용할 수 있는 유동적인 접근제어가 필요하다. 기존의 역할기반 접근제어 모델과 속성기반 접근제어 모델은 역할의 확장성과 속성의 복잡성의 문제점을 해결하지 못하였으며 이로 인해 클라우드 컴퓨팅 환경에 적용하기에는 아직까지 부적합하다. 따라서 본 연구에서는 이러한 문제점들을 해결하는 방법으로 클라우드 컴퓨팅 환경의 특성에 맞게 정적인 속성과 동적인 속성을 분류하고 각 속성의 성향에 따라 역할기반 혹은 속성기반 접근제어 모델을 이용하여 정책을 생성하는 통합 접근제어 모델을 제안하였다. 향후 연구에서는 본 논문에서 제안한 통합 접근제어 모델을 토대로 클라우드 컴퓨팅 환경에 가장 적합한 접근제어 솔루션을 구현하고 평가하고자 한다.

참고문헌

[1] Abdul Raouf Khan, “Access Control in Cloud Computing Environment”, APPN Journal of Engineering and Applied Sciences, Vol.7, No.5, pp.613-615, 2012

[2] ByungRae Cha, JaeHyun Seo and JongWon Kim, "Design of Attribute-Based Access Control in Cloud Computing Environment", In Proceedings of the 2rd International Conference on IT Convergence and Security, pp.41-50, 2012

[3] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, "Role-Based Access Control Models", Computer, Vol.29, No.2, pp.38-47, 1996

[4] Eric Yuan and Jin Tong, “Attributed Based Access Control (ABAC) for Web Services”, In Proceedings of the 3rd IEEE International Conference on Web Services, pp. 561-569, 2005

[5] Xu Feng, Xie Jun, Huang Hao and Xie Li, "Context-Aware Role-Based Access Control Model for Web Services”, In Proceedings of the 3rd International Conference on Grid and Cooperative Computing, pp.430–436, 2004

[6] Zhuo Tang, Juan Wei, Ahmed Sallam, Kenli Li and Ruixuan Li, “A New RBAC Based Access Control Model for Cloud Computing”, In Proceedings of the 7th international conference on Advances in Grid and Pervasive Computing, pp.279–288, 2012

[7] Hassan Takabi, James Joshi and Gail-Joon Ahn, “Security and Privacy Challenges in Cloud Computing Environments”, IEEE Security & Privacy, Vol.8, pp-24-31, 2010

[8] Ravi Sandhu, David Ferraiolo and Richard Kuhn, “The NIST Model for Role-Based Access Control: Towards a Unified Standard”, In Proceedings of the 5th ACM workshop on Role-based access control, pp.47-63, 2000

[9] Richard Kuhn, Edward Coyne and Timothy Weil, "Adding Attributes to Role Based Access Control”, Computer, Vol.43, No.6, pp.79-81, 2010