분산 컴퓨팅에 적합한 제한적인 위임 명세

은 승 희[†] · 김 용 민[††] · 노 봉 남[†††]

요 약

권한의 위임은 그리드 컴퓨팅과 같은 대규모의 분산 환경에서 사용자가 원하는 작업을 처리하기 위해 해당 노드에 권한을 부여하는 중요한 과정 중의 하나이다. 하지만, 기존의 권한 위임 기법들은 작업에 대한 적절한 권한을 부여하지 못하거나, 위임의 범위를 세분화하지 못하고, 작업의 처리 과정에 대한 위임이 아닌 자원 자체에 대한 접근 권한의 위임만이 존재한다. 또한 시스템 자원의 예약이나 실행 전·후의 호스트 접근 등 작업의 실행 전·후의 위임이 필요한 과정에 대하여 적절한 위임을 적용하지 못한다. 따라서 본 논문에서는 분산 환경에서의 제약적인 위임을 위한 방법 및 명세를 제안한다. 제안한 방법은 위임을 작업 측면과 권한 측면으로 분리하고, 위임의 명세 및 절차를 XML 스키마와 UML을 이용하여 표현하며, 분산 컴퓨팅 환경에서의 제한적인 위임 시나리오를 제시한다.

키워드 : 분산 신뢰 환경, 위임 명세, 위임 프로토콜

A Specification for Restricted Delegation to suitable on Distributed Computing

Seung-Hee Eun[†] · Yong-Min Kim[††] · Bong-Nam Noh[†††]

ABSTRACT

A delegation of privileges is one of important processes that empower authority to relevant node to process job that user wants in large-scale distributed environment such as Grid Computing. However, existing delegation methods do not give suitable privilege about job, and do not atomize range of delegation and exists delegation of access privilege for only resources itself that is not delegation about executing process of job itself. Also, they do not apply about process that needs delegation before and after execution of job such as reservation of system resources or host access before and after execution. Therefore, this paper proposes a method and specification for restricted delegation in distributed environment. Proposed method separates delegation for job side and privilege side, and express specification and procedure of delegation using XML schema and UML, and present restricted delegation scenario in distributed computing environment.

Key Words : Distributed Trust Environment, Delegation Specification, Delegation Protocol

1. 서 론

인터넷이 대중화되고, 컴퓨터 및 네트워크의 성능이 향상되면서, 정보의 공유가 보편화 되었다. 특히 인터넷에 존재하는 정보를 쉽게 사용할 수 있는 기술인 웹 서비스는 현재 없어서 안 될 중요한 정보 공유 수단으로 자리 잡게 되었다. 웹 서비스 기술의 발달로 모든 자원들을 공유하여 효율적으로 사용하고자 하는 분산 서비스의 연구가 활발하게 진행되고 있다. 그리고 이러한 연구들은 인터넷을 통하여, 분산 시스템 자원을 단일한 자원으로 활용하고자 하는 그리드 컴퓨팅의 연구로 발전되고 있다. 분산된 컴퓨팅 자원을 활용하기 위해서는, 자원을 이용하려는 외부 사용자의 신원을 확인하기 위한 인증(Authentication) 서비스, 사용자가 자원 접근에 대한 권한의 유무를 결정하기 위한 접근통제(Access Control) 서비스 등과 같은 보안 문제에 대한 기술적인 기반이 뒷받침되어야만 효과적이고 안전한 서비스를 제공할 수 있다[1, 2].

이러한 보안 요구사항 중 한 부분인 권한의 위임(Delegation)은 분산된 신뢰 환경에서의 일반적인 작업 처리 방식이다. 분산 환경의 작업은 일반적으로 여러 노드로 분배되어 자체적으로 실행되며, 작업 결과는 이들의 처리 결과를 조합함으로써 이루어진다. 이 때 자신의 작업 중 일부 혹은 전부를 다른 노드에게 맡기는 것을 위임이라고 한다. 복잡하고 다양한 자원을 관리하고 보호하기 위해서는 작업을 위임할 때, 해당 작

※ 본 연구는 정보통신부 대학 IT연구센터 육성, 지원사업의 연구결과로 수행되었습니다.
† 준 회 원 : 전남대학교 정보보호협동과정 석사과정
†† 정 회 원 : 여수대학교 정보기술학부 전임강사
††† 종신회원 : 전남대학교 전자컴퓨터정보통신공학부 교수
논문접수 : 2005년 9월 1일, 심사완료 : 2005년 11월 7일

업을 처리하기 위해 충분한 권한을 주는 것이 필요하다. 그리고 이 때, 보안을 위하여 필요한 최소한의 권한만을 위임해 주는 것이 중요하다. 최소한의 권한만을 위임하기 위해 처리할 자원에 제한된 권한을 위임하는 방법을 제한적인 권한 위임(Restricted Delegation)이라고 한다[3, 4].

현재 대부분의 분산 환경은 프록시 인증서(Proxy Certificate)를 통한 위임을 제공하고 있지만, 위임 요청자의 정보, 접근하려는 서버의 정보, 권한 및 타임스탬프 기술 정보에 의한 사용자 인증이 이루어지면, 해당 작업에 대한 모든 권한을 부여함으로써, 작업에 따른 세부적인 권한의 부여가 불가능하다. 그리고 단순히 인증만 확인되면 해당 사용자의 모든 권한으로 자원을 소유할 수 있어, 최소한의 권한만을 부여할 수 있는 어떠한 제한 수단이나 정의가 없다. 또한 제한적인 권한 위임의 정의 자체도 자원에 대한 권한 만으로 한정되어 있어서, 실행되어야 할 작업의 사전 처리 과정이나 컴퓨팅 자원 예약 등의 위임이 필요한 경우에 대해서는 적용이 어렵다. 본 논문에서는 분산 환경의 특징에 따라, 위임을 작업 측면과 권한 측면으로 구분하고, 위임할 권한 및 위임 모드를 분류한다. 또한, 현재에도 사용되고 있는 분산 객체 기술인 Microsoft의 COM/DCOM, OMG의 CORBA, JAVA Bean 등의 호환성 문제를 용이하게 처리하기 위하여 데이터 표현의 통합화 및 표준화된 채널을 제공하는 XML을 기반으로 위임 명세를 기술하고, UML로 위임 절차를 도식화한다. 그리고 분산 환경에서의 제안된 위임 방법을 적용한 시나리오를 제시하고 기존의 위임 방법과 제안된 위임 방법을 비교한다.

2. 관련 연구

2.1 분산 신뢰 환경에서의 위임

L. Kagal et al.는 기존 분산 신뢰 환경에서 시스템의 보안과 신뢰 관리를 용이하게 하기 위해 이종 에이전트들의 네트워크 내에서의 인가 문제 완화 및 인가 정보의 위임을 위한 메커니즘을 제시하고, 인증을 위한 인증서의 식별 및 위임의 캡슐화를 위하여 X.509 인증서를 사용 했다[5]. 그는 실행할 권한을 가진 행동(Action)을 실행하거나 실행 권한을 위임할 수 있는 에이전트를 정의하였고, 만약 계속적인 위임이 인가된다면, 위임 받은 에이전트는 소유한 권한을 또 다른 에이전트에게 위임할 수 있다. 하지만 에이전트는 임의의 위임 가능한 권한만을 위임할 수 있다. 이것은 위임 체인(Delegation Chain)을 구성하며, 만약 그 체인에서 위임이 유효하지 않다면, 접속을 거절한다.

〈표 1〉 위임의 타입

위임 종류	설명
Time Bound	특정 시간 동안만 위임
Group	임의의 조건을 만족하는 에이전트 그룹에게 위임
Action Restricted	행동 실행 전 사전 조건을 만족시 위임
Redelegatable	실행할 권한과 재위임이 가능한 권한을 동시에 위임
Strictly Redelegatable	실행할 수 있는 권한을 제외한 재위임이 가능한 권한만을 위임

<표 1>은 그가 제안한 위임의 제한 타입으로서 위임 방법은 시간, 그룹, 행동 조건, 재위임 여부를 통하여 위임을 제한한다. 하지만, 이러한 제한 타입은 실제 위임 받을 권한의 종류를 세부적으로 제어하기에는 한계가 있다. 권한 외적 요인(시간, 그룹, 행동 조건, 재위임 여부)들과 더불어 권한 자체의 세부적 분류도 고려할 필요가 있다.

2.2 그리드 환경에서의 위임

분산 신뢰 환경 기반의 그리드 컴퓨팅의 위임은 그리드 미들웨어를 통하여 이루어진다. 그 중 대표적인 것으로 GT3 (Globus Toolkit 3)[6, 7]와 Legion[8]이 있다.

GT3는 그리드 보안 구조(Grid Security Infrastructure: GSI)를 이용하여 공개키 시스템, 상호 인증, 인증서 위임, 단일 인증 등의 보안 문제를 해결할 수 있도록 도움을 준다. 이러한 특징 중 인증서 위임은 프락시 인증서라고 불리는 인증서에 의해 지원되며, 한 사용자가 다른 사용자를 대신하여 업무를 수행하는 것을 허용한다. 위임 모드는 위임을 허용하지 않는 위임 불가(*no delegation*), 제한적인 위임을 허용하는 제한적 위임(*restricted delegation*), 어떠한 제한 조건도 존재하지 않고 권한 전체에 대한 위임을 허용하는 전체 위임(*fully delegation*)을 제공하고 있다. 하지만 제한적인 위임의 범위에 대한 상세한 정의는 없다. 또한, 위임 자체가 프락시 인증서를 통한 사용자와 상호 인증이 완료된 후, 로컬의 한 사용자의 권한으로 작업을 수행할 수 있지만, 단순히 인증 여부만 확인할 뿐 재위임 여부 등의 작업에 맞는 적절한 권한 위임은 불가능하다.

다음으로 Legion은 이종의 분산되어 있고, 독립적으로 관리되는 자원들을 이용하는 객체-기반의 그리드 운영 시스템이다. Legion에서의 자원들은 활성화 객체들로 표현하고, 자원의 생성과 관리를 담당하는 클래스의 인스턴스이다. 또한 X.509 기반의 인증서를 사용하여 사용자를 인증하고, 접근 제어를 결정하며, 접근통제 리스트(ACLs)에 기반한 기본적인 보안 구조를 제공함으로써, 각각의 객체에 대한 인가를 결정한다. Legion은 하나의 객체가 또 다른 서비스를 호출할 때, 위임을 위한 권한을 결정하는 객체, 메소드, 시간 제한 위임을 논의하였다[3]. 이것은 위임 인증서가 임의 또는 모든 연산을 수행하는 것을 허용하는 대신, 인증서 생성자가 명시적으로 임의 시간대에 허용 가능한 객체, 메소드를 개별적으로 목록화하여 열거할 수 있다. 하지만, 객체와 메소드의 수가 많아서 사용자가 수동적으로 열거하여 제한하기 힘들고, 객체와 메소드, 시간만으로는 해당 작업에 필요한 위임을 효율적으로 표현하기 힘들다. 또한, 객체지향 시스템이 아닌 경우에 적용하기 힘든 단점이 있다.

3. 제한적인 위임 프로토콜

현재의 위임 프로토콜들은 단순히 권한의 범위(일부 혹은 전부)만을 제한한 경우가 많아 특정한 작업에 대한 세부적인 위임 제어는 불가능하다. 이 장에서는 위임을 작업(Job)과 권

한(Privilege)으로 구분하여 각 위임들을 정의하고, 권한의 위임 형식에 따른 권한의 위임 모드를 제안한다. 또한, 분산 컴퓨팅 환경에서의 제한적인 위임을 위한 시스템 구성과 해당 시스템 내에서의 제한적인 위임의 과정을 보인다.

3.1 제한적인 위임 방법

3.1.1 작업 위임(Job Delegation)과 권한 위임(Privilege Delegation)

분산 신뢰 환경의 위임은 일반적으로 사용자의 요청 작업이 실행되는 동안 자원에 접근하기 위한 권한의 위임이다. 즉, 작업이 실행되는 해당 노드에 권한을 부여하고, 권한을 부여받은 노드는 작업을 실행하는 동안 파일이나 메소드와 같은 자원에 접근할 수 있다. 그러나 작업이 실행되는 동안이 아닌, 실행되기 전 준비 단계 및 작업 종료 후, 임의 노드의 접근이 필요한 경우나, 거대한 데이터의 흐름을 생성하는 작업을 위해 미리 컴퓨팅 자원을 예약하는 경우 등은, 해당 노드가 임의의 호스트로의 접근이나 예약을 위한 허가권을 자동적으로 가지지 않는다. 따라서 사용자의 권한을 사용할 수 있도록 명시할 수 있는 기능이 필요하다.

또한, 기존의 분산 환경에서 위임은 <표 1>과 같이 시간, 그룹, 행동조건, 재위임과 같이 작업 자체의 측면이 아닌 작업 외적인 기준을 통하여 위임을 구분하였기 때문에, 작업 실행 전에 자원을 예약하는 것과 같은 특정한 작업 행동을 제한하기에는 한계가 있었다. 따라서 본 논문에서는 특정한 작업에도 적용이 가능하도록 위임을 설계하였다. 본 논문에서는 작업 측면의 위임(Job Delegation)과 권한 측면의 위임(Privilege Delegation)으로 구분하여 각각 적용하도록 제안하여, 위임을 좀 더 세부적으로 제어할 수 있도록 하였다.

작업 위임은 분산 환경에서의 작업이 실행 전·후 임의의 호스트 접근이 필요한 경우나, 작업 처리 시 요구되는 컴퓨팅 자원의 예약이 필요한 경우 적절한 권한을 해당 노드에 위임한다. 위임되는 권한은 호스트 접근(Access) 권한과 예약(Reserve) 권한이다. 이러한 접근 권한과 예약 권한은 재위임 시에 남용될 수 있어 보안상 취약하므로 재위임은 불가능해야 한다[9].

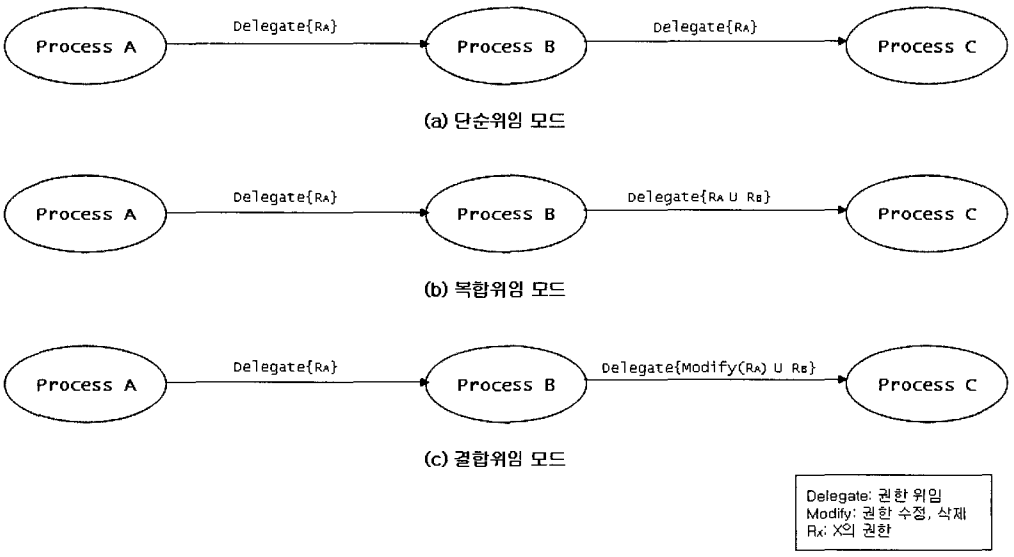
권한 위임은 작업이 실행되는 동안 원격 자원에 대한 접근 권한을 작업 프로세스에게 위임하는 것으로서, 자원을 읽고(Read), 쓰고(Write), 다른 작업 프로세스를 실행(Execute)하는 권한을 위임한다. 이러한 권한은 다른 작업 프로세스에게 재위임 할 수 있다. <표 2>는 작업 위임과 권한 위임을 비교한 것이다.

〈표 2〉 작업 위임과 권한 위임 비교

위임의 종류	작업 위임	권한 위임
위임 요청자	Job Node	Job Process
위임 권한	Access, Reserve	Read, Write, Execute
권한 소유자	User	User
재위임 가능	불가	가능(위임 모드 적용)

3.1.2 권한 위임 모드(Privilege Delegation Mode)

기존의 권한에 대한 위임은 대부분 작업에 맞는 특정한 권한을 위임하는 것이 아닌, 사용자가 소유한 모든 권한을 위임함으로써 최소 권한의 원칙에 취약하였다. 따라서 본 논문에서는 분산컴퓨팅 기술 중의 하나인 CORBA에서의 위임 모드를 이용하여 사용자의 권한 집합 중 작업에 적합한 권한을 세부적으로 위임할 수 있는 권한 위임 모드를 제안한다. 권한 위임 모드는 단순위임 모드(Simple Delegation Mode), 복합위임 모드(Composite Delegation Mode), 그리고 결합위임 모드(Combined Delegation Mode) 중 하나를 명시하여 수행한다. 제안된 위임 모드에서 사용자는 자신의 권한 중 일부 혹은 전부를 위임할 수 있다. 위임 요청자(Job Process)는 위임 모드에 따라서, 사용자가 위임한 권한 중 일부를 수정 또는



(그림 1) 위임 모드 적용 시 권한의 구성

삭제하여 제 3자에게 위임할 수 있다[10].

단순위임 모드에서 위임 요청자는 사용자에게 위임 받은 권한의 수정, 삭제, 추가가 불가능하며, 재위임이 가능할 경우, 해당 권한을 그대로 제 3자에게 위임한다. 복합위임 모드는 위임한 권한의 수정과 삭제는 불가능하지만, 추가는 가능하다. 그리고 재위임시 권한을 추가하여 제 3자에게 위임할 수 있다. 결합위임 모드는 위임한 권한의 수정, 삭제, 추가가 모두 가능하다. <표 3>과 (그림 1)은 위임 모드의 비교와 위임 모드 적용 시 권한 구성을 나타낸다.

〈표 3〉 위임 모드

위임 모드	권한의 수정	권한의 삭제	권한의 추가	비고
Simple	X	X	X	strictly
Composite	X	X	O	-
Combined	O	O	O	loosely

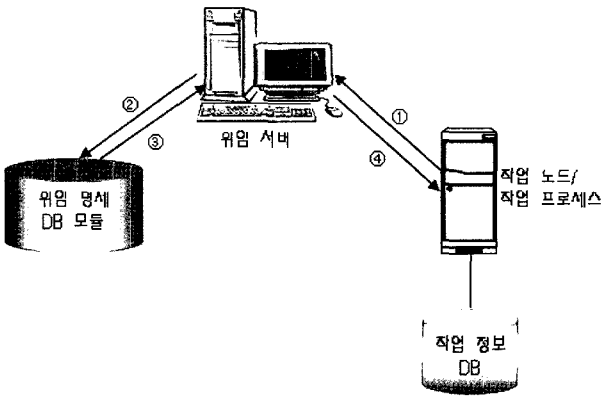
3.2 제한적인 권한 위임의 절차

3.2.1 제한적인 권한 위임 시스템의 구성

제한적인 권한 위임 시스템은 크게 위임 서버, 위임 명세 DB 모듈, 작업 노드, 작업 프로세스로 구성하며 각 기능은 다음과 같다. 위임 서버는 작업 노드에서 위임 요청이 들어올 경우, 위임 명세 모듈과의 상호작용을 통하여 해당 노드에서 요청한 위임이 적절한지 판단 한 후, 위임 인증서를 발행한다. 위임 명세 DB 모듈(4장에서 설명)은 위임 명세서를 처리하고, 요청된 위임의 수용 여부를 판단하는 모듈이다. 임의의 작업이 사용자에 의하여 생성될 때, 작업 위임 명세서를 통하여 해당 작업에 대한 작업 위임과 권한 위임의 세부사항이 설정되어 DB에 저장된다. 그리고 위임 명세 DB 모듈은 DB에 질의를 하여 요청한 위임이 적절한지를 판단한다. 사용자는 작업의 수행을 원할 때, 작업 노드에게 수행하고자 하는 작업을 제출하고, 작업 노드는 제출된 작업을 실행할 작업 프로세스에 전달하기 전, 작업 정보 DB를 통하여 작업의 사전/사후 준비 작업 및 예약 작업 여부를 결정한다. 그 후 작업 위임이 필요한 경우 위임 서버에 요청한다. 이러한 작업 위임이 종료된 후, 작업 프로세스에 해당 작업을 전달한다. 작업 정보 DB는 사용자에 의해 작업이 노드에 생성될 때, 작업의 전·후 준비 및 예약 등의 작업에 대한 세부사항이 설정되어 저장된다. 작업 노드나 작업 프로세스는 작업 정보 DB를 통해 작업에 대한 질의를 할 수 있다. 작업 프로세스는 사용자가 제출한 작업을 수행하기 위해 작업 노드에 의하여 생성되며, 작업을 수행하는 도중, 해당 작업에 대한 권한 위임이 필요할 경우 위임 서버에 위임 인증서를 요청한다.

3.2.2 제한적인 위임 프로토콜

(그림 2)에서 제한적인 위임은 단계별로 이루어지는데, 크게 요청(*Request*), 검증(*Verify*), 결과(*Result*), 발행(*Issue*)의 4 단계로 구분 한다. 먼저, 요청 단계는 위임 요청이 이루어지는 단계(①)로서, 위임 요청자는 위임이 필요한 경우 위임 서버에 위임을 요청한다. 다음으로 검증 단계는 요청된 위임의 검증 단계(②)로, 위임 서버는 요청된 위임을 위임 명세 DB 모듈과 상호 작용을 통하여 위임이 적절한지 검증한 후 "Delegation Accept/Reject"를 판단한다. 다음 결과 단계는 위임 명세 DB 모듈이 검증 단계에서 검증된 결과를 위임 서버에 전달하는 단계(③)이다. 마지막 발행 단계는 위임 인증서를 발행하는 단계(④)로, "Delegation Accept"의 결과가 나오면, 위임 요청자에게 요청한 위임에 대한 인증서를 발행한다.



(그림 2) 제한적인 위임 프로토콜

이러한 프로토콜에서의 위임 메시지의 구조는 (그림 3)와 같다.

(1) **Request Message (작업 노드/작업 프로세스 → 위임서버)**

① 작업 위임: {C_U, C_{JN}, Job, RACCEPT/RRESERVE}

② 권한 위임: {C_U, C_{JP}, Job, RREAD/RWRITE/REXECUTE }

(2) **Result Message (위임서버 ←→ 위임 명세 DB)**

① **Accept**
   a. 작업 위임:{C_U, C_{JN}, DC_J}
   b. 권한 위임:{C_U, C_{JP}, DC_P}

② **Reject:** {"Delegation Request Rejected"}

(3) **Issue Message (위임 서버 → 작업 노드/작업 프로세스)**

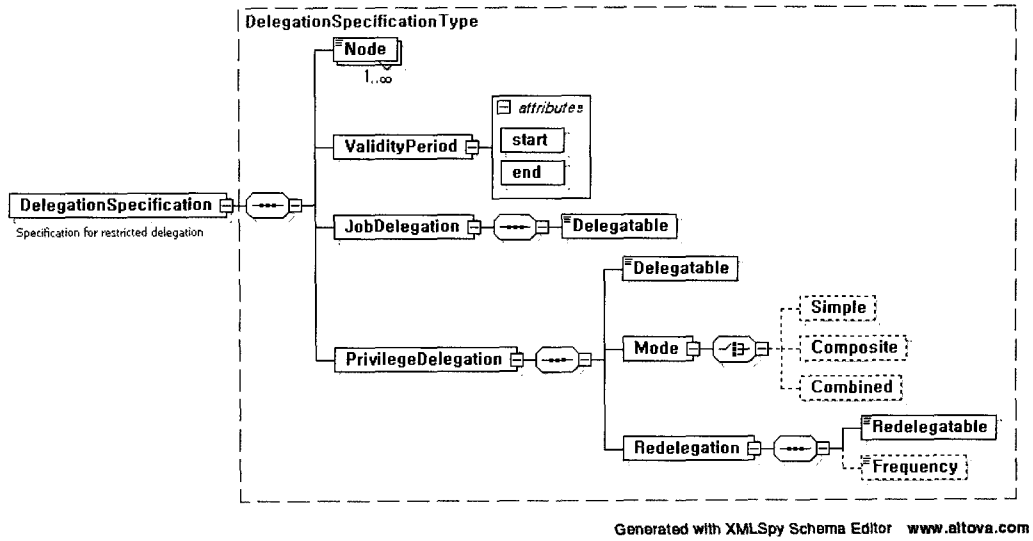
① Accept: {K_{DS} (DC_J)/K_{DS}(DC_P)}

② Reject: {"Delegation Request Rejected"}

C_U: 사용자 인증서
C_{JN}: 작업노드 인증서, C_{JP}: 작업프로세스 인증서,
DC_J: Job Delegation 인증서, DC_P: Privilege Delegation 인증서,
R_X: 권한 X

(그림 3) 위임 메시지의 구조

위임 메시지의 구조는 위임 프로토콜의 단계에 따라 크게 요청 메시지(*Request Message*), 검증 메시지(*Verify Message*), 결과 메시지(*Result Message*), 발행 메시지(*Issue Message*)로 구분된다. 요청 메시지는 위임을 요청하는 메시지이다. 작업 노드/작업 프로세스는 사용자의 인증서와 사용자가 제출



(그림 4) 위임 명세서의 XML 스키마

한 작업, 작업 노드/작업 프로세스의 인증서와 함께 위임하고자 하는 권한(Access, Reserve, Read, Write, Execute)을 요청 메시지에 포함하여 위임 서버에게 요청한다. 다음으로 위임서버는 검증을 요청하는 검증 메시지를 위임 명세 DB 모듈에 전달하고, 위임 명세 DB 모듈은 결과 메시지 구조로 Accept시에 사용자 인증서, 작업 노드/작업 프로세스의 인증서와 더불어 해당 위임에 대한 위임 인증서를 작성하여 위임 서버에 전달한다. 위임 거절(Reject) 시에는 위임 요청이 거절되었다는 메시지를 전송한다. 발행 메시지의 경우 위임 명세 DB에서 받은 메시지를 위임 서버가 위임 요청자에게 전달한다. 이 때 위임 서버는 위임이 위임 승인(Accept) 시에는 자신의 개인키로 서명한 위임 인증서를 전달하며, 위임 거절(Reject) 시에는 거절 메시지를 전달한다.

4. 제한적인 위임의 명세

이 장에서는 호환성 문제를 용이하게 처리하기 위하여 제안한 프로토콜을 바탕으로 작업 위임과 권한 위임에 관한 명세를 데이터 표현의 통합화 및 표준화된 채널을 제공하는 XML 스키마를 통해 명세서 및 인증서 구조로 표현하고, 예제 시나리오를 통한 제한적인 위임의 절차를 UML을 이용하여 보인다.

4.1 위임 명세서를 통한 제한적인 위임의 명세

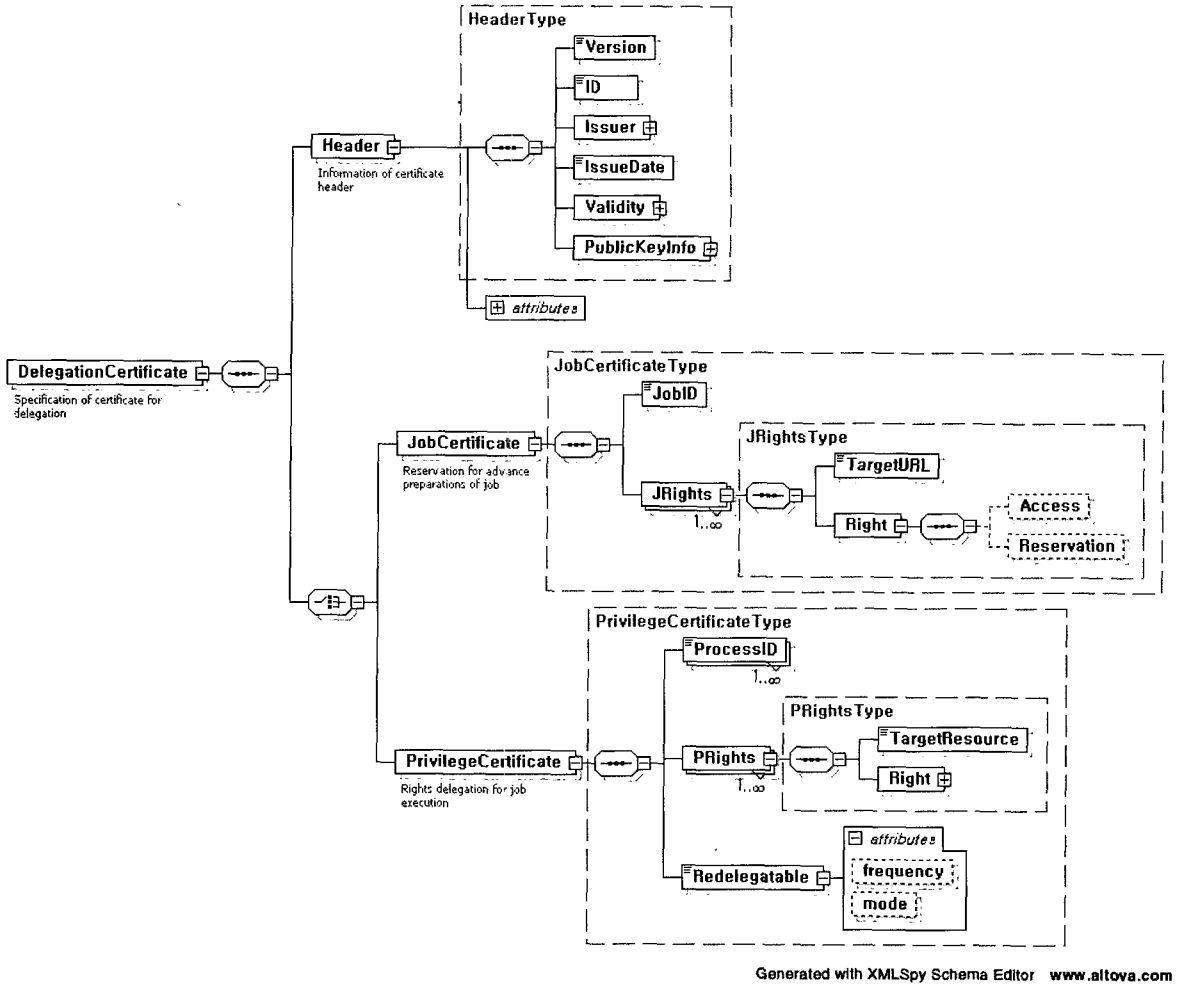
사용자는 제한적인 위임이 필요한 작업을 생성 시, 해당 작업에 알맞은 위임방식을 위임 명세서를 통하여 명세할 수 있다. 이렇게 명세 된 위임은 웹 포탈 등의 매개 방식을 통하여 작업위임 명세 DB 모듈로 전송되고, 모듈은 전송된 위임 명세서를 변환하여 DB에 저장한다. 이렇게 저장된 명세 데이터는 이후에 작업 노드나 작업 프로세스의 위임 요청 검증 시 사용된다.

위임 명세서의 기본적인 XML 스키마는 (그림 4)에 보인다. 위임 명세서의 구조는 크게 *Node*, *ValidityPeriod*, *JobDelegation*, *privilegeDelegation*으로 나눌 수 있다.

Node 부분은 위임을 받을 수 있는 작업 노드/작업 프로세스의 목록을 기술하는 부분으로, 위임 받을 개체를 제한하는 부분이다. *ValidityPeriod* 부분은 위임 인증서를 발행 시, 인증서의 기본적인 사용 기간을 명시하는 부분으로써, 실질적으로 위임이 가능한 시간을 나타낸다. *JobDelegation* 부분은 해당 작업에 대한 작업 위임의 가능 여부를 사용자가 설정하는 부분이다. 만약 이 부분을 "False"로 명세할 경우, 작업 노드가 작업 실행 전후의 호스트 접근이나 사전 예약 등에 대한 작업 위임을 요청 시 위임 요청이 거부된다. 마지막으로, *privilegeDelegation* 부분은 해당 작업에 대한 권한 위임에 대한 설정 부분으로, 권한 위임 여부와, 권한 위임 가능 시 적용 모드(Simple, Composite, Combined), 그리고 재위임 가능 여부(Redelegatable) 및 횟수(Frequency)를 설정할 수 있다.

4.2 위임 인증서의 구조

작업 노드/작업 프로세스로부터 위임 요청이 들어오면, 인증서버는 작업 위임 명세 DB 모듈에 요청된 위임을 전달한다. 작업 위임 명세 DB는 이렇게 전달된 위임을 DB와의 상호 작용을 통하여 적절한지 판단한 후, 위임 인증서를 생성하게 되고, 생성된 위임 인증서를 위임 서버에게 전달한다. 위임 서버는 전달 받은 위임 인증서를 자신의 개인키로 서명한 후, 위임을 요청한 작업 노드/작업 프로세스에게 전달한다. 이때의 위임 인증서는 위임의 종류에 따라 작업위임 인증서와 권한위임 인증서 2가지로서 표현될 수 있다. 작업위임 인증서는 작업의 위임을 위하여 작업 노드에 부여되는 위임 인증서로써, 작업과 관련된 시스템 자원 예약 및 작업 시작 전후 혹은 종료 후에 호스트로의 접근에 사용되게 된다. 이와 달리 권한위임 인증서는 실제 작업이 실행되는 작업 프로세스에 전달되는 위임 인증서로써, 작업의 실행 권한이 사용자가 기술한 위임 명세서에 의해 제한적으로 부여되고, 작업 실행 도



(그림 5) 위임 인증서의 XML 스키마

중에 필요한 자원 요청을 위한 위임 및 다중 작업의 호출을 위하여 사용된다. 또한, 재위임(redelegation)이 필요할 경우, 단순, 복합, 결합 모드를 통하여 위임 체인(Delegation Chain) 내에서의 권한 부여 모드를 설정할 수 있다.

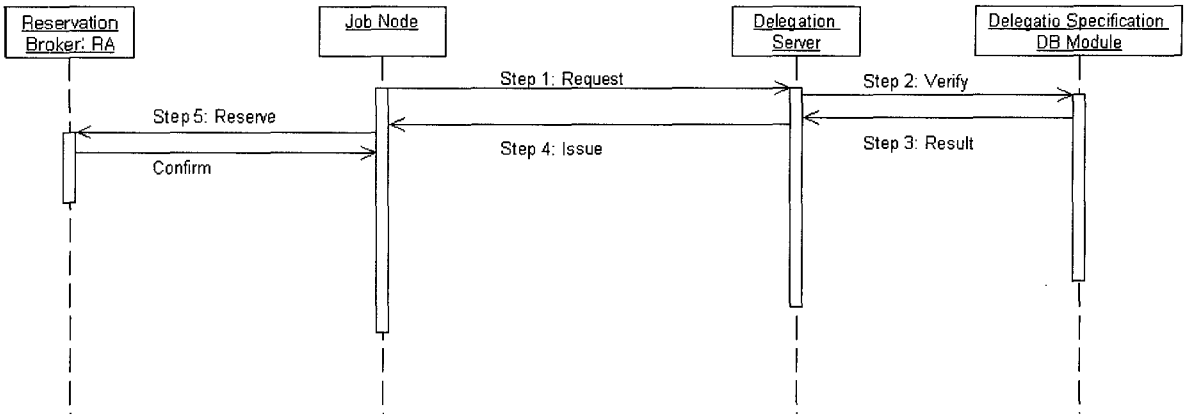
사용자의 작업이 제출된 후 작업 노드/작업 프로세스는 위임이 필요한 경우 위임 서버에서 위임 인증서를 발급 받아, 위임을 수행한다. 위임 인증서의 XML 스키마는 (그림 5)와 같다. 위임 인증서의 구조는 크게 *Header*와 인증서 부분으로 나눌 수 있으며, 인증서 부분에는 위임 목적에 따라, 작업 위임 인증서인 *JobCertificate*와 권한 위임 인증서인 *Privilege Certificate*로 나누어진다.

Header 부분에는 X.509 인증서와 유사한 정보가 포함된다. 인증서 버전, 일련 번호, 발행자, 발행 날짜, 유효기간, 공개키 정보 등이 들어가고, 유효기간에는 사용자가 사전에 명세한 위임 기간이 포함된다[11]. *JobCertificate* 부분은 관련된 작업의 ID와 작업 실행 전·후 접근할 호스트나 예약할 시스템 자원의 위치, 접근이나 예약을 구별할 수 있는 권한(Access/Reservation)으로 표현된다. *PrivilegeCertificate* 부분은 실질적으로 작업을 실행하는 도중 작업 프로세스가 사용자를 대신하여 또 다른 작업을 요청하거나, 자신의 권한 중 일부 혹은 전부를 다른 프로세스에게 위임해야 할 경우에 사용되며, 실제 위임을 받게 될 프로세스 ID의 목록과 요청할 자원의 위치, 그리고 그에 따른 권한(Read, Write, Execute) 및 위임 체인(Delegation Chain)을 위한 인증서의 재위임 여부가 표현된다.

이러한 인증서를 통해 위임 서버는 작업 노드나 작업 프로세스에게 해당 작업에 대한 권한의 위임을 좀 더 명확하고, 세분화화여 제한할 수 있으며, 위임 요청자는 이러한 위임 인증서의 명세를 통하여 위임 작업을 원활히 수행할 수 있게 된다.

4.3 제한적인 위임 적용 시나리오

이 절에서는 제한적인 위임을 적용한 시나리오를 예시한다. 작업 생성 환경은 분산 신뢰 환경이라고 가정한다. 사용자는 $HOST_A$의 작업 노드에서 실행할 J_X라는 작업을 생성한다. J_X라는 작업은 실행 전 시스템 자원 R_A의 예약과, 실행 프로세스와는 별도로 작업 노드 $HOST_B$에서의 프로세스 P_Y의 실행이 필요하다고 가정한다. 또한 작업 J_X를 실행하는 프로세스 P_X는 $HOST_A$의 모든 자원에 대한 읽기와 쓰기 권한을 가지며, 예약과 관련된 상세한 정보는 작업 생성 시 작업



(그림 7) J_X에 대한 작업위임 절차 순서다이어그램

노드의 작업 정보 DB에 저장되어 있다. 작업 생성과 더불어 사용자는 위임 명세서를 (그림 6)과 같이 작성하여 위임 명세 DB 모듈에 전송한다.

```
<?Xml version="1.0" encoding="UTF-8"?>
<DelegationSpecification>
        <Node>HOSTA</Node>
        <Node>HOSTB</Node>
        <ValidityPeriod start="2005-08-14" end="2005-08-15"/>
        <JobDelegation>
                <Delegatable>TRUE</Delegatable>
        </JobDelegation>
        <PrivilegeDelegation>
                <Delegatable>TRUE</Delegatable>
                <Mode><Composite></Composite></Mode>
                <Redelegation>
                        <Redelegatable>TRUE</Redelegatable>
                </Redelegation>
        </PrivilegeDelegation>
</DelegationSpecification>
```

(그림 6) JX에 대한 HOSTA에서 HOSTB로의 위임 명세서

(그림 6)에서 사용자는 자신의 권한을 위임 받을 수 있는 노드를 $HOST_A$와 $HOST_B$로 제한하였고, 위임 가능 기간을 1일로 설정하였다. 그리고 작업 위임과 권한 위임 모두 가능하며, 권한 위임의 경우에는 복합 모드로 위임하고 재위임이 가능하도록 위임 사항을 명세하였다.

4.3.1 작업위임 적용 시나리오

작업 J_X는 작업 실행 전 시스템 자원 R_A의 예약을 필요로 한다. 시스템 자원에 대한 예약은 노드의 권한 만으로는 어렵기 때문에, 작업 위임과 관련된 사용자 권한을 위임 서버에 요청해야 한다.

1. 작업 노드는 작업 정보 DB를 참조하여 작업 명세를 얻고, 위임 서버에 작업 위임을 요청한다.
2. 위임 서버는 요청된 작업위임을 위임 명세 DB 모듈에 전달하여 위임 검증을 받는다.
3. 위임 명세 DB 모듈은 DB에 저장되어 있는 Jx에 대한 위임 명세를 참조하여 검증 후 자원 R_A에 대한 작업 위임 인증서를 생성하여 위임 서버에 전달한다.
4. 위임 서버는 작업위임 인증서를 서버의 개인키로 서명하여 위임을 요청한 작업 노드에 전달한다.
5. 작업 노드는 전달된 작업 위임 인증서를 첨부하여 시스템 자원 R_A에 대한 예약을 수행한다.

<표 4>는 시나리오에 대한 작업 위임 구문을 단계별로 설명하였고, (그림 7)은 각 단계에 대한 절차를 UML을 통하여 표현하였다[12]. 이렇게 처리된 작업 위임의 결과는 (그림 8)과 같이 J_X에 대한 작업 위임 인증서로 나타난다.

〈표 4〉 J_X에 대한 작업 위임 구문

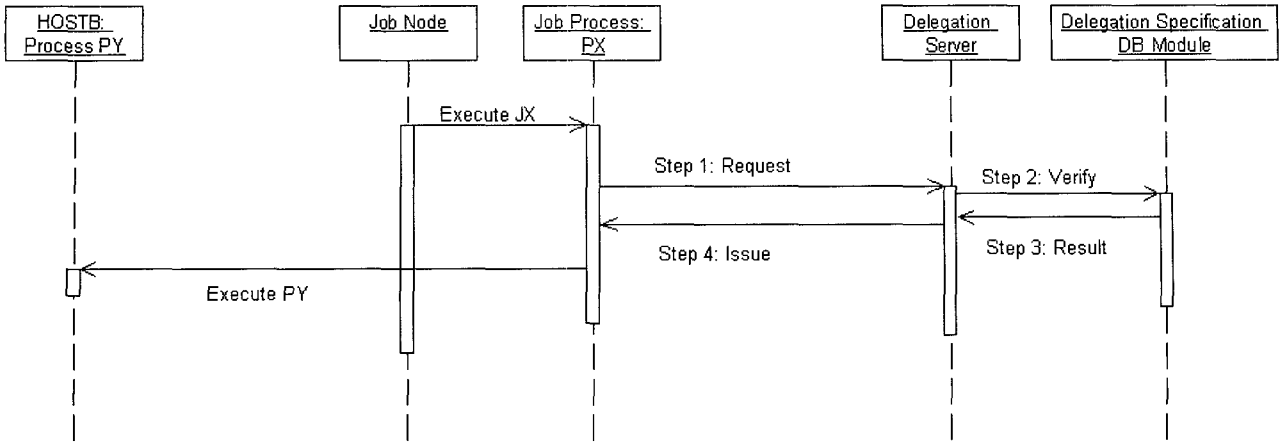
단계	작업 구문	설명
1	Request(C_U, C_{JN}, $J_X(R_A)$, $R_{RESERVATION}$)	R_A에 대한 예약을 위한 위임 서버에 작업 위임 요청
2	Verify(Request Message)	검증을 위해 요청 메시지를 위임 명세 DB 모듈에 전달
3	Result(C_U, C_{JN}, DC_J)	위임 요청에 대한 결과로 작업 위임 인증서를 위임 서버에 전달
4	Issue($KD_S(DC_J)$)	위임 서버의 개인키로 서명된 작업 위임 인증서 발행
5	Reserve($KD_S(DC_J)$, R_A)	작업 위임 인증서를 통하여 R_A 예약

```
<?Xml version="1.0" encoding="UTF-8"?>
<DelegationCertificate>
        <Header>
                ...(생략)
        </Header>
        <JobCertificate>
                <JobID>JX</ProcessID>
                <JRights>
                        <TargetResource>ReservationBroker:RA</TargetResource>
                        <Right><Reservation></Reservation></Right>
                </JRight>
        </JobCertificate>
</DelegationCertificate>
```

(그림 8) 작업 JX에 대한 작업 위임 인증서

4.3.2 권한위임 적용 시나리오

작업 노드는 작업 J_X에 대한 사전 예약이 완료 후 작업을 실행시킬 작업 프로세스 P_X를 생성한다. 작업 프로세스 P_X는 $HOST_A$에 대한 자원의 읽기, 쓰기 작업을 실행하는 중, 별도



(그림 9) J_X에 대한 권한위임 절차 순서다이어그램

의 $HOST_B$에서의 프로세스 P_Y를 실행시켜야 한다. 하지만, 다른 사이트의 프로세스를 실행시킬 권한은 프로세스 P_X 가 가지고 있지 않기 때문에, P_X는 위임 서버에 $HOST_B$의 P_Y를 실행시킬 권한 위임을 요청한다.

1. $HOST_A$의 작업 프로세스 P_X는 위임 서버에 $HOST_B$의 프로세스 P_Y를 실행시킬 권한 위임을 요청한다.
2. 위임 서버는 요청된 권한 위임을 위임 명세 DB 모듈에 전달하여 위임 검증을 받는다.
3. 위임 명세 DB 모듈은 DB에 저장되어 있는 J_X에 대한 위임 명세를 참조하여 $HOST_B$의 프로세스 P_Y에 대한 실행 권한을 복합(Composite) 모드로 허용함을 검증한 후, 권한위임 인증서를 생성하여 위임 서버에 전달한다.
4. 위임 서버는 전달 받은 권한 위임 인증서를 서버의 개인키로 서명하여 위임을 요청한 프로세스 P_X에 전달한다.
5. 프로세스 P_X는 전달받은 위임 인증서를 첨부하여 $HOST_B$에서 인증을 받은 후, 프로세스 P_Y를 실행시킨다.

〈표 5〉 J_X에 대한 권한 위임 구문

단계	작업 구문	설명
1	Request(C_U, C_{JP}, $J_X(HOST_B(P_Y))$, $R_{EXECUTE}$)	$HOST_B$의 프로세스 P_Y를 실행할 권한을 위임 서버에게 요청
2	Verify (Request Message)	검증을 위해 요청 메시지를 위임 명세 DB 모듈에 전달
3	Result(C_U, C_{JP}, DC_R)	위임 요청에 대한 결과로 권한 위임 인증서를 위임 서버에 전달
4	Issue($KD_S(DC_R)$)	위임 서버의 개인키로 서명된 권한 위임 인증서 발행
5	Execute($KD_S(DC_R)$, $HOST_B(P_Y)$)	권한 위임 인증서를 통하여 $HOST_B$의 P_Y를 실행

<표 5>는 시나리오에 대한 권한 위임 구문을 단계별로 설명하였고, (그림 9)는 각 단계에 대한 절차를 UML을 통하여 표현하였다. 이렇게 처리된 권한 위임의 결과는 (그림 10)과 같이 J_X에 대한 권한 위임 인증서로 나타난다.

```
<?Xml version="1.0" encoding="UTF-8"?>
<DelegationCertificate>
        <Header>
                ...(생략)
        </Header>
        <PrivilegeCertificate>
                <ProcessID>HOST:PX</ProcessID>
                <PRights>
                        <TargetResource>HOSTB:PY</TargetResource>
                        <Right><Read></Read></Right>
                        <Right><Write></Write></Right>
                        <Right><Execution></Execution></Right>
                </PRight>
                <Redelegatable>true</Redelegatable>
        </PrivilegeCertificate>
</DelegationCertificate>
```

Composite Mode (P_X 권한 + 위임권한)

(그림 10) 작업 JX에 대한 권한 위임 인증서

4.4 기존의 위임과 제안한 위임의 비교

〈표 6〉 제한적인 위임 비교

제한적인 위임	분산 신뢰 환경	그리드 환경		제안한 위임
		Globus	Legion	
모든 권한의 위임	O	O	O	O
작업 실행 중 권한 위임	X	△	△	O
권한의 세부적 위임	△	X	△	O
재위임	O	O	X	O
자원 예약	X	X	X	O
작업 수행 전·후 위임	X	X	X	O

<표 6>은 제한적인 위임의 요구사항으로 기존의 방식들과 본 논문에서 제안한 방식을 비교한 것으로 다음과 같이 설명될 수 있다.

작업 실행 중 권한 위임의 경우 기존의 방식에서는 작업이 실행되는 시점부터 종료 시점까지 주어진 권한을 모두 사용할 수 있기 때문에, 인증서가 악의적인 사용자에 의해 사용될 경우, 인증서에 부여된 모든 권한을 사용할 수 있는 보안 취약성이 존재한다. 하지만, 본 논문에서 제안한 위임의 경우에는 작업 노드와 프로세스만이 해당 권한을 사용할 수 있으므로, 권한의 무조건적인 오용을 피할 수 있다.

권한의 세부적인 위임의 경우, 본 논문에서는 위임을 작업 측면과 권한 측면으로 분류하여 권한을 부여하기 때문에 작업 특성에 알맞고, 꼭 필요한 최소한의 권한만을 위임한다. 그러나 분산 신뢰 환경에서의 위임은 권한 외적 측면만을 강조함으로써, 작업 특성에 맞는 권한 부여가 이루어지지 못하고, Legion의 경우 이를 지원하지만, 객체 기반의 시스템 특성으로 인하여 분산 환경이나 그리드 환경에 적합하지 못한 확장성 문제를 가진다.

재위임의 경우에는 대부분의 방식들이 이를 허용하고 있지만, 자원 예약이나 작업 수행 전·후의 위임의 경우에는 권한만을 강조하는 기존의 방식들에서 적용이 불가능하였다. 그러나 본 논문에서는 작업 특성에 맞게 위임을 분류하였기 때문에 적용 가능하게 되었다.

5. 결론 및 향후 연구

분산 컴퓨팅 환경은 계속적으로 확장 및 발전하고 있고 그로 인한 시스템의 보안 위험성 또한 증가하고 있다. 시스템 보안에 있어서 인증과 접근제어는 가장 중요한 부분 중의 하나이고, 이들을 효율적으로 정의, 구현하는 것이 필요하다. 특히 위임은 인증과 접근제어를 필요로 하는 기술이지만, 현재 위임에 대한 세부적 범위 및 위임 가능한 권한 정의가 확실하지 않아 보안에 커다란 취약점을 가지고 있다.

본 논문에서는 위임에 대한 보안 취약점을 극복하기 위하여 분산된 신뢰 환경에서 작업 실행 시 사용될 수 있는 제한적인 위임 프로토콜 및 위임의 명세를 제안하였다. 또한, 위임 자체를 작업 측면과 권한 측면으로 분리하여 작업 수행 전·후의 호스트 접근, 자원 예약 및 실행 중 권한 위임의 자세한 표현을 가능하게 하였다. 본 논문에서 제안한 방법은 무조건적인 권한 위임에 대한 권한 남용 및 오용의 피해를 최소화할 수 있고, 사용자가 작업에서 사용할 자신의 권한에 대해 좀 더 세부적으로 제어할 수 있다. 그리고 데이터 표현의 통합 및 표준을 제공하는 XML을 통하여 위임을 명세함으로써, 다양한 분산 환경 기술에 호환성 문제를 해결하였다.

현재 대표적인 분산 신뢰환경인 그리드 환경에서는 제약적인 위임 기능이 지원되지 않고 있다. 그러므로 향후 본 논문을 확장하여 그리드 환경에 적합하도록 하나의 분산 신뢰 도메인 내에서의 위임뿐만 아니라 다중 분산 신뢰 도메인 사이에서의 위임 또한 연구할 필요성이 있다.

참 고 문 헌

[1] 김학두, 김진석, "그리드 미들웨어 : 자원 관리 및 원격 데이터 접근 기술 동향", 정보과학회지, 제 20권 제 2호, 2002

[2] R. Welch, D. Engert, I. Foster, S. Tueke, J. Volmer and G. Kesselman, "A National-Scale Authentication Infrastructure", IEEE Vol.33, Issue 12, pp.60~66, 2000

[3] G. Stoker, B. S. White, E. Stackpole, T. J. Highley and M. Humphrey, "Toward Realizable Restricted Delegation in Computational Grids", European High Performance Computing and Networking(HPCN) Europe, pp.32~41, 2001.

[4] 김승현, 김 종, 홍성제, 김상완, "그리드 환경에서 제약적인 권한 위임을 고려한 아키텍처", 한국정보과학회 컴퓨터시스템연구회 추계 학술발표회, 2002.

[5] L. Kagal, T. Finin and Y. Peng, "A Delegation Based Model for Distributed Trust", Proceeding of The IJCAI-01 Workshop on Autonomy, Delegation, and Control, 2001.

[6] B. Sotomayor, "The Globus Toolkit 3 Programmer's Tutorial"

[7] 김승현, 김 종, 홍성제, 김상완, "그리드 환경에서의 작업 시나리오에 기반 한 제약적 권한 위임", 한국정보보호학회 종합학술발표회 논문집, Vol.12, No.1, 2002.

[8] A.Grimshaw, A. Ferrari, F. Knabe and M. Humphrey, "Wide-Area Computing : Resource Sharing on a Large Scale", IEEE Computer, 32(5) : 29-37, 1999.

[9] M. Humphrey and M. Thompson, "Security Implications of Typical Grid Computing Usage Scenarios", GFD-I.12, 2000.

[10] Object Management Group, "CORBAservices : Common Object Services Specification", Dec., 1998.

[11] 오홍룡, 염홍열, "XML 서명을 이용한 접근 제어 모델", 한국정보보호학회 동계정보보호학술대회 논문집 Vol.13, No.2, 2003.

[12] M. Ahsant, J.Basney and O. Mulmo, "Grid Delegation Protocol", UK Workshop on Grid Security Experiences, Oxford 8th and 9th, 2004.

은 승 희

e-mail : hyunmu7@lsrc.chonnam.ac.kr

2002년 전남대학교 컴퓨터정보학부(이학사)

2004년~현재 전남대학교 정보보호협동과정 석사과정

관심분야 : 시스템 및 네트워크 보안, 접근통제 기술 등

김 용 민

e-mail : bluearain@yosu.ac.kr

1989년 전남대학교 전산통계학과(이학사)

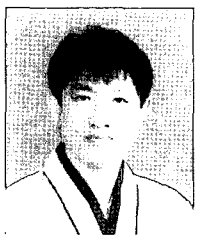
1991년 전남대학교 전산통계학과(이학석사)

2002년 전남대학교 전산통계학과(이학박사)

2003년~2004년 전남대학교 리눅스시스템 보안연구센터 Post-doc.

2004년~현재 여수대학교 정보기술학부 전임강사

관심분야 : 시스템 및 네트워크 보안, 전자상거래보안 등



노 봉 남

e-mail : bongnam@chonnam.ac.kr

1978년 전남대학교 수학교육과(학사)

1982년 KAIST 전산과(석사)

1994년 전북대학교 전산과(박사)

1983년~현재 전남대학교 전자컴퓨터정보통신공학부 교수

2000년~2003년 리눅스보안연구센터 소장

2004년~현재 시스템보안연구센터 소장

관심분야 : 컴퓨터와 네트워크 보안, 정보보호시스템, 전자상거래 보안, 사이버사회와 윤리