Repository 를 활용한 CORBA Security 의 보안정보관리 시스템 설계

유기영*, 김기봉**,진성일*, 인소란***
*충남대학교 컴퓨터과학과, **대전보건대학 컴퓨터정보처리과,***(주)니츠
e-mail : gyyoo@cs.cnu.ac.kr

The Design of Security Information Management System of CORBA Security with Using Repository

Ki-young Ryu*, Ki-bong Kim**, Seong-il Jin*, So-ran In***
*Dept. of Computer Science, Chungnam National University
**Dept. of Computer Information Processing, Taejon Health Sciences College
*** NITZ(Next Information Technology Zone) Corp.

요 약

분산객체 처리 환경에서 CORBA 를 기반으로 한 응용 객체간에 전달되는 정보에 대하여 보안을 유지하기 위해 요구 되는 보안 정보를 관리하는 기법을 제안하고 이들을 관리 하기 위한 보안정책 들의 객체들을 효율적으로 관리하기 위하여 Repository 응용 방안을 제시하였다.

1. 서론

최근 정보처리 기술 중 하나인 분산객체 처리 환경은 여러 통신망으로 연관된 자원들을 공유하는 분산처리 환경과 상속, 다양화, 캡슐화, 재사용 등의 장점을 지닌 객체 지향 처리 환경을 결합한 것으로 주어진 환경 조건을 기반으로 융통성 있게 통신 분야에 신뢰성 있는 새로운 형태로 주목을 받고 있다. 그러나 정보처리 환경은 정보의 공유가 확산됨과 동시에 정보보안의 역기능적인 측면의 부작용도 증가하여 심각한 문제로 대두되고 있다.

분산객체 처리 환경에서 정보보안을 지원하기 위한 기술들이 많이 개발 중에 있고 분산 객체 처리 환경에서 CORBA 를 기반으로 하는 응용 객체간의 정보보안을 지원하기 위하여 OMG CORBA Security 명세서를 제안하고 있다[6][10].

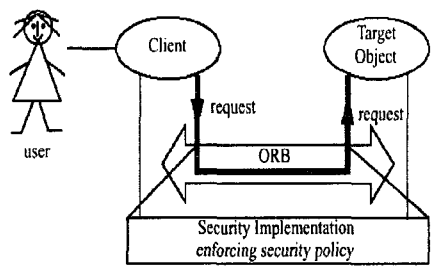
또한 OMG CORBA Security 명세서에 있는 보안 정보 관리 부분만 생각해 볼 때 각각 보안 정보 관리를 하기 위한 정책 정보와 보안 정보의 관리가 어렵고 단순한 테이블 형태로 존재하여 각 테이블을 단순 조인하는 검색으로 이루어 지고 있다. 본 논문에서는 보안 정보관리의 정확성, 무결성, 효율성을 기하기 위한 Repository 활용 방안을 제시하였다.

2. CORBA 보안 구조

2.1 보안 참조 모델

보안 참조 모델은 안전성을 필요로 하는 시스템이 보안 정책을 어디서, 어떻게 시행하는가를 기술한다. 보안 정책은 객체를 접근할 수 있는 조건, 사용자 또는 Principal 이 누구이며 허용된 일이 무엇이고 그들의 권한을 위임할 수 있는지 여부를 보여주는 인증에 대한 정보, 객체간 통신의 안전성 품질(Quality of Protection 등), 보안 관련 행위들에 대하여 어떤 책임이 요구 되는지 등을 정의한다.

보안 정책에는 접근 제어 정책(Access Control Policy), 감사정책(Auditing Policy), 인증 정책(Authentication Policy), 보안 호출 정책(Security Invocation Policy), 부인봉쇄 정책(Non-Repudiation Policy), 위임 정책(Delegation Policy) 등이 있을 수 있다[2].



<그림 1>객체 시스템을 위한 보안 모델[2]

CORBA 를 기반으로 한 객체 시스템의 보안 모델은 <그림 1>에서 보는 바와 같이 나타낼 수 있다. 모든 작업은 반드시 보안 정책을 시행하는 적절한 구현 모듈을 거쳐서 수행 되도록 하고(by-pass impossible),

* 이 논문은 '99 산업자원부에서 시행하는 "민군겸용기술사업"의 과제로 수행되었습니다.

보안 기능들은 그 자체가 도중에 불법으로 변경되거나 절취 되지 않도록 안전성을 유지해야 하며, 보안 정책에 의해 항상 호출될 수 있어야 한다.

대부분의 응용 객체들은 보안 정책이 어떻게 되어 있는지 또는 내부적으로 어떻게 처리되는지 알지 못한다(unaware of security). 사용자는 응용 클라이언트를 호출하기 전에 사용자로서의 자격을 인증 받고, 계속해서 보안 서비스가 자동으로 수행된다. 어떤 응용 객체들은 시스템이 제공하는 보안 정책의 통제를 받고 있지만 그 자신이 보안 기능을 수행하지는 않으며. 어떤 응용 객체는 자신이 정한 보안 정책(예를 들면, 그들 자신의 데이터에 대한 접근 제어나 보안 관련 감시 활동)들을 수행한다[4][5][10].

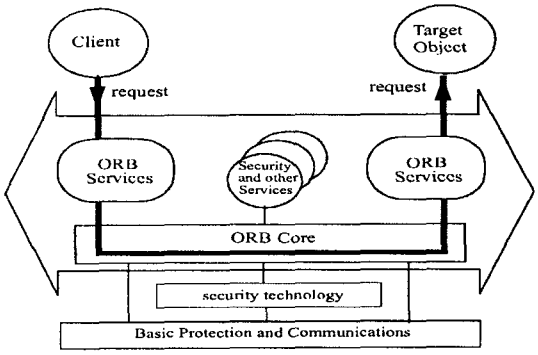
ORB 는 보안 정책을 위배하지 않는 정당한 요구에 대해서는 최소한 올바른 처리를 해주도록 해야 하며 보안 정책에 의해 요구된 보안 서비스를 수행해야 한다.

보안 모델은 일반적으로 보안 정책들의 특정 집합을 정의한다. OMA(Object Management Architecture)는 서로 다른 시장의 요구를 만족하는 다양한 보안 정책을 광범위하게 지원해야 하므로 단일 보안 모델의 제안은 적합하지 못하고 많은 다른 종류의 정책들을 만들어 낼 수 있는 기본 골격(framework)을 제공하는 보안 참조모델(또는 meta-policy)을 정의하는 것이 필요하다. Meta-policy 는 보안 구조에서 제공하는 추상화된 인터페이스와 가능한 보안 기능들을 정의하고 유연성 있는 지침을 제공한다[2].

2.2 보안 구조 모델

CORBA 에서 object invocation 시 보안 서비스를 제공하기 위한 structural model 은 아래 와 같이 4 개의 구성 요소로 구성된다.(<그림 2>참조)

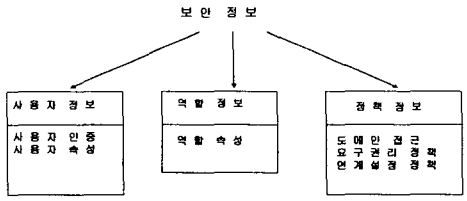
- Application-level components
- 특정 보안 technology 에 독립적인 보안 서비스 components
- 특정 보안 technology 처리 components
- Basic protection & communication components



<그림 2> Structural model 의 구성[2]

3. 보안 정보 관리 기법

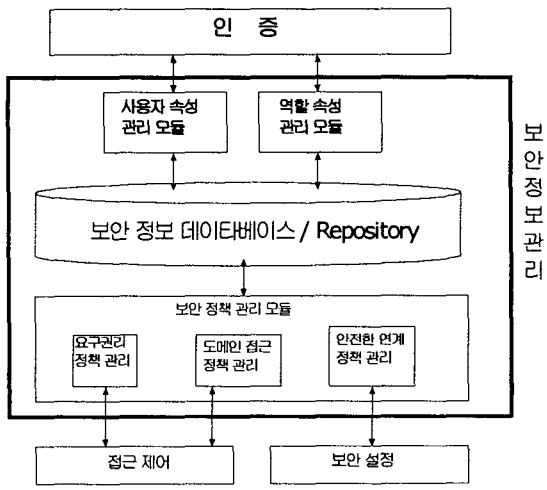
CORBA Security 환경에서 관리해야 하는 보안 정보는 사용자의 인증 정보 및 속성 정보와 역할 속성 정보, 접근 제어를 위한 정책 정보, 안전한 연계를 설정하기 위한 정책 정보로 구성 할 수 있다.



<그림 3>보안 정보 관리 객체 환경에 있는 보안 정보

사용자 인증 정보는 사용자의 암호와 비밀 키 등이 있고, 사용자 속성 정보는 사용자의 역할 이름과 속성 권한 등이 있다. 또한 역할 속성 정보는 이 역할에 존재하는 타겟과 위임 여부 정보 등이 있다. 접근 제어 정보는 사용자가 응용 객체에 기술된 인터페이스의 특정한 메소드를 실행하는 여부를 나타낸 요구권리(RequiredRights) 정책 정보와 사용자의 권한 속성에 특정한 권리를 결합시킨 형태로 권한 속성과 위임 프래그와 권리로 구성되는 도메인 접근 정책(DomainAccessPolicy) 정보 등이 있고, 응용 클라이언트와 응용 서버사이에 데이터를 주고 받는데 요구되는 데이터의 보안 수준을 나타내는 안전한 연계를 설정하는 정책(SecureInvocationPolicy)정보가 있다.

보안 정보 권리 구조는 사용자 속성 관리 모듈, 역할 관리 모듈, 접근제어 정책 관리 모듈, 안전한 연계 설정 정책 관리 모듈로 구성된다.



<그림 4> 보안 정보 관리 구성 요소[6]

사용자 인증 및 속성 관리 모듈은 사용자 데이터베이스(Repository)에 정의되어 있는 사용자 인증 정보 및 속성 정보를 관리하며, 역할 속성 관리 모듈은 역할 데이터베이스(Repository)에 정의 되어 있는 역할 속성 정보를 관리한다. 이들 두 관리 모듈은 사용자 인증을 위하여 아래와 같은 관리 정보를 제공하는 인

터페이스를 갖는다[6][7].

```
//User authentication and attribute management
    void UserAuthentication:get_user_auth();
//Role privilege management
    void UserPrivilegeAttribute::add_user();
    void UserPrivilegeAttribute::delete_user();
    void UserPrivilegeAttribute::replace_user();
    void UserPrivilegeAttribute::get_user();
```

보안 정책관리 모듈은 접근 제어를 위한 정책 정보와 안전한 연계 설정을 위한 정책 정보를 관리하는 인터페이스와 객체에 대하여 사용자가 접근이 허용되는 지를 결정하거나 응용 클라이언트와 응용 서버 사이의 안전한 연계 설정을 위하여 요구되는 아래와 같은 각 정책 정보를 제공하는 인터페이스를 갖는다.

```
//RequiredRights Policy management
      void RequiredRights::get_required_right();
      void RequiredRights::set_required_right();
// Domain Access Policy management
      void DomainAccessPolicy::grant_right();
      void DomainAccessPolicy::get_right();
      void DomainAccessPolicy::replace_right();
      void DomainAccessPolicy::revoke_right();
//Security Invocation Policy management
      void SecurityInvocationPolicy::get_association();
      void SecurityInvocationPolicy::set_association();
```

CORBA 보안의 다른 서비스(인증, 접근제어, 안전한 연계설정 서비스)는 자신의 기능을 수행하기 위해서 우선 원격에 있는 보안 정보 관리 객체에 바인딩하며[1], 이후 적절한 인터페이스를 이용하여 보안 정보 데이터베이스(Repository)로부터 보안 정보를 얻어 온다. 또한 이러한 보안 정보 관리에 대한 관리는 특정한 관리자 계정을 두어 관리 하도록 한다.

응용 객체 클라이언트는 응용 객체를 호출할 때에는 정보에 대하여 보안을 유지하기 위하여 필요한 보안 정보를 얻는다. 각 보안 정보의 유형에 따라 적절한 보안 정보 제공 객체를 호출한다. 보안 제공 객체는 사용자 정보, 역할 정보, 도메인 접근 정책 정보, 요구권리 정책 정보, 안전한 연계 설정 정책 정보 등의 객체가 있다.

4. Repository 적용

4.1 보안 정보 관리를 위한 Repository

보안 정보 관리 객체를 설계할 때 각각의 필요한 사용자 정보, 역할 정보, 도메인 접근 정책 정보, 요구권리 정책 정보, 안전한 연계 설정 정책 정보 등의 정보들을 단순한 데이터베이스에 있는 테이블 형태로 존재하여 관리 하는 것 보다 Repository 를 두어 관리함으로써 관리자의 관리 형태나 설계 후 시스템 구현 작업 측면에서도 많은 효과를 볼 수 있다

각각의 Object Policy 별 연계설정 시 정책 들의 연관성을 지어 줄 수 있고, 요구권리, 역할 정보, 정책 정보 등의 수정, 추가 등의 History 를 버전 별로 관리하고 정리 할 수 있다

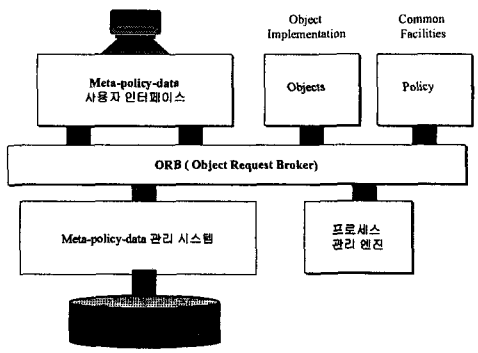
보안 정보 관리를 위한 Repository 는 데이터베이스만으로 보안 정보 관리 시스템을 구현하여 사용하는 것 보다 많은 효율성이 있다. 데이터베이스만으로는 각각의 Object Policy 의 형상정보를 명확히 표현하지 못하고, Object Policy 별 도메인 멤버들의 관리가 하나하나 이루어 지기가 어렵다. 또한 Object Policy 들의 수정, 추가 등의 작업이 발생하였을 때 해당 Object Policy 들의 이력 정보를 관리 할 수 가 없는 실정이다.

CORBA 기반의 Repository 를 이용한 경우는 Repository 가 보안 정보 관리 시스템의 Object Policy 정보를 관리. 저장되는 곳으로, 각 Object 들, 각 보안 단계들, 사용자들, 응용프로그램 사이의 시스템 정보를 연관성을 지을 수 있도록 해준다. 이러한 Repository 는 보안 정보 관리 시스템 유지 보수를 용이하게 하여 보안 정보 관리 시스템의 관리자로 하여금 관리적인 면에서는 시간적, 시스템의 구성적인 면에서는 공간적 비용의 감소라는 큰 효과를 가지고 있기 때문에 위에서 말한 데이터베이스 만으로 보안 정보 관리 시스템을 설계하고 구현하여 발생하는 시스템 구현 측면의 어려운 점과 관리적인 측면에 발생하는 문제점을 보완 할 수 있다. Repository 에 저장되는 정보의 종류로는 Object Policy 로서 각각 하나하나의 Policy 를 Meta-policy-data 로 보고 논리적 데이터 및, Policy 데이터의 규칙, Policy 간의 관계 등이 존재한다

4.2 CORBA 기반의 Repository 시스템 적용

Repository 는 기본적으로 데이터베이스 서비스를 사용하고 그 위에서 작동 한다. Repository 는 전형적으로 메타데이터와 관리 측면에 비중을 두어 다룬다[3].

본 논문에서 제안 하고자 하는 보안 정보 관리 역시 관리적인 측면으로 Repository 전형적인 구조에 상당히 많은 점이 부합 된다고 할 수 있다. 보안 정보 관리를 위한 Repository 의 주요 서비스는 Meta-policy-data 와 스키마의 관리, 변경 관리, Policy 들의 통합과 위치정보 등이다. Repository 는 데이터베이스에 비해 적은 데이터를 다루면서 데이터베이스와 함께 공존하는 형태를 가지고 있다.[9]



<그림 5>분산객체 기반의 Repository 시스템 구조[9]

각각의 보안 관리 정보들을 메타데이터라고 보고 이들에 대한 메타데이터의 상호 교환, 저장, 관리 등의 기능을 포함 한다. 관리자는 Meta-policy-data 인터

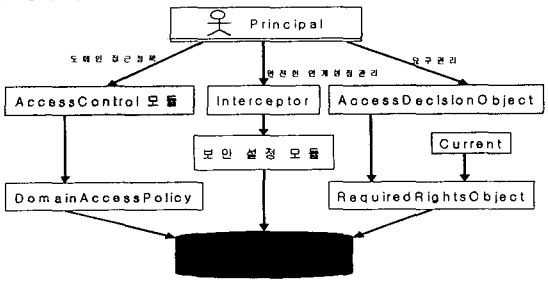
* 이 논문은 '99 산업자원부에서 시행하는 "민군겸용기술사업"의 과제로 수행되었습니다.

페이스를 통하여 쉽고 빠른 관리 환경을 제공하고 앞에서 제시한 보안 정보관리 기법에 맞추어 관리할 수 있다.

ISO IRDS 표준안을 근간으로 설계한 서비스 인터페이스는 시스템 운영에 관한 서비스, 메타데이터 단계에 독립적인 서비스, 메타스키마 단계에 독립적인 서비스로 크게 3 가지로 구분 할 수 있다[8]. 시스템 운영에 관한 서비스는 시스템의 활성화 및 종료 등 Repository 시스템 운영에 관한 서비스로 Repository 개방 서비스, Repository 폐쇄 서비스, 메타스키마 생성 서비스, 메타스키마 제거 서비스가 있다. 메타데이터 단계에 독립적인 서비스는 메타데이터 단계의 자료를 조작할 때 사용되는 서비스로 메타데이터 객체/관계 추가 서비스, 메타데이터 객체/관계 삭제 서비스, 메타데이터 객체/관계 수정 서비스, 메타데이터 출력 서비스, 메타데이터 생성 서비스, 메타데이터 제거 서비스가 있다. 메타스키마 단계에 독립적인 서비스는 메타스키마 단계의 자료를 조작할 때 사용되는 서비스로 메타스키마 객체/관계 추가 서비스, 메타스키마 객체/관계 삭제 서비스, 메타스키마 객체/관계 수정 서비스, 메타스키마 출력 서비스가 있다.

사용자 인터페이스는 Meta-policy-data 관리 시스템을 탐색할 수 있는 인터페이스로 메타정보의 계층적 구조와 관계 형식으로 표현하고 이러한 인터페이스의 기능으로는 메타 정보의 저장, 삭제 등에 관련된 Repository 기능, 객체나 타입의 정보, 그들 사이의 관계, 속성, 연산, 버전 정보 등을 표현하기 위한 View 기능이 있다.

이러한 Meta-policy-data 의 정보를 저장하는 정보 저장 테이블의 구조는 크게 각각의 보안 정보들의 데이터가 들어가 있는 내부테이블과 해당 Meta-policy-data 정의 및 연관된 Meta-policy-data 들 상에서 제공되는 서비스들을 제어하기 위한 환경 테이블, 모든 Meta-policy-data 정의와 Meta-policy-data 의 명명, 버전 제어 및 기타 공통 기능 관리를 위한 공통 테이블로 구성된다.



<그림 6> Repository 를 적용한 보안정보관리 메커니즘

[요구권리]

요구권리는 보안정보관리 메커니즘을 통해서 각각 거쳐야 할 Object 들의 정보를 Repository 에서 Meta-policy-data 정보를 얻고 추가, 삭제 등의 작업을 수행하게 된다. Domain 들은 서로 같은 Policy 를 가지고 있을 수 있다. 이렇게 중복된 Policy 의 변경 혹은 삭제로 인한 여러 개 그리고 다차원적인 Domain 을 별도로 관리 하면서 같이 변경하여 줄 수 있고, 만약 변경된 domain 과 Policy 를 적용하여 요구권리를 하였을 때는 정상적인 운용이 가능해진다. 따라서 Repository 를 사용하게 되면 domain 과 Policy 들을 별도로 관리할 수 있게 인터페이스를 추가하여 만들어 줄 필요가 없게 된다.

[안전한 연계설정관리]

연계설정관리도 요구권리와 같은 방식으로 보안정보관리에 필요한 각각의 Object 의 정보를 Repository 에서 Meta-policy-data 정보를 획득한다. 각각의 객체들끼리의 연계설정에서 오는 Version 정보유지는 과거에 연계를 지어준 정보가 존재 하지 않음으로 해서오는 반복적인 작업과 History 정보를 무시하고 같은 레코드에 Update 하는 낭비를 줄일 수 있다. 보안 설정 모듈을 통해서 들어오는 객체들은 접근할 때 마다 연계설정을 수시로 해주어야 한다. 보안객체의 History 정보로 인한 반복적인 작업이 없어지고 또한 매번 Update 에서 발생되는 트랜잭션도 감소한다.

[도메인 접근정책]

도메인 접근정책도 앞에서 제시한 요구권리와 연계설정관리 메커니즘과 같은 방식으로 Meta-policy-data 정보를 Repository 에서 추가, 삭제 등의 형식으로 관리 되어진다. Principal 의 순수성을 검증할 수 있도록 Level 를 두어 앞 단계에서 어떤 작업으로 인증 되어 거쳐 왔는지를 알 수 있고 신임할 수 있는지 여부를 판단 할 수 있다. Domain 과 Policy 들 사이의 연관 관계성을 나타낼 수 있고 그들 간의 스키마 관리도 가능해진다.

5. 결론

본 논문은 분산객체 처리 환경에서 보안 서비스를 제공하기 위해 필요한 보안 정보들을 관리하기 위한 기법을 제시하고 설계 하였다. 본 논문에서 제시한 보안 정보 관리 기법은 CORBA Security 명세서를 따라 설계 하였다. 그리고 이 보안 정보 관리를 관리자 측면에서 보다 더 효율적이고 정확하고 투명하게 관리하기 위해서 보안 정보 관리를 데이터베이스만을 기반을 둔 단순 테이블을 사용하지 않고 Repository 를 사용하여 CORBA 보안 정보 관리에 적용할 수 있는 시스템으로 설계 하였다.

참고문헌

[1] OMG. "CORBAservice : Common Object Specification",1996
[2] OMG. "CORBA Security service Specification",1995
[3] ISO/IEC, "Information technology IRDS framework", *ISO/IEC, 1993*
[4] Randy Otte, "Understanding CORBA", Prentice Inc., 1996
[5] Thomas J. Mowbray, "The Essential CORBA ", Wiley & Sons Inc., 1995
[6] 나중찬," SCAP 에서의 보안 정보관리" ,제 1 회 개방형보안기술과 응용 워크샵, 통신정보보안학회, 1997
[7] 송영기, "초고속정보통신 기반 안전성지원 플랫폼개발", 제 1 회 개방형보안기술과응용워크샵,통신정보보안학회, 1997
[8] 한국전산원," IRDS 서비스 표준 연구", 한국전산원, 1995
[9] 염태진, 박재형, 리자, 김기붕, 진성일, "분산 환경에서의 IRDS 기반 정보저장소 설계및구현," 한국정보처리학회, 1998
[10] 이권일," 분산객체환경에서의 보안서비스 구현" ,통신정보보안학회논문지, 1998