CORBA기반의 보안구조에 관한 연구

이수연*, 김창련**, 김석우**, 정진욱*

* 성균관대학교 정보공학과
** 한국전자통신 연구소

A Study on Security Architecture in CORBA Environment

Suyoun Lee*, Changryeon Kim, Seokwoo Kim, Jinwook Chung* ,
* Dept. of Information Engineering , SungKyunKwan University
** ETRI

요 약

개방형 분산 컴퓨팅 환경에 관한 표준구조인 CORBA를 기반으로 하여 CORBA에서 보안 요구사항 중 사용자가 자신의 시스템에 접근하는 것과 분산되어 있는 객체에 접근하기 위한 선결조건으로 인증을 수행하는 데 필요한 인증 메커니즘으로 본 논문에서는 분산환경을 위해 개발 된 Kerberos 버전5의 사용과 이를 위한 보안 인터페이스로 GSS-API를 이용함으로 CORBA 보안요구를 만족하는 보안구조를 제시하고자 한다.

1. 서 론

정보 처리 기술과 컴퓨터 네트워크의 발달은 분산 컴퓨팅(Distributed Computing)이라는 새로운 컴퓨팅 환경을 만들게 되었고 객체 지향 프로그래밍(Object-Oriented Programming)의 발달과 필요성에 따라 분산 된 통신망에서도 객체지향 기술이 적용되는 분산 객체 컴퓨팅 환경이 생겨났다. 이러한 분산 객체 컴퓨팅(Distributed Object Computing) 은 분산 컴퓨팅의 기본 단위를 객체(Object)로 한 것으로 이 기종 컴퓨터 시스템으로 구성된 클라이언트-서버 환경에서 분산 객체간의 상호운영성(Interoperability)과 이식성(Portability)을 증진시키는 것을 목적으로 하고 있다.

현재 관련업체들간의 협의체인 OMG(Object Management Group)에서는 분산 객체 컴퓨팅 시스템을 표준화하려는 작업의 하나로 분산 객체 컴퓨팅 표준 구조인 CORBA(Common Object Request Broker Architecture)를 제안하고 600여개의 업체가 지원하고 있으며 제품으로도 나왔다. CORBA는 객체지향 분산 컴퓨팅 환경의 실제적인 표준으로 자리잡아가고 있다. 그러나 분산 컴퓨팅 환경에서는 정보를 보다 효율적으로 이용하고 다른 통신망의 사용자들과 원활한 통신을 하기 위해서는 자원을 공유해야만 한다. 이러한 정보와 자원의 공유는 불완전한 통신 채널을 통해 이루어지기 때문에 악의를 가진 사용자로부터 고의적인 공격을 받을 수 있다. 특히, 인증 된 사용자로 가장(masquerade)한 사용자의 침입, 정보의 노출(information disclosure), 비인가 된 사용자로부터의 무결성 훼손(integrity violation) 등은 대표적인 위협으로 여겨지고 있다. 이러한 문제들로부터 CORBA의 안전성을 보장하기 위해서는 인증, 무결성, 비밀성 등의 보안 서비스들이 CORBA의 구조에 최소한의 영향을 주도록 설계되어야 한다. OMG에서는 분산 컴퓨팅 환경에서 사용자의 실수나 고의적인 침입에 대하여 안전성을 부여하기 위하여 보안 구조와 제공 될 보안 서비스들에 대한 작업을 진행하고 있다. 현재 OMG에서는 CORBA 의 보안과 관련하여 White Paper On

Security 와 Object Services RFP3(Request For Proposal)를 발표해 놓고 있다. White Paper On Security 에서는 분산 객체 컴퓨팅 환경에서 요구되는 보안 성질들을 일반적 요구와 기능적 요구로 나누어 기술하였고 CORBA의 구성 요소들이 보안 성질에 대한 요구를 만족하기 위하여 보안 서비스들을 이용하는 구조를 보였다. 그리고 Object Services Request For Proposal3(RFP3)에서는 CORBA에 제공 될 보안 서비스의 사양을 표준화하기 위한 제안을 나타내고 있다.

본 논문에서는 개방형 보안구조의 표준인 CORBA를 기반으로 하여 이에 적합한 보안 메커니즘을 소개하고자 한다. 먼저, 2장에서는 OMG에서 제안한 OMA 구조와 중심 역할을 하는 CORBA의 각 요소와 이에 대한 동작원리를 기술하고, 3장에서는 CORBA에 필요한 보안 요구사항과 보안 서비스를 살펴보고 보안서비스를 만족시키기 위한 보안 메커니즘 중 대표적인 인증 메커니즘인 Kerberos 버전5와 GSS-API를 살펴보겠다. 4장에서는 CORBA에서 제공되어지는 소프트웨어에 보안 기능의 추가로 인한 영향을 최소화하기 위한 개방형 보안메커니즘과 향후 방향을 제시하고자 한다.

2. CORBA 개요

2.1 OMA 개념 및 구조

1989년 4월 응용 프로그램들을 결합한 객체지향 표준을 제정하기 위해 OMG(Object Management Group)라는 비영리 단체가 나타나게 되었다. OMG는 마이크로소프트사를 포함한 수많은 컴퓨터 관련 단체들이 참여해 객체지향 기술을 기반으로 이종의 분산 된 환경 하에서 응용 프로그램들이 서로 통합할 수 있는 표준기술로 객체 관리 구조(Object Management Architecture : OMA)를 그림 2.1과 같이 4 가지로 구성하였다. 이 OMA는 크게 이종의 분산 환경에서 통신을 담당하는 CORBA와 객체를 조작하는 데 필요한 각종 기본 기능을 정의하고 있는 COSS(Common Object Service Specification), 그리고 추가로 제공되는 공통 기능 서비스가 있다.

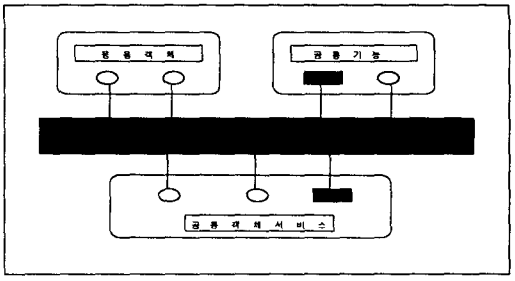

그림 2.1 객체 관리 구조(OMA)

OMA에서 가장 중요한 CORBA는 분산 된 객체들 사이의 투명한 요청과 응답을 제공하는 하나 이상의 객체 요청 중재자를 이용하여 분산 객체 컴퓨팅을 지원해 주는 메커니즘이다.

2.2 CORBA의 구조

OMG에서 제안한 CORBA는 실제 구현에 관한 기술보다는 사양을 정의하였고 이는 구체적으로 구현된 것이 아니라 분산 객체 컴퓨팅 구조의 사양만을 기술하고 있다. 현재 OMG로부터 정의 된 CORBA 2.0의 전체적인 조는 그림 2.2와 같다. 그림2.2에서 나타난 것처럼 CORBA는 객체 요청자

중개자 코어(ORB Core), 인터페이스 정의 언어(Interface Definition Language: IDL), 동적 호출 인터페이스(Dynamic Invocation Interface: DII), ORB 인터페이스(ORB Interface), 객체 어댑터(Object Adaptor), 인터페이스 저장소(Interface Repository), 구현 저장소(Implementation Repository) 등으로 이루어져 있다.

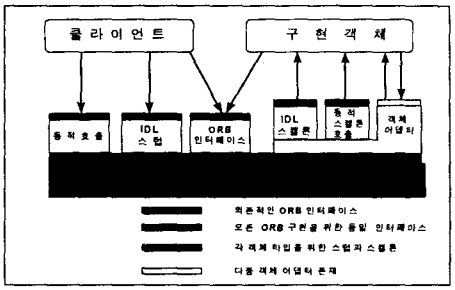

그림 2.2 CORBA 구조

3. CORBA 보안

3.1 개 요

운영체제나 데이터베이스 등의 보호로부터 시작 된 보안은 다중레벨 보안(Multilevel Security)이라는 개념으로 확장되었다. 이 MSL보안은 다른 레벨에서 작동되는 시스템간의 상호운영을 가능하게 하였고 같은 시스템에서도 다중레벨의 보안을 지원한다. 통신에서의 보안문제는 일반적인 보안의 개념보다 훨씬 복잡하다고 할 수 있다. 특히 개방형 환경에서 발생되는 문제는 일반적인 보안 문제들뿐만이 아니라 CORBA와 같은 개방형 분산 컴퓨팅 환경에서는 분산되어 운용되는 응용들에 대해서는 적절한 보호 방법이 필요하다. 이것은 앞으로 분산 컴퓨팅 환경에서 CORBA의 채택이 높아질 것을 감안할 때 매우 중대하다. CORBA는 이질적인 시스템들 사이의 분산 환경임으로 신뢰 있는 분산 시스템 구축 환경을 제공하기 위해서는 보안 서비스가 중요하다. 이러한 보안서비스는 전체 OMA의 구조를 만족시키며 다른 객체 서비스들 사이에서 보안의 특성인 비밀성, 무결성, 책임성, 유용성 등을 만족시킨다. 또한 이러한 보안의 기능으로는 인증(authentication) 기능, 접근제어(access control) 기능, 무결성(integrity) 기능 , 비밀성(confidentiality) 기능 등이 있다. 그리고 객체 응용들에게 보안 기능의 추가로 인한 영향을 최소화하여야 한다.

보안 서비스는 각 계층에 의해 제공되며 시스템이나 데이터 전송에 적당한 보안을 보장하기 위한 서비스이다. 보안 서비스의 종류는 다음과 같다.

① 인증 서비스(Authentication Service)
② 접근 제어 서비스(Access Control Service)
③ 정보의 비밀성 서비스(Information Confidentiality Service)
④ 정보의 무결성 서비스(Information Integrity Service)
⑤ 회계 감사 서비스(Auditing Service)

3.2 보안 서비스 요구사항

OMG의 White paper on Security에서 나타난 보안 서비스에는 인증(authentication) 서비스, 특권(Privilege) 서비스, 키 분배, 감사(auditing), 접근제어(access control) 서비스 등이 있

다. 보안 서비스에서 요구되는 사항은 일치성(Consistency),확장성(Scalability),가용성(Availability),강제성(Enforceability), 융통성(Flexiability),상호 운영성(Interoperability), 이용성 (Usability), 보안 기법의 독립성(Security Mechanism Independence) 이다.

3.3 CORBA 보안 구조

OMA 구조에 CORBA 보안 구조는 다른 객체 서비스들 사이에서 보안의 특성인 비밀성, 무결성, 책임성, 유용성 등을 만족시킨다. 이것은 사용자의 접근만을 제어하는 대신에 객체들의 접근도 제어하여야 한다. 그림 3.3은 OMG에서 제안한 CORBA에 적합한 보안 서비스를 제시한 보안 구조이다. 또한 서비스 기능에 대한 사양을 RFP에 발표 한 후 1995년 말에 잠정 규격을 만들었다. 보안 구조에서 객체 응용 보안 서비스로는 인증(authentication) 서비스, 접근 제어(access control) 서비스, 감사(auditing) 서비스가 있다. 선택사항 서비스로는 키 분배(key distribution)와 특권(privilege) 서비스 등이 있다.

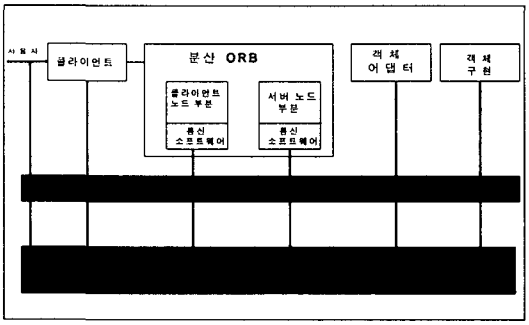

그림 3.3 OMG White Paper 보안 구조

3.4 보안 메커니즘

3.4.1 Kerberos 버전5

Kerberos는 네트워크 상에 비밀키 암호체계를 사용하는 인증 서비스를 제공한다. Kerberos는 MIT의 Needham과 Schroeder 모델과 타임 스탬프를 첨가한 Denning과 Saaco 모델을 기반으로 개발된 분산 된 인증 시스템이다. Kerberos 버전4가 현재까지의 환경에서 많이 사용되었으나 개방형 환경에서의 인증 메커니즘으로는 부족하여 최근에 이에 적합한 버전 5가 개발되었다.

다른 분산 시스템과 같이 Kerberos 인증 시스템에 관한 모델은 클라이언트에 의해서 사용되는 서비스들로 구성된다. 어떤 클라이언트가 서비스를 사용하기를 원하는 경우에 클라이언트는 티켓 부여서버로 부터 서비스를 위한 티켓을 받아야 한다. 이것은 초기 로그 인 단계에서 티켓부여서버에게 인증 받기 위한 티켓을 Kerberos 인증서버로부터 받는다. Kerberos 의 구조와 인증절차는 그림 3.4에서와 같다.

Kerberos 서버는 반드시 각 서버와 비밀키를 공유하여야 한다. 모든 서버는 Kerberos 서버에 등록이 되어 있어야 한다. 그러한 환경을 영역(realm)이라고 한다. Kerberos 는 상호 영역 인증 메커니즘을 제공한다. 상호 운영 영역에 있는 Kerberos 서버는 비밀키를 다른 영역에 있는 서버와 공유하고 두 Kerberos 서버는 서로 등록되어 있어야 한다.

다음은 Kerberos 버전4와 Kerberos 버전5의 차이점을 살펴보겠다. 먼저 Kerberos 버전4가 개방형 환경에 부족한 이유는 다음과 같다.

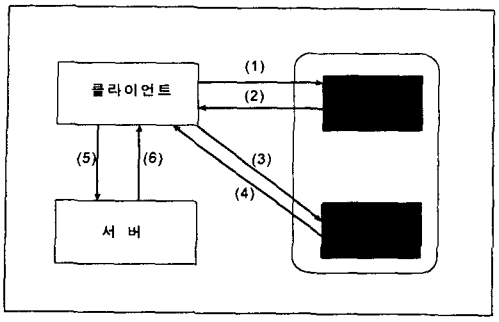

그림 3.4 Kerberos의 구조와 인증절차

① 버전 4는 DES 암호만이 사용되며 이는 암호 메커니즘에 대한 의존성이 높다.
② 버전 4는 인터네트 프로토콜(IP) 주소의 사용이 요구된다. 즉 인터네트 프로토콜에만 의존한다는 것이다.
③ 티켓에 대한 유효시간 값은 5분 단위로 8비트로 표현되어 있어서 21시간 정도로 제한되어 있다. 이것은 어떤 응용프로그램에서도 불충분하다.
④ 인증서버에서 클라이언트로 보내는 정보와 티켓 부여서버로 부터 클라이언트로 보내는 정보에 있는 티켓은 암호화가 중복되어진다.
⑤ 암호화는 PBC(Plain Block Chaining)의 DES 비표준모드를 사용한다. 이것은 암호문 블록의 교환에 관계 된 공격에는 취약하다는 단점을 가지고 있다.
⑥ 영역(realm)에 대한 관계가 각 영역마다 직접적으로 연결되어 있어서 n개의 영역이 존재한다 n^2만큼의 복잡한 관계가 형성되어져 있다.
⑦ 각 티켓은 그 티켓과 관련 된 서비스로 보내어지는 인증자를 암호화하기 위해 클라이언트가 사용하는 세션키를 가지고 있다. 이것은 세션 동안 전달되는 정보를 막기 위해 클라이언트와 서버에 의해 연속적으로 사용 될 수 있다. 그러나 같은 티켓이 특정한 서버로 부터 서비스를 받기 위해 연속적으로 사용될 수 있기 때문에 해커에 의해 변조되거나 재전송 할 위험성이 있다.
⑧ 인증서버에서 클라이언트로 가는 정보에는 클라이언트의 패스워드에 기반을 둔 비밀키로 암호화 된 내용이 있다. 해커는 이 정보를 가로채서 여러 가지 패스워드 복호를 시도 할 수 있다.

이러한 버전 4의 단점을 보안하기 위해 버전 5가 나왔는 데 인증 절차는 버전 4와 마찬가지로 6 단계의 과정을 거친다. 그러나 개방환경을 위해 필요한 요소가 몇 가지 추가되었다. 각 과정을 살펴보면 다음과 같다.

(1) 인증 서비스 교환 : 티켓 부여 허가 티켓을 얻는다.
(클라이언트-인증 서버)

① $C \rightarrow AS$: $Options \,||\, ID_C \,||\, Realm_C \,||\, ID_{tgs} \,||\, Times \,||\, Nonce_1$

② $AS \rightarrow C$: $Realm_C \,||\, ID_C \,||\, Ticket_{tgs} \,||\, E_{Kc}[K_{C,tgs} \,||\, Times \,||\, Nonce_1 \,||\, Realm_{tgs} \,||\, ID_{tgs}]$

* $Ticket_{tgs} = E_{Ktgs}[Flags \,||\, K_{C,tgs} \,||\, Realm_C \,||\, ID_C \,||\, AD_C \,||\, Times]$

① 단계에서는 서비스를 받고자 하는 클라이언트가 인증 서버에게 요청을 하게 된다. 자신의 ID와 자신이 속해있는 영역(realm), 서비스를 받고자 하는 티켓 인증 서버에 대해 명시해야 한다. 또 티켓이 유효한 시간을 지정할 수 있는 Time, 교환과정에서 오류가 발생하여 다시 보내야 할

때 구별해주기 위한 Nonce를 보내게 된다. Time과 Nonce는 Version 5에서 새로 개발 된 부분이다. ① 단계에서 요청을 받은 인증 서버는 클라이언트가 티켓 허가 서버와의 교환을 수행할 수 있는 티켓을 클라이언트에게 주게 된다.

② 단계에서 클라이언트가 전달받는 티켓은 티켓 허가 서버의 마스터키로 암호화되어 있기 때문에 클라이언트는 내부를 볼 수는 없다. 그리고 클라이언트가 티켓 허가 서버와의 교환에 쓸 세션키를 클라이언트의 마스터키로 암호화하여 전달된다. 또 티켓과 동시에 클라이언트의 영역, ID를 다시 전달하여 인증 서버가 받았던 내용이 정확한지를 확인할 수 있게 된다.

(2) 티켓 허가 서비스 교환 : 서비스 허가 티켓을 받는다. (클라이언트-티켓 허가 서비스)

③ $C \rightarrow TGS$: $Options \,||\, ID_V \,||\, Times \,||\, Nonce_2 \,||\, Ticket_{tgs} \,||\, Authenticator_C$

④ $TGS \rightarrow C$: $Realm_C \,||\, ID_C \,||\, Ticket_V \,||\, E_{Kc,tgs}[K_{C,V} \,||\, Times \,||\, Nonce_2 \,||\, Realm_V \,||\, ID_V]$

* $Ticket_{tgs} = E_{Ktgs}[Flags \,||\, K_{C,tgs} \,||\, Realm_C \,||\, ID_C \,||\, AD_C \,||\, Times]$
* $Ticket_V = E_{Kv}[Flags \,||\, K_{C,V} \,||\, Realm_C \,||\, ID_C \,||\, AD_C \,||\, Times]$
* $Authenticator_C = E_{Kc,tgs}[ID_C \,||\, Realm_C \,||\, TS_1]$

③ 단계에서는 인증 서버로부터 받은 티켓을 티켓 허가 서버에게 전달해준다. 또한 클라이언트가 서비스를 받을 서버의 ID, 시간, Nonce와 클라이언트를 인증할 수 있는 인증자를 전달한다.

④ 단계에서는 인증을 확인한 티켓 허가 서버가 클라이언트와 서버가 사용할 세션키를 서버에게 전달될 티켓에, 그리고 클라이언트와 티켓 허가 서버 사이의 세션키로 암호화 된 곳에 포함시켜 전달한다.

(3) 클라이언트/서버 인증 교환 : 서비스를 받는다. (클라이언트-서버)

⑤ $C \rightarrow V$: $Options \,||\, Ticket_V \,||\, Authenticator_C$

⑥ $V \rightarrow C$: $E_{Kc,v}[TS_2 \,||\, Subkey \,||\, Seq\#]$

* $Ticket_V = E_{Kv}[Flags \,||\, K_{C,V} \,||\, Realm_C \,||\, ID_C \,||\, AD_C \,||\, Times]$
* $Authenticator_C = E_{Kc,v}[ID_C \,||\, Realm_C \,||\, TS_2 \,||\, Subkey \,||\, Seq\#]$

⑤ 단계에서는 클라이언트가 티켓 허가 서버로부터 받은 티켓을 서버에게 전달하게 된다. 또한 클라이언트를 인증 할 수 있는 인증자도 전달된다. 마지막으로 ⑥단계에서는 서버와 클라이언트 사이의 세션키로 암호화된 메시지를 보내어 클라이언트가 보낸 메시지를 잘 받았다는 확인을 알려주게 된다. 이 단계에서 보조키(Subkey)는 클라이언트가 암호화 키를 선택하기 위한 키이며 Seq#는 메시지에 대한 일련의 번호로 전달과정에서 오류가 생겼을 때 다시 보내야 하는 부분을 구별하기 위한 것이다.

티켓에 포함되어있는 플래그는 전달과정에 있어서의 필요한 정보를 나타내는 역할을 한다. 이 플래그에 대한 정보를 클라이언트가 인증서버나 티켓 허가 서버에게 알려주는 역할을 하는 것은 선택사항이다. 버전5 에서는 이전의 버전4와는 달리 클라이언트가 서비스를 받을 서버들에 대한 영역의 정보를 가지고 있을 필요가 없기 때문에 인증 서버나 티켓 허가 서버가 클라이언트에게 주는 메시지에는 그 다음 단계에서 클라이언트가 접해야 하는 서버의 영역이 포함된다.

Kerberos에서 클라이언트의 마스터키는 패스워드가 담당하게 된다. 클라이언트가 패스워드를 자주 사용하게 되면 패스워드에 대한 침범이 가능하게 되며 또 침입자의 입장에서 패스워드를 알아내게 되면 침입을 막을 수 없다는 단점이 있다. Kerberos 버전5에서는 패스워드의 변경을 가능하게 하여 패스워드를 자주 사용하는 것에 대한 문제점을 해결하였다.

3.4.2 GSS-API(Generic Security Service Application Programming Interface)

IETF(The Internet Engineering Task Force)에 의해 제안 된 인터넷 표준으로 응용 프로그램에 안전성 서비스(security service) 에 관련 일반적인 인터페이스를 정의하였다. GSS-API 의 사양은 IEFT의 RFC 1508(Request For Comments 1508)과 RFC 1509 에 기술되어 있다.

RFC1507 의 Kerberos는 메커니즘을 사용자가 매우 자세히 알 필요가 있어야 한다. 그러나 GSS-API는 보안 메커니즘의 영향을 받지 않는 일반적인 인터페이스를 제공한다. 또한 응용 프로그램이 사용하는 프로토콜에 대해서도 독립적이며 연결지향 통신에서 적합하다.

GSS-API는 분산환경하에서 보안 서비스에 대한 일반적인 인터페이스를 제공하는 것으로 크게 인터페이스와 메커니즘의 집합으로 나누어 생각할 수 있다.

인터페이스는 주체들에게 응용 코드와 연결되어 있는 라이브러리를 통해 GSS-API 호출에의 접근을 가능하게 하는 것으로 주요 역할은 데이터의 변환과 메커니즘과의 교류에 관련된 호출 등이다. GSS-API 구조는 그림 3.5와 같다. GSS-API는 라이브러리 형태를 가지고 있다. 따라서 사용자는 라이브러리 함수를 호출시 GSS-API 하부구조의 복잡한 내용을 알지 않아도 된다.

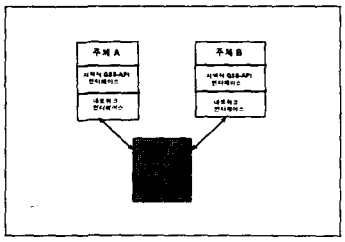

그림3.5 GSS-API 구조

그림 3.5에서와 같이 GSS-API는 하나 또는 복수개의 메커니즘들에 대하여 최상위 계층에 위치하며 응용 프로그램들이 보안 메커니즘으로부터 서비스를 받을 수 있는 인터페이스들을 제공한다. 메커니즘은 Kerberos와 같은 보안 서비스를 제공하는 것으로 각 세션이 시작할 때 알맞은 메커니즘이 선택되어진다. 인터페이스가 각 사용자 시스템에 존재하는 것에 반해 메커니즘은 신뢰할 수 있는 서버나 보안이 되는 호스트에 위치하고 있어서 인터페이스와 메커니즘사이에 네트워크가 필요하게 된다. 그러난 이러한 네트워크는 사용자의 응용 프로그램이 알 필요없는 정보의 교류가 일어나게 된다. GSS-API의 중요한 목적은 각 시스템에서 서로 다른 보안 서비스를 사용하는 것에 대비하여 메커니즘에 독립성을 부여하는 것과 통신 프로토콜에 독립성을 부여하는 것이다. GSS-API의 보안메커니즘을 이용하기 위한 함수는 크게 4가지 종류로 구분할 수 있다.

(1) 신뢰 관리 호출 (Credential Management Calls)

주체가 신뢰를 얻거나 해제시키는 일을 하며 또한 많은 신뢰에 대한 정보를 조사하는 일을 담당한다.

① gss-acquire-cred() : 보안 문맥의 초기화, 허가에 대한 신뢰를 얻는다.
② gss-release-cred() : 문맥의 수행이 끝난후 신뢰구간을 해제시킨다.
③ gss-inquire-cred() : 신뢰에 대한 정보를 얻는다.

(2) 문맥 관리 호출 (Context Management Calls)

문맥의 초기화, 허가, 제거 등을 담당한다.
① gss-init-sec-context() : 외부에 대한보안 문맥을 초기화한다.
② gss-accept-sec-context() : 내부로 향하는 보안 문맥을 받아들인다.
③ gss-delete-sec-context() : 문맥이 끝났을 때 문맥적 정보를 없앤다.
④ gss-process-context-token() : 각 사용자로부터의 제어 토큰을 수행한다.
⑤ gss-context-token() : 문맥상에서의 남은 시간에 대한 유효성을 결정한다.

(3) 메시지 관련 호출(Per-Message Calls)
무결성과 신뢰성 등의 기능에 대한 억세스를 제공한다.
① gss-sign() : 메시지에 서명을 적용시킨다.
② gss-verify() : 서명된 메시지를 확인한다.
③ gss-seal() : 메시지를 캡슐화, 서명, 암호화(선택적) 시킨다.
④ gss-unseal() : 메시지를 인증하고 요구에 따라 복호화 시킨다.

(4) 지원 호출 (Support Calls)
일반적으로 필요한 작업을 제공하며 할당 된 메모리나 이름의 해제 등의 작업을 지원한다.
① gss-display-status() : 상태 코드를 인쇄가능한 형태로 바꾼다.
② gss-compare-name() : 두 이름의 동일성을 비교한다.
③ gss-indicate-mechs() : 구현에서 사용할수 있는 메커니즘의 형태를 나타낸다.
④ gss-display-name() : 이름을 인쇄 가능한 형태로 바꾼다.
⑤ gss-import-name() : 사용자가 읽을 수 있는 이름을 내부적 표현으로 변화시킨다.
⑥ gss-release-buffer() : 할당되었던 버퍼 저장소를 해제시킨다.
⑦ gss-release-name() : 이름 저장소를 해제시킨다.
⑧ gss-release-oid-set() : OID 저장소를 해제시킨다.

GSS-API는 다중 레벨로 이루어져 있다고 할 수 있다. 가장 상위 레벨에 응용 프로그램 입장에서 직접적으로 접촉이 가능하다고 볼 수 있는 API가 존재하여 하위레벨의 보안에 관련 된 여러 가지 일(예: 인증, 키생성, 관리, 데이터의 신뢰성과 무결성 보장 등)을 하는 보안 서비스들과의 인터페이스를 수행하게 된다. GSS-API의 인터페이스는 정보에 대한 무결성, 비밀성, 키, 신용장 그리고 세션 관리 인터페이스를 제공함으로서 안전한 세션을 요구하는 클라이언트/서버 기반의 응용 프로그램을 제공한다. 즉, 보안 서비스는 GSS-API가 일반적인 인터페이스를 제공하는 기본 시스템에 의해서 제공되고 이러한 기본 시스템은 메커니즘으로 기술된다. 이와 유사한 개념을 통신 메커니즘인 TCP/IP 소켓으로 볼 수 있는 데 이는 유닉스에서 하나의 통신 메커니즘으로 실질적인 서비스 영역을 제공하는 것과 같이 보안 메커니즘 또한 실질적인 서비스 영역을 제공한다.

3.4.3 GSS-API를 이용한 Kerberos 버전5 인증 메커니즘

분산환경에 있는 사용자와 객체들은 위에서 기술한 GSS-API 인터페이스를 통해 Kerberos 버전5의 인증 과정을 거쳐 시스템과 분산환경에 있는 객체들에 접근을 할 수 있는 것이다. 이러한 인증과정을 설명하기 위하여 GSS-API를 적용한 예를 살펴보겠다. 먼저, 사용자는 인증을 받기 위해 GSS-API 즉, gss-init-sec-context를 호출하여 아래의 형식에 따라 인증 메커니즘 중 kerberos 버전5를 선택하여 인증을 받은 후 시스템의 자원을 이용할 수 있다.

```
InitialContextToken ::=
[APPLICATION 0]  IMPLICIT SEQUENCE {
        thisMech      MechType
```

```
innerContextToken ANY DEFINED BY thisMech
}
```

여기서, thisMech는 kerberos 버전5를 기술하여 주고, innerContextToken은 Kerberos V5 에 대한 메시지를 나타낸다. 메시지 토큰의 형식은 KRB_AP_REQ, KRB_AP_REP,KRB_ERROR로 구분할 수 있다.
KRB-REQ에서는 Kerberos V5라는 것과 메시지 타입을 나타내며 option과 티켓,인증자가 포함되어 있다. 티켓은 요청자의 영역, 이름, 데이터가 포함되어 있으며 암호화 된 티켓에는 플래그, 키, 요청자의 영역, 이름이 포함되어 있어서 인증자의 내용과 비교하여 인증의 과정이 수행된다. 또 티켓에는 티켓의 허용시간이 표시되어 있으며 시작시간, 재생시간, 호스트주소 등이 선택적으로 포함되어 있다. 인증자에는 인증의 과정에서 필요한 내용과 동시에 보조키, 행렬 번호, 인증 데이터 checksum등이 선택적으로 포함되어 있다.
AP_REP는 AP_REQ의 구성과 거의 동일하며 보조키, 행렬번호가 선택적으로 포함되어 있다.
KRB-ERROR에서는 시간, 영역, 이름등이 포함되어 있으며 에러의 종류를 나타내고 잘못된 부분의 수정이 표현되어 있다.

2) 메시지 관련과 문맥 해제 토큰

메시지 관련 토큰을 생성하고 수행하기 위해 문맥 키, 신임장 키,MD2.5 seed 키 등이 사용된4 다. 메시지 관련 토큰(MIC)은 GSS_GetMIC() 함수를 호출하여 받은 데이터의 무결성을 확인한다. 그리고 문맥 해제 토큰은 GSS_Delete_sec_context()에 의해 발생하며 ID, 무결성에 사용되는 알고리즘의 종류, fillller, 행렬번호, checksum등이 포함되어 있다.

3) 이름 유형과 객체 식별자

이름 유형은 GSS Import_name() 호출의 입력값, 관련된 ID값으로 사용된다. 호환성을 지원하는데에 있어서의 인터페이스 요소를 정의한다. 이름의 형식은 Kerberos 주체 이름 형식과 서비스 기반 호스트 이름 형식으로 구분이 된다.

4. 결 론

현재 개방형 환경에서 정보 보호의 기능에 대해 일반적인 보안 구조 및 보안 서비스로는 충분하지 못하다. 특히 개방형 환경에서 분산되어 처리되고 있는 객체들에 대해서는 적절한 방법이 존재하지 않고 있다. 따라서 본 논문에서는 개방형 환경에 적합한 구조인 OMG 에서 표준화 한 CORBA를 기반으로 하여 객체 응용에 대한 보안 구조 , 요구 사양 및 보안 서비스를 Object Services RFP3(Request For Proposal), White Paper On Security 에서 분석하였다
OMG white paper는 네 가지 다양한 보안 기능들에 대해 필요한 인터페이스들의 레벨에서 다루어진다. 첫째, 응용레벨에서 보안 파라미터들은 IDL에 보안 알고리즘, 서비스 등을 선택하기 위해 추가로 정의되어 질 수 있다. 둘째, ORB는 다양한 보안 메커니즘을 사용하여 보안 환경을 만들고 관리할 수 있다. 셋째, 보안서비스 인터페이스는 소프트웨어가 사용할 수 있는 보안 메커니즘과 서비스에 대해 명확해야하며 동시에 보안 환경에 의해 숨겨져 있어야 한다. 넷째, 보안 서비스 제공자 인터페이스는 다른 메커니즘들과 서비스를 선택하는 데 사용되어지며 인터페이스의 외부에 노출되어 있는 부분을 지원하는 데 사용되므로 표준화되어있는 것이 적당하다.
따라서, CORBA에 적용시키기 위한 보안 서비스는 사용자 뿐 아니라 객체들에 대해서도 적용되어야 하며 보안 기능의 추가로 인한 보안 구조와 각 소프트웨어들에 대해 최소의 영향을 줄 수 있도록 해야 한다.

본 논문에서는 인증메커니즘으로 Kerberos 버전5를 사용하고 이 메커니즘을 분산환경에서 사용할 수 있도록 다른 인증메커니즘을 이용할 수 있는 시스템간의 상호운영을 위해 GSS-API를 고려

하여 보았다. 이 GSS-API는 UNIX의 소켓 메커니즘처럼 하나의 인터페이스 시스템 호출자로서 사용자가 메커니즘에 독립적인 특징을 가진다는 것을 알 수 있었다.

앞으로의 연구방향은 CORBA와 같은 분산개방환경의 보안 요구사항 중 인증 메커니즘으로 GSS-API를 이용한 Kerberos 버전5를 위한 인터페이스를 구현함으로서 다른 보안 요구사항 즉, 무결성, 접근제어, 회계감사 등을 만족시킬 수 있는 보안구조를 설계하고자 한다.

참고 문헌

[1] The Common Object Request Broker : Architecture and Specification, OMG Document Number 1995.July

[2] CORBA Security, OMG Document Number 95.12.1

[3] Object Management Group, Framinghsm, " White Paper on Security:,Document Number 1994.4.16

[4] Belinda Fairthorne, " OMG White Paper on Security", OMG Security Working Group, April, 1994

[5] Warwick Ford, "Computer Communication Security", PTR Prentice Hall,1994

[6] Charlie Kaufman, Radia Perlman, Mike Speciner, " Network Security",PTR Prentice Hall, 1995

[7] Thomas J.Mowbray, Ron Zahavi, " The Essential CORBA- Systems Integration Using Distributed OBJECT", John Wiely & Sons, Inc, 1995

[8] Robert Orfali, Dan Harkey, Jeri Edwards, " The Essential Distributed Objects Survival Guide ", John Wiely & Sons, Inc, 1996

[9] RFC 1507

Distributed Authentication Security Service

[10] RFC 1508

Generic Security Service Application Programming Interface

[11] RFC 1509

Generic Security Service Application Programming Interface: C bindings

[12] RFC 1510

The Kerberos Network Authentication Service(V5)

[13] RFC 1511

Common Authentication Technology Overview

[14] D.P. Barton, L.J.O Connor, Implementing Generic Security Services in a Distributed Environment