情報保護學會論文誌
第14券 第3號, 2004. 6

보안운영체제 환경에서의 신뢰채널 지원을 위한 모듈의 설계 및 구현

유 준 석[a)†‡], 임 재 덕[a)], 나 재 훈[a)], 손 승 원[a)]
한국전자통신연구원[a)]

The Design and Implementation of Module supporting Trusted Channel in Secure Operating System Environment

Joon-suk Yu[a)†‡], Jae-deok Lim[a)], Jae-hoon Nah[a)], Sung-won Sohn[a)]
Electronics and Telecommunications Research Institute[a)]

요 약

보안운영체제는 운영체제에 내재된 보안상의 결함으로 인하여 발생할 수 있는 각종 해킹으로부터 시스템을 보호하기 위해서 기존 운영체제에 다양한 보안기능을 추가한 운영체제이다. 보안운영체제는 시스템 보안을 목적으로 고안되었으나 전체 시스템의 안전을 위해서는 네트워크 측면에서의 보안이 필수적이다. 이를 위해서 IPsec이나 SSL과 같은 네트워크 보안 프로토콜들이 개발되어 사용되고 있으나 정책이나 키 관리에 많은 주의를 필요로 하고 보안운영체제의 특성을 반영하기 어렵다. 본 논문에서는 보안운영체제 사이에서 안전한 통신을 제공하기 위한 간단한 신뢰채널 메커니즘을 소개한다. 본 신뢰채널은 네트워크 트래픽에 대해 기밀성과 인증 서비스를 제공하며, 보안운영체제에 사용되는 특정 보안정보를 전달할 수 있는 구조를 가진다. IP 계층의 커널 수준에서 구현된 신뢰채널은 단순한 처리구조를 통하여 신뢰채널 처리과정에서 발생할 수 있는 오버헤드를 줄일 수 있다.

ABSTRACT

Secure operating system is a special operating system that integrates some security functions(i.e. access control, user authentication, audit-trail and etc.) with normal operating system in order to protect system from various attacks. But it doesn't consider any security of network traffic. To guarantee the security of the whole system, network traffic must be protected by a certain way and IPsec is a representative technology for network security. However, it requires administrator's carefulness in managing security policies and the key management mechanism is very heavy as well as complicated. Moreover, it doesn't have a suitable framework for delivery of security information for access control mechanism. So we propose a simple trusted channel mechanism for secure communication between secure operating systems. It provides confidentiality and authentication for network traffic and ability to deliver security information. It is implemented at the kernellevel of IP layer and the simplicity of the mechanism can minimize the overhead of trusted channel processing.

Keywords: *Secure OS, trusted channel, IPsec, access control, MAC, RBAC*

Ⅰ. 서 론

현재의 유닉스나 리눅스와 같은 다중 사용자 환경

접수일: 2003년 8월 29일; 채택일: 2004년 2월 26일
† 주저자, ‡ 교신저자 : jsyu92@etri.re.kr

에는 여러 가지 보안 위협 요소와 취약점들이 존재한다. 그 중에서 시스템 관리자, 즉 루트 사용자에게 모든 권한이 집중되어 있으므로 발생할 수 있는 문제나 트로이 목마와 같은 것들은 대부분 운영체제 자체에 내재된 보안상 결함에 기인하고 있으며, 이러한 문제들은 다양한 접근제어 정책을 적용함으로써 효과적으로 대처할 수 있다.

보안운영체제는 기존 운영체제에 몇 가지 보안 기능을 추가한 운영체제로서 일반적으로 다양한 접근제어 메커니즘 및 부가적인 사용자 인증, 그리고 감사추적 등의 기능을 가진다. 따라서 앞에서 언급한 운영체제 자체의 결함으로 발생할 수 있는 보안상의 문제들에 대해서 보안운영체제를 사용한다면 적절하게 대응할 수 있다. 하지만 보안운영체제는 스푸핑이나 스니핑과 같은 네트워크 차원의 다양한 공격에 대해서는 전혀 고려하지 않고 있다.

오늘날 대부분의 시스템들은 인터넷과 같은 개방형 네트워크에 연결되어 있고 따라서 통신 내용이 외부에 불법적으로 노출되거나 변조될 위험을 항상 지니고 있다. 또한 네트워크를 통해 처리되는 정보의 가치가 나날이 커지고 있는 현실을 감안할 때 보안운영체제를 통한 시스템 보안만으로는 다양한 환경에서 이루어지는 공격들로부터 전체 시스템의 안전을 보장할 수 없으며, 시스템 보안과 더불어 네트워크 측면에서의 보안 수단도 함께 강구되어야 할 것이다.

현재 네트워크 보안을 위한 수단으로 IPsec이나 SSL 등의 다양한 프로토콜들이 개발되어 사용되고 있으며, IP 계층에서 통신 트래픽에 대한 인증 및 기밀성 등의 보안 서비스를 제공하는 IPsec은 대표적인 네트워크 보안 프로토콜로써 널리 사용되고 있다. 하지만 IPsec을 사용하는 데에 있어서 보안정책에 세심한 주의가 필요하고 키 관리를 위한 오버헤드가 크다는 등의 관리상 어려움이 존재한다.[1,2] 또한 보안운영체제로 구성된 네트워크 환경에서 파일 등과 같은 객체의 보안특성을 일관되게 유지하려면 객체 전송 시에 보안정보를 상대 시스템으로 함께 전달하여 관리되도록 하여야 할 것이다. 하지만 IPsec이나 SSL은 이러한 사항들이 고려되어 설계되지 않았으므로 구조적으로 보안정보의 전송을 지원하기 어렵다.

따라서 본 논문에서는 통신 트래픽에 대해 기밀성 및 인증 서비스 제공뿐만 아니라 보안운영체제의 특정 보안정보 전달 기능을 제공할 수 있는 신뢰채널 구현에 대해서 소개한다. 구현한 신뢰채널에서는 필요한 기능들을 제공하기 위해서 새로운 헤더를 정의하여 사용하며, 헤더 및 처리 구조의 단순화를 통해 신뢰채널 처리 과정에서 발생할 수 있는 오버헤드를 최소화하였다.

구현한 신뢰채널은 공개 운영체제인 FreeBSD 4.3 커널에 다양한 접근제어 메커니즘 및 보안기능을 통합시킨 보안운영체제에 기반하며, 상세한 내용은 본 논문의 나머지 부분에서 기술하도록 한다.[3]

II. 관련 연구

최근 들어 컴퓨터 시스템은 다양한 방법에 의해 해킹되어 그 피해가 늘고 있으며, 이에 따라 보안 솔루션에 대한 사용자들의 요구 또한 늘고 있다. 방화벽이나 침입탐지 시스템 등이 이러한 침입에 대한 대안으로 개발되었지만 침입탐지 시스템은 새로운 방법의 공격으로부터 시스템을 보호할 수 없고 방화벽 또한 해커가 이미 시스템 내에 침입해 있다면 소용이 없게 된다. 따라서 다양한 공격으로부터 시스템 자원을 보호하기 위해 커널 수준에서 접근을 제어하는 방법이 사람들의 관심을 끌게 되었고 다양한 보안 커널, 즉 보안운영체제들이 개발되었다.

앞에서 언급한 바와 같이 보안운영체제는 운영체제 자체에 내재된 보안상의 허점으로 인하여 발생할 수 있는 각종 해킹으로부터 시스템을 보호하기 위해서 기존 운영체제 내에 다양한 보안기능을 추가한 운영체제이다. 일반적인 보안운영체제의 기능으로는 접근제어, 사용자에 대한 인증, 감사추적 등이 있으며, 이 중에서 접근제어가 가장 핵심적인 기능이라고 할 수 있다.

접근제어는 그 정책에 따라서 임의적 접근제어(DAC: Discretionary Access Control), 강제적 접근제어(MAC: Mandatory Access Control), 그리고 역할기반 접근제어(RBAC: Role-Based Access Control) 등으로 나눌 수 있으며, 이들 정책은 많은 곳에서 구현되고 표준화 되어왔다.[4-7]

DAC은 파일이나 디렉토리 등과 같은 객체에 대한 접근을 객체가 속해 있는 사용자 혹은 그룹의 신분에 기반하여 제한하는 방식으로 신분기반 접근제어라고도 한다. 이는 가장 일반적이고 대표적인 접근제어 정책으로 모든 유닉스류의 운영체제에서 사용되고 있다.

또한 DAC의 확장된 모델인 ACL(Access Contorl List)은 self, group, other 외에 임의

의 사용자나 그룹에 퍼미션을 추가적으로 할당할 수 있게 한다. DAC이나 ACL 모두 객체에 대한 접근 허용 여부를 전적으로 주체의 신분에 기반하여 판단하기 때문에 보안상 취약점이 있는 것으로 지적되고 있다.

한편 정부나 군사 기관에서는 조직의 특성을 반영하고 이에 따라 기밀문서를 처리할 수 있는 보안 시스템을 필요로 하였고 [8]에서는 이에 대한 대안으로 MLS(Multi-Level Security)라고 불리는 강제적 접근제어 모델을 제안 하였다.이는 한 컴퓨터 시스템 내에서 다중 수준의 정보 관리를 위한 정책을 설명하고 있다.

MAC에서는 한 시스템 내의 모든 객체와 주체에 보안 라벨, 즉 보안 속성이 할당되며, 객체에 대한 주체의 접근은 사용자의 신분과는 관계없이 각 주체와 객체가 가지고 있는 보안 속성에 의해 결정된다. 주체가 가지는 보안 속성은 주체와 연결된 신뢰의 정도를 나타내고 객체가 가지는 보안 속성은 해당 객체에 접근하기 위해서 주체가 가져야만 하는 신뢰의 정도를 나타내는데 이는 다시 보안등급(class)과 보안범주(category)로 나눌 수 있다. 보안등급은 top secret, secret, confidential, unclassified 등과 같이 비밀성의 중요 정도를 나타내며, 보안범주는 총무부, 인사부 등과 같이 그룹의 의미를 가진다.

주체가 객체에 접근할 때, 주체와 객체의 보안 속성을 서로 비교하고 시스템은 특정 규칙, 즉 simple-security property와 *-property에 따라 접근여부를 결정한다. 이 규칙에 의하여 정보는 낮은 수준에서 높은 수준으로 흐를 수는 있으나 높은 수준에서 낮은 수준으로 흐를 수는 없으며, 이를 통해 트로이 목마 공격에 대한 보호수단을 제공할 수 있다.

MAC이 정부나 군사 기관의 특성을 반영하는 데에 적합한 정책이지만 기업과 같은 상업적인 조직의 사용에는 부적합한 특성을 지니며, 이러한 이유로 RBAC이 소개되었다.

RBAC은 객체에 대한 사용자의 접근을 신분이 아니라 개인의 직무에 따라서 결정하는 방법이다. 이 방법에서는 사용자, 역할, 그리고 퍼미션의 3가지 기본요소를 가지고 접근제어를 수행하며, 해당 파일에 접근할 수 있는 역할을 가진 사용자만이 객체에 접근하여 읽기, 쓰기, 실행 등의 허가된 작업만을 할 수 있도록 한다. RBAC을 사용하면 시스템 관리, 웹 관리, 보안 관리 등과 같이 특정 역할을 두고 해당 역할에 속한 사용자만이 특정 관리기능을 수행하게 함으로써 루트 사용자의 권한을 분리시킬 수 있을 뿐 아니라 시스템 관리의 편의성도 도모할 수 있다.

본 논문에서 설명하는 신뢰채널은 한국전자통신연구원에서 개발한 보안운영체제(이하 SOS)에 기반하여 구축되었다. SOS는 FreeBSD 4.3 커널에 다양한 보안기능을 통합시킨 보안운영체제로써 접근제어 정책으로 MAC과 RBAC, 그리고 신분기반 접근제어의 일종인 ACL을 포함하며, 이 외에도 스마트카드를 이용한 사용자 인증 및 암호화 파일시스템, 감사추적 기능 등이 구현되어 있다.[3]

SOS에서는 RBAC을 통하여 시스템관리자 외에 보안관련 업무를 담당할 보안관리자라는 사용자를 별도로 두고 있는데 이는 기존 운영체제에서 시스템관리자인 루트 사용자에게 집중되어 있는 권한을 분리시킴으로써 시스템관리자 권한을 정당하게 혹은 부당하게 획득한 사용자에 의해 발생할 수 있는 자료유출 등의 문제를 방지할 수 있다. 보안관리자 역할은 시스템 설치 시에 미리 특정 사용자가 가질 수 있도록 설정되며, 해당 사용자는 로그인 시에 보안관리자 역할을 가지고 인증을 수행함으로써 보안관리자 역할을 획득할 수 있다.

SOS에서 보안관리자는 일반 사용자들에게 MAC에서 사용하는 보안등급과 보안범주, 그리고 RBAC을 위한 역할을 할당하게 되며, 각 사용자들은 로그인 시에 보안관리자로부터 이미 할당받은 보안등급 및 보안범주, 그리고 역할의 범위 내에서 인증을 수행할 수 있다. 시스템은 사용자가 로그인 시에 입력한 이들 정보에 기반하여 사용자가 특정 객체에 접근할 권한이 있는지를 판단하게 된다. 이와 같이 접근제어가 적용될 경우에는 다양한 접근제어 정책과 접근규칙으로 인해 불법적인 사용자나 비인가자가 시스템 자원에 접근하는 것을 막을 수 있다.

하지만 SOS와 같이 접근제어를 통하여 시스템 보안을 제공하는 보안운영체제로는 네트워크 측면의 보안은 제공할 수 없으므로 이를 위한 별도의 메커니즘이 필요하다. 본 논문에서는 두 단말 사이에서 전송되는 데이터에 대한 기밀성, 인증 등의 기능을 제공함으로써 스니핑 및 스푸핑과 같이 네트워크 상에서 이루어지는 공격으로부터 트래픽을 보호하는 것을 신뢰채널이라 정의하여 사용한다. 현재 신뢰채널을 제공할 수 있는 메커니즘으로는 IPsec이 대표적이며, 널리 사용되고 있다. IPsec은 AH와 ESP 보안 프로토콜을 통해 네트워크 계층에서 다양한 보안 서비스를 제공할 수 있는데 이를 위해서는 각 트래픽

마다 보안 서비스 제공에 사용될 암호키, 인증키, 사용 알고리즘 등을 명시한 보안연계(Security Association)가 정의되어 있어야 한다. 보안연계는 사용자에 의해 수동으로 설정되거나 IKE(Internet Key Exchange)를 통하여 자동으로 설정될 수 있는데 IKE의 경우에 하나의 보안연계 생성을 위해 9번의 메시지 교환이 필요하여 오버헤드가 크다는 문제가 있다. 수동 설정의 경우에도 입/출력 패킷에 대한 보안연계를 각각 설정해야 하고 각 보안연계 마다 사용되는 키 및 알고리즘과, 기타 파라미터들을 정의해야 하므로 관리가 상당히 부담스러울 수밖에 없다. 본 논문에서 기술하는 신뢰채널은 보안 서비스 제공을 위해 필요한 설정 과정 및 파라미터들을 최대한 단순화시킴으로써 관리상의 부담을 줄이고 있다.

Ⅲ. 설계 및 구현

3.1 기능 개요

본 눈문에서 설명하는 신뢰채널은 SOS 시스템들간 통신이 이루어질 때 사용자에 상관없이 자동으로 제공되며, 상위 프로토콜이나 애플리케이션과는 무관하게 IP 계층을 통해 전달되는 트래픽에 대해서 모두 적용된다. 신뢰채널을 통해 제공되는 주요 기능은 다음과 같다.

- 기밀성 서비스
- 메시지 인증 서비스
- 보안정보(MAC class, MAC category)의 전달

본 기능들을 수행하기 위해서 IP 계층에서는 출력 패킷의 암호화를 수행하고 암호화된 패킷에 대한 인증정보를 추가한다. 이 과정에서 전송되는 데이터의 MAC 관련 보안정보도 새로 정의한 헤더에 실려서 함께 전달된다. 반대로 수신된 패킷은 IP의 상위 프로토콜로 전달되기 전에 인증정보에 대한 검증과 복호화가 수행된다.

송신측에서 패킷에 신뢰채널을 적용해야 하는지에 대한 판단은 패킷의 목적지 주소를 기반으로 이루어진다. 즉, 목적지의 주소가 각 SOS 시스템이 가지고 있는 신뢰시스템 목록 내에 포함되어 있을 경우에만 신뢰채널이 적용된다. 신뢰채널을 적용할 경우에는 IP 헤더의 다음 프로토콜 필드에는 신뢰채널 적용을 나타내는 플래그가 설정되고 수신측에서는 이 값을 통하여 수신된 패킷에 신뢰채널이 적용되었는지를 판단하여 처리하게 된다.

신뢰채널의 적용이 결정되어 사용될 때 신뢰채널의 인증과 기밀성 보장에 사용되는 키 및 알고리즘은 사전에 설정되어 있다고 가정하며, 시스템에 대한 자세한 내용은 계속되는 절에서 설명하도록 한다.

3.2 구조 및 일반사항

구현하는 신뢰채널은 IP 계층에 통합되어 신뢰채널 적용이 결정된 패킷에 대한 암호화 및 인증작업을 처리한다. 이를 위해서 IP 계층에서 패킷 처리를 담당하는 루틴, 즉, ip_input()과 ip_output()에 신뢰채널 처리를 위한 코드가 추가되었다. 신뢰채널 처리를 통하여 IP 패킷에 별도의 신뢰채널 헤더가 추가되지만 IP 계층에서는 이를 IP 페이로드로 인식하게 되므로 IP 계층의 기본적인 패킷 처리 절차에는 영향을 미치지 않는다. 그림 1은 시스템의 전체적인 처리 구조를 도식화하여 간략하게 보여주고 있다.

패킷의 기밀성 보장을 위해서는 64비트 블록암호 알고리즘인 blowfish를 CBC(Cipher Block Chaining) 모드로 사용하며, 패킷의 인증을 위해서는 임의 길이의 입력을 받아 128비트의 고정 길이 출력을 내는 HMAC-MD5를 사용한다. 기밀성 보장에 사용되는 blowfish는 현재 널리 사용되고 있는 DES나 IDEA보다 속도면에서 우수한 특성을 지니는 것으로 알려져 있으며, 가변 길이의 키를 사용할 수 있다.[9,10] 하지만 구현의 용이성을 위해 현재 키 길이는 128비트로 고정하여 사용하며, 사용되는 알고리즘은 모듈 형태로 지원되므로 향후 필요에 따라서 수월하게 확장 및 변경이 가능하다.

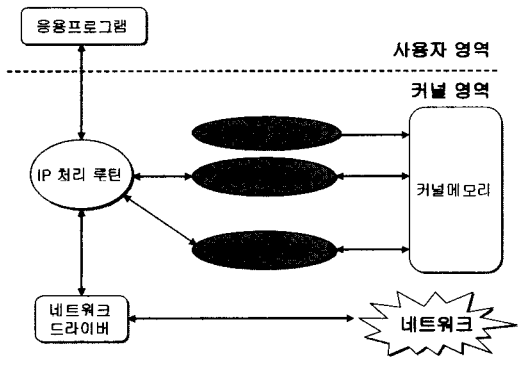


그림 1. 신뢰채널 시스템의 처리 구조

신뢰채널을 사용하기 위해서는 몇 가지 초기 설정이 이루어져야 하며, 초기설정과 관련된 정보들은 다음과 같은 세 종류의 파일을 통하여 유지된다.

- 암호키 파일: 패킷 암/복호에 사용될 128 비트의 암호키를 저장
- 인증키 파일: 패킷 인증에 사용될 128 비트의 인증키를 저장
- 호스트 파일: 신뢰채널 서비스가 적용되어야 할 호스트들의 IP 주소를 저장. 즉, SOS가 설치된 신뢰시스템의 주소 목록

본 파일들은 SOS가 설치된 시스템에서 공통적으로 사용되는데 시스템 설치 시에 생성된다. 또한 설정파일들은 보안관리자만이 접근할 수 있는 위치에 저장되어 안전하게 보관되며, 부팅 시에 커널 메모리로 적재되어 사용된다.

3.3 신뢰채널 헤더 및 패킷 입/출력

3.3.1 신뢰채널 헤더

신뢰채널이 적용되는 패킷에는 신뢰채널 헤더가 기존 IP 패킷에 추가된다. 그림 2는 신뢰채널 헤더가 추가된 전체 패킷의 형태와 제공되는 보안 서비스의 영역을 보여주고 있으며, 그림에서 신뢰채널 헤더는 음영으로 나타나 있다.

- Authentication data: 인증 제공영역에 대한 해쉬값
- IV(Initial Vector): 기밀성 제공영역의 암/복호에 사용되는 초기벡터
- next-hdr.: IP 헤더의 next-protocol 필드의 값을 가짐
- HLEN: 신뢰채널 헤더의 길이
- PLEN: 패딩 길이
- MAC class: 전송되는 객체의 보안등급
- MAC category: 전송되는 객체의 보안범주

인증 데이터는 패킷의 인증 제공영역에 대한 해쉬값으로써 송신측에서 계산되어 패킷에 추가되며, 수신측에서는 이 정보를 통하여 패킷을 인증한다. 즉, 수신측에서 인증 제공영역에 대하여 해쉬값을 계산한 후 송신측에서 계산하여 패킷에 추가한 인증 데이터와의 동일성 여부를 검사함으로써 패킷의 위/변조 여부를 확인한다. Next-hdr.는 IP 헤더의 next-protocol 필드값을 가지며, 이는 수신측에서 신뢰채널 헤더를 제거하고 원래의 IP 패킷을 만들어 낼 때 next-protocol 필드 값을 복원하기 위해 사용된다. HLEN과 PLEN 필드는 수신측에서 수신한 패킷에서 신뢰채널 헤더와 패딩을 제거하여 원래의 패킷으로 복원할 때 필요한 정보이다. Blowfish나 DES와 같은 블록암호 알고리즘은 일정 크기의 블록단위로 암/복호를 수행하므로 패킷을 암호화하기 위해서는 패킷이 블록의 정수배 크기가 되어야하며, 패딩은 이러한 목적으로 암호화 전에 패킷에 덧붙여지고 복호 후에는 제거된다. MAC class와 MAC category는 전송되는 객체의 보안특성을 나타내는 값으로써 보안운영체제 시스템들이 서로 연결된 컴퓨팅 환경에서 객체 전송 시 이를 함께 전송하여 처리함으로써 객체의 보안특성에 대한 일관성을 유지시킬 수 있다. 즉, 시스템 A에서 top secret이었던 파일은 다른 시스템 B로 전송된 후에도 top secret 특성을 유지하여 불법적인 정보의 흐름을 방지할 수 있다. 만약 이러한 보안 정보가 함께 전송되지 않는다면 시스템 A에서 top secret이었던 파일은 시스템 B로 전송된 후에는 보안특성을 상실하게 되며, 결과적으로 낮은 등급 사용자들에 의한 파일 접근이 가능하게 된다.

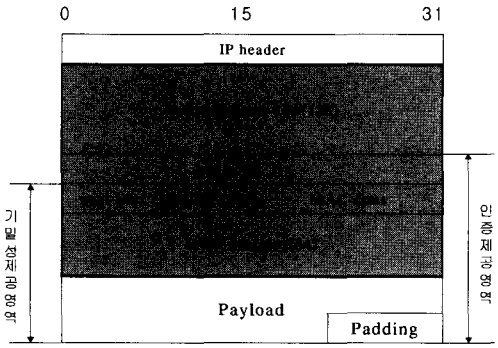


그림 2. 신뢰채널 헤더 구조 및 서비스 제공 영역

3.3.2 패킷 출력

패킷의 출력과 입력과정에서 신뢰채널 서비스를 처리하기 위해 이루어지는 동작은 각각 상이하다.

우선 출력과정에서 사용자 프로세스로부터 데이터 송신 요청이 발생하게 되면 해당 요청은 상위 프로토콜 처리를 수행하는 루틴을 거쳐 IP 계층에서 출력을 담당하는 ip_output()까지 전달된다. 이 루틴에

서는 일정한 IP 처리 절차를 수행하고 하위 계층으로 패킷을 넘기기 전에 패킷에 대한 검사합 계산 및 단편화를 수행한다.[11,12,13] 신뢰채널 적용여부의 판단과 판단 결과에 따른 신뢰채널 적용은 패킷의 검사합 계산 및 단편화 작업 바로 전에 수행된다. 이 때 신뢰채널 적용여부의 판단은 패킷의 목적지 주소를 기반으로 이루어진다. 즉, IP 패킷 헤더의 목적지 주소가 호스트 파일로부터 커널 메모리에 적재된 주소 목록에 있을 경우에만 신뢰채널을 적용하고 그렇지 않을 경우에는 기존의 IP 출력 처리를 수행한다.

신뢰채널을 적용하는 단계에서는 우선 신뢰채널 헤더 추가를 위한 공간을 확보하고 암호화를 위한 패딩정보를 덧붙인 후 지정된 알고리즘을 통하여 패킷을 암호화한다. 이 과정에서 IP 헤더의 next-protocol 필드는 해당 패킷에 신뢰채널이 적용되었음을 수신측에서 판단할 수 있도록 특정 값으로 설정되고 원래 값은 신뢰채널 헤더의 next-hdr. 필드에 보관된다. 이는 그림 3에서 도식화하여 설명하고 있다.

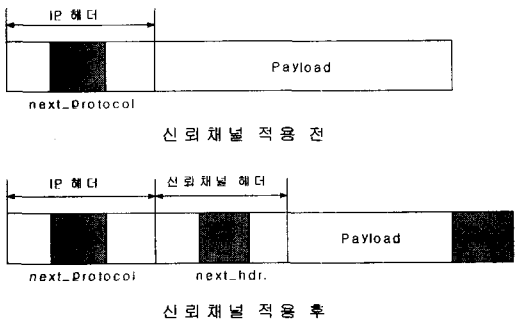


그림 3. 상위 프로토콜 처리를 위한 헤더 정보 설정

신뢰채널 적용단계에서 패킷을 암호화한 후에는 인증정보를 계산하여 신뢰채널 헤더의 해당 필드를 설정하는데 계산되는 인증정보는 암호화된 패킷에 대한 해쉬값이다. 또한 신뢰채널 적용 단계에서는 신뢰채널 헤더를 추가함으로써 기존 패킷에 영향을 미치는 패킷 길이와 같은 정보를 재계산하여 IP 패킷 헤더의 해당 필드를 재설정함으로써 기존 IP 처리 루틴의 변경 없이 신뢰채널을 처리할 수 있도록 하고 있다. 이상의 신뢰채널 처리가 완료되면 제어는 기존 IP 출력 루틴으로 넘어가고 IP 출력 루틴은 필요시 패킷의 단편화를 수행한 후 하위 계층으로 패킷을 전달한다. 그림 4는 ip_output() 내에서 신뢰채널 처리를 수행하는 함수들이다.

```
int
ip_outPut(m0, opt, ro, flags, imo)
{
        . . .
#ifdef SECUROS_TC
        sostc_ok = sostc_tdb_decision(m, 1);
        if(sostc_ok)
        {
                . . .
                sostc_err = sostc_tPb_outPut(&m_state, &sostc_mac);
                . . .
        }
#endif /* SECUROS_TC */
        . . .
}
```

그림 4. ip_output( ) 루틴에서의 신뢰채널 처리 관련 함수

3.3.3 패킷 입력

목적지에서 패킷이 수신되면 하위 프로토콜의 입력 루틴을 거쳐 IP 계층에서 입력을 담당하는 ip_input()까지 전달된다. 패킷을 전달받은 IP 입력 루틴은 패킷의 재조합 및 기타 처리를 수행하고 패킷을 상위 계층으로 넘기기 직전에 해당 패킷에 신뢰채널이 적용되었는지를 판단하게 된다.[11, 12, 13] 이는 IP 헤더의 next-protocol 필드를 확인함으로써 이루어진다. 만약 신뢰채널이 적용된 패킷이라고 판단되면 패킷 해당 패킷에 대한 인증 검사가 수행되며, 이를 통과한 패킷에 대해서만 패킷을 복호화 한다. 복호화된 패킷은 신뢰채널 헤더와 패딩이 제거되고 상위 프로토콜로 전달된다. 이 때 IP 헤더의 next-protocol 필드는 신뢰채널 헤더의 next_hdr. 필드의 값으로 대체되며, 패딩이 제거된 전체 패킷의 길이정보가 재 계산되어 IP 패킷 헤더에서 패킷 길이를 나타내는 필드에 설정된다. 만약 신뢰채널이 적용되지 않은 패킷이라고 판단되면 패킷은 기존 IP 처리와 동일하게 그대로 상위 계층으로 전달된다.

```
void
ip_inPut(struct mbuf *m)
{
        . . .
#ifdef SECUROS_TC
        sostc_ok = sostc_tdb_decision(m, 0);
        if(sostc_ok)
        {
                . . .
                sostc_tPb_inPut(m, hlen);
                . . .
        }
#endif /* SECUROS_TC */
        . . .
        (*inetsw[ip_Protox[ip_>ip_P]].Pr_inPut)(m, off, nh);
}
```

그림 5. ip_input( ) 루틴에서의 신뢰채널 처리 관련 함수

그림 5는 ip_input() 내에서 신뢰채널 처리를 수행하는 함수들을 보여주며, 그림 6은 패킷의 입출력 과정에서의 신뢰채널 처리 절차를 도식화하여 보여주고 있다.

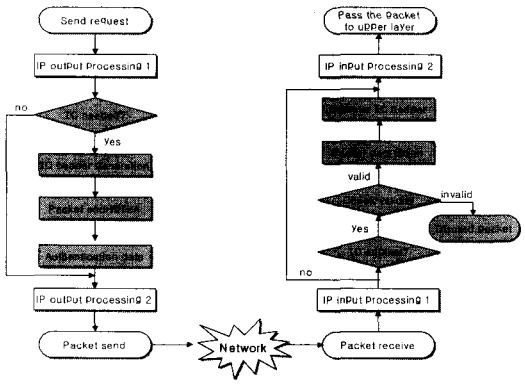


그림 6. IP 계층에서의 신뢰채널 처리 절차

Ⅳ. 고려사항

4.1 안전성

앞에서 언급한 바와 같이 신뢰채널의 사용을 위해서 세 가지 설정파일이 이용되는데 이 파일들의 안전은 전체 신뢰채널 서비스의 안전성 보장을 위한 핵심적인 부분이다. 결과적으로 전체 시스템의 안전성을 보장하기 위해서는 설정파일을 불법적인 접근으로부터 보호할 필요가 있다.

구현한 신뢰채널에서는 설정파일들을 불법적인 접근이나 노출로부터 보호하기 위하여 보안관리자만이 접근할 수 있는 특정 디렉토리에 보관하며, 이는 SOS에서의 RBAC 기능을 이용함으로써 가능하다. 즉, 해당 디렉토리는 시스템 설치 시에 보안관리자 역할을 가진 사용자에 대해서만 모든 권한을 허용하도록 설정되며, 따라서 정당한 보안관리자만이 신뢰채널의 설정파일들에 접근하여 필요한 작업을 수행할 수 있다. 그림 7은 RBAC에 의해서 설정파일이 보관된 특정 디렉토리(/sos)가 보호되고 있는 것을 보여준다.

그림 7의 (가)는 보안관리자에 대해서만 rwx 권한이 주어진 /sos 디렉토리에 해당 역할을 가지고 있지 않은 사용자는 루트 사용자라고 하더라도 접근이 허가되지 않음을 보여주고 있다. 반면에 (나)에서는 보안관리자 역할을 가진 사용자는 동일한 디렉토

```
root@sostest5# id
uid=0(root) gid=0(wheel) groups=0(wheel), 2(kmem), 3(sys), 4(tty),
5(operator), 20(staff), 31(guest)
root@sostest5# getprole
root@sostest5# getfrole /sos
#security_manager : -rwx
#login_manager : -r-x
root@sostest5# cd /sos
/sos: Operation not permitted.
root@sostest5# ls /sos
ls: sos: Operation not permitted
root@sostest5#
```

(가) 불법적인 접근

```
manager@sostest5# id
uid=999(manager) gid=0(wheel) groups=0(wheel)
manager@sostest5# getprole
 #security_manager
manager@sostest5# getfrole /sos
#security_manager : -rwx
#login_manager : -r-x
manager@sostest5# cd /sos
manager@sostest5# ls /sos
access_acl.dat          sos_key_id
default_acl.dat         sostc.auth
mac.dat                 sostc.enckey
rbac.dat                sostc.hosts
role.dat                user.dat
sos_key
manager@sostest5#
```

(나) 정당한 접근

그림 7. RBAC에 의한 접근제어

리에 접근하여 원하는 작업을 수행할 수 있음을 보이고 있다. RBAC으로 접근이 제어되는 디렉토리에는 신뢰채널을 위한 설정파일 외에도 시스템 운용과 관련하여 보호되어야 할 파일들이 함께 보관된다.

신뢰채널 설정파일들은 시스템 부팅 시에 자동적으로 커널 메모리로 로딩되도록 구현되었다. 파일의 내용을 커널 메모리로 적재시켜서 사용하는 것은 시스템 운용 중에 빈번히 발생하는 디스크 접근을 줄여 시스템의 효율성을 높여줄 뿐만 아니라 타 메모리 영역보다 높은 보안성을 제공할 수 있다는 장점을 지닌다.

구현한 신뢰채널을 적용하면 패킷에는 인증정보가 추가되고 암호화되어 전송된다. 이와 같이 제공되는 인증 및 기밀성 서비스의 안전성은 기반하는 해쉬 및 암호 알고리즘의 안전성에 기반하므로 해당 알고리즘이 안전하다면 제공되는 인증 및 기밀성 서비스 또한 안전하다 할 수 있다. 그림 8은 신뢰채널이 적용되어 전송되는 패킷을 캡쳐하여 보여주고 있다. 그림 8에서 보는 바와 같이 129.254.242.65와 129.254.62.22의 주소를 가지는 호스트에는 SOS가 설치되어 있고 따라서 두 호스트 사이에 전송되는 패킷들이 암호화되는 것을 확인할 수 있다.

```
root@sostest5# ./packet  p

        129.254.242.65 -->      129.254.62.22.  NEXT Prcto:55, SOSTC
        129.254.242.65 -->      129.254.62.22.  NEXT Proto:55, SOSTC
        129.254.62.22 -->       129.254.242.65.         NEXT Proto:5
5, SOSTC
        129.254.242.65 -->      129.254.62.22.  NEXT Proto:55, SOSTC
```

그림 8. 신뢰채널에 의한 패킷 보호

4.2 성 능

신뢰채널의 구현에서 트래픽에 대한 기밀성을 제공하기 위해서는 암호화가 필수적이며, 이에 따른 어느 정도의 성능저하는 필연적으로 발생하게 된다. 그림 9는 신뢰채널이 적용된 경우와 그렇지 않은 경우에 대해서 FTP를 이용한 데이터 전송시간의 차이를 비교한 결과이다. 단, 여기서 언급하는 전송시간은 전송경로 상에서의 지연 뿐 아니라 신뢰채널을 적용하는 데에 부가적으로 소요되는 암/복호화 등의 처리시간을 모두 포함한 것이다. 본 실험에는 서로 상이한 서브넷에 위치하는 세대의 호스트가 사용되었다. 두 대의 호스트에는 SOS를 설치하여 데이터 전송시에 신뢰채널이 적용되게 하였고, 나머지 한 대에는 FreeBSD를 설치하여 신뢰채널이 적용되지 않도록 하였다. 각 호스트로는 펜티엄 IV 1.4GHz 프로세서와 256MB 메모리가 장착된 PC를 사용하였으며, 실험 데이터로는 50MB와100MB 크기의 파일을 사용되었다. 각각의 데이터에 대해서는 다섯 번씩의 전송을 통한 평균시간을 측정하였다.

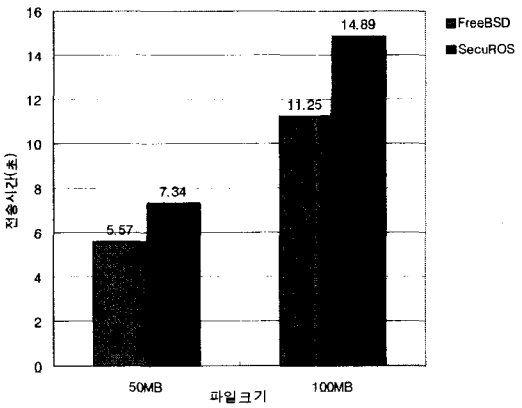


그림 9. 신뢰채널 적용 시 파일 전송성능 비교

위 그림에서 보는 바와 같이 신뢰채널이 적용된 경우는 그렇지 않은 경우에 비하여 약 75% 정도의 성능을 보이고 있다. 일반적인 환경에서 특수한 경우를 제외한다면 사용자들이 네트워크를 통해 전송하는 데이터의 대부분이 실험 데이터 보다 작은 것이 현실이다. 이는 기존 네트워크 성능 대비 70%~80% 정도의 성능이라면 특별한 경우를 제외하고는 사용자들이 크게 느끼지 못 할 수준이며, 큰 불편 없이 보안 서비스를 받을 수 있을 것으로 예상된다. 만약 암호 연산을 위한 전용 하드웨어의 사용이 적용된다면 좀 더 좋은 성능을 기대할 수 있을 것이다.

V. 결 론

다양한 접근제어 메커니즘을 기존 운영체제의 커널 수준에서 통합시킨 보안운영체제는 기존 운영체제가 근본적으로 지니고 있는 시스템 자체의 취약성에 대해서 일정 수준의 대안을 제시하고 있으나 스니핑이나 스푸핑과 같은 네트워크 상에서 이루어지는 공격들에 대해서는 무방비로 노출되어 있는 상태이다.

IPsec이나 SSL 등의 네트워크 보안 프로토콜들이 다수 존재하지만 이들을 사용하기 위해서는 보안정책이나 키의 관리를 위해 많은 노력이 소모될 뿐만 아니라 보안운영체제의 보안특성을 지원하기 어려운 구조를 가지고 있다. 본 논문에서는 이러한 점들을 해결하고자 개발된 간단한 구조의 신뢰채널에 대해서 설명하였다.

개발한 신뢰채널은 SOS 사이의 통신에 대한 인증과 기밀성 서비스를 제공하며, 한 시스템 내에서 데이터 객체의 보안정보를 상대 시스템으로 전달할 수 있는 기능을 제공한다. 또한 본 신뢰채널의 사용을 위해서 사용자가 별도로 개입하지 않으며, 데이터의 전송이 사용자에게 투명하다. 하지만 SOS가 설치된 모든 시스템에서 동일한 설정파일을 미리 수동적으로 설정해야하고 설정파일의 내용이 바뀌었을 경우에 이를 시스템에 적용하기 위해서 시스템의 재부팅이 필요하다는 점은 관리적인 측면에서 오버헤드로 작용할 수 있다. 또한 현재 사용하고 있는 암호 알고리즘에 고정 길이의 키를 사용하는 것은 해당 알고리즘의 특징을 제대로 반영하지 못하고 있다. 지금까지 설명한 신뢰채널이 프로토타입의 성격을 가지고 있다는 점을 감안한다면 향후 개발에 있어서는 이러한 관리적인 측면 및 세부적인 개선이 수반되어야 할 것이다.

참 고 문 헌

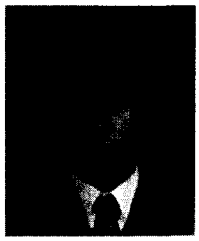
[1] S. Kent and R. Atkinson, Security Architecture for the Internet Protocol, RFC 2401, 1998.

[2] C. Kaufman and et al., Code-preserving simplications and improvements to IKE, draft-kaufman-ipsec-

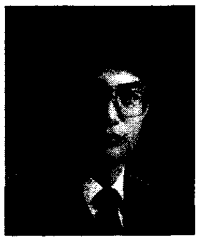
improveike-00.txt, 2001.
[3] J. G. Ko, J. N. Kim, & K. I. Jeong, Access Control for Secure FreeBSD Operating System, Proceedings of WISA2001, The Second International Workshop on Information Security Applications, 2001.
[4] Mark Funkenhauser, B1 TUNIS: A Kernel for a Secure UNIX System, Canadian Computer Security Conference, 1989.
[5] Roos Lindgreen, Herschberg I. S. On the Validity of the Bell-Lapadula Model, Computer & Security, Vol. 13, 1994.
[6] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based Access Control models, IEEE Computer, 29(2), 1996.
[7] D. Ferraiolo, J. Cugini, and D. R. Kuhn. Role-based Access Control: Features and motivations, In Annual Computer Security Applications Con ference. IEEE Computer Society Press, 1995.
[8] Bell, David Elliott, and Leonard J. La Padula. Secure computer system: Unified exposition and multics interpretation, MITRE Technical Report 2997. MITRE Corp, 1975.
[9] B. Schineier, Description of a New Variable-Length Key, 64-Bit Block Cipher(Blowfish), Cambridge Security Workshop Proceedings, Fast Software Encryption, Springer-Verlag, 1994.
[10] B. Schineier, Applied Cryptography, John Wiley & Sons, 1996.
[11] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, The Design and Implementation of the 4.4BSD Operating System, Addison-Wesley Publishing Company, 1996.
[12] FreeBSD 4.3-RELEASE Source Code.
[13] Behrouz A. Forouzan, TCP/IP Protocol Suite, McGrawHill, 2002.

〈著者紹介〉

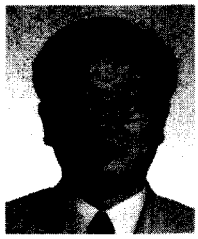
유 준 석 (Joon-suk Yu) 정회원
1999년 2월 : 성균관대학교 정보공학과 졸업
2001년 2월 : 성균관대학교 전기전자 및 컴퓨터공학부 석사
2001년 1월~현재 : 한국전자통신연구원 연구원
〈관심분야〉 정보보호, 암호이론, 무선인터넷 보안



임 재 덕 (Jae-deok Lim)
1999년 2월 : 경북대학교 전자공학과 졸업
2001년 2월 : 경북대학교 전자공학과 석사
2000년 12월~현재 : 한국전자통신연구원 연구원
〈관심분야〉 시스템 보안, 네트워크 보안, 정보보호, 운영체제



나 재 훈 (Jae-hoon Nah)
1985년 2월 : 중앙대학교 컴퓨터공학과 졸업
1987년 2월 : 중앙대학교 컴퓨터공학과 석사
1987년~현재 : 한국전자통신연구원
〈관심분야〉 네트워크보안, IPsec, 무선인터넷 보안



손 승 원 (Sung-won Sohn)
1984년 2월 : 경북대학교 전자공학과 졸업
1994년 2월 : 연세대학교 전자공학과 석사
1999년 2월 : 충북대학교 컴퓨터공학과 박사
현재 : 한국전자통신연구원 정보보호연구단 단장
〈관심분야〉 정보보호, 네트워크 보안