신뢰경로가 보장되는 보안커널 설계 및 구현

이해균*, 김재명**, 조인준*

*배재대학교 컴퓨터공학과, **시큐브 (주)

The design and implementation of security kernel assured trusted path

Hae-gyun Lee*, Jae-myoung**, In-june Jo*

*Department of Computer Engineering Paichai Univ., **Secuve Ltd.

요 약

MAC(Mandatory Access Control)이나 MLS(Multi Level Security) 보안정책이 적용된 보안운영체제는 주체와 객체에 보안등급(Security Level)과 영역(Category) 값을 부여하고, 부여된 이들 정보에 의해 객체에 행해지는 주체의 접근을 제한한다. 하지만, 이러한 MAC과 MLS 보안이 적용된 경우라 하더라도 시스템의 보안정책을 위반하며 불법적 정보를 유통하게 하는 경로가 있을 수 있다.

본 논문에서는 불법적 정보의 유통경로가 되고 있는 IPC(Inter Process Communication) 메커니즘과 스토리지에 의한 비밀채널을(Covert Channel) 문제 해결 위해 커널 수준의 설계와 구현을 시도하였다. IPC 메커니즘에 의한 불법적 정보흐름 재거를 위해 IPC 메커니즘에 MLS 보안정책을 적용하였고, 스토리지 비밀채널는 시스템 콜 명세를 분석하여 이를 식별, 감사, 지연처리가 가능토록 하였다.

ABSTRACT

Security operating system applied to MAC(Mandatory Access Control) or to MLS(Multi Level Security) gives both subject and object both Security Level and value of Category, and it restrict access to object from subject. But it violates Security policy of system and could be a circulated course of illegal information. This is correctly IPC(Interprocess Communication)mechanism and Covert Channel.

In this thesis, I tried to design and implementation as OS kernel in order not only to give confidence of information circulation in the Security system, but also to defend from Covert Channel by Storage and IPC mechanism used as a circulated course of illegal information. For removing a illegal information flow by IPC mechanism. I applied IPC mechanism to MLS Security policy, and I made Storage Covert Channel analyze system call Spec. and than distinguish Storage Covert Channel. By appling auditing and delaying, I dealt with making low bandwidth.

I. 서론

접근통제 기술에는 DAC(Discretionary Access Control)과 MAC(Mandatory Access Control) 기술이 있다. DAC은 일반적으로 UNIX시스템에서 사용하는 접근통제 기술이다. 객체에 대한 접근권한이 사용자 그룹에 의해 결정되기 때문에 사용에 편의성은 있으나 해킹이나 트로이 목마와 같은 공격에 취약하다. MAC은 객체와 주체에게 변경될 수 없는 보안속성을 가진다. 따라서 정보에 대한 접근여부는 주체와 객체간의 보안속성이 보안정책에 부합되는지를 비교하여 이루어진다. 일반적으

로 사용되는 MAC 기술은 MLS(Multi Level Security)이다. 주체가 생성한 객체는 주체의 보안등급과 동일한 보안등급이 부여되고 이러한 객체의 보안속성은 보안관리자만이 변경할 수 있다.

일반적으로 MAC 기술이 적용된 운영체제는 파일에 의한 불법적 정보유통을 방지할 수 있으나 IPC 메커니즘이나 비밀채널에 의한 불법적 정보유통 방지에는 미비하다. 본 논문은 MAC이 적용된 운영체제에서 불법적 정보 유통 경로(IPC 메커니즘, 스토리지 비밀채널) 문제 해결을 위한 연구이다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 필요한 MLS, 비밀채널등에 대해 조사 분석 하였고, 3장은 신뢰성 있는 경로가 보장되는 보안커널을 설계하였다. 4장에서는 이를 구현하였다. 5장에서는 본 연구의 의미와 향후 진행되어야 할 연구방향에 대해 기술하였다.

II. 보안커널 설계 및 구현을 위한 기술 분석

1. MLS 분석

1) MLS 개요

MLS(Multi Level Security)에 대한 보안 연구는 미국방성에 의해서 1960년대 후반 최초로 시작되었다[1]. 국방성에서 사용하는 문서에는 보안등급이 있었으며, 문서를 읽기 위해서는 문서의 보안등급과 같거나 높은 보안등급이 필요했다. 컴퓨터의 발전으로 종이 형태로 보관되던 정보는 컴퓨터로 옮겨지게 되었으며, 종이 문서의 보안등급이 컴퓨터에 저장된 정보에도 적용되어야 했다. MLS 보안정책은 컴퓨터에서의 정보와 사용자간의 보안정책을 명시하고 있다.

2) 군사 보안정책(Military Security Policy)

MLS에서 사용하는 보안정책은 군사 보안정책에서 가져왔다. 보안정책은 간단히 문서에 대한 보안등급과 문서에 접근하고자 하는 사람의 보안등급을 비교해 주는 것이다. 다음은 정보와 정보를 얻고자 하는 사람에 대해 주어지는 보안속성의 예이다.

· 보안등급 - UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP SECRET

· 영역 - NATO, UNCLEAR,

보안등급은 문서에 주어지는 보안상의 등급을 의미하며, 그 구조는 계층적이다. 영역은 정보를 취급 할 수 있는 부류(Compartment)를 나타내고 영역내의 요소들은 각각 대등하다. 보안등급이 SECRET이고, 영역이 NATO, NUCLEAR, CRYPTO인 경우 보안속성은 다음과 같이 표기된다.

· {SECRET;NATO,NUCLEAR,CRYPTO}

3) 보안등급간 수학적 관계

보안등급은 다음과 같은 수학적 관계가 정의된다.

· UNCLASSIFIED < CONFIDENTIAL < SECRET < TOP SECRET

올바른 객체와 주체간의 보안등급과 영역간의 수학적 관계는 다음과 같이 정의된다.

· $S_L \geq O_L$ (S_L 주체의 보안등급, O_L 객체의 보안등급)

· $S_C \supseteq O_C$ (S_C 주체의 영역, O_C 객체의 영역)

주체의 보안등급이 객체의 보안등급과 같거나 클 경우에 접근이 허가되며, 주체의 영역은 객체의 영역을 포함해야 접근이 허가된다.

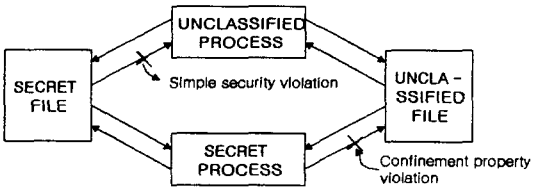
4) MLS 보안정책

MLS의 기본 보안정책만을 본다면 매우 간단하다. 그러나 기본 정책만으로는 문제가 있을 수 있다. 예를 들어 보안등급이 UNCLASSIFIED인 프로세스와 파일이 있고 보안등급이 SECRET인 프로세스와 파일이 있다고 할 때 기본 보안정책만을 본다면 SECRET인 프로세스는 UNCLASSIFIED인 파일에 쓰기를 할 수 있다. 이것은 SECRET인 프로세스에서 UNCLASSIFIED 파일로의 정보 흐름을 의미한다. 그리고 기본 보안정책에서는 UNCLASSIFIED 프로세서에서 SECRET파일로의 쓰기 권한을 제한하고 있는데, 이것은 정보의 편의적인 이동을 생각하면 UNCLASSIFIED 프로세서에서 SECRET파일로의 쓰기 권한을 제한 할 이유가 없다. 그래서 MLS에서는 기본 보안정책 외의 제한 속성을 정의하고 있다. MLS에서의 보안정책을 정리하면 다음과 같다.

· 단순 보안 : 주체는 보안등급이 같거나 낮은 객체에 읽기를 할 수 있으나 높은 보안등급의 객체에는 읽기를 할 수 없다.

· 제한 속성 : 주체는 보안등급이 같거나 큰 객체에 쓰기를 할 수 있으나 낮은 보안등급의 객체에는 쓰기를 할 수 없다.

단순 보안정책과 제한 속성을 도식화하면 [그림 1]과 같다. 프로세스에서 파일로의 화살표는 쓰기를 의미하며, 파일에서 프로세서로의 화살표는 읽기를 의미한다[1].



[그림 1] MLS 보안정책

2. 비밀채널

비밀채널은 시스템의 정상적인 정보흐름 경로를 이탈한 채널을 의미한다. 이러한 비밀채널은 모니터링 시스템을 갖춘 네트워크, MAC이 적용된 시스템에서 의도적으로 정보 유출을 위해 발생할 수 있다.

1) 비밀채널 개요

TCSEC에 정의한 비밀채널의 정의는 "프로세스가 시스템의 보안정책을 위반하여 정보를 전송할 수 있는 통신채널이다"이다

비밀채널의 정의나 발생 환경을 고려해 본다면 몇 가지 비밀채널의 속성을 알 수 있다[2].

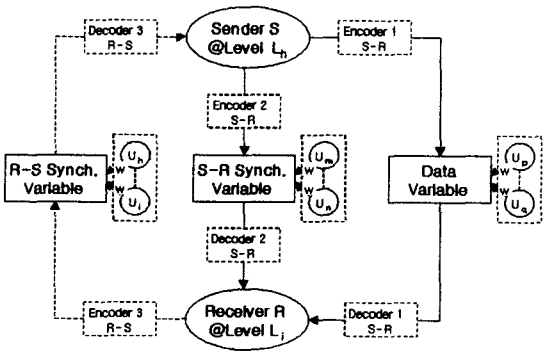
· 임의적 정책 모델의 부적절정

· 비임의적 보안정책 모델에 대한 종속성

· 비밀 모델 및 무결성 모델 모두에 대한 관련성

· TCB(Trusted Computing Base) 명세에 대한 종속성

비밀채널은 스토리지 채널과 타이밍 채널로 분류를 한다. 이들간에는 이론적인 구별이 이루어지지 않지만 비밀채널 시나리오가 작성될 경우 두 분류로 나뉘게 된다. 송신자와 수신자가 [그림 2]와 같이 동기화 되어 위치할 경우 비밀채널이 발생한다. 비밀채널을 설명하면 [그림 2]와 같다[2]. 스토리지 채널과 타이밍 채널은 동기화에 사용되는 매체가 무엇이냐에 의해 나뉘게 된다. 스토리지를 동기화 매체로 사용하면 스토리지 채널이라 하고, 시스템 시간을 동기화 매체로 사용하면 타이밍 채널이라 한다.



[그림 2] 송신자와 수신자간의 비밀채널 표현(L_h>L_i또는L_h><L_i)

보안등급이 상이한 프로세서 S와 R은 데이터 전송을 동기화하기 위한 변수와 전송할 데이터를 나타내는 변수를 가지게 되고, 전송되는 데이터는 임의적인 인코딩과 디코딩 형식을 가지게 된다.

III. 신뢰경로가 보장되는 보안커널 설계

1. 신뢰경로를 보장하는 IPC 설계

1) 개요

UNIX 운영체제에서는 프로세스간 통신 메커니즘으로 IPC 메커니즘을 사용한다. IPC 메커니즘은 message queue, semaphore, sheared memory 로 구성된다. 각 메커니즘은 그들이 사용할 자료구조를 정의하고 있으며, 각 자료구조는 메커니즘에서 사용할 접근권한을 담당하는 공통된 자료구조를 가지고 있다. 접근정책은 일반 UNIX 시스템에서 사용하는 것과 같다[3, 4, 5].

IPC 생성 시스템 콜이 호출되면 IPC에서 사용되는 자료구조의 내용이 초기화되고, 접근권한에 대한 자료구조도 생성한 사용자의 UID와 GID로 초기화된다. 프로세스간 통신을 원하는 프로세스는 생성되어 있는 IPC의 기술자(Descriptor)를 얻어 통신을 하게 된다. IPC에 읽기, 쓰기 위한 시스템 콜에서는 시스템 콜을 호출한 프로세스가 접근권한이 있는지를 검사해 주게 된다.

기본적인 접근제한이 있는 IPC 메커니즘에 MLS를 적용하기 위한 기본적 설계는 다음과 같

이 세 부분으로 구성된다.

첫째, 각 IPC 메커니즘의 구조체에서 사용하는 공통된 퍼미션 구조체(ipc_perm 구조체)에 IPC의 보안등급(clearance)과 영역(categories) 정보를 넣을 수 있는 변수를 추가한다.

둘째, IPC 생성 시스템 콜이 호출되고 IPC 구조체가 초기화 될 때, 상기에서 추가되었던 퍼미션 구조체의 보안등급과 영역 변수를 현 프로세스의 보안등급과 영역으로 초기화한다.

셋째, IPC 읽기, 쓰기 시스템 콜에서 보안등급과 영역이 MLS 보안정책에 적합한지 검사한다. MLS 보안정책 검사에서 실패하면 접근거부 에러를 발생한다.

2) 자료형 추가

자료형 추가를 위해 Linux kernenl 2.2.x에서 linux/ipc.h 파일의 struct ipc_perm 구조체에 다음의 보안등급과 영역에 대한 변수를 추가한다.

· unsigned int clearance;

· unsigned int categories;

ipc_perm 구조체는 3개의 IPC 메커니즘에서 모두 사용한다.

3) 보안등급, 영역 초기화

각 IPC가 생성될 때 보안등급과 영역도 초기화한다. 초기화 값은 현 프로세스의 보안등급과 영역과 같다

가. Message queue

message queue 메커니즘에서 message queue가 생성될 때 초기화 함수는 ipc/msg.c 파일의 newque 이다. 이 함수에 보안등급과 영역 변수 초기화를 위한 코드로 다음을 추가한다.

· msq->msg_perm.clearance = current->clearance;

· msq->msg_perm.categories = current->categories;

나. Semaphore

semaphore를 생성할 때 수행되는 함수는 ipc/sem.c 파일의 newary 이다. 이 함수에 보안등급과 영역 변수 초기화를 위한 코드로 다음을 추가한다.

· sma->sem_perm.clearance = current->clearance;

· sma->sem_perm.categories = current->categories;

다. Shared memory

shared memory를 생성할 때 수행되는 함수는 ipc/shm.c 파일의 newseg 이다. 이 함수에 보안등급과 영역 변수를 초기화하기 위한 코드를 다음과 같이 추가한다.

· shp->u.shm_perm.clearance = current->clearance;

· shp->u.shm_perm.categories = current->categories;

4) MLS 보안정책 적용

위에서 설정해준 보안등급과 영역을 MLS의 보안정책에 의해 통제 될 수 있도록 각 메커니즘의 읽기, 쓰기 시스템 콜에서 검사해야 한다. MLS의 보안정책은 주체 보다 보안등급이 높은 객체는 읽지 못하며, 주체 보다 보안등급이 낮은 객체에는 쓰지 못한다. Linux 커널에서는 IPC 퍼미션 검사를 ipc/util.c 파일의 ipcperms 함수에서 한다. 따라서 MLS 보안정책도 ipcperms 함수에 추가하여 효과를 얻을 수 있다. ipcperms 함수의 원형은 다음과 같다.

· int ipcperms (struct ipc_perm *ipcp, short flag)

ipcperms 함수는 flag 값이 S_IRUGO 일 경우 읽기 권한을 검사하며, S_IWUGO 일 경우 쓰기 권한을 검사한다. 그러므로 flag 값이 S_IRUGO이면 주체의 보안등급과 객체의 보안등급을 비교하여 주체의 보안등급이 객체의 보안등급 보다 작을 경우 -1 값을 리턴 하여 접근이 제한되었음을 표시한다. 이와 마찬가지로 flag 값이 S_IWUGO 이고 주체의 보안등급이 객체의 보안등급보다 클 경우 -1 값을 리턴 하여 보다 높은 등급의 정보가 낮을 보안등급으로 흐르지 못하게 한다.

2. 스토리지 비밀채널 해결방법 설계

1) 설계 개요

비밀채널이 발생할 수 있는 시스템 요소는 많다. 발생 가능한 비밀채널을 모두 제거한다는 것은 불가능 할 것이다. 특히 비밀채널을 고려를 하지 않고 개발되어 사용되고 있는 커널 일 경우 비밀채널을 제거하기 더욱 곤란하다. 왜냐하면 비밀채널은 시스템 콜과 시스템 콜 협약에 의존적이기 때문이다. 본 연구에서는 스토리지에 의한 비밀채널 중 일반적으로 발생 가능한 경우를 대상으

로 이를 해결하였다. 다음은 본 연구의 대상이 된 스토리지 비밀채널이다.

- 파일의 속성에 의한 비밀채널
- 파일의 존재 여부에 의한 비밀채널
- 공유자원(스토리지 관련)에 의한 비밀채널

비밀채널 해결방법은 이들이 발생하지 않도록 제거하는 것과 발생한 비밀채널을 감사 기록하여, 잡음이나 대역폭을 감소시키는 방법을 적용하였다.

2) 파일속성에 의한 비밀채널 해결방법 설계

파일속성에 의한 비밀채널 해결을 위해 파일속성을 알아내는 시스템 콜에 MLS 보안정책을 적용하였다. 파일과 프로세스(객체와 주체)의 보안등급과 영역이 만족할 때만 파일의 정보를 보여주기 위해 파일의 속성을 알아내는 시스템 콜인 fs/stat.c 파일의 sys_newstat, sys_newlstat 시스템 콜에 보안등급과 영역을 검사하는 과정을 추가하였다. 보안등급과 영역을 검사하는 과정은 inode에서 stat 구조체로 정보를 가져오는 cp_new_stat 함수에서 추가된다. 보안정책이 옳으면 cp_new_stat 함수를 수행하고 그렇지 않으면 ENOENT 에러 메시지를 발생하도록 하였다.

3) 파일존재 여부에 의한 비밀채널 해결 방법 설계

파일의 존재 유무를 확인할 수 있는 방법은 많이 존재한다. 'ls' 명령으로 확인할 수 있고, 파일 생성과정이나 디렉토리 삭제과정에서도 파일의 존재 여부를 확인할 수 있다. 그러나 본 연구에서는 'ls' 명령에 의해 확인 할 수 있는 파일의 존재 여부만 처리한다

파일존재에 의한 비밀채널을 처리하기 위해 fs/readdir.c파일의 sys_getdents시스템 콜을 수정하였다.[6] sys_getdents시스템 콜은 디렉토리 엔트리를 반환한다. sys_getdents시스템 콜의 끝 부분에 반환되는 디렉토리 엔트리(디렉토리가 가지고 있는 파일이나 디렉토리)를 검색하는 과정을 추가하고, 과정에서 파일(Object)의 보안등급과 현 프로세스(Subject)의 보안등급을 비교한다. 비교한 파일과 프로세스의 보안등급이 MLS 보안정책에 맞지 않으면, 반환되는 엔트리에서 파일의 정보를 삭제 한 후 반환한다. 그렇게 하면 현 프로세스와 다른 보안등급의 파일은 'ls'명령으로 존재 유무를 확인 할 수 없다.

4) 공유자원에 의한 비밀채널 처리 설계

공유자원에 의한 비밀채널은 프로세스들이 공유하고 있는 자원 즉, 메인 메모리나 보조 기억장치를 통해 일어 날 수 있는 비밀채널을 의미한다. 공유자원에 의해 발생하는 비밀채널의 형태는 한정된 자원의 소비와 이미 사용하고 있는 자원의 요구로 발생하는 형태이다. 프로세스들이 공유하고 있는 자원은 한정되어 있고, 한정되어 있는 자원은 모두 소비 될 수 있다. 자원이 모두 소비되면 더 이상 프로세스는 생성되지 못한다. 남은 자원에 따라 프로세스 생성은 성공할 수도 있고 실패할 수도 있다.

공유자원에 의한 비밀채널은 TCB 명세의 수정에 의해 처리해 주면 다른 프로그램과 시스템에 오동작을 발생하게 만들 소지가 있다. 따라서 본 연구에서는 감사를 통해 비밀채널을 식별하고 처리해 주기 위해 채널의 대역폭을 줄이는 지연 방법을 사용한다.

공유자원에 의한 비밀채널을 처리하기 위해 다음의 단계를 설계하였다.

단계 1: 공유자원에 의한 비밀채널이 발생 가능한 시스템 콜과 시스템 콜 협약 정의 및 정리

단계 2: 단계 1에서 정의한 비밀채널이 발생 가능한 시스템 콜 협약이 발생할 경우 감사를 할 수 있도록 기록(log)을 남긴다.

단계 3: log 감사에 이용할 비밀채널 기준을 세운다.

단계 4: 단계 2에서 남겨진 log를 단계 3의 기준에 적용하여 비밀채널 여부 판단

단계 5: 단계 4에서 비밀채널이라고 판단될 경우 지연을 추가한다.

가. 시스템 콜과 시스템 콜 협약 정의 및 정리

비밀채널이 발생할 수 있는 공유자원 관련 시스템 콜과 시스템 콜 협약은 「비밀채널을 생성할 수 있는 인터페이스의 조건은 정보를 보내고자 하는 프로세스가 다른 프로세스에게 영향을 미칠 수 있는 환경을 만들어야 하며 정보를 받는 프로세스에서는 정보를 보내고자 하는 프로세스가 만든 환경에 따라 성공 혹은 실패 여부가 분명히 나타나야 한다.」로 정의하였다.

다음은 위의 정의에 부합하는 대표적인 시스템

콜과 시스템 콜 협약을 linux kernel 2.2.x에서 찾아 정리한 것이다[표 1].

시스템 콜	시스템 콜 협약	설 명
sys_execve	ENOMEM	메인 메모리 부족
	ENFILE	시스템 안에 빈 디스트립터가 없을 때
	EMFILE	process을 위한 빈 디스크립터가 없을 때
sys_fork	EAGAIN	메모리를 할당할 수 없을 때
	ENOMEM	메모리 부족
sys_mount	EBUSY	장치나 자원 사용 중
sys_pipe	EMILE	시스템 안에 빈 디스트립터가 없을 때
	ENILE	process을 위한 빈 디스크립터가 없을 때

[표 1] 비밀채널 발생 가능 시스템 콜과 시스템 콜 협약

나. LOG 작성

[표 1]의 시스템 콜에서 시스템 콜 협약이 발생하면 감사를 할 수 있도록 log를 남기게 된다. log는 두 가지 형태로 작성된다. 빠른 분석을 위해 커널 전역에서 사용할 수 있는 배열과 사람이 분석할 수 있는 파일의 형태이다.

다음은 파일에 기록되는 log의 형태이다.

· log가 발생한 시간 - 커널 버전 - 시스템 콜 이름 - 프로세스 번호 - 시스템 콜 협약(error number) - 시스템 콜 발생 시간(초) (100000분의 1초)

다. 비밀채널 판단 기준

정형적인 비밀채널의 허용 대역폭에 대한 정의는 없으며, 운용되는 시스템에 따라 허용 대역폭을 결정해야만 한다. 따라서 본 연구의 비밀채널 판단 기준은 1초 동안 비밀채널 발생 가능한 시스템 콜 협약이 2개 이상 발생할 경우로 정했다. 이런 경우 ASCII 문자 코드 셋을 사용할 경우 최대 대역폭을 1cps로 제한하는 것이다.

라. 비밀채널 처리 정책

비밀채널 판단 기준을 적용하여 비밀채널로 판단될 경우 지연 시간을 주어 대역폭을 감소시킨다. 대역폭 감소를 위해 주어지는 지연 시간은 1초로 하였다. 지연 시간을 1초로 하였을 경우 대역폭을 1cps로 제한할 수 있게 된다. 지연 시간에 대한 정의 역시 운용되고 있는 시스템에 따라 달리 설정할 수 있다.

IV.커널 구현

1. 구현환경

본 연구에서 개발된 보안 커널은 Open source 운영체제인 리눅스 커널을 이용하였다. 개발 환경 역시 리눅스가 운용될 수 있는 일반 PC 워크 스테이션을 사용하였다. [표 2]은 보안 커널 개발 환경이다

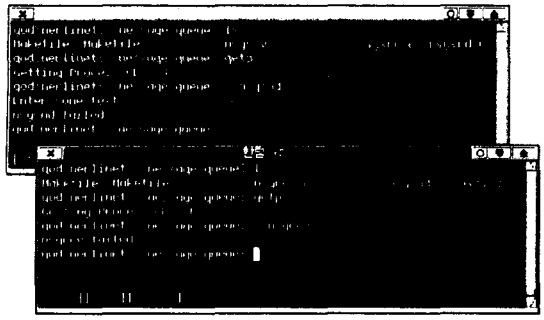
	구 성
하드웨어	· CPU : Intel Pentium II 500 DUAL · 주기억장치 용량 : 1G · 보조기억장치 용량 : 17.5 GB
운영체제	· 운영체제(OS) : Linux · OS 패키지명 : Mandrake Linux · 패키지 버전 : 7.0
개발 도구	gcc 2.95
라이브러리	klib
커널	시큐브 (주)에서 제작한 MLS 적용된 linux kernel 2.2.14

[표 2] 개발 환경

커널 수준의 파일 출력을 하기 위해 Linux Kernel Library Project로 작성된 Klib를 사용하였다.[7]

2. 신뢰경로를 보장하는 IPC 구현

[그림 3]은 MLS 보안정책을 IPC메커니즘 중 message queue를 나타낸 것이다. 보안등급이 0인 프로세스로 'msgrcv' 명령을 수행하여 보안등급이 0인 message queue 메모리 객체를 만들고, 보안등급이 3인 프로세서로 'msgsnd' 명령을 수행하여 보안등급 0인 message queue 메모리 객체에 '쓰기'하려 하면 'failed' 메시지가 출력된다.



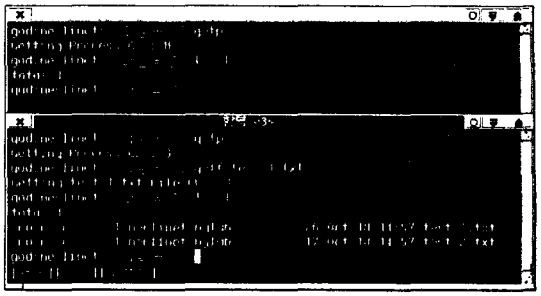
[그림 3] MLS 보안정책이 적용된 IPC(쓰기)

[그림 4]는 보안등급이 낮은 프로세스에서 보안등급이 높은 메모리 객체를 읽으려 할 때 MLS 보안정책으로 '읽기' 연산에 'failed' 메시지를 리턴하는 모습이다.

[그림 4] MLS 보안정책이 적용된 IPC(읽기)

3. 스토리지 비밀채널 처리 구현

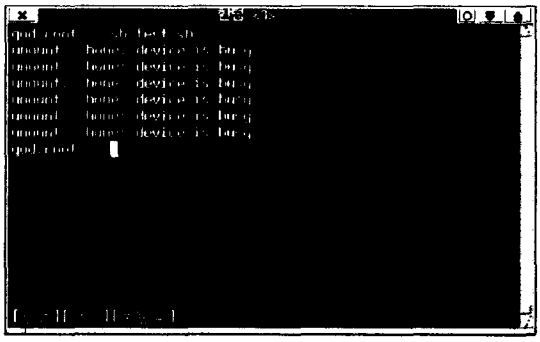
파일속성에 의한 비밀채널은 보안등급이 낮은 프로세스에서는 보안등급이 높은 파일의 속성을 알 수 없다. [그림 5]는 파일속성과 파일존재에 의한 비밀채널 처리를 나타낸 것이다.



[그림 5] 파일속성과 존재 여부에 의한 비밀채널

파일존재에 의한 비밀채널은 보안등급이 낮은 프로세스에서는 보안등급이 높은 파일의 존재 여부를 확인할 수 없어야 한다. 파일의 존재 여부 확인은 여러 경우가 있으나 본 연구에서는 'ls'명령에 의한 확인만 처리하였다.

공유자원에 의한 비밀채널은 비밀채널 발생가능 시스템 콜 협약이 비밀채널 판단기준 이상으로 발생할 경우 지연(Delay) 시간이 주어지게 되고, 비밀채널의 대역폭이 낮아진다. [그림 6]은 sys_umount 시스템 콜이 이상적으로 많이 발생한 경우를 한 것으로 테스트한 것으로 지연 삽입으로 대역폭이 낮아지는걸 확인할 수 있었다.



[그림 6] 공유자원에 의한 비밀채널 처리

V. 결론 및 향후 연구과제

본 연구에서는 MLS 보안정책에서 문제가 되었던 불법적 정보흐름을 처리하기 위한 방법론을 제시하였다. 제시된 방법론으로 시스템 내에서 정보흐름의 수단이 되었던 IPC 메커니즘에 MLS 보안정책을 적용한 것과 TCB 명세에 의해 발생할 수 있는 비밀채널 중 스토리지에 위해 발생 가능한 비밀채널을 처리하였다.

향후 연구해야 할 과제로는 파일존재 여부에 의한 비밀채널에서 보다 근본적이고, 안정적인 채널 제거를 위한 노력이 필요하고 공유자원에 의한 비밀채널 처리를 위한 본 연구의 방법론을 대표적인 시스템 콜만 적용하는 것이 아니라 보다 확대하여 적용 해야할 것이다. 또한 비밀채널 식별에서 식별 기준 정의를 위해 다양한 시스템 콜에 관한 분석과 통계, 대역폭에 관한 연구가 이루어져야 할 것이다.

참고문헌

[1] Morrie Gasser, "Building a secure computer system", VNR
[2] "A Guide to Understanding Covert Channel Analysis of Trusted System", NCSC-TG-030
[3] 한동훈, 이만용 역, "초보자용 리눅스 프로그래밍", 도서 출판 대림
[4] 김치하 역, "Advanced programming int the UNIX environment", 홍릉과학출판사
[5] Chris Brown, "UNIX Distributed programming", PRENTICE HALL
[6] Pragmatic, "(nearly) Complete Linux Loadable Kernel Modules", 03/1999
[7] klib, http://linuxkernel.net