情 報 保 護 學 會 誌
第 15 卷 第 2 號, 2005. 4

보안운영체제를 위한 자원관리기법 연구

김 동 근*, 김 정 순*, 이 승 용*, 김 민 수**, 노 봉 남***

요 약

각종 침해사고에 대하여 보안에 대한 관심이 높아졌고 최근 많은 기술들이 안정된 서버의 운영을 위하여 연구되었다. 하지만 이들 기술과 운영체제의 통합에 대한 연구는 미진한 상황이다. 보안기능이 강화된 운영체제인 보안운영체제는 각종 침입에 대비하여 효율성과 보안성을 가져야 한다. 이를 위해서 운영체제가 제공해야할 중요한 기능 중 하나인 자원관리는 효율성과 보안성을 모두 만족시켜야 한다. 본 고에서는 현재의 리눅스와 유닉스의 자원관리기법을 살펴보고 보안운영체제가 제공해야할 자원관리기법에 대해 고찰한다.

Ⅰ. 서 론

컴퓨터 범죄의 발생 빈도가 점점 증가하면서 안정된 서버의 운영에 대한 관심이 증폭되고 있다. 이는 각종 서비스를 제공하는 서버 애플리케이션의 보안과 서버에 설치된 운영체제 자체에 대한 보안의 연구로 이어졌다.

보안운영체제는 기존운영체제에서 지원되는 전형적인 보안기능이 아닌 부가적인 보안 메커니즘이 제공되도록 운영체제를 수정하거나 보안 기능을 강화시킨 운영체제이다. 보안운영체제는 강제적 접근통제(Mandatory Access Control) 메커니즘을 제공하며, 폭넓은 시스템 접근통제 정책, 접근권한에 대한 규칙 등을 제공한다. 그러나, 대부분의 운영체제는 자원과 서비스를 보호하기 위하여 임의적 접근통제(Discretionary Access Control) 기법이 적용되고 있다. 그래서, 최근에 강제적 접근통제 방법이나 역할기반접근통제(Role-Based Access Control) 방법이 적용된 운영체제가 연구되어 개발되었다. 이러한 보안이 강화된 운영체제에서 효율적인 자원관리에 대한 필요성이 증대되고 있지만 연구는 미진한 실정이다.

운영체제의 자원관리는 시스템의 효율을 높이고 안정성을 제공해야 한다. 이러한 자원들에는 메모리, CPU 시간, 프로세스, 파일 시스템 등이 해당된다. 최근의 운영체제에서는 사용자마다 자원 제한 방법을 제시하여 과도한 사용을 방지하고 있다. 보안운영체제는 서비스 거부공격(Denial of Service)과 같은 컴퓨터 보안 위협에 대비하여 보안을 위한 자원관리 기법과 가용성과 신뢰성을 보장할 수 있도록 침입감내 기법을 고려해야 한다.

본 고에서는 침입감내시스템과 유닉스, 리눅스의 자원관리기법, 클래스기반 커널 자원관리기법에 대하여 살펴보고 보안운영체제를 위한 자원관리기법에 대한 연구의 방향을 고찰한다.

Ⅱ. 침입감내기술

2.1 침입감내기술의 정의

보안위협에 대응하기 위한 정보보호 기술은 지금까지 비밀성 보장을 위한 암호기술, 침입차단기술, 접근통제와 같은 기술이 주된 기술이었다. 이러한 기술들은 중요정보의 비밀성 보장 측면에서 매우 효과적이었지만 가용성은 매우 떨어지는 단점이 있다. 이러한 단점은 침입탐지시스템과 공격대응기술 등의 정보보호기술을 사용함으로써 일부 해결되었다.

그러나 침입차단시스템이 차단하거나, 침입탐지시스템이 탐지할 수 있는 공격 또는 침입의 비율은 전체의

본 연구는 정보통신부 대학 IT연구센터 육성, 지원사업의 연구결과로 수행 되었습니다.
* 전남대학교 리눅스보안연구센터 ({guri, cybersun, birch}@lsrc.jnu.ac.kr)
** 목포대학교 정보보호학과 (phoenix@mokpo.ac.kr)
*** 전남대학교 전자컴퓨터정보통신공학부 (bbong@jnu.ac.kr)

일부분에 지나지 않는다. 또한 침입차단시스템에 의한 차단이 어려운 내부 공격자들에 의한 보안 사고나 침입탐지기술로 탐지하지 못하는 알려지지 않은 취약점을 이용하는 공격에 대한 대응은 매우 어려우므로, 이러한 공격에 대한 고려가 필요하게 되었다. 침입감내기술은 이러한 문제를 해결하기 위한 기술이며, 방지기술과 탐지기술로 미처 발견하지 못한 공격이나 침입이 있는 경우에도 서비스를 정상적으로 제공하기 위한 새로운 분야의 정보보호기술이다.[10-12]

2.2 침입감내기술의 특징

침입감내기술이 가지는 특징은 가용성과 신뢰성, 그리고 보안성으로 대표되는 의존성의 확보이다. 가용성의 확보를 위한 기술들은 시스템의 클러스터링 기술이나 보안시스템의 통합적 사용을 위한 ESM 기술의 확장된 사용, 침입대응 및 복구기술의 사용 등을 통하여 부분적으로 이루어 질 수 있다. 그러나 침입감내시스템은 서비스의 가용성 제공과 더불어 제공되는 정보의 신뢰성과 보안성, 그리고 적시성이 동시에 만족되어야 하는 특징이 있다. 그러므로 부분적인 문제 해결을 제공하는 기존의 기술들과 많은 부분에서 차이를 보인다.

2.3 침입감내기술의 목적

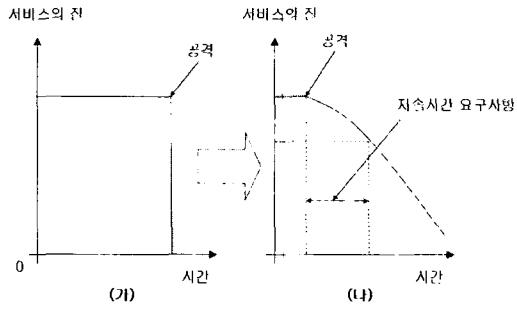
침입감내기술은 가용성, 안전·신뢰성 강화를 위한 기술을 개발함으로써 사이버공격 대응체제를 구축하는데 목적이 있다. 침입감내시스템은 원하는 서비스의 품질을 일정시간 지속적으로 제공할 수 있어야 한다. 그림 1은 침입감내기술이 제공해야할 지속시간 요구사항을 설명한다. 그림 1의 (가)에서 서비스 거부 공격이 발생하는 경우 수행 중이던 서비스의 품질이 급격히 감소하여 정상적으로 서비스를 받을 수 없는 상태가 된다. 침입감내 시스템은 그림 1의 (나)에서와 같이 서비스의 품질이 이러한 상태로 전이되는 것을 방지할 수 있도록 침입감내 요구사항, 즉 지속시간 요구사항과 품질 요구사항을 제시하고, 이 요구사항을 만족시킬 수 있도록 구축된다.[1]

Ⅲ. 솔라리스 자원 관리자

3.1 솔라리스의 자원관리 컴포넌트

솔라리스의 자원관리 컴포넌트는 다음과 같다.

- 자원 제어
- 프로젝트와 태스크
- 확장된 어카운팅
- 자원 풀



〔그림 1〕 침입감내 요구사항

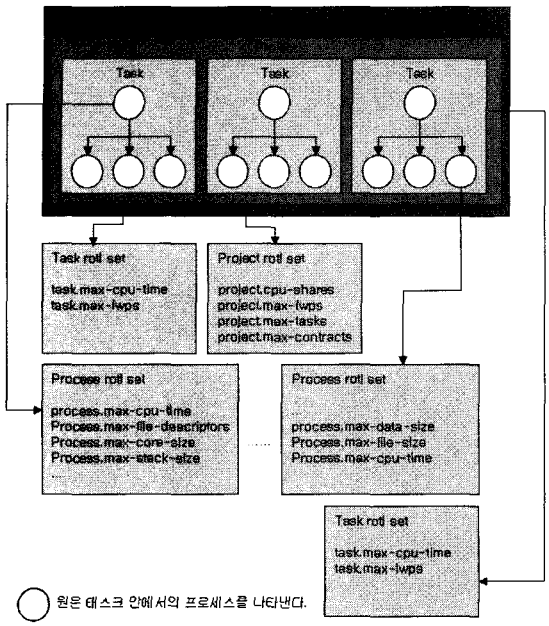
3.1.1 자원제어

자원 제어는 자원의 사용을 제한하기 위하여 사용된다. 자원 제한은 자원의 과도한 사용을 방지한다. 솔라리스의 자원제어는 다음과 같은 특징을 가진다.

- 동작 중인 시스템에 대한 동적인 제어가 가능하다.
- 프로젝트, 태스크, 프로세스 수준의 제어가 가능하다.
- 실제 자원의 한도보다 높은 사용 한계 값을 설정하려고 시도해도 사용 한계 값은 변하지 않는다.

3.1.2 프로젝트와 태스크

구역(zone)은 응용프로그램의 실행을 위해 가상의 독립된 환경을 생성하는 운영체제의 서비스를 제공한다. 이러한 독립된 환경의 제공은 한 구역에서 실행되는 프로세스가 다른 구역에서 실행되는 프로세스를 모니터링하거나 영향을 미치는 것을 방지한다. 태스크는 워크로드(Workload)를 표현하는 프로세스들의 집합이다. 프로젝트는 전체 워크로드를 표현하는 태스크들의 집합이다. 어느 한 시점에 한 프로세스는 오직 한 태스크와 한 프로젝트에 속한다. 두 프로젝트 이상에 연관되어진 사용자는 동시에 여러 프로젝트의 프로세스를 실행할 수 있다. 프로세스로부터 실행된 모든 프로세스는 부모프로세스로부터 프로젝트속성을 상속받는다. 시작 스크립트에서 새로운 프로젝트로 전환한다면 그에 따르는 모든 프로세스는 새로운 프로젝트에서 실행된다. 실행중인 프로세스는 연관된 사용자 아이디, 그룹 아이디, 프로젝트 아이디를 가진다. 프로세스의 속성은 태스크 수행문맥을 형성하기 위하여 사용자 아이디, 그룹 아이디, 프로젝트 아이디로부터 상속



〔그림 2〕 솔라리스의 프로세스, 태스크, 프로젝트의 자원제어집합

된다. 그림 2는 솔라리스의 프로세스, 태스크, 프로젝트의 자원제어집합이다.

3.1.3 확장된 어카운팅

확장된 어카운팅은 작업이 완료된 프로젝트에 사용기록 라벨을 남긴다. 태스크나 프로세스의 시스템과 네트워크의 자원 사용에 대한 유연한 기록시스템을 제공한다. 사용되는 자원의 양을 실시간으로 알려주는 모니터링 툴과는 달리 관리자는 확장된 어카운팅으로 추후에 워크로드로부터 요구될 자원을 평가해볼 수 있다.

3.1.4 자원 풀

자원 풀은 프로세서와 스레드 스케줄링 클래스에 대한 관리를 제공한다. 자원 풀은 시스템의 자원을 각 워크로드에 분할하여 자원의 사용이 겹치지 않도록 한다.

3.2 솔라리스의 자원관리 항목

다음 표 1은 솔라리스의 자원관리 항목이다.

〔표 1〕 솔라리스의 자원관리 항목

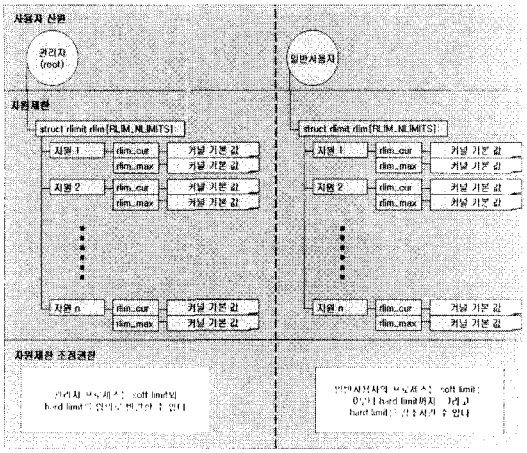
시스템 자원	설명
CPU shares	제한을 명시하는 lnode에 할당된 CPU 시간의 양. 솔라리스 자원관리자는 모든 가용한 시스템 자원을 lnode에 할당한다. lnode는 자원이 사용가능 상태라면 할당받은 양보다 많은 자원을 사용할 수 있다.
CPU accrual	현재 lnode에 해당하는 그룹 안의 모든 lnode들의 CPU 사용 제한
Memory limit	lnode에 연관된 모든 프로세스에게 허용된 최대의 가상메모리 제한. 이 제한은 제한 데이터베이스에 명시된 고정된 값이다. 0은 제한이 없음을 나타낸다.
Process memory limit	한 프로세스에게 허용된 최대 가상메모리 제한. 이 제한은 제한 데이터베이스에 명시된 고정된 값이다. 0은 제한이 없음을 나타낸다.
Physical memory limit	lnode에 연관된 모든 프로세스에게 허용된 물리메모리 제한. 이 값은 양수여야만 한다.
Memory accrual	일정 시간간격에 따른 메모리 사용 제한.
Number of processes	사용자가 동시에 실행시킬 수 있는 프로세스 수의 제한.
Number of logins per user	사용자에 대한 동시 로그인 세션 제한.
Connect-time	사용자의 로그인 시간 제한.

Ⅳ. 리눅스 커널의 자원제한관리

4.1 리눅스 커널의 프로세스 자원제한

리눅스 커널의 프로세스 자원제한관리에서 자원제한에 대한 제어 권한은 관리자 권한 프로세스와 일반 권한 프로세스로 구분된다. 일반 권한의 프로세스는 관리자 권한 프로세스와 달리 제한적인 자원제한 조절 권한을 가지게 된다. 따라서 리눅스 커널은 사용자 또는 관리자가 지정한 우선순위나 응용프로그램의 중요도와 각 응용프로그램의 특성에 따른 자원의 활용을 보장하지 않는다. 리눅스 커널의 프로세스 자원제한 관리는 그림 3과 같다. 각 프로세스에 대하여 soft limit와 hard limit는 커널의 기본 설정에 따라서 정해진다. 관리자 권한 프로세스는 soft limit와 hard limit을 증가 또는 감소시킬 수 있다. 관리자 외의 일반 사용자 프로세스는 soft limit를 0부터 hard limit까지 변경시킬 수 있으며, hard limit를 감소시킬 수 있다.[5]

각 프로세스마다 프로세스가 사용할 수 있는 자원의 양을 지정하는 여러 자원 제한이 있다. 자원에는 CPU사용시간과 파일의 크기, 실행할 수 있는 프로세스의 수 등이 있다. 이러한 제한을 통해 사용자가 자원을 과도하게 사용하는 것을 막을 수 있다. 리눅스 커널에서는 다음과 같은 자원 제한을 지원한다.[6-7]



〔그림 3〕 리눅스 커널의 프로세스 자원제한 관리

4.1.1 RLIMIT_AS

프로세스 주소 공간의 최대 크기(바이트 단위). 커널은 프로세스가 malloc이나 자신의 주소 공간을 늘리는 관련 함수를 호출할 때 이 값을 검사한다.

4.1.2 RLIMIT_CORE

최대 코어 덤프 파일 크기(바이트 단위). 커널은 프로세스가 어떤 문제로 종료될 때 프로세스의 현재 디렉토리에 core 파일을 만들기 전에 이 값을 검사한다. 제한 값이 0이라면 커널은 이 파일을 만들지 않는다.

4.1.3 RLIMIT_CPU

프로세스가 사용할 수 있는 최대 CPU 시간(초 단위). 프로세스가 이 제한 시간을 넘어 동작하면 커널은 프로세스에게 SIGXCPU 시그널을 보낸다. 이 시그널에 대한 기본 반응은 프로세스를 종료하는 것이다. 그러나 이 시그널은 실행을 계속하도록 프로세스에 의해 처리될 수 있다. 만약 프로세스가 CPU 시간을 계속 사용한다면 커널은 SIGXCPU 시그널을 hard limit에 도달할 때까지 프로세스에게 매초마다 보낸다. 그래도 프로세스가 종료하지 않으면 SIGKILL시그널을 보낸다.

4.1.4 RLIMIT_DATA

프로세스의 데이터 세그먼트의 최대 크기(바이트 단위). brk와 sbrk호출 시에 soft limit을 넘어서게 되면 ENOMEM를 반환한다.

4.1.5 RLIMIT_FSIZE

사용할 수 있는 최대 파일 크기(바이트 단위). 프로세스가 이 값보다 큰 파일을 만들려고 하면 커널은 SIGXFSZ 시그널을 보낸다. 이 시그널에 대한 기본 반응은 프로세스를 종료하는 것이다. 하지만, 이 시그널은 프로세스에 의해서 처리될 수 있다. 시그널이 프로세스에 의해 처리될 경우에 함수의 호출은 EFBIG 에러를 반환하며 실패한다.

4.1.6 RLIMIT_LOCKS

최대 파일 잠금 개수. 커널은 프로세스가 flock 또는 fcntl 시스템 호출로 파일에 잠금을 걸려고 할 때 이 값을 검사한다.

4.1.7 RLIMIT_MEMLOCK

스왑할 수 없는 메모리의 최대 크기(바이트 단위). 커널은 프로세스가 mlock이나 mlockall시스템 콜을 이용하여 메모리에서 페이지 프레임을 잠그려고 할 때 이 값을 검사한다.

4.1.8 RLIMIT_NOFILE

최대로 열 수 있는 파일 디스크립터의 수. 커널은 open, pipe, dup 등의 시스템호출 시에 이 값을 검사하여 이 제한 값에 도달하게 되면 EMFILE을 반환하도록 한다.

4.1.9 RLIMIT_NPROC

사용자가 생성할 수 있는 최대 프로세스 개수. 이 제한 값에 도달하게 되면 fork시스템 호출은 EAGAIN에러와 함께 호출이 실패한다.

4.1.10 RLIMIT_RSS

프로세스가 소유할 수 있는 최대 페이지 프레임 수. 커널은 프로세스가 malloc이나 자신의 주소 공간을 늘리는 관련 함수를 호출할 때 이 값을 검사한다.

4.1.11 RLIMIT_STACK

최대 스택 크기(바이트 단위). 커널은 프로세스의 사용자 모드 스택 크기를 늘리기 전에 이 값을 검사한다. 제한 값에 도달하게 되면 SIGSEGV 시그널이 발생한다.

4.2 setrlimit, getrlimit 시스템 호출

getrlimit과 setrlimit시스템 호출은 각각 프로세스가 사용할 시스템자원에 대한 제한을 읽거나 설정할 수 있게 해준다. 다음은 setrlimit, getrlimit 시스

템 호출 함수의 원형이다.[5]

int getrlimit(int resource, struct rlimit *rlim);
int setrlimit(int resource, const struct rlimit *rlim);

각 자원에 대하여 hard limit과 soft limit의 두 가지 제한이 존재한다. soft limit은 hard limit을 넘지 못하고, 관리자 권한을 가진 프로세스만이 hard limit을 변경할 수 있다. 전형적으로, 응용프로그램은 사용할 자원을 스스로 제한하기 위하여 soft limit을 감소시킨다. RLIM_INFINITY 값은 자원을 무제한으로 사용할 수 있도록 설정할 때 사용된다.

getrlimit과 setrlimit은 모두 자원 제한 타입과 struct rlimit변수를 파라미터로 받는다. getrlimit은 넘겨준 변수에 값을 채워주는 반면에, setrlimit은 넘겨준 변수의 값에 따라 자원제한을 변경하여 준다. rlimit 구조체는 두 개의 멤버변수를 갖는다. rlim_cur는 soft limit을, rlim_max는 hard limit을 각각 나타낸다. 이 시스템 호출은 성공 시에는 0을, 실패 시에는 errno에 해당 에러번호를 저장하고 -1을 반환한다. 표 2는 에러번호에 해당하는 에러내용이다.

〔표 2〕 setrlimit, getrlimit의 에러번호

EFAULT	rlim이나 usage가 접근할 수 없는 메모리 영역을 가리키고 있다.
EINVAL	getrlimit나 setrlimit가 잘못된 resource에 대하여 호출되었거나, getrusage가 잘못된 who에 대하여 호출되었다.
EPERM	관리자(root)가 아닌 사용자가 setrlimit를 시도하여 soft limit나 hard limit을 증가시키려고 했거나, 관리자가 RLIMIT_NOFILE을 커널의 최대제한보다 높게 하려고 시도하였다.

4.3 getrusage 시스템 호출

getrusage 시스템 호출은 현재 사용 중인 자원의 양을 알 수 있게 해준다. 다음은 getrusage 시스템 호출 함수의 원형이다〔Man01〕.

int getrusage(int who, struct rusage *usage);

who매개변수에 RUSAGE_SELF를 명시하면 현재 프로세스의 자원 사용량을, RUSAGE_CHILDREN을 명시하면 종료를 기다렸던 자식프로세스의 자원 사용량을 알아내는데 사용된다. 호출이 성공하면 두 번째 매개변수에 프로세스의 자원 사용량이 저장된다.

4.4 ulimit 명령

ulimit 명령은 쉘과 쉘이 실행하는 프로세스들의 자원에 대한 제어를 제공한다. 다음은 ulimit명령의 사용법이다.

ulimit 〔-SHacdflmnpstuv 〔limit〕〕

-H와 -S옵션은 각각 자원에 대한 hard limit과 soft limit을 명시하는데 사용된다. 수퍼유저 권한이 없는 프로세스는 hard limit을 증가시킬 수 없지만 soft limit는 hard limit까지 증가시킬 수 있다. 만약 -H 또는 -S옵션이 모두 주어지지 않는다면 soft limit과 hard limit이 모두 설정된다. 제한 값으로 자원에 대한 숫자 또는 현재의 hard limit값을 나타내는 'hard', 현재의 soft limit값을 나타내는 'soft', 제한이 없음을 나타내는 'unlimited'를 쓸 수 있다. 만약 제한 값이 생략되고 -H옵션이 주어지지 않는다면 현재의 soft limit값을 출력한다. 한 가지 이상의 자원에 대하여 명시하면, 자원의 이름과 단위가 값의 앞에 출력된다. 기타 옵션은 표 3과 같다.

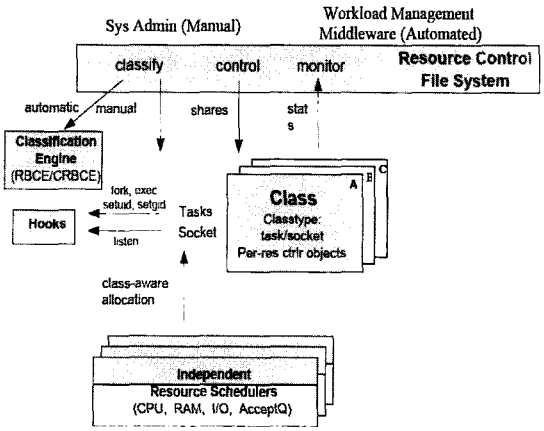
〔표 3〕 ulimit 명령의 옵션

옵션	설 명
-a	모든 자원의 현재 제한을 출력한다.
-c	core파일의 최대 제한 크기를 출력한다.
-d	프로세스의 최대 제한 데이터 세그먼트 크기를 출력한다.
-f	쉘에 의해 생성될 최대 제한 파일의 크기를 출력한다.
-l	잠글 수 있는 메모리의 최대 크기
-m	최대 상주집합 크기
-n	열 수 있는 최대 파일의 수 (대부분의 시스템에서 이 값을 설정하는 것을 허락지 않는다.)
-p	512-바이트 블록의 파이프 크기
-s	최대 스택 크기
-t	최대 CPU 사용 시간
-u	사용자가 생성할 수 있는 최대 프로세스의 수
-v	쉘이 사용하는 최대의 가상 메모리 크기

새로운 제한 값이 주어지면, 그것은 명시된 자원에 대한 새로운 제한 값이 된다. -a 옵션은 출력만을 위한 것이다. 아무 옵션도 주어지지 않으면, -f옵션이 주어진 것으로 가정된다. 잘못된 옵션이나 매개변수를 제공하거나 새로운 제한 값을 설정하는 중에 에러가 발생하지 않는다면 0을 반환한다.

V. 클래스기반 커널자원관리

클래스기반 커널자원관리(Class-based Kernel Resource Management, CKRM)는 리눅스 커널 2.6에서 자원관리를 수행하는 시스템이다. 그림 4는 클래스기반 커널자원관리의 구조를 설명한다.[4] 클래스기반 커널자원관리는 커널 오브젝트를 태스크 클래스와 소켓 클래스의 두 가지의 클래스타입으로 구분한다. 태스크 클래스는 CPU 사용시간, 물리 메모리 페이지, 디스크 I/O와 대역폭에 대하여 관리를 제공하고, 소켓 클래스는 들어오는 네트워크 대역폭에 대하여 관리를 제공한다. 클래스기반 커널자원관리는 태스크 클래스와 소켓 클래스의 각각에 대하여 동적으로 클래스를 정의하여 자원관리를 수행한다. 동적으로 생성된 클래스는 자원의 사용에 대하여 명시하고 이는 커널의 내부에서 자원관리를 적용하는데 사용된다.[8,9]

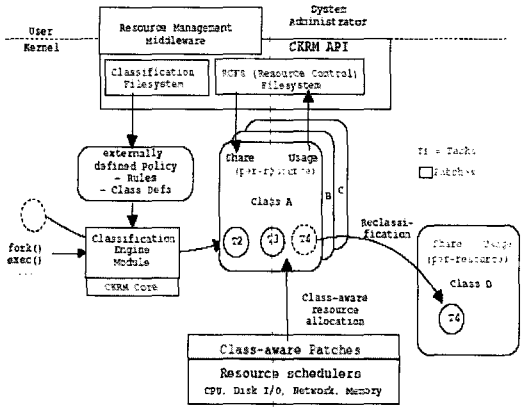


〔그림 4〕 클래스기반 커널자원관리의 구조

5.1 구성요소

클래스기반 커널자원관리의 구성요소는 다음과 같다.

- 핵심 (Core)
- 분류엔진 (Classification Engine)
- 자원관리자 (Resource Controller)



〔그림 5〕 클래스기반 커널자원관리의 프레임워크

- 자원관리 파일시스템 (Resource Control File System)

그림 5는 클래스기반 커널자원관리의 프레임워크 보여준다. 시스템 관리자는 자원관리 미들웨어를 사용하여 클래스 분류 파일시스템에 클래스 분류정책을 정의하고 분류된 클래스에 대한 자원제어정책을 자원제어 파일시스템을 통해 정의한다. 클래스분류엔진 모듈에서는 프로세스의 fork, exec등의 이벤트에 대하여 정의된 클래스 분류정책을 적용하여 클래스를 분류한다. 자원 스케줄러에서는 분류된 클래스에 대하여 자원제어정책을 적용한다. 다음 절에서 클래스기반 커널자원관리의 각 구성요소에 대하여 살펴보겠다.

5.1.1 핵심

클래스기반 커널자원관리의 핵심은 다른 모든 컴포넌트들을 연결한다. 분류엔진에서 사용될 클래스타입을 정의하고 분류엔진을 가동시켜서 커널 오브젝트를 분류하도록 한다.

5.1.2 분류엔진

클래스기반 커널자원관리에서 관리되는 각각의 커널 오브젝트는 항상 특정 클래스와 연관되게 된다. 만약 아무런 클래스도 정의되어있지 않다면 모든 커널 오브젝트는 해당 클래스타입(classtype)의 기본설정 클래스로 분류된다. 커널 오브젝트의 속성이 변경되는 fork, exec, setuid, listen과 같은 중요한 커널의 이벤트 발생 시에 클래스기반 커널자원관리는 분류 엔진에게 분류를 요청하고 분류된 결과는 커널 오브젝트에 연관된다. 클래스기반 커널자원관리는 관리자가 커널 오브젝트의 속성에 따라 클래스로의 사상을 할 수

있도록 해주는 규칙기반 분류엔진을 제공한다.

5.1.3 자원관리자

각 클래스타입은 연관된 자원관리자를 가진다. 태스크 클래스는 CPU 사용시간, 물리 메모리 페이지, 디스크 I/O와 대역폭에 대하여 관리를 제공하고, 소켓 클래스는 들어오는 네트워크 대역폭에 대하여 관리를 제공한다. 자원관리자는 서로 독립적이며, 따라서 한 자원에 대하여 관리를 수행하며 다른 자원에 대하여 관리를 적용하지 않을 수 있다. 예를 들면 CPU사용시간에 대하여 자원관리를 수행하고 물리 메모리 페이지에 관하여서는 자원관리를 수행하지 않을 수 있다.

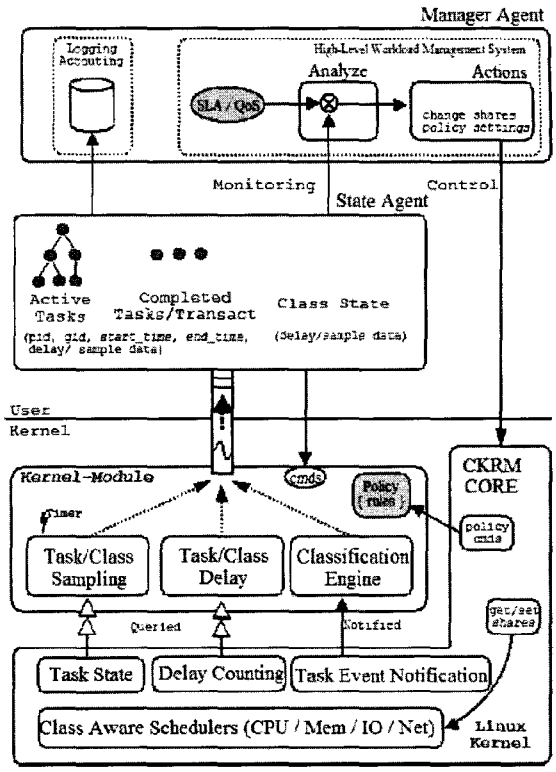
5.1.4 자원관리 파일시스템

자원관리 파일시스템은 클래스기반 커널자원관리의 주요 인터페이스이다. 이는 가상파일시스템으로 구현되었다. 이는 커널레벨과 사용자레벨을 연결한다. 디렉토리와 파일의 계층을 제공한다. 일반적인 파일과 디렉토리에 대한 연산인 open, close, read, write, mkdir, rmdir, unlink 를 사용하여 관리된다. 자원관리 파일시스템에서 디렉토리는 클래스와 대응된다. 가상파일에 대한 쓰기연산은 클래스기반 커널자원관리의 함수를 호출하여 설정을 변경한다. 가상파일에 대한 읽기연산은 현재의 설정을 알아보는 기능으로 사용된다.

5.2 자원 모니터링

자원모니터링에서는 자원관리자에 포함된 상태에이전트 (state-agent)에 상태정보를 유지하여 자원사용을 감시한다. 클래스기반 커널자원관리는 주기적으로 각 태스크에 대하여 샘플링(Sampling)을 수행하여 태스크가 기다려온 자원과 지금까지 사용한 자원과 속해있는 클래스에 대한 정보를 저장한다.

그림 6은 클래스기반 커널자원관리의 자원 모니터링과 제어에 대한 흐름을 보여준다. 프로세스가 생성되면 이 프로세스는 분류엔진의 정책설정 또는 관리자의 설정작업에 의해 특정 클래스로 분류된다. 상태에이전트는 각 클래스의 상태와 해당 클래스 내의 프로세스의 자원사용에 대한 정보를 관리하고 이 정보에 대한 로깅, 어카운트를 수행하며 워크로드매니저 에이전트(Workload Manager Agent)에 자원사용에 대한 정보를 전달한다. 운영체제는 종료된 프로세스에 대한 정보를 저장하지 않기 때문에 이 정보 또한 상태에이전트에서 저장하여 관리하게 된다. 응용프로



〔그림 6〕 클래스기반 커널자원관리의 자원 모니터링과 제어

그램으로 구현된 워크로드 매니저 에이전트는 SLA (Service Level Agreements)/QoS(Quality of Service)와 상태에이전트로부터 전달받은 정보를 분석하여 공유자원 또는 제어정책을 변경하여 클래스기반 커널자원관리의 핵심에 반영한다. 커널에서는 워크로드 매니저 에이전트로부터 전달받은 설정을 바탕으로 자원공유와 클래스 분류를 수행한다. 프로세스의 자원사용정보와 분류된 클래스에 대한 정보는 커널모듈과 상태에이전트 사이의 단일 양방향 통신채널을 통하여 상태에이전트에 전달된다.

클래스기반 커널자원관리는 프로세스단위로 이루어진다. 자원모니터링은 분류엔진의 분류이벤트에 반응하여 동작한다. 모니터링은 다음을 고려하여 설계되었으며 이를 특징으로 갖는다.

- 이벤트 동작방식 (Event-driven)
- 통신채널 (Communication Channel)
- 최소 커널상태 (Minimal Kernel State)

5.2.1 이벤트 동작방식

커널에서 태스크의 상태에 영향을 미치는 모든 종

요한 이벤트는 기록되고 상태에이전트에게 전달된다. 이러한 이벤트는 프로세스의 fork 그리고 exit, 재분류, 샘플링(Sampling) 등이다.

5.2.2 통신 채널

상태에이전트와 커널모듈사이에 한 개의 통신채널이 유지되며, 이를 통해 모든 명령과 자료가 전송된다. 대부분의 자료 흐름은 이벤트에 대한 결과를 커널공간으로부터 사용자공간으로의 전달하는 것이다.

5.2.3 최소 커널상태

클래스기반 커널자원관리의 자원 모니터링에서 커널이 유지해야할 프로세스단위 상태정보를 최소화하여 설계되었다. 대부분의 상태정보는 커널로부터 상태에이전트에게 전달되어 유지된다.

5.3 사용자 인터페이스

클래스기반 커널자원관리는 가상 파일시스템(Virtual File System)을 사용하여 구현된 자원관리 파일시스템을 통하여 클래스들을 관리한다. 이 파일시스템에서 태스크 클래스와 소켓 클래스는 각각 /rcfs/task_class와 /rcfs/socket_class의 디렉토리로 관리된다. 태스크 클래스와 소켓 클래스 내의 각 디렉토리는 클래스를 표현한다. 클래스를 표현하는 디렉토리 내의 파일들은 다음과 같은 기능을 한다.

- members: 어떤 프로세스가 해당 태스크 클래스 또는 소켓 클래스에 속하여 있는지 읽을 수 있다.
- config: 태스크 클래스 또는 소켓 클래스에 대한 설정을 읽고 쓸 수 있다.
- target: 태스크 클래스의 경우 pid 값을 이 파일에 쓰면 해당 pid를 가진 프로세스가 이 클래스로 분류된다. 소켓 클래스의 경우에는 IP 주소와 포트번호를 쓰면 해당 프로세스가 이 클래스로 분류된다.
- shares: 태스크 클래스 또는 소켓 클래스의 자원공유에 대한 설정을 읽고 쓸 수 있다.
- stats: 태스크 클래스 또는 소켓 클래스의 자원사용에 대한 통계정보를 읽을 수 있다. 이 파일에 대한 쓰기는 통계정보를 초기화한다.

디렉토리 /rcfs/ce는 분류엔진에 대한 사용자 인터페이스이다. 이 디렉토리는 다음과 같은 파일들을 포함한다.

- reclassify: 태스크 클래스의 경우에는 pid, 소켓 클래스의 경우에는 IP주소와 포트번호를 이 파일에 쓰면 해당 프로세스의 속성이 변경되었을 때 분류엔진이 재분류를 수행하도록 한다.
- state: 분류엔진의 작동을 중지시키거나 동작하도록 할 수 있다.
- Rules: 분류정책에 대한 설정을 읽고 쓸 수 있다.

Ⅵ. 결 론

지금까지 침입감내시스템의 개념과 특징, 유닉스, 리눅스의 자원관리기법에 대하여 살펴보았다. 운영체제의 동작에서 자원관리는 보안상 매우 중요한 요소이며, 보안운영체제 설계시 고려해야 한다. 보안운영체제가 제공해야 할 자원관리 기법은 각종 침해사고에 대하여 안정된 서버를 운영하기 위해 침입감내시스템의 요구사항인 가용성과 신뢰성, 그리고 보안성을 가져야하며, 시스템의 효율성을 높이는 동시에 각종 위협에 대해 안정된 서비스를 제공할 수 있어야 한다.

운영체제보안에 대한 연구가 진행됨에 따라 접근통제 프레임워크, 침입차단시스템, 방화벽 등의 많은 보안기술이 운영체제에 통합되었다. 결국, 안정된 서버를 운영하기 위해서는 이들 기술과 운영체제가 효율적으로 동작하도록 설계되어야 하며, 각종 보안기술과 운영체제의 자원관리기법의 통합에 대한 연구가 진행되어야 한다.

참 고 문 헌

[1] KISA, 기반보호기술 (침입감내기술), http://www.kisa.or.kr/ISTD/ISTD_m_01_02.html, 2004

[2] Sun Microsystems, Inc., Solaris 10 Resource Manager Developer's Guide, January 2005

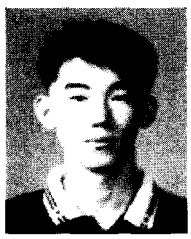
[3] Sun Microsystems, Inc., Resource Management in the Solaris 9 Operating Environment, September 2002

[4] Shailabh Nagar, Rik van Riel, Hubertus Franke, Chandra Seetharaman, Vivek Kashyap, Haoqiang Zheng, 'Improving Linux resource control using CKRM', July 21th-24th 2004

[5] Daniel P. Bovet & Marco Cesati, Un-

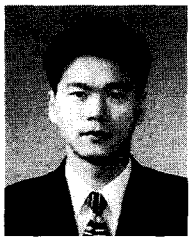
derstanding the Linux Kernel 2nd Edition, O'Reilly, September 2002
[6] Linux Programmer's Manual, Linux Man Pages - setrlimit, getrlimit, rusage, ulimit
[7] Mark Mitchell, Jeffrey Oldham, and Alex Samuel, Advanced Linux Programming, New Riders, June 2001
[8] Hubertus Franke, Shailabh Nagar, Chandra Seetharaman, Vivek Kashyap Rik van Riel, IBM Corp., RedHat., Advanced Workload Management Support for Linux, 2004
[9] Shailabh Nagar, Hubertus Franke, Jonghyuk Choi, Chandra Seetharaman, Scott Kaplan, Nivedita Shinhvi, Vivek Kashyap, Mike Kravetz, Class-based Prioritized Resource Control in Linux, IBM Corp., 2003
[10] Malicious and accidental-fault tolerance for internet applications, http://maftia.org. IST Programme RTD Research Project, 1999
[11] L. Blain and Y. Deswarte. An intrusion tolerant security server for an open distributed system. In 1st European Symposium in Computer Security, Toulouse, France, 1990
[12] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In 2002 IEEE International Conference on Dependable Systems and Networks (DSN'02), Washington D.C, USA, June 2002

〈著 者 紹 介〉



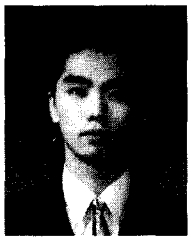
김 동 근 (Dong-geun Kim)

2003년 2월 : 전남대학교 컴퓨터정보학부 졸업
2005년 2월 : 전남대학교 대학원 정보보호협동과정 석사
2005 3월~현재 : 전남대학교 대학원 정보보호협동과정 박사과정
〈관심분야〉 보안운영체제, 침입탐지시스템



김 정 순 (Jung-sun Kim)

1998년 2월 : 전남대학교 전산학과 졸업
2000년 2월 : 전남대학교 전산학과 석사
2000년 3월~현재 : 전남대학교 전산학과 박사과정
〈관심분야〉 정보보호, 보안운영체제

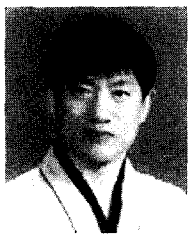


이 승 용 (Seung-yong Lee)

1995년 : 전남대학교 전산학과 졸업
1997년 : 전남대학교 전산통계학과 석사
2004년 : 전남대학교 정보보호협동과정 박사
1997년 1월~1998년 4월 : 금호정보통신연구소
1998년 5월~2002년 10월 : 핸디소프트 기술연구소
2005년~현재 : 전남대학교 객원교수
〈관심분야〉 보안운영체제, 유비쿼터스 컴퓨팅 보안

김 민 수 (Minsoo Kim)

1993년 2월 : 전남대학교 전산통계학과 졸업
1995년 2월 : 전남대학교 전산통계학과 석사
2000년 2월 : 전남대학교 전산통계학과 박사
2005년 3월~현재 : 목포대학교 정보보호학과 교수
〈관심분야〉 침입탐지시스템, 보안운영체제, 데이터마이닝



노 봉 남 (Bong-nam Noh)

1978년 2월 : 전남대학교 수학교육과 졸업
1982년 2월 : 한국과학기술원 전산학과 석사
1994년 2월 : 전북대학교 전산통계학과 박사
1983년~현재 전남대학교 전자컴퓨터정보통신공학부 교수
〈관심분야〉 컴퓨터 네트워크 보안, 사이버 사회와 윤리