LxBSM: C2 수준의 감사 자료 생성을 위한 리눅스 기반 동적 커널 모듈

(LxBSM: Loadable Kernel Module for the Creation of C2 Level Audit Data based on Linux)

전 상 훈 † 최 재 영 †† 김 세 환 ††† 심 원 태 ††††
(Sanghoon Jeun) (Jaeyoung Choi) (Sehwan Kim) (Wontae Sim)

요 약 현재 대부분의 상용 운영체제는 운영체제의 보안성을 높이기 위하여 높은 수준의 감사 기능을 제공한다. Linux의 성능 및 안정성은 기존 상용 운영체제에 뒤떨어지지 않지만, 감사 기능을 거의 제공하지 못하고 있다. Linux를 서버 운영체제로 사용하기 위해서는 C2 수준 이상의 보안성을 필요로 하며, 이를 만족시키기 위해서는 시스템 콜에 대한 감시와, 감사 이벤트가 요구된다. 본 논문의 LxBSM은 Linux 커널에서 C2 수준의 감사 기능을 제공하는 커널 모듈이다.

LxBSM은 SunShield BSM의 감사 자료와 호환되는 C2 수준의 감사 자료를 제공하며, 동적 커널 모듈(Loadable Kernel Module) 방식으로 구현되어 운용성을 높였다. 또한 사용자 프로세스에 대한 감사 자료를 생성함으로써, 기존의 Linux 기반 감사 모듈보다 풍부한 감사 자료를 제공한다. 파이프와 파일로 감사 자료의 출력이 가능하여 감사 자료를 활용하는 침입 탐지 시스템의 연계성을 높였다. LxBSM의 성능을 측정한 결과, fork, execve, open, close와 같이 감사 자료를 생성하는 시스템 콜이 호출될 때의 응답 시간은 지연되었으나, 그 외의 다른 성능 감소 현상은 나타나지 않았다.

키워드 : 리눅스 보안, 보안 LKM, 커널 감사시스템

Abstract Currently most of commercial operating systems contain a high-level audit feature to increase their own security level. Linux does not fall behind the other commercial operating systems in performance and stability, but Linux does not have a good audit feature. Linux is required to support a higher security feature than C2 level of the TCSEC in order to be used as a server operating system, which requires the kernel-level audit feature that provides the system call auditing feature and audit event. In this paper, we present LxBSM, which is a kernel module to provide the kernel-level audit features.

The audit record format of LxBSM is compatible with that of SunShield BSM. The LxBSM is implemented as a loadable kernel module, so it has the enhanced usability. It provides the rich audit records including the user-level audit events such as login/logout. It supports both the pipe and file interface for increasing the connectivity between LxBSM and intrusion detection systems (IDS). The performance of LxBSM is compared and evaluated with that of Linux kernel without the audit features. The response time was increased when the system calls were called to create the audit data, such as fork, execve, open, and close. However any other performance degradation was not observed.

Key words : Linux Security, Security LKM, Kernel Audit System

† 비 회 원 : 엠엠씨테크놀로지 연구원
shjeun@mmctech.com
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
choi@comp.ssu.ac.kr
††† 비 회 원 : 티맥스소프트 연구원
sehwan@tmax.co.kr
†††† 비 회 원 : 한국정보보호진흥원 팀장
wtsim@kisa.or.kr
논문접수 : 2003년 3월 21일
심사완료 : 2003년 12월 5일

1. 서 론

현재 운용되는 대부분의 상용 운영체제는 감사 기능을 포함한 C2 수준의 보안 기능을 제공한다. Linux는 91년 Linus Tovalds에 의하여 처음 개발된 이후 지난 10여년간 전세계의 해커들의 지원으로 폭발적으로 성장하였다. Linux는 운영체제로서의 안정성을 인정받으면서 시장 점유율도 크게 증가하고 있으며, 임의적 접근

제어(Discretionary Access Control)를 지원하여 C2 수준의 보안 성능을 제공하지만, 감사 기능은 C2 수준의 요구사항을 충족시키지 못한다[1]. 90년대 후반에 LinuxBSM, Snare와 같이 Linux 환경에서 C2 수준의 감사 모듈을 개발하려는 공개 프로젝트가 있었다. LinuxBSM은 시제품만 개발하고 개발이 중단되었으며[2], Snare는 C2 수준의 감사 기능을 거의 대부분 제공하지만, SunShield BSM 등 상용 감시 모듈보다 사용자 인증 및 식별에 대한 감사 자료가 부족하다[3].

C2 수준의 감사 모듈은 운영체제에서 발생하는 모든 사건을 기록할 수 있어야 한다. 따라서 감사 기능을 수행하는 모듈은 커널 내부에 존재한다. C2 수준의 감사 모듈은 시스템 호출에 대한 감사 기능과 사용자의 접속 과정, 그리고 특정 응용 프로그램에 대한 사건들을 기록할 수 있어야 한다[1,5]. 그러나 Linux 기반의 감사 모듈은 아직까지 그러한 기능을 제공하지 않는다[2,3,6,7]. Linux의 LKM(Loadable Kernel Module) 구조는 커널 모듈을 동적으로 관리할 수 있으므로 LKM 방식으로 커널 모듈을 구현하는 것이 개발 및 유지 보수에 유리하다[4].

본 논문에서 제안하는 LxBSM은 완전한 동적 커널 모듈 구조로 구성되어 기존 커널에 영향을 주지 않으면서, 커널 수준에서 감사 작업을 수행한다. LxBSM은 C2 수준을 충족시키며, SunShield BSM의 감사 자료 형식을 사용함으로써 감사 자료의 유용성을 높였다. 또한 다른 감사 모듈과 달리 커널에서 커널 감사 이벤트 뿐만 아니라 사용자 수준의 감사 이벤트도 제공한다. 그러나 감사 자료 생성을 필요로 하는 응용 프로그램에 감사 루틴을 포함시키기 위하여 커널을 변경할 필요가 없게함으로써 감사 모듈의 이식성을 높였다. LxBSM은 SunShield BSM의 감사 자료를 표현하기 위한 추가적인 정보가 필요하다. BSM Hash 큐를 통하여 Linux의 태스크 구조체를 변경하지 않으면서 감사 작업에 필요한 정보들을 프로세스 단위로 관리할 수 있다. 또한 버전 2.2대와 버전 2.4대 Linux 커널을 모두 지원하여, LxBSM의 활용 범위를 넓혔다. 마지막으로 LMBench와 LTP를 사용하여 LxBSM이 시스템의 성능 및 안정성에 미치는 영향을 검사하였다.

본 논문의 구성은 다음과 같다. 2장은 기존의 상용 운영체제의 감사 자료 생성 방법과 이전에 개발된 Linux 기반의 감사 자료 생성 모듈 등 관련 연구에 대하여 기술하며, 3장에서는 감사 모듈의 전체적인 구조에 대하여 LKM과 System Call Wrapping 기술을 중심으로 기술하였다. 4장에서는 SunShield BSM 감사 자료 형식의 구조와 본 감사 모듈에서 감사 자료를 생성하는 과정에 대하여 기술하였으며, 5장에서는 실시간으로 감사 자료를 연동하기 위한 감사모듈의 구조에 대하여 기술하였다. 6장에서는 본 감사 모듈의 성능 분석을 다루고 있으며, 마지막으로 7장은 결론으로 구성되었다.

2. 관련연구

C2 수준을 만족시키는 감사 모듈은 아래와 같은 기능이 요구된다. 우선 사용자 식별 및 인증 작업, 시스템 콜, 그리고 시스템 사용자 및 시스템 관리 작업에 대한 감사 이벤트를 생성할 수 있어야 한다. 또한 각각의 감사 이벤트는 감사 레코드로 기록되며, 각 감사 레코드는 이벤트가 발생한 시간, 사용자, 이벤트의 종류, 그리고 작업의 성공 및 실패에 대한 정보를 포함해야 한다. 식별 및 인증 이벤트들은 이벤트에 대한 기원이 되며, 모든 감사 레코드들은 자신의 기원에 대한 정보를 가지고 있어야 한다. 마지막으로 감사 모듈을 관리하는 시스템은 어떤 특정 사용자 혹은 그룹별로 선택적으로 감사 이벤트를 생성할 수 있도록 해야 한다[20].

Linux의 보안 수준은 상용 운영체제와 비교해도 뒤떨어지지 않는다. 임의적 접근 제어를 제공하여, C2 수준에 근접한 보안 성능을 제공한다. 그러나 Linux의 감사 기능은 C2 수준의 운영체제에서 요구되는 기능을 제공하지 못하고 있다. 그 이유는 감사 모듈로 사용되는 syslog가 네크워크에 관련된 감사 자료를 충분히 제공하지만, 시스템 콜에 대한 감사 자료를 생성하지 못하며, 또한 syslog는 감사 이벤트 형식을 제공하지 않기 때문이다[21]. 따라서, Linux에 보다 높은 수준의 보안 능력을 제공하기 위한 연구가 많이 시도되었다.

미국 NSA(National Security Agency)는 Linux에 강제적 접근 제어(Mandatory Access Control) 기능을 추가한 seLinux(Security Enhanced Linux)를 개발하였으며, Zie Huagang과 Phillippe Biondir는 커널 수준에서 강제적 접근 제어와 포트 스캐너, 파일 및 프로세스 접근 방지 등의 작업을 수행하는 LIDS를 개발하였다[6,7]. 상용 제품으로는 NPS MLS(Multi-Level Security) Linux, WireX의 Immunix, CA사의 eTrust Access Control 등의 사례가 있다. MLS Linux는 다중 등급 보안 체계를 통한 접근 제어에 초점이 맞추어져 있으며, WireX의 Immunix는 응용 수준의 침입 탐지 작업을 수행하는 보안 도구들을 모아놓은 배포판이다. 또한 eTrust Access Control은 가장 우수한 기능을 제공하지만 Linux보다는 상용 유닉스 시스템을 지원하는데 초점이 맞춰져 있으며, 최근 많이 사용되는 2.4 버전의 커널을 지원하지 않는다[6,22,23,24]. 순수하게 C2 수준의 감사 자료 생성만을 위한 개발 사례로 LinuxBSM과 Snare가 있다[2,3].

seLinux, LIDS는 커널 코드를 직접 수정하는 방식으

로 Linux의 보안 성능을 향상시켰다. 이와 같은 커널을 직접 수정하는 방식은 개발하기가 용이하지만, Linux와 같이 빈번하게 버전의 변경이 이루어지는 경우는 변경이 이루어질 때마다 매번 새로운 커널에 해당 코드를 이식하는 작업이 발생하는 문제점이 있다. 반면, 감사 자료의 생성만을 고려하여 개발된 LinuxBSM과 Snare는 동적 커널 모듈(LKM: Loadable Kernel Module) 형태로 구현되어 커널의 동작 중에도 감사 모듈을 커널에 로딩하여 사용하거나 제거할 수 있다. 그러나 두 시스템 모두 감사 이벤트는 사용하지만 감사 자료의 호환성은 없다[2,3,8]. LinuxBSM은 시제품으로 execve 시스템 콜에 대한 감사 이벤트만을 제공한다[2]. Snare는 주요 시스템 콜에 대해서 감사 이벤트를 생성할 수 있지만, login, logout과 같은 사용자 식별 및 인증에 대한 감사 이벤트와 Network과 관련된 감사 이벤트를 생성하지 못하기 때문에 C2 수준에 미치지 못한다[3].

현재 Linux에 기반한 호스트 기반 침입 탐지 시스템은 커널의 구조적 변경없이 버퍼 오버플로우와 같은 침입 행위를 탐지할 수 없다[9,12,13]. 상당수의 Linux를 기반의 침입 탐지 시스템은 커널 패치 방식으로 자신이 필요로 하는 감사 자료를 얻는다[7,9]. 대부분의 상용 운영체제들은 - SunShield BSM(Basic Security Module)과 IBM MVS의 RACF를 비롯하여, HP-UX, IRIX, Windows NT, Digital Unix, Ultrix, AS-400 - C2 수준 이상의 보안 수준을 만족시키기 위한 커널 수준의 감사 자료 생성 모듈을 제공한다[1,11]. 현재 운용되는 거의 대부분의 호스트 기반 침입 탐지 시스템은 운영체제가 제공하는 감사 모듈이 생성하는 자료 기반으로 하므로 Linux의 보안성을 높이기 위해서는 효율적인 감사 모듈이 필요하다.

3. LxBSM 감사 자료 생성 모듈의 구조

LxBSM은 드라이버, 랩퍼(Wrapper), 그리고 감사 데몬(Audit Daemon)으로 구성되어 있다. 실제 감사 자료의 생성은 랩퍼에서 이루어지며, 드라이버는 생성된 감사 자료를 감사 데몬을 통해 사용자 영역으로 전송한다. 감사 데몬은 드라이버로부터 받은 감사 자료를 LxBSM의 설정에 따라 감사 파일로 생성하거나 파이프로 보내어 다른 시스템이 커널에서 생성한 감사 자료를 활용할 수 있도록 한다. 그림 1은 각 모듈의 관계를 나타낸다.

랩퍼는 시스템 콜 함수를 가로채어 감사 이벤트를 생성한다. 랩퍼는 원래의 시스템 콜 대신 이를 대체하는 함수를 호출하면서 시스템 콜을 호출한 프로세스에 대한 감사 자료를 수집하고, 수집된 자료는 드라이버 모듈로 전송한다. 드라이버는 랩퍼에서 수집된 감사 자료를 버퍼에 수집하며 버퍼에 대한 인터페이스를 제공한다. 감사 자료가 수집되면 감사 데몬이 처리할 수 있도록

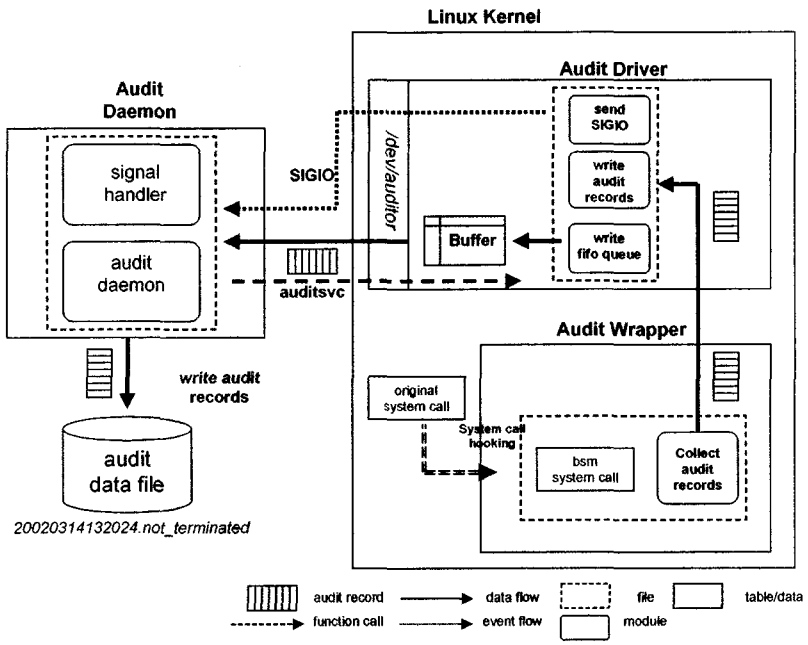

그림 1 감사 자료 생성 과정

신호를 보낸다. 신호를 받은 감사 데몬은 드라이버의 공유 버퍼로부터 감사 자료를 읽어 온다. 감사 데몬은 LxBSM의 설정에 따라 자료를 감사 파일로 남기거나 파이프에 남긴다.

랩퍼는 실제 커널에서 감사 자료의 수집을 담당하며 Linux 커널의 원래 시스템 콜의 내부 함수를 가로채서 감사 자료를 생성할 수 있는 함수로 대체한다. Linux 커널에서 제공하는 시스템 콜의 내부 함수들은 sys_call_tables 배열에 등록되어 있다. execve 시스템 콜의 내부 함수 sys_execve는 sys_call_tables 배열의 __NR_execve 매크로로 지정된 위치에 등록되어 있다. 커널에서 제공하는 모든 시스템 호출은 __NR_syscall_name의 형식의 매크로로 sys_call_tables 배열에서 위치가 정해져 있다[4]. 따라서 해당 위치에 동일한 인터페이스를 가지는 다른 함수를 등록시켜 원래의 시스템 호출을 대체할 수 있다[4,8].

드라이버는 문자 디바이스 드라이버 형태로 구현되었다. 문자 디바이스 드라이버는 키보드, 모뎀, 사운드 카드 등 입출력 데이타의 크기가 고정되어 있지 않은 형태의 디바이스 드라이버에 사용된다[14]. 랩퍼는 감사 자료를 수집한 후에 드라이버에서 제공하는 audit_event 함수를 호출한다. audit_event 함수는 랩퍼가 수집한 자료를 드라이버의 버퍼로 복사한다. LxBSM의 동작 모드가 파일 모드로 설정되면 버퍼가 가득찼을 때 신호를 발생시키고, 실시간 모드로 설정되면 감사 이벤트가 발생할 때마다 신호를 발생시킨다. 신호를 받은 감사 데몬은 mmap 시스템 콜을 사용하여 버퍼의 데이타를 감사 데몬으로 복사한다.

감사 데몬은 드라이버가 감사 자료를 받을 때마다 발생시키는 신호에 대한 처리기를 가지고 있다. 감사 데몬이 새로운 감사 자료에 대한 신호를 드라이버로부터 받으면 감사 데몬은 mmap으로 버퍼의 감사 자료를 가져온다. 데몬은 커널 영역에 존재하는 드라이버의 버퍼 영역에 직접 접근할 수 없다. 파일 모드로 동작할 때는 드라이버가 맵핑 영역을 할당함으로써 감사 데몬이 직접 버퍼에 접근할 수 있도록 하였으며, 실시간 모드로 동작할 때에는 auditsvc 시스템 콜을 호출하여 파이프에 감사 자료를 전송한다. 데몬 모듈은 드라이버 모듈과 랩퍼 모듈을 안정적으로 동작할 수 있게 조절한다.

4. 감사 자료 생성

Solaris의 감사 시스템인 SunShield BSM은 C2 수준의 감사 자료를 제공한다. LxBSM은 시스템 콜의 호출 기록, 사용자 식별 및 인증, 그리고 시스템 관리에 대한 감사 자료를 제공하므로써, C2 수준을 만족시키며, SunShield BSM의 감사 자료 형식을 사용한다. SunShield BSM의 감사 자료는 커널 이벤트와 사용자 이벤트로 구분된다. 커널 이벤트는 시스템 콜이 호출될 때 발생되며, 사용자 이벤트는 응용 프로그램에서 발생된다[5]. SunShield BSM은 동일한 이벤트에 대해서 LinuxBSM이나 Snare보다 상세한 자료를 제공하기 때문에 호스트 기반 침입 탐지 시스템에 적합하다. 또한 SunShield BSM을 바탕으로 개발된 호스트 기반 침입 탐지 시스템을 Linux로 쉽게 이식할 수 있다[15].

4.1 랩퍼 모듈

각 시스템 콜의 감사 자료는 랩퍼의 시스템 콜 랩핑(wrapping) 함수에서 생성된다. 모든 시스템 콜 랩핑 함수는 감사 작업 중간에 다른 작업으로 문맥 전환이 되지 않도록 임계 구간을 설정한다. 그리고 현재 시스템 호출의 감사 이벤트를 생성해야 하는지를 판단하기 위하여 비트 마스크를 사용한다. 감사 작업이 진행되면 header 토큰을 저장할 수 있는 필드를 생성하고 초기값들을 저장한다. 그러나 전체 감사 자료의 크기는 다른 토큰들이 모두 생성되어야 알 수 있으므로 비워두고 return 토큰을 제외한 나머지 감사 토큰들을 생성한다.

그리고 sys_call_table 배열의 주소를 가로채기 전에 저장한 원래의 시스템 콜의 내부 함수의 포인터를 이용하여 원래 시스템 콜을 호출한다. 이와 같은 방법으로 대부분의 시스템 콜을 처리할 수 있지만 execve는 메모리 관리 문제로 내부 함수 do_execve를 감사 모듈에서 직접 호출한다. 내부 함수의 반환 값을 이용하여 return 토큰을 생성하고, 생성된 감사 자료의 총 길이를 header 토큰에 삽입한다. 이렇게 생성된 감사 레코드는 audit_event 함수를 호출하여 드라이버의 버퍼로 전송한다.

4.2 BSM Hash Queue

현재 Linux의 태스크 구조는 SunShield BSM에 정의된 감사 자료를 생성하기에 적합하지 않다. 예를 들어 사용자가 접속한 IP 주소, 특정 프로세스의 로그인 아이디 등을 추적하기 위한 auid(Audit Identifier)를 각 프로세스마다 가지고 있어야 한다. 그러나 Linux의 태스크 구조체는 이와 같은 정보들을 제공하지 못한다[4,5]. Linux의 태스크 구조체에서 제공하지 못하는 부족한 정보들을 제공하기 위하여 아래와 같이 BSM Hash Queue(bsm_queue)를 설계·구현하였다.

BSM Hash Queue의 각 노드들은 프로세스 식별자, 감사 식별자(audit id, auid), 사용자가 접속한 터미널의 IP 주소, 사용자 이벤트를 위한 플래그 변수, setuid count 등의 정보를 가지고 있다. BSM Hash Queue는 Linux pidhash와 동일한 구조를 가진다. 따라서 BSM Hash Queue에 속한 노드에 접근하는 속도는 pidhash 노드의 접근 속도와 동일하다. BSM Hash Queue에 포함된 각 노드들은 새로운 프로세스가 생성될 때마다 만

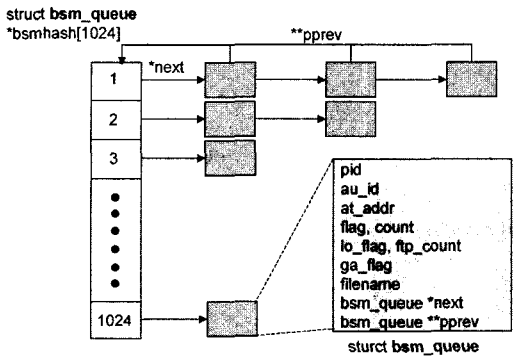

그림 2 BSM Hash Queue의 구조

들어진다.

fork 시스템 콜 랩퍼 함수에서 감사 자료가 모두 생성된 다음에 프로세스가 정상적으로 생성되었다면, 새로운 프로세스 노드를 생성하고 현재 동작중인 프로세스에 대한 포인터인 current를 이용하여 fork를 호출한 프로세스의 이름을 확인한다. fork를 호출한 프로세스가 login과 ftpd인 경우에는 BSM Hash Queue의 노드에 속한 flag를 설정한다. login과 ftpd에 대한 flag는 login과 ftp에 관련된 이벤트를 생성할 때 사용된다. SunShield BSM은 login, ftpd, telnetd, rlogind 등의 응용 프로그램에서 직접 감사 자료를 생성할 수 있도록 감사 자료 생성을 위한 API를 제공하고 있으며, 각 응용 프로그램은 감사 시스템이 활성화되면 감사 자료를 생성한다. 그러나 Linux에서는 배포판마다 포함되어 있는 응용 프로그램이 다르기 때문에 응용 프로그램의 코드를 수정하지 않고, LxBSM에서 직접 처리하도록 하였으며 데비안과 레드햇 6.2, 7.0, 7.2에 적용되었다.

4.3 login/logout 이벤트

C2 수준의 감사 작업을 진행하기 위해서는 사용자에 대한 추적 기능이 필요하다. 기존의 LinuxBSM, Snare는 동적 모듈 구조로 이루어져 있지만, Linux의 태스크 구조체에서 직접 얻을 수 있는 정보만으로 감사 자료를 생성하였다. 따라서 SunShield BSM에 비하여 표현할 수 있는 감사 이벤트의 종류가 부족하다. console, telnet, ftp, rlogin 등의 login 관련 이벤트들은 해당 응용 프로그램에서 감사 자료 생성 루틴을 제공해야 하지만, LinuxBSM과 Snare는 커널 영역에서 시스템 콜만 감사하는 모듈이다. 따라서 사용자 영역에서 생성되는 login, logout과 관련된 감사 자료를 제공하지 못한다. LxBSM은 각 프로그램이 커널에 요청하는 서비스를 BSM Hash Queue를 바탕으로 사용자 프로세스의 이벤트에 관련된 감사 자료를 생성한다.

그림 3은 사용자가 텔넷(telnet)을 이용하여 로그인하는 과정을 나타낸다. 외부 사용자의 요구가 inetd에 발생하면 in.telnetd가 구동되면서 pseudo slave 터미널을 생성하여 사용자 암호를 검사한다. 암호 검사가 완료되면 login shell을 생성한다. 이때 시스템이 PAM(Personal Authentication Module)을 사용하면 pam.d를 호출하고, 그렇지 않으면 바로 사용자 쉘 프로세스를 생성함으로써 로그인 과정을 마친다[16]. 각 단계가 진행될 때마다 fork/exec 시스템 콜을 호출하므로, fork 랩퍼에서는 현재 프로세스가 login인지를 검사하여 생성된 BSM Hash Queue 노드에 login 플래그를 설정한다. login 프로세스에서 패스워드 검사가 완료되고 setuid시스템 콜이 성공적으로 수행되어야 로그인 과정이 끝나게 되므로 login과 관련된 감사 자료는 setuid 시스템 콜 랩퍼 함수에서 생성된다.

setuid 시스템 콜의 랩퍼 함수 lxbsm_setuid에서 원래 setuid 시스템 콜의 수행이 완료되면 현재 수행중인 프로세스가 login 인지를 검사한다. 수행중인 프로세스가 login이면 감사 식별자를 설정한다. 사용자가 로그인

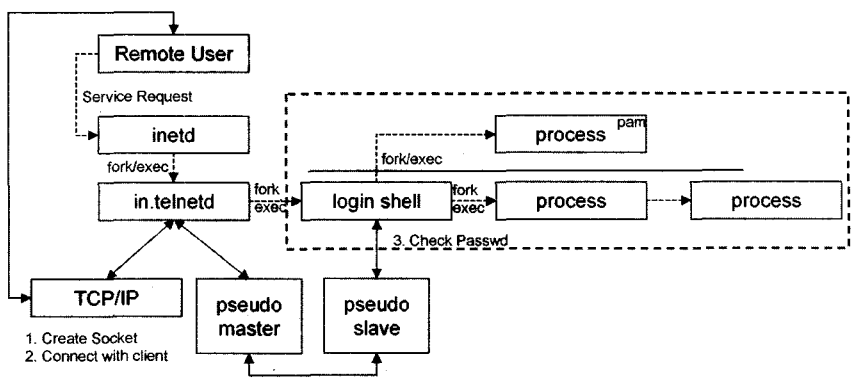

그림 3 텔넷의 로그인 처리 과정

할 때 얻는 사용자 식별자(user id)가 감사 식별자가 되며 해당 프로세스가 setuid 시스템 호출을 통해 식별자가 바뀌어도 감사 식별자는 변하지 않는다. 그림 3에서 보면 login에서 fork를 호출하는 것을 알 수 있다. login 프로세스에서 fork 시스템 콜이 호출될 경우, fork 랩퍼는 flag를 설정하고 setuid 랩퍼에서 flag가 설정되어 있는지를 검사한다. flag가 설정되어 있으면 부모 프로세스가 in.telnetd 혹은 in.rlogind인지를 검사하고, 여기에 해당되지 않는다면 콘솔 로그인으로 간주하여 각 이벤트를 AUE_telnet 또는 AUE_rlogin, AUE_login으로 정한다. 그리고 각 이벤트 토큰들을 생성하여 드라이버로 보낸다. setuid에서 login 이외에도 su인지를 검사하여, AUE_su 이벤트도 함께 생성한다. setuid의 경우 동일한 프로세스에서 여러 번 반복 호출되며, BSM Hash Queue 노드는 이를 위한 count 변수를 가지고, setuid 랩퍼는 count 변수에 설정된 값에 따라서 감사 토큰의 생성 여부를 결정한다.

4.4 lxbsm_sys_socketcall

Linux에서 네트워크 인터페이스는 socket 함수를 이용한다. Linux socket 함수의 커널 내부 함수 sys_socketcall은 lxbsm_sys_socketcall 함수로 대체된다. lxbsm_sys_socketcall 함수는 네트워크 이벤트에 대하여 감사 자료를 생성하고, 원격으로 접속한 사용자를 추적하기 위하여 IP 주소를 BSM Hash Queue에 저장한다. 외부에서 telnet, rlogin, ftp 등을 통하여 접근한 사용자를 추적하기 위해서는 해당 사용자의 IP 주소가 필요하다. 그러나 Linux 커널에서 pid를 이용하여 해당 사용자의 IP 주소를 얻기 어렵기 때문에 lxbsm_sys_socketcall에서 얻은 IP 주소를 BSM Hash Queue의 노드에 각 프로세스마다 저장한다

*int sys_socketcall(int call, unsigned long *args);*

sys_socketcall 함수는 위와 같은 형식을 가진다. 각 socket 함수들은 call 변수 값으로 구분된다. call 변수에는 SYS_SOCKET, SYS_ACCEPT와 같이 미리 지정된 값이 들어간다. 예를 들어 socket() 시스템 콜이 사용자 수준에서 호출되면 SYS_SOCKET이 sys_socketcall 함수의 call 변수에 설정된다. sys_socketcall에서는 socket 시스템 콜의 기능을 수행한다. lxbsm_sys_socket- call 함수는 우선 AUE_ACCEPT 이벤트만 생성한다. AUE_ACCEPT는 accept에 대한 감사 이벤트이다. 외부에서 해당 컴퓨터에 접근하기 위해서는 반드시 accept를 호출하므로, 침입 탐지 시스템이 공격을 탐지하기 위해서는 accept 시스템 콜의 감사 이벤트만으로도 충분하기 때문이다[15].

커널에서 socket 관련 자료는 socket 버퍼를 통해 따로 관리되며 socket 버퍼는 i-node 테이블을 통해서 접근할 수 있다. sys_socketcall 함수의 리턴 값이 해당 socket 버퍼의 i-node의 파일 지시자(fd: File Descriptor)로 사용된다[17]. 물론 리턴 값은 sys_socketcall 함수를 성공적으로 수행하였을 때만 얻을 수 있다. socket 버퍼로부터 IP 주소, port 정보를 얻은 후 감사 자료를 생성한다. AUE_ACCEPT는 파일 지시자에 따라서 다양한 결과를 나타낸다. 예를 들어 socket이 연결되지 않거나 주소 체계가 인터넷이 아닌 경우, 그리고 파일 지시자가 잘못된 경우 등에 따라 다른 구조의 감사 자료를 생성한다.

5. 실시간 감사 자료 연동

LxBSM은 침입 탐지 시스템이 즉시 침입 결과를 판단할 수 있도록 이벤트가 발생할 때마다 감사 자료를 전달하게 하였다. 커널에서 생성된 감사 자료를 파이프로 전송하기 위하여 감사 데몬에게 새로운 감사 자료가 생성되었다는 시그널을 발생시킨다. Linux의 시그널은 윈도우의 메시지와 달리 반복되는 시그널을 무시하고 바로 넘겨버리기 때문에, 빈번하게 시그널이 발생할 때에는 시그널의 처리가 지연되다가, 결국에는 발생한 시그널을 잃어버려서 감사 모듈에서 생성된 감사 자료가 손실되는 현상이 발생한다. 이와 같은 문제를 해결하기 위하여 LxBSM은 시그널의 발생 빈도에 상관없이 한번만 시그널을 받으면 감사 데몬이 드라이버 버퍼에 존재하는 모든 감사 이벤트를 파이프에 전송하도록 처리하였다.

그림 4는 LxBSM의 드라이버가 감사 자료를 처리하는 구조를 나타낸다. LxBSM은 감사 파일 모드와 실시간 모드를 제공하며, 감사 파일에 감사 자료를 전송할 때는 mmap 시스템 콜을 통하여 감사 버퍼에 감사 데몬이 접근하여 감사 자료를 감사 파일에 남긴다. 실시간 모드에서는 감사 자료를 파이프로 전송한다. 실시간으로 감사 자료를 전송하기 위해서 auditsvc 시스템 콜을 만들었다. auditsvc 시스템 콜은 감사 데몬이 시그널을 받으면 호출되어 감사 버퍼에 남아있는 모든 지연된 이벤트를 파이프로 전송함으로써 지연된 이벤트의 손실없이 감사 자료를 실시간으로 다른 프로세스에 전달한다. LxBSM의 동작 모드는 모듈이 동작할 때 audit_control 설정 파일에 정의된 모드에 따라 결정된다.

6. 성능 측정

본 감사 모듈의 성능 측정은 Linux 커널 2.4.9와 2.2.18에서 이루어졌다. 성능 측정에 이용된 시스템은 인텔 펜티엄 III-600Mhz, 메모리는 387MB를 가지며, Hitachi Travelstart 40GB IDE 하드 디스크를 ATA

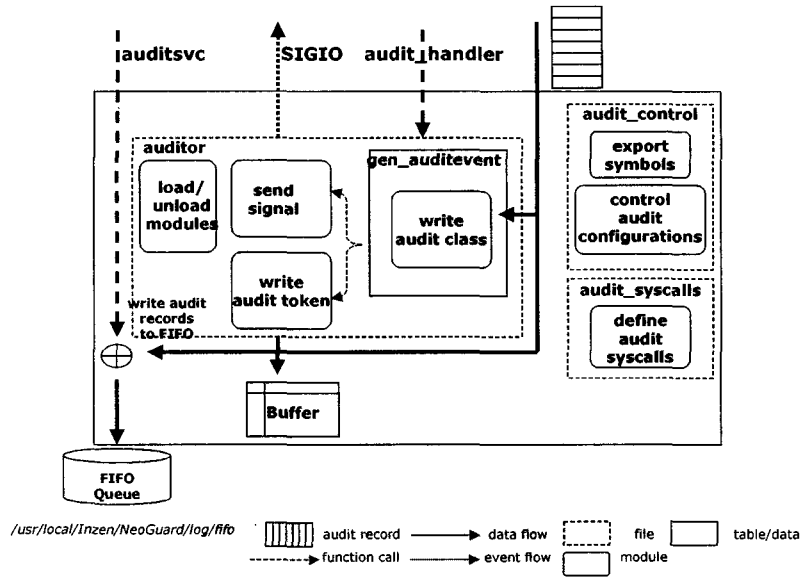

그림 4 감사 드라이버 모듈의 전송 구조

100 모드로 사용하였다. 성능 측정 도구로 LMBench 2.0.4을 사용하였다[18]. 표 1과 2는 각 커널을 감사 모듈없이 동작시킬 경우(Normal)와 실시간 모드(FIFO)와 파일 모드(FILE)로 동작시킬 경우로 나누어 성능을 측정하였으며, 각 표의 항목은 LMBench의 검사 항목을 그대로 나타내었다. 각 표에서 진하게 나타난 부분이 LxBSM에서 직접 처리하는 시스템 콜 함수이다. 테스트 중 일부 항목에서 LxBSM을 사용하였을 때 더 좋은 결과치를 내는데, 이는 캐쉬 효과에 따른 오차이다. 대부분의 오버헤드는 각 토큰을 생성할 때 사용하는 가변 버퍼를 동적으로 할당하는 과정에서 발생하며, BSM Hash를 생성하고 불필요한 Hash 테이블 노드를 제거할 때 발생한다. 따라서 고정 버퍼방식으로 변경하면 성능 향상효과를 얻을 수 있을 것으로 기대된다. 각 표에서 진하게 표시된 부분이 LxBSM이 직접 영향을 미치는 부분이다. BSM Hash 테이블의 상태변화를 확인하기 위하여, 로그를 남기도록 하였기 때문에 특정 부분에서의 오버헤드가 상당히 높게 나타났으나, 로그 생성루틴을 제거하면 이러한 수치는 훨씬 줄어들 것으로 보인다.

표 1은 시스템 콜 함수 호출에 대한 처리 지연 시간의 변화를 보여준다. 표 1의 각 항목은 시스템 콜 호출에 따른 지연 시간을 나타낸다. 표 1에서 진하게 표시된 부분을 제외하면 미묘한 오버헤드가 발생하며, null I/O의 경우 FILE에서 오버헤드가 조금 더 발생한다. 이는 잦은 파일 입출력 작업으로 인한 오버헤드로 보인다. null call은 전반적인 시스템 호출 루틴에서의 지연시간을 나타낸다. 2.2.18에서만 약간 증가하였으나, 2.4.9에서는 변화가 없었다. 시스템 콜 핸들러가 2.4에서 더 최적화된 것으로 추정된다. 이와 같은 지연시간의 증가는 단위가 microseconds임을 감안한다면 전체 응답시간에 큰

표 1 시스템 호출 처리 지연 시간(단위: microseconds)

Linux 2.4.9	*null call*	*null I/O*	*Stat*	*open close*	*select TCP*	*sig inst*	*sig handle*	*fork proc*	*exec proc*	*sh proc*
Normal	0.5	0.75	2.64	[illegible]	26.33	1.52	4.3	[illegible]	793.5	[illegible]
FIFO	0.5	0.763	2.687	[illegible]	28.93	1.527	4.308	[illegible]	5367.2	52K
FILE	0.5	0.754	2.645	[illegible]	27.29	1.524	4.303	[illegible]	2236	52.5K

Linux 2.2.18	*null call*	*null I/O*	*stat*	*open close*	*select TCP*	*sig inst*	*sig handle*	*fork proc*	*exec proc*	*sh proc*
Normal	0.49	0.71	2.82	[illegible]	26.6	1.5	1.85	[illegible]	687	29K
FIFO	0.52	0.761	2.996	[illegible]	27.59	1.534	1.849	[illegible]	2065.3	59K
FILE	0.522	0.784	2.876	[illegible]	25.88	1.572	1.858	[illegible]	1728	55K

표 2 지역 통신 지연 시간(단위: microseconds)

Linux 2.4.9	PIPE	AF UNIX	UDP	RPC/UDP	TCP	RPC/TCP	TCP Conn
Normal	6.364	10.95	20.4	49.325	30.23	65.775	105.5
FIFO	6.575	11.8	21.82	50.6	30.55	72.1	413
FILE	6.528	11.04	21.81	50.475	30.53	66.3	156.38

Linux 2.2.18	PIPE	AF UNIX	UDP	RPC/UDP	TCP	RPC/TCP	TCP Conn
Normal	5.54	11.25	18.65	41.15	28.1	58.6	95.5
FIFO	5.611	11.44	19.81	42.486	28.44	61.786	353.71
FILE	5.59	11.92	19.7	43.24	28.1	59.32	149.75

표 3 문맥 교환 지연 시간 (단위: number of process/in cache size, microseconds)

Linux 2.4.9	2P/0K ctxsw	2P/16K ctxsw	2P/64K ctxsw	8P/16K ctxsw	8P/64K ctxsw	16P/16K Ctxsw	16P/64K ctxsw
Normal	1.295	5.815	20.33	9.26	189	38.075	190.8
FIFO	1.3	5.923	24.55	11.58	191	39.317	191.67
FILE	1.349	5.931	21.74	9.433	190	39.95	191.36

영향을 미치지 않음을 알 수 있다.

실제 감사작업을 수행하는 시스템 호출인 open/close, fork, exec, sh 항목은 지연 시간이 크게 증가되었다. 특히 FIFO 모드에서 지연시간 증가치가 두드러지는데, FIFO 모드에서 매번 감사 이벤트가 발생할 때마다 파이프에 대하여 읽기/쓰기 작업이 발생하기 때문에 시그널 처리를 위한 문맥 교환 횟수가 증가하여 FILE 모드보다 지연 시간이 크게 증가하였다. 2.2.18에서 보면 시그널 핸들러에 대한 지연시간이 FIFO에서 아주 미묘하게 감소하였는데, 이것은 FIFO 모드에서 동일한 시그널 핸들러를 자주 호출함으로써, 캐쉬 효율이 좋아져서 이와 같은 결과를 나타낸 것으로 보인다.

표 2는 지역 통신 지연 시간을 나타낸다. 전반적으로 2.4.9가 2.2.18보다 약간 더 좋은 낮은 지연시간을 가지는 것을 알 수 있다. LxBSM에 동작하는 부분은 accept 항목밖에 없음에도 불구하고, 다른 항목에서도 미묘한 변화가 감지되었다. FIFO에서 약간 더 오버헤드가 더 발생하였으며, 실제 accept 시스템 호출이 영향을 미치는 TCP Connection 항목에서 오버헤드 증가가 두드러진다. FIFO 모드에서 나타난 오버헤드가 2.4.9, 2.2.18 모두 FILE 모드보다 훨씬 높다. 이와 같은 비율은 진하게 표시된 부분에서 대부분 나타나는 수치로서, 리눅스에서 시그널을 통해서 대용량의 이벤트를 전달하는 것이 효과적이 못하다는 것을 보여준다.

표 3은 문맥교환에 따른 지연시간 증가에 대한 오버헤드를 보여준다. LxBSM을 사용하였을 때 약간의 오버헤드가 발생함을 알 수 있다. 캐쉬의 크기가 64K 이하인 경우에는 FILE 모드에서 오버헤드가 약간 더 높게 나타난다. 메모리 처리 지연시간보다 FILE 모드에서 잦은 FILE 입출력 작업이 영향을 준 것으로 보인다.

마지막으로 시스템 콜을 가로채는 경우 커널의 원래의 시스템 콜 루틴에서 수행하는 에러 상황 검사 및 허가권 검사를 중복하여 수행해야 할 경우가 있다. LTP 도구를 사용하여 각 시스템 콜에게 비정상적인 인자를 넘겨주고 이에 대해 적절한 에러 코드와 결과 값을 되돌려주는지를 확인할 수 있다[19]. 예를 들어 execve의 인자 중 filename에 -1이나 프로세스 영역 밖의 값이 넘어갈 경우 EFAULT를 에러 값으로 되돌려주어야 되는데, 랩퍼 함수에서 이를 검사하지 않으면 시스템에서 Oops 메세지가 발생한다[4]. 비정상적인 인자를 넘겨받은 LxBSM의 여러 랩퍼 루틴들에서 이에 대한 처리 루틴을 추가하여 정상적인 결과 값과 에러 코드 설정하게 하였다. 반면 Snare와 LinuxBSM은 비정상적인 인자에 대한 처리 루틴이 부족하여, LTP 검사를 통과하지 못한다.

7. 결론 및 향후 계획

본 논문에서 제안한 감사 모듈은 Linux에서 C2 수준의 보안 기능을 제공하기 위하여 개발되었다. 동적 커널 모듈로 개발되었기 때문에 커널의 버전의 변화에 종속적이지 않으며, 기존에 개발된 Linux 기반 감사 모듈과 비교하여 C2 수준의 감사 자료를 쉽게 생성할 수 있다. 또한 실시간 모드와 파일 모드를 분리함으로써 LxBSM을 사용하는 침입 탐지 시스템의 운용성을 높일 수 있다. BSMhash라는 SunShield BSM과 같은 상용 운영체제의 감사 모듈보다 훨씬 유연한 구조를 채택하여,

login/logout과 같은 사용자 프로세스가 생성하는 이벤트들도 커널내의 감사 모듈에서 생성할 수 있다. 또한 SunShield BSM의 감사 포맷과 호환되기 때문에 기존의 SunShield BSM 기반으로 개발된 호스트 기반 침입 탐지 시스템을 Linux에서 바로 활용할 수 있다. Lx-BSM은 상용 유닉스 시스템용으로 개발된 인젠의 Neo-Guard@ESM에 사용되어 Solaris용 호스트 기반 침입 탐지 시스템을 Linux 환경으로 이식하였다. 또한 커널 2.2, 2.4 모두 지원하여 기존의 다른 모듈에 비하여 활용성을 더욱 높였다.

앞으로 LxBSM의 성능을 향상시키기 위한 추가적인 연구가 요구된다. 특히 지연 시간에 관련된 문제가 잦은 시그널 처리로 문맥 교환이 증대하였기 때문이므로, 실시간 모드에서 시그널 발생을 최소로 줄이고 감사 자료를 전달하기 위한 연구가 필요하다. 또한 사용자 수준의 이벤트를 추적할 때 동일한 기능을 제공하더라도 Linux 배포판에 따라 다른 구조의 프로그램을 사용하기 때문에 커널에서 이를 확인하고 추적하기 어렵다. 이러한 문제를 해결하여야 하며, 마지막으로 보다 많은 이벤트에 대하여 감사 자료를 제공하기 위한 연구와 더불어, B1급 이상의 보안 커널에 요구되는 강제적 접근 제어 기능 또는 역할 기반 접근 제어 기능을 Linux 환경에서 제공하기 위한 연구를 추가적으로 수행할 예정이다. LxBSM의 BSMhash 구조는 동적 커널 모듈의 방식의 감사 시스템을 개발할 때 필요한 정보를 유지하는데 효과적으로 활용될 수 있으며, 앞으로 보다 향상된 접근 제어 모듈을 만드는데 활용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] Boran Consulting, *IT Security Cookbook - OS Overview,* http://www.boran.com/security/it15-os-overview.html, 2000.

[2] LinuxBSM, http://linuxbsm.sourceforge.net/, 2000.

[3] Snare, http://www.intersectalliance.com/projects/Snare/, 2001.

[4] D. P. Bovet, M. Cesati, *Understanding The Linux Kernel,* O'Reilly, 2001.

[5] Sun Microsystems, SunShield Basic Security Module Guide, Sun Microsystems, 1998.

[6] SELinux, http://www.nsa.gov/selinux/, 2001.

[7] Linux Intrusion Detection System(LIDS), http://www.lids.org/.

[8] Pragmatic/THC, (nearly) Complete Linux Loadable Kernel Modules, http://www.thehackerschoice.com/papers/LKM_HACKING.html, 1999.

[9] ImSafe, http://imsafe.sourceforge.net/.

[10] Counterpane Corporation, "Syslog Overview," http://www.counterpane.com/syslog-overview.pdf.

[11] M. Bishop, "A Standard Audit Trail Format," Dept. CS. UC. Davis, 1996.

[12] J. Xu, Z. Kalbarczyk, S. Patel and R. K. Iyer, "Architecture Support for Depending Against Buffer Overflow Attacks," http://www.crhc.uiuc.edu/~junxu/Papers/EASY_02_arch_support_stack_overflow.pdf.

[13] Kaladix Linux, http://www.kaladix.org/docs/information.shtml, 2001.

[14] A. Rubini, J. Corbet, *Linux Device Driver,* 2nd Edition, O'Reilly, 2001.

[15] Inzen Corporation, http://www.inzen.com/.

[16] W. R. Stevens, *Unix Network Programming,* 2nd Edition, Prentice Hall, 1998.

[17] J. Crowcroft, L. Philips, *TCP/IP and Linux Protocol Implementation,* Johns & Wiley, 2002.

[18] LMBench, http://www.bitmover.com/lmbench/, 1998.

[19] Linux Test Project, http://ltp.sourceforge.net/, 2002.

[20] "Department of Defense Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, 1985.

[21] U. Flegel, "Pseudonymizing Unix Log Files," http://ls6-www.informatik.uni-dortmund.de/issi/archive/literature/2002/, 2002.

[22] P. C. Clark, "Policy-Enhanced Linux," 23rd NISSC, 2000.

[23] Immunix, http://immunix.org/.

[24] Computer Associates, eTrust Access Control for UNIX, 2001.

전 상 훈

1998년 숭실대학교 컴퓨터학부(학사). 2000년 숭실대학교 컴퓨터학과(석사). 2003년 숭실대학교 컴퓨터학과 박사과정 수료 2003년~현재 (주)엠엠씨테크놀로지 주임연구원. 관심분야는 시스템 소프트웨어, 보안커널, 리눅스, 침입 대응

최 재 영

1984년 서울대학교 제어계측공학과(학사) 1986년 미국 남가주대학교 전기공학과(석사). 1991년 미국 코넬대학교 전기공학과(박사). 1992년~1994년 미국 국립 오크리지연구소 연구원. 1994년~1995년 미국 테네시 주립대학교 연구교수. 2001년~2002년 미국 국립 수퍼컴퓨팅 응용센터(NCSA) 초빙연구원. 1995년~현재 숭실대학교 정보과학대학 컴퓨터학부 부교수. 관심분야는 시스템 소프트웨어, 침입 복구 및 대응, 병렬/분산처리, 고성능컴퓨팅

김 세 환

1997년 한국과학기술원 전자계산학과 (학사). 2004년 서울대학교 컴퓨터공학부 (석사). 1997년~2000년 AIO Corp. 연구원. 2000년~2003년 (주)인젠 선임연구원. 2003년~현재 티멕스소프트(주) 책임연구원. 관심분야는 내장형 시스템, 시스템 소프트웨어, 보안커널, 리눅스

심 원 태

1987년 서울대학교 계산통계학과(학사) 1989년 한국과학기술원 전자계산학과(석사). 1989년~2000년 데이콤 개발팀장 2000년~2003년 (주)인젠 연구소장. 2003년~한국정보보호진흥원(KISA) 팀장. 관심분야는 보안 운영체제, 침입탐지, 침입대응