리눅스 커널 기반 사용자 키스트로크 로깅 모듈 설계 및 구현

정계옥**, 김정순*, 노봉남*

**전남대학교 정보보호협동과정, *전남대학교 컴퓨터정보학부

User Keystroke Logging Module Design and Implementation on the Linux Kernel

GyeOk Jung**, JungSoon Kim*, BongNam Noh*

**Department of Information Security, *Department of Computer Science Chonnam Univ.

요 약

일반적으로 시스템들은 관리자를 위한 많은 로깅 기능을 제공한다. 이러한 로깅 기능에는 사용자 행위를 파악하는 부분도 제공하고 있으나 정작 사용자들이 입력하는 명령어를 직접 로깅하는 기능은 없거나 매우 미약하다. 시스템 사용자들이 입력한 명령이는 시스템 자체에서 사용자가 어떤 행위를 하였는가를 가장 확실히 보여주는 중요한 단서이다. 본 논문에서는 리눅스 커널을 기반으로 하여 사용자 키스트로크를 로그로 남길 수 있는 방법을 제안하고 구현한다.

I. 서론

유닉스 운영체제의 로그는 크게 커널 로그정보와 응용프로그램 로그정보로 분류되며 이 모든 로그들은 syslogd 라고 하는 응용프로그램에 의해 정리되고 기록되어진다[1]. 기존의 로그 정보는 단순한 시스템 모니터링을 위한 보조적인 역할정도만을 수행할 뿐, 시스템을 악의적으로 공격하는 공격자들에 대한 정보를 전혀 제공하지 못하는 문제점을 드러내고 있다. 이에 따라 매우 정확하고 자세한 로그의 필요성이 제기되었다. 그중에서 사용자 키스트로크의 저장은 매우 효과적인 정보이지만, 사생활 침해와 유닉스 시스템에서의 구현 어려움으로 사용되기 어렵다.

리눅스의 커널구조는 유닉스와 달리 매우 유연한 모듈구조를 지원하고 있어 응용프로그램 수준의 로그가 아닌 커널 수준의 로그를 제공할 수 있다. 커널수준의 로그는 응용프로그램 수준의 로그와는 달리 관리자가 원하는 기능을 자유롭게 구현할 수 있고, 임의의 공격자가 발견하기 쉽지 않은 것이 장점이다[2].

HoneyPot 시스템을 구축하는 경우에는 사용자 키보드 입력을 수집하여 데이터베이스화하는 것이 필수적인 요소이고, 이상탐지(Anomaly Detection)를 수행하는 IDS(Intrusion Detection System)를 운용하고자 할 때, 정상적인 사용자 명령어들의 패턴을 데이터베이스로 구축하는 과정에서도 사용자 키스트로크 로그정보는 각 사용자들의 특성을 가장 잘 나타내 주기 때문에 필수불가결하다.

본 논문에서는 리눅스 커널에 모듈로서 사용자들의 키스트로크를 가로채어 저장하는 방법을 제안하고 이를 구현하여 정상적인 관리자 도구로써 운영하는 것을 보이고자 한다.

본 논문의 구성은 2장에서 기존의 리눅스 로깅 데몬을 소개하고 그 문제점을 지적한 뒤, 3장에서 본 논문이 제안하는 리눅스 커널 기반 사용자 키스트로크 로깅 모듈의 설계 및 구현을 설명한다. 4장에서는 결론과 더불어 앞으로의 향후 연구 계획을 기술한다.

II. 관련연구

1. 리눅스 로그 데몬

그림 1에서와 같이 리눅스에서 로깅을 수행하는 데몬은 크게 klogd와 syslogd 두 가지로 분류된다. 두 데몬의 로그는 동일한 syslogd.conf 라는 설정파일에 따라, 여러 파일로 나뉘어 저장된다. 일반적인 경우, 대부분의 커널메시지들은 messages 파일에 기록되며, 응용프로그램 로그정보는 종류에 따라 maillog, secure, message, boot.log, lastlog, xferlog, wtmp 등으로 기록된다[1].

리눅스의 로그파일 중에서 사용자들에 대한 정보를 남기는 로그파일에는 wtmp, lastlog 가 있다. wtmp는 who 라는 명령어를 사용해서 볼 수 있는 바이너리 형식의 파일로써, 현재 로그인한 사용자 정보를 담고 있다. *lastlog* 파일은 *last* 라는 명령어를 사용해서 볼 수 있는 바이너리 형식의 파일로써, 사용자별 로그인, 로그아웃 정보를

볼 수 있다[1].

2. 기존의 사용자 명령어 로깅 방법

기존의 유닉스 시스템은 프로세스 accounting이라고 하여, 모든 사용자에 의해 실행된 모든 단일 명령어를 acct 또는 pacct 파일에 기록으로 남긴다. acct 및 pacct 파일은 사용자가 직접 읽을 수 없는 이진 형태로 사용자가 수행한 단일명령어의 정보를 기록하고 있으므로, 사용자는 lastcomm이나 acctcom 명령어에 사용하여 명령어의 정보를 확인 가능하다. 그러나 acct/pacct 파일은 사용자별로 사용한 명령어를 구분하는데 유용하게 사용될 수 있지만, 사용된 명령어의 argument와 그 명령어가 언제 시스템 내 어느 파일시스템의 어느 경로상에서 실행되었는지는 기록하지 않는다[3].

또한, Shell의 history기능이 사용자가 입력한 명령어들의 최근순서를 저장하고 있지만, 사용자 스스로 내용을 조작할 수도 있고, 이 역시 언제 명령어가 실행되었는지 기록하지 않는다. Shell의 history기능은 로그라기보다는 사용자의 키 입력 편의를 위한 보조 기능 성격이 강하다.

III. 모듈 설계 및 구현

1. 기본 개념

본 논문은 문자 디바이스 드라이버(Character Device Driver)인 키보드 드라이버를 중심으로 접근하고 있다. 그림 1은 리눅스 키보드 드라이버가 동작하는 방식을 나타내는 것이다.

먼저 사용자가 키보드상의 키를 누르면, 키보드는 키에 할당되어진 scancode를 키보드 드라이버에게 보낸다. 하나의 키 입력은 최대 6개까지의 scancode를 발생시킬 수 있다. 키보드 드라이버안에 있는 handle_scancode() 함수가 여러 개의 scancode들을 처리하여, keycode라고 불리는 키 누름, 키놓기의 동작상태들로 변경하여 준다.

키보드상의 각각의 키는 1에서 127까지의 유일한 keycode값 k를 가지며, 키를 누를 때 keycode 값 k가 발생하며, 키를 놓을 때 k+128의 keycode 값이 발생한다. 다음 keycode들은 적절한 키 맵상에 있는 키 심볼들로 변환된다. 키를 누를 때, 활성화된 변수와 여러 배타적인 잠금들이 사용될 키맵을 정한다. 위의 과정을 지나면, 확정된 문자들이 tty queue안의 tty_flip_buffer로 넣어진다. tty line discipline 안에서, receive_buf() 함수는 tty_flip_buffer로부터 문자들을 얻어오기 위해 주기적으로 호출된다. 그리고 그것들을 tty read queue로 넣는다. 사용자 프로세스가 사용자 입력을 요구하면, 프로세스의 stdin으로 read() 함수를 호출한다. sys_read() 함수는 입력된 문자들을 읽기 위해서 연관되어진 tty의 file_operations 구조체 안에 정의되어 있는 read() 함수(tty_read로 지정되어진)를 호출한다[4].

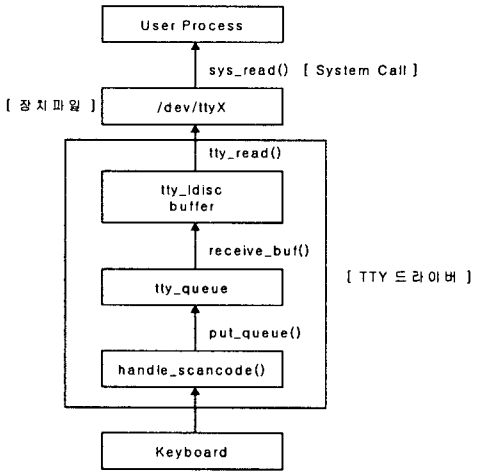

그림 1: 리눅스 키보드 드라이버의 동작 방식

2. 키입력을 가로채는 방법

리눅스 커널상에서 키입력을 가로채는 방법은 크게 다음 두 가지 범주로 분류된다.

1) 인터럽트 핸들러 수정

인텔 아키텍처하에서 키보드를 제어하는 IRQ번호는 1이다. 인터럽트 핸들리를 수정해 놓으면, 키보드 인터럽트가 생길 때, 수정된 인터럽트 핸들리는 scancode와 키보드 상태값을 읽는다. 인텔 아키텍처하에서 키보드 사건들은 키보드 데이터 레지스터인 port 0x60과 키보드 상태 레지스터인 0x64를 통해서 읽거나 쓰여진다. 그러나, 이 방식은 매우 시스템 의존적이다. 그래서, CPU나 운영체제마다 다른 코드로 작성해야 하며, 인터럽트를 다룰 때 매우 주의해야 한다. 그렇지 않으면 컴퓨터가 정지상태로 되어버린다.

2) 키보드 드라이버 함수 가로채기

앞의 그림 1을 참고로 하여, 함수 가로채기를 사용하여 사용자 키 입력을 로그할 수 있다. 가로챌 수 있는 함수는 handle_scancode(), put_queue(), receive_buf(), tty_read(), sys_read() 다섯가지이다.

3. 리눅스 커널 모듈 설계

키입력을 가로채기 위해서 인터럽트 핸들러를 사용하는 방법은 시스템 의존적이며, 잘못된 코드 작성시에는 시스템에 미치는 영향이 너무 크다.

그리고, handle_scancode() 함수와 put_queue() 함수를 가로채는 방법은 tty가 열려진 상태에 상관없이 콘솔이나 X상에서의 키스트로크도 로깅할 수 있다는 것과 Control, Alt, Shift, PrtSc 등의 키보드 상에 있는 특수한 키들을 모두 포함하여 정확히 어떤 키가 눌리졌는지 확실하게 알 수 있다는 장점을 가지는 대신에, 역시 시스템 의존적

이어서 플랫폼간의 호환이 되지 않는다. 또한, 원격 세션의 키스트로크는 로깅할 수 없는 매우 큰 단점과 다양한 기능을 확장시키기가 매우 어렵다는 문제점을 안고 있다[5].

마지막으로, sys_read/sys_write 시스템 콜을 가로채는 경우에는 시스템 전체에 너무 많은 부하를 주고 시스템을 불안정하게 한다. 왜냐하면, sys_read 와 sys_write는 매우 일반적이고 필수적인 read/write 함수이고, 이 함수들은 프로세서들이 읽거나 또는 쓰고자 할 때, 계속적으로 불려지는 함수이기 때문이다. tty_read 함수의 경우는 sys_read 함수와의 관련성이 높기 때문에 역시 사용하기가 쉽지 않다[6].

그러므로, 본 논문에서는 로컬 세션과 원격 세션을 모두 로그할 수 있고, 매우 쉽게 시스템콜을 가로챌 수가 있으며, 시스템 전체에 부하를 거의 주지 않는 receive_buf() 함수 가로채기 방법을 이용하여 모듈을 설계한다.

1) receive_buf() 함수 가로채기 모듈

그림 2는 본 논문이 제안하는 리눅스 커널 모듈의 구조도이며, 모듈은 네 단계로 구성된다.

① 시스템 콜 테이블에 있는 기존의 sys_open 시스템 콜을 저장하고, 새롭게 작성한 new_sys_open 함수로 교체한다.

② new_sys_open() 함수에서는 기존의 sys_open 시스템 콜을 먼저 호출하고, tty가 할당되어질 때, tty의 receive_buf() 함수를 새롭게 작성한 new_receive_buf() 함수로 대체한다.

③ new_receive_buf() 함수는 사용자 키스트로트를 저장하는 logging() 함수를 호출하여 지정한 로그파일로 키스트로크를 저장한다.

④ 위의 과정이 모두 끝나면, 정상적인 receive_buf() 함수를 호출하여 원래의 정상적인 동작으로 돌아간다.

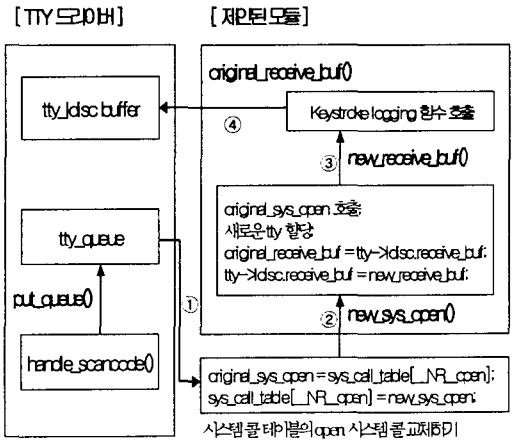

그림 2: 제안된 모듈이 동작하는 방식

2) logging 함수

new_receive_buf() 함수가 내장하는 logging 함수는 그림 3과 같이 tty의 receive_buf() 로 입력되는 키값들을 라인 버퍼로 넣는다. 버퍼내의 키값들은 한 줄의 명령어가 될 때까지 차례로 검사된다. 한 줄의 명령이가 되면 버퍼내의 내용은 로그파일로 형식에 맞추어 저장된다.

3) 로그 파일 형식

로그파일의 형식은 "< 로깅날짜 - 시간 사용자id 현재 수행 프로그램명> 입력한 키스트로크" 순이다.

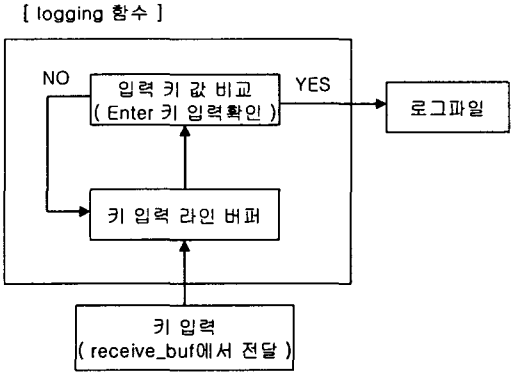

그림 3: logging 함수의 동작 방식

4) 실행 결과

본 논문이 제안하는 리눅스 커널 모듈을 사용하여, 한 사용자의 키스트로크를 가로채면 그림 4와같은 로그를 얻어낼 수 있다. 사용자가 키스트로크를 한 정확한 날짜와 시간, 사용자 ID, 그리고 사용하고 있는 프로그램을 기준으로 하여 키스트로크된 정확한 명령어를 로깅하고 있는 것을 볼 수 있다.

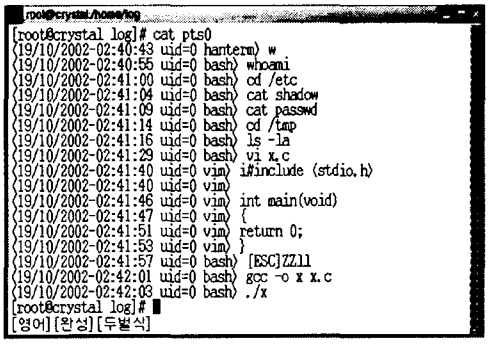

그림 4: 실행 결과

IV. 결론

본 논문에서는 리눅스 키보드 드라이버의 동작 과정 중에서 receive_buf() 함수를 가로채는 방법을 사용하여 사용자 키스트로크를 로깅할 수 있는 방법을 제안하였다. 다른 여러 키보드 입력처리 함수보다 시스템에 전혀 부하를 주지도 않고, 로컬상에서 로그인한 것뿐만 아니라 원격에서 접속한 사용자의 키스트로크를 모두 로깅할 수 있는 능력을 보여주었다. 이런 로깅 방식은 관리자에게 필요에 따라 사용자들의 행동을 파악하는데 많은 도움을 줄 것이다.

향후로는 사용자의 키스트로크를 가로채어 로깅하는 것을 호스트상에 데이터로 남기는 것만 아니라, 네트워크상의 로그서버로 안전한 채널을 통하여 전송시키는 것이 필요하다. 또한, 리눅스 커널 모듈은 여러 기능을 할 수 있는데, 그 가운데 특이한 기능을 사용하면 정상적인 명령어를 가로채어 일부의 결과값을 숨길 수 있다. 이 기능을 이용하여 관리자가 설치한 로깅 모듈을 숨기는 방법을 구현해보고자 한다. 이것은 임의의 공격자가 시스템에 침입한 후에 관리자의 로깅 모듈을 전혀 발견하지 못하고, 관리자는 공격자의 모든 행동을 매우 은밀하고 안전한 상태에서 추적할 수 있다.

참고문헌

[1] 송창훈, "레드햇 리눅스 완벽가이드 Ver.5.2", 사이버출판사, pp 1062-1069 , 1999년 5월

[2] 권수호, "리눅스 프로그래밍 바이블", 글로벌출판사, pp 17-23, 2002년 6월

[3] 한국정보보호진흥원 기술문서, "로그파일 위변조 방지 기술"
http://www.kisa.or.kr/technology/sub3/logfile.html

[4] rd, "Writing Linux Kernel Keylogger", PHRACK 59호, 2002년 6월
http://www.phrack.org/phrack/59/p59-0x0e.txt

[5] Silvio Cesare, "Kernel function hijacking", 1999년 11월
http://www.r-fx.net/docs/kernel-hijack.txt

[6] Halflife, "Abuse of the Linux Kernel for Fun and Profit", PHRACK 50호, 1997년 4월
http://www.phrack.com/phrack/50/P50-05