자체 보호 기능을 갖는 안전한 로깅 시스템

김 민 수† · 노 봉 남††

요 약

감사 로깅 시스템은 시스템 사용 내역과 통신망을 통한 접근 내역을 기록한다. 이 내역은 불법적인 시스템 자원의 사용이나 통신망을 통한 불법 접근이 발생하였을 때, 그 경로를 추적하는 자료로 사용된다. 따라서, 로깅 시스템은 침입자의 일차적인 공격목표가 될 수 있다.

우리는 UNIX 시스템을 기반으로 로그인 정보와 명령어 수행 정보를 로그파일에 기록하는 로깅 시스템을 개발하였다. 또한, 로깅 시스템에 대한 침입자의 공격을 방어하기 위해서 자체 보호 장치를 두고 있다. 그것은 로깅 프로세스와 로그 파일을 보호하는 것이다. 로깅하는 프로세스를 보호하기 위해서 프로세스 ID를 계속해서 변경하므로써 침입자의 공격을 회피하도록 하였다. 로그 파일을 보호하기 위해서 하드링크와 강제적 파일 잠금을 사용하여 로그 파일에 대한 삭제나 변경을 할 수 없도록 하였다.

Secure logging system with self-protecting function

Min-Soo Kim† · Bong-Nam Noh††

ABSTRACT

The audit logging system is to write the details of system use and access on networks. These details are used for trailing the route, when illegal access or using system resource is occurred on networks. The logging system, therefore, might be the first target of intruder.

We developed the logging system which writes the information of login and command execution on UNIX system. And we prepared the self-protecting functions of blocking intruder's attack on the logging system. They are protecting the logging process and the log file. To protect the logging process, we made it keep changing the process ID to avoid the intruder's attack. To protect the log file, we use hard link and mandatory file locking, so it can make it impossible to delete or change log file.

1. 서 론

감사 로깅 시스템은 시스템 사용 내역을 기록하여 시스템의 안정성을 높이기 위한 도구이다. 일반적으로 로깅 시스템에서는 시스템 내에서 수행된 각종 응용 프로세스에 대한 내역은 물론, 외부로부터 통신망을 통해 접근하여 수행된 작업 내역 등의 정보를 기록한다. 수집된 정보는 추후에 시스템 무결성을 확인하거나, 제반 운영 정책의 효과적인 적용여부를 확인하기 위한 분석자료를 제공하는 목적으로 구축 운영된다[2, 6]. 아울러, 시스템의 불법 사용, 통신망을 통한 불법 접근 등이 발생했을 때, 사용 및 접근 경로를 추적하여 불법 사용자를 적발하거나, 시스템 보안의 취약점을 식별하여 보안 체계를 강화하기 위한 목적으로도 사용된다[7].

† 정 회 원 : 전남대학교 전산학과 박사과정
†† 종신회원 : 전남대학교 전산학과 교수
논문접수 : 1998년 8월 20일, 심사완료 : 1999년 7월 2일

본 논문에서 제시한 로깅 시스템은 UNIX 시스템을 기반으로 하여 로그인 정보와 명령어 수행 정보를 로그 파일에 기록한다. 이 정보는 감사 작업의 입장에서 가장 근간이 되는 것으로 침입자가 어디에서 접속했으며 어떠한 작업을 수행했는지를 일목요연하게 볼 수 있다.

감사 자료에는 침입자의 침입 기록이 남아있기 때문에 침입자의 공격목표가 될 수 있다. 따라서, 제시한 로깅 시스템에서는 침입자의 공격으로부터 대응하기 위한 자체 보호 장치를 두고 있다. 로깅 프로세스는 프로세스 ID를 계속해서 변경하여 침입자가 로깅 프로세스를 강제 종료시킬 수 없도록 하였다. 그리고, 로그 파일은 강제적 파일 잠금(mandatory file locking)을 하고 하드 링크(hard link)로 연결하여 로그 파일의 수정과 삭제가 이루어 질 수 없도록 하였다.

2장에서는 기존의 보안 대책에 대하여 설명하였다. 3장에서는 본 연구에서 개발한 감사 로깅 시스템을 설명하였다. 그리고, 4장에서 결론을 맺었다.

2. 로깅 시스템

2.1 로깅 시스템 필요성

로깅 시스템은 시스템 내에서 수행된 각종 응용 프로세스에 대한 내역과 외부로부터 통신망을 통해 접근하여 수행된 작업 내역 등의 정보를 기록한다. 수집된 정보는 추후에 시스템 무결성을 확인하거나, 제반 운영 정책의 효과적인 적용여부를 확인하기 위한 분석자료로 사용된다. 또한, 불법적인 시스템 자원의 사용이나 통신망을 통한 불법 접근 등의 사건이 발생하였을 때, 사용 및 접근경로를 추적하는 감사 추적(audit trail) 자료로 사용된다.

위와 같은 감사 자료는 침입자의 흔적을 기록하기 때문에 침입자의 공격 목표가 될 수 있다. 따라서, 침입자에 대응할 수 있는 안전한 로깅 시스템이 필요하다.

2.2 로깅 시스템 응용

2.2.1 정보 수집

로깅 시스템에서 얻어지는 데이터는 일반적으로 누구나 얻을 수 있는 정보로서 시스템 사용 내역, 시스템 호출(system call) 정보, 컴퓨터 통신에 사용되는 패킷 등이다. 이러한 정보는 침입탐지 시스템에서 사용되는 자원으로 이용되거나 사후 감사 추적을 할 때 사용된다[9].

2.2.2 감사 자료 축약

정보 수집 단계에서 모아진 정보를 감사 자료라고 한다. 이 자료는 시스템을 사용하는 모든 사용자에 대한 흔적이며 시스템 사용 시간이 많을수록 자료 파일의 크기는 엄청나게 커지게 된다. 따라서, 이 자료에서 필요한 정보를 추출하고 압축하는 방법이 필요하다.

2.2.3 침입 탐지

침입 탐지 시스템 (Intrusion Detection Systems)은 일반적으로 컴퓨터 시스템의 운용체제가 제공하는 로깅 시스템을 이용하여 만들어진 감사 자료를 이용한다[3]. 이러한 감사 데이터를 분석하여 구분할 수 있는 침입의 종류를 허가 받지 않은 외부인에 의한 침투, 허가 받은 내부인에 의한 침투, 그리고 바이러스나 트로이 목마와 같은 경우로 분류된다.

UNIX 시스템에서 최근 침입 탐지 시스템에 대한 연구들이 진행되고 있다. 시스템의 불법접근을 차단하기 위한 방화벽(firewall)과 시스템의 취약점을 조사하는 진단 프로그램에 대한 연구와 개발이 이루어지고 있다. 그러나, 침입 탐지 부분에선 아직도 시스템 관리자의 능력이나 직관에 의존하고 있다.

2.2.4 감사 추적 및 조치

침입 여부를 판정하여 침입으로 판단되면, 침입 탐지 시스템은 이 사실을 관리자에게 알린다. 관리자는 침입이 확인되면 어느 경로를 통해서 어떠한 방법으로 침입하였는지 감사 자료를 분석하여 조사한다. 만약 시스템에 위험한 상황이 있을 때는 시스템의 피해를 최소화하도록 시스템을 정지시키는 조치를 취할 수 있다.

2.3 UNIX 로깅 시스템

2.3.1 로깅 시스템 종류

기존 UNIX 시스템에서 사용하는 로깅 방법을 알아본다. 이러한 로깅에는 사용자가 언제 접속하였는지를 기록하는 내용과 사용자가 무슨 일을 했는지를 기록하는 것으로 나누어 볼 수 있다.

사용자가 언제 어느 호스트에서 로그인하여 얼마동안 사용하고 로그아웃 했는지를 알 수 있는 파일로는 *utmp* (*/var/adm/utmp*)와 *wtmp* (*/var/adm/wtmp*) 두 파일이 있다[5]. 이것은 시스템의 재부팅(reboot)이나

셧다운(shutdown) 등의 정보도 알 수 있다. *wtmp* 파일의 경우는 시간이 지남에 따라 파일이 커지기 때문에 정기적인 백업이 필요하다. *lastlog* (*/var/adm /lastlog*)는 주로 사용자가 로그인 할 때 위 파일에서 사용자 ID 정보를 찾아 어느 호스트에서 언제 접근했는지에 대한 최후 정보를 보여준다[5].

사용자가 사용한 명령어 내역을 기록하는 파일로는 */var/adm/pacct*가 있다. 기록 내용은 사용자 ID, 명령어, 시간 등이다. *accton*이라는 명령어에 의해서 시작되며, 프로그램 내에서도 함수를 호출할 수 있다. 단, 많은 명령어가 기록되기 때문에 로그 파일이 커지지 않도록 백업하는 것이 필요하다.

UNIX에서는 최근에 사용한 명령문의 내역을 *$HOME /.history*라는 파일에 기록한다. 일반적으로 침입자가 이 파일의 존재를 알지 못하거나 간과하기 쉽기 때문에 유용하게 이용될 수 있다.

*syslogd*라는 데몬 프로그램에 의해서 작동되는 *syslog* 프로그램은 메시지를 받으면 메시지에 해당하는 파일을 검사하여 */var/adm/message*, */var/adm/sulog*, */var/log/syslog* 등에 기록한다. *message* 파일에는 전체적으로 *syslog*에서 지정하는 메시지에 대한 내용을 기록하고 *sulog*에는 *su* 명령어 활동을 기록한다. 또한, 모든 *sendmail* 사용 내역을 */var/log/syslog* 파일에 기록하여 메일에 관련된 버그를 악용하여 SMTP 포트에 접근하는 것을 알아 낼 수 있다[5].

그 외 *trace*는 사용자 프로세스나 명령어에서 사용되는 시스템 호출을 추적한다. 임의의 프로세스를 추적하기 시작하면 그 프로세스가 종료될 때까지 어떠한 시스템 호출을 이용했는지를 기록한다. 따라서, 의심이 가는 프로세스나 명령어를 조사해 볼 수 있다. 그러나, 전체 시스템의 시스템 호출을 기록하는 것이 아니라 하나의 프로세스나 명령어를 지정하여 시스템 호출을 기록하기 때문에 프로세스의 관계를 알아내는 것은 어렵다.

2.3.2 UNIX 로깅 시스템의 장단점

UNIX에서는 사용자의 로그인, 로그아웃 정보와 명령어 실행 정보를 기록하고 있다. 이러한 작업은 커널이 담당하고 있다. 그렇기 때문에 시스템을 사용한 모든 내역이 기록된다[8].

그러나, UNIX에서의 로그 파일에는 문제점이 있다. 로그 파일을 보호하는 조치가 없는 것이다. 즉, 슈퍼유저 권한만 있으면 모든 로그 파일을 삭제할 수 있다. 특히, 뒷문(backdoor) 프로그램에서는 사용자가 슈퍼유저의 권한으로 로그인 할 수 있도록 허용하고, 로그아웃 하면 침입자의 모든 사용 내역을 로그 파일로부터 삭제한다. 따라서, 이러한 위협에 능동적인 방어를 할 수 있는 로깅 시스템이 필요하다.

2.4 기타 보안 정책

보안은 상황에 따라 다양한 의미를 내포할 수 있다. 인가되지 않는 사용자에 의해서 자신의 시스템의 내용이 변경되는지, 그리고 네트워크에 연결된 다른 시스템에서 자신의 시스템에 접근 권한을 얻을 수 있는지 등이다. 네트워크의 보안 측면에서 관리자는 외부의 접근으로부터 네트워크 내의 자원을 보호하기 위해 방화벽 시스템을 설치할 수 있다. 그 외에도 네트워크 내에서 비인가된 접근으로부터 네트워크 내의 시스템을 보호하는 조치를 취할 수 있다.

2.4.1 방화벽 시스템

방화벽 시스템은 외부로부터의 불법적인 접근을 봉쇄하는 강력한 보안 시스템을 말한다. 일반 시스템에서 보안을 강화하는 경우 시스템 성능저하라는 단점이 있기 때문에, 방화벽 시스템은 보안 전담 시스템으로서 보안을 강화한 시스템이다[1].

방화벽은 인터넷과 조직 내의 네트워크 접속, 전송기능을 갖지 않게 설정한 호스트를 중간에 위치하여 수행한다[11]. 따라서 내부 조직으로부터 인터넷으로 나가는 경우는 반드시 방화벽의 역할을 가진 호스트에 로그인한 다음에 수행하게 된다. 방화벽은 외부로부터의 부당한 접근을 막는 아주 중요한 수단이라고 말할 수 있다.

2.4.2 인터넷 보안

인터넷에는 전세계적으로 매우 많은 시스템들이 포함되어 있다. 매우 많은 사용자들이 사용하고 있지만, 그 반면에 여기에는 어떠한 알 수 없는 종류의 위험성이 내포되어 있다.

시스템을 네트워크에 참여시킨다는 것은 정보를 쉽게 접근할 수 있고 자유롭게 통신을 하기 위해서이다. 네트워크를 이용하여 다른 시스템에 자료를 손쉽게 주고받을 수 있으며, 또한 자기 시스템에 없는 자원(예, 프린터)들을 공유를 할 수 있다[7]. 그러나 실제로 네

트워크에 접속을 할 때는 반드시 보안에 관해서 세심한 주의를 기울일 필요가 있다.

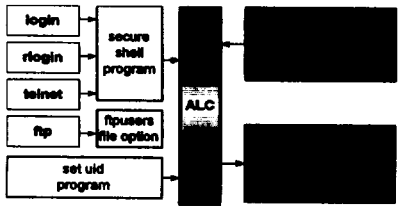
잘 알려진 인터넷 보안 도구로 Tcp Wrapper가 있다. 이것은 네트워크 서비스(예를 들어 *telnet, finger, rlogin, rsh* 등)에 대한 로그 파일을 만들며 몇 가지 부가적인 보안 체크를 하는 보안 강화 도구이다. Tcp Wrapper를 사용하면 *in.telnetd*를 구동하기 전에 *tcpd*라는 데몬 프로세스(daemon process)를 구동시켜 *telnet*에 대한 로그 파일을 기록한 후 *in.telnetd*를 구동시킨다. 이러한 방식을 해커가 시스템에 들어오기 위해 자주 사용하는 *finger, rsh, tftp* 등에 사용하면 해커도 모르게 그의 자취를 남겨 놓을 수 있다[11].

네트워크 상에 돌아다니는 패킷을 모두 가로채서 내용을 훔쳐볼 수 있는 프로그램으로 sniffer가 있다. 이것을 이용하여 네트워크 상에 흘러 다니는 패킷들을 검사하여 해커가 불법적인 작업을 하고 있는가를 알아낼 수도 있다.

기타 보안 도구로 Swatch(Simple Watcher), Crack, Tripwire, COPS(Computer Oracle and Password System), MD5(New Message Digest Algorithm), CPM(Check in Promiscuous Mode), LSOF(LiSt Open Files), Traceroute, *sudo*, 그리고 ISS(Internet Security Scanner) 등이 있다[10].

3. 감사 로깅 시스템

본 연구에서 개발한 감사 로깅 시스템의 전반적인 구조는 (그림 1)과 같다. 그림에서 ALC(Audit Log Collec -tor)는 감사 로깅 프로그램으로 섹션 4에서 설명한다. 시스템에 대한 접근 방법으로 시스템에 접근하면 그 정보를 UNIX 커널에서 수집하여 안전한 로그 파일에 기록하는 시스템이다.



(그림 1) 감사 로깅 시스템의 구조

3.1 시스템 요구사항

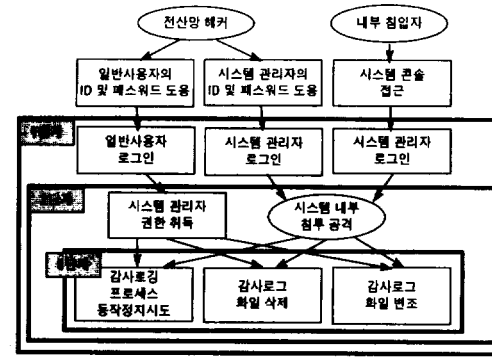
이 시스템은 크게 두 가지로 분류된다. 먼저, 추후에 불법 침입자를 추적하는데 유용한 정보를 효과적으로 수집하기 위한 감사 로깅 시스템을 설계하고 구현하는 작업이다. 두 번째는, 감사 로깅 시스템 자체를 불법 침입자로부터 보호하기 위한 자체 보호 장치를 설계하고 구현하는 작업이다.

UNIX 시스템에서 지원하는 명령어 로깅의 경우 침입자를 추적하는데 유용한 정보를 충분히 지원하지 않는다. 따라서, 효과적으로 정보를 수집하기 위해서 커널에서 직접 정보를 수집하는 방법을 구현할 수 있다.

로깅 시스템은 침입자의 흔적을 추적하는 자료이기 때문에 침입자의 주요 공격목표가 되기 쉽다. 따라서, 이에 대한 보호 조치를 마련하여야 한다. 그림 2에서는 침입자의 시스템 침입 단계를 도식적으로 그려보았다. 안전한 로깅 시스템은 위와 같은 단계적인 침입에 보호되어야 한다.

3.2 방어 대책

여기에서 일반적인 UNIX 시스템에서 침투하는 단계를 나누어 보면 (그림 2)와 같이 3단계로 나누어 볼 수 있다. 그리고, 각 단계의 침투 유형을 볼 수 있다. 그러면 각 침투 방법에 대한 감사 로깅 시스템의 대응 방법을 알아보자.



(그림 2) 시스템 침입 단계

3.2.1 1단계 침투

① 일반 사용자 로그인

시스템은 일단 허락 받는 누구나 사용할 수 있도록 하고 있다. 따라서, 본 로깅 시스템 역시 일반 사용자 ID를 이용한 로그인은 어느 곳이든 허락하고 있다.

② 콘솔에서의 슈퍼유저 로그인

콘솔은 보통 시스템 관리자가 사용하는 시스템이다.

따라서, 콘솔에서 슈퍼유저로 로그인하는 것은 허용한다.

③ 콘솔이 아닌 곳에서 슈퍼유저 ID를 사용하여 로그인
감사 로깅 시스템에서는 먼저 슈퍼유저로 시스템에 접근하려는 것을 제한한다. 본 시스템에서는 슈퍼유저는 콘솔에서만 접근할 수 있도록 한다.

3.2.2 2단계 침투

① 일반 사용자로 접근하여 슈퍼유저 권한을 얻는 경우
일반적으로 *su* 명령어를 사용하거나 setuid 프로그램을 이용하여 슈퍼유저 권한을 얻는다. 이러한 경우 로깅 프로세스에서는 각 프로세스 정보를 커널에서 가져올 때마다 UID와 TTY를 검사하여 콘솔이 아닌 곳에서 일반 사용자가 슈퍼유저의 쉘 권한을 획득하는 것을 감시한다.

② 시스템 내부 침투 공격
시스템을 사용한 모든 내역은 로깅 프로세스가 기록하여 내부적인 활동을 감시한다.

3.2.3 3단계 침투

① 감사 로깅 프로세스의 작동을 중지하려는 시도
불법 침입자는 자신의 활동이 기록되는 것을 거북하게 느낄 것이다. 따라서 일반적인 공격형태는 로깅 프로세스를 종료시키거나 로그 파일을 변경 또는 삭제하려 할 것이다. 본 시스템에서는 이러한 공격에 대응하기 위하여 로깅 프로세스가 쉽게 공격당하지 않도록 하고 있다.

② 감사 로그 파일을 불법적으로 삭제하려는 경우
일반적으로 침입자는 감사 로그 파일을 삭제하여 자신의 행적을 숨기려 할 것이다. 이러한 공격에 대응하여 본 시스템은 슈퍼유저 권한을 갖는 사용자라도 감사 로그 파일을 강제로 삭제할 수 없도록 하고 있다.

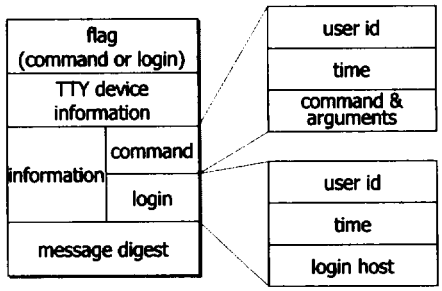
③ 감사 로그 파일을 변조하려는 경우
감사 로그 파일에서 자신의 행적을 변경하려는 시도가 있을 수 있다. 이러한 공격에 대비하여 감사 로그 파일을 수정할 수 없도록 로그 파일에 강제적 파일 잠금하고 있다.

3.2.4 기 타

사용자가 어떠한 프로그램을 백그라운드 환경에서 수행시키고 로그아웃 하더라도 그 프로세스를 추적할 수 있도록 하고 있다.

3.3. 감사 로그 내용

감사 로깅 시스템은 다음과 같은 두 가지 종류의 시스템 접근내역을 감사 로그에 기록한다. 이 시스템에서 기록하고 있는 내용은 (그림 3)과 같다.



(그림 3) 로그 파일 내용

3.3.1 사용자 로그인 및 로그아웃 정보

컴퓨터 시스템을 사용하기 위해서는 기본적으로 컴퓨터 시스템 내에 등록된 계정으로 로그인 해야 하고, 사용이 종료되면 로그아웃을 하게 된다. 각 사용자별로 로그인하여 로그아웃 할 때까지의 기간을 하나의 로그인 세션으로 취급할 수 있는데, 시스템 사용자의 모든 활동은 원칙적으로 각 로그인 세션 내에서만 이루어진다[4]. 따라서, 각 사용자별로 로그인 시간, 로그아웃 시간, 사용한 터미널 위치 등의 정보는 감사작업의 관점에서 가장 근간이 되는 중요한 정보이다.

다음 두 가지 경우에는 로그인 세션과는 별도로 시스템 자원을 접근할 수 있다. 첫째, 일반적으로 사용자가 로그아웃을 하게 되면 프로세스 관리자는 그 사용자의 로그인 프로세스로부터 생성된 모든 자손 프로세스를 종료시킨다. 그렇지만, 특별히 사용자가 백그라운드 환경에서 수행시킨 프로세스는 로그아웃 이후에도 계속 수행된다. 이러한 프로세스는 사용자가 로그아웃한 이후에도 터미널 디바이스(TTY)를 유지하고 있다. 이러한 이유로 본 시스템에서는 백그라운드 환경의 프로세스를 계속해서 감시할 수 있다.

둘째, 통신망을 통해 클라이언트-서버 방식으로 시스템에 접근한 경우이다. 클라이언트-서버 방식으로 통신망을 통해 시스템에 접근한 경우에는 해당 시스템에서는 서버 프로세스만 동작하는 것이므로, 그 서버 프로세스에 대한 정보만 관리된다. 따라서, 여러 사용자들이 자신의 클라이언트를 통해 임의적으로 접근하여 다양한 논리적 로그인 세션이 생성, 소멸되었

다 하더라도, 시스템 내부에서는 계속적으로 서버 프로세스만 동작하고 있는 것으로 기록되므로, 로그인 세션이 아닌데도 시스템 자원에 대한 접근이 이루어진다고 볼 수 있다. 예를 들어, 사용자가 FTP 접속을 하게 되면 서비스를 제공하는 서버 시스템에서는 *ftpd* 라는 데몬 프로세스만 동작한다. 그리고, 사용자가 업로드(up-load)나 다운로드(down-load)하는 정보는 알 수 없다.

3.3.2 각 응용 프로그램의 수행 내역

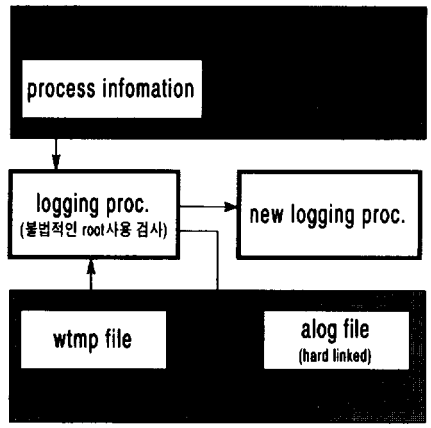
시스템 사용자는 로그인 세션 동안에 다양한 응용 프로그램을 수행한다. 로그인할 때 *.cshrc, .profile*등에 미리 규정된 응용 프로그램을 기본 수행함을 물론, shell 인터페이스를 통해 응용 프로그램의 수행을 지시하거나, 한 응용 프로그램 내부에서 *fork, exec* 등을 통해 다른 응용 프로그램을 구동시킬 수 있다. 원칙적으로 응용 프로그램의 수행 없이 시스템 자원에 대한 접근이 절대 이루어 질 수 없으므로, 사용자별 응용 프로그램 수행 내역은 감사 로깅 시스템에서 시스템에 대한 불법 접근을 추적할 수 있는 매우 유용한 정보가 된다. 특히, 응용 프로그램 내부에서 다른 응용 프로그램을 구동시키는 경우는 전산시스템 불법 이용의 대표적인 사례인 트로이 목마형의 접근이 이루어질 수 있다. 또한 UNIX 시스템의 응용 프로그램 검색 정책 즉, path 환경변수에 의한 순차적 검색 방식을 악용하여 동일한 이름의 다른 응용 프로그램을 불법적으로 수행할 수 있도록 조작할 수도 있다.

따라서, 본 시스템에서는 운영체제의 커널로부터 프로세스 정보를 수집한다. 프로세스는 **exec**, *fork* 등을 통해 다른 프로세스를 생성할 수도 있으므로, 응용 프로그램의 수행내역에 대한 정보는 shell과 같은 외부 인터페이스를 통해서 얻는 것보다 운영체제의 커널로부터 직접 입수해야 하는 게 효과적이다.

3.4 개발 시스템 세부구조

3.4.1 감사 로깅 프로그램

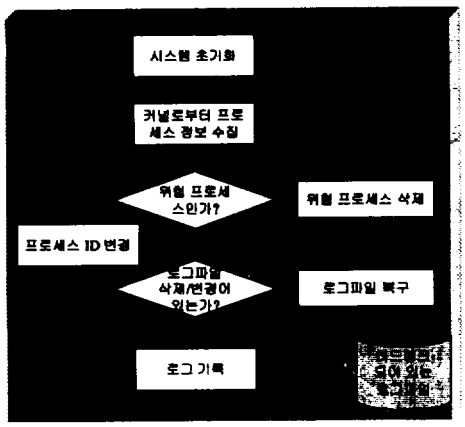
로그인 세션정보, 응용 프로그램 수행내역, TCP/IP 접근 내역을 일괄적으로 수집하는 감사 로깅 프로그램(이하 ALC)이다. 특히 감사 로깅 프로그램은 불법 침입자로부터의 공격대상이 될 수 있기 때문에 ALC는 자체적인 보호 장치를 부착한다. ALC는 스스로 보호하기 위해서 자신의 프로세스 ID를 계속해서 변경한다. 즉, 침입자가 *ps* 등의 명령어로 ALC 프로그램의 프로세스 ID를 알아내어 강제 종료시키려는 공격에 대한 방어책이다. 프로세스 ID를 계속해서 바꾸어 로깅 프로세스를 강제로 종료시키려고 할 때 사용하는 프로세스 ID가 타당하지 않게 하려는 의도이다. 로깅 프로세스는 계속해서 생성하여 부모 프로세스는 종료하고, 자식 프로세스가 일을 이어받아 수행하는 구조로 되어 있다(그림 4). 이렇게 함으로서 로깅 프로세스는 프로세스 ID를 계속해서 바꿀 수 있다.



(그림 4) 감사 로깅 프로그램

로깅 프로세스에서는 로깅 파일을 보호하기 위해 강제적 파일 잠금과 하드 링크를 사용한다. 하드 링크를 사용하여 같은 파일 위치를 가리키는 다른 이름의 파일을 생성한다. 이러한 방법은 별도의 큰공간을 차지하지 않고도 같은 파일을 여러개 가지고 있는 효과를 발휘한다. 본 시스템에서는 이러한 하드 링크를 여러 개 생성하여 파일 삭제가 어렵도록 하였다. 만약의 경우에 어떤 불법적인 침입자가 로깅 파일을 지우려고 시도하여 몇 개의 링크 파일을 삭제한다면, 로깅 프로세스는 삭제하는 프로세스를 종료시키고 손상된 링크 파일을 복구시킨다.

ALC에서 수행하는 절차를 (그림 5)에서 흐름도로 보여준다. 시스템을 초기화한 후 커널에서 프로세스 정보를 가져와 위험 프로세스(잘못된 슈퍼유저 권한 프로세스)인지 검사하여 처리하고 로그파일의 무결성을 검증한다. 프로세스 정보를 로그파일에 기록한 후 프로세스 ID를 변경한다.



(그림 5) ALC의 수행 흐름도

3.4.2 감사 로그 관리자(Audit Log Manager)

감사 로그의 내용을 해독하여 분석할 수 있도록 해주는 감사 로그 관리자 프로그램이다. 이것은 로깅 프로세스에서 기록한 로그 파일을 읽어 사용자가 원하는 형태로 보여주는 역할을 한다.

3.4.3 슈퍼유저 권한 획득 제한

이것은 두 가지로 나누어 볼 수 있다. 하나는 사용자 로그인을 제한하는 것이다. 즉, 슈퍼유저로 로그인하는 경우는 콘솔로만 제한한다. 처리 방법은 *.login*과 같은 로그인 쉘 스크립트 파일에 슈퍼유저이면서 TTY가 콘솔인지를 검사하는 프로그램을 삽입한다. *login, rlogin, telnet* 등으로 시스템에 접속하는 모든 사용자를 검사할 수 있다. ftp는 */etc/ftpusers*에 root를 삽입함으로써 슈퍼유저 권한으로 ftp 접속하는 것을 막을 수 있다.

두 번째는 일반 사용자로 접속한 후 슈퍼유저 권한을 얻는 경우이다. 대표적인 예는 *su* 명령어를 사용하는 것이다. 처리 방법으로 쉽게 생각할 수 있는 것은 *su* 명령어의 실행권한을 root로만 제한하면 된다. 다른 예로는 setuid 프로그램을 이용하는 방법이다. 여기에서 처리는 *su*처럼 사용 권한을 제한하면 일반 사용자가 시스템 자체를 사용할 수 없기 때문에, 커널에 있는 프로세스 정보를 읽어와 검사한다. 이러한 역할을 하는 것은 로깅 프로세스로 로깅 파일에 기록하는 과정에서 검사한다.

3.5 수행결과

감사 로깅 시스템을 작동시킨 결과를 스크립트로 받아 보았다(그림 6). 어떠한 사이트에서 로그인을 하게 되면 그림 6에 라인 2와 라인 5에서처럼 pts/5 디바이스에 *azalea*라는 시스템에서 *is*라는 UID로 로그인하는 정보를 볼 수 있다.

로그 파일을 강제로 삭제하려는 시도가 있을 수 있다. 이에 따른 결과는 라인 10처럼 파일 삭제를 명령할 경우 라인 11-13과 같은 메시지가 나오면서 파일 복구가 수행된다.

```
    PID [UID] TTYD    COMMAND
 1: 1806 [  0] ?      (in.telnetd) in.telnetd
 2: 1810 [  0] pts/5  (login) login -d /dev/pts/5 -h azalea TERM=vt10
0
 3: 1836 [108] pts/5  (tty) tty
 4: 1825 [  0] ?      (rquotad) rquotad
 5: 1810 [108] pts/5  (csh) -csh
 6: <login>  USR=is DEV=pts/5 HST=azalea
 7: 1849 [108] pts/5  (vi) vi .cshrc
 8: 1873 [108] pts/5  (telnet) telnet chonnam
 9: <logout> USR=is DEV=pts/5 HST=azalea
10: 1997 [  0] consol (rm) rm -i lk060 lk068
11: Log file is deleting!!!
12: * log file is deleted *
13: File repaired. [/UPA/new/lk060]
14: 2002 [  0] consol (logging) logging
15: 2023 [106] pts/1  (vi) vi
16: 2105 [  0] ?      (in.telnetd) in.telnetd
17: 2111 [  0] pts/9  (login) login -d /dev/pts/9 -h spring TERM=vt10
0
18: 2111 [  0] pts/9  (csh) -csh
19: process[2111] killed!!
```

(그림 6) 로깅 프로세스 수행 예

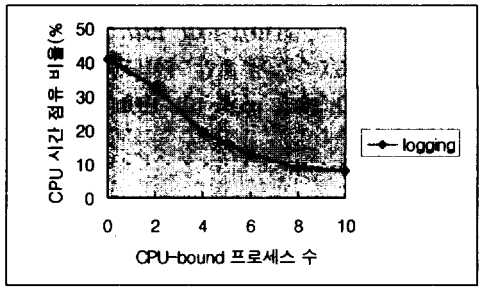
라인 17에서는 콘솔이 아닌 곳에서 슈퍼유저로 로그인하는 경우를 보여준다. 라인 19처럼 해당 프로세스를 바로 종료시켜 버린다.

이 시스템은 로그인 정보를 모두 기록할 수 있다. 다만, 명령어의 경우 아주 짧은 시간을 필요로 하는 프로그램(예를 들면, *cd*)의 수행을 인식하지 못하는 경우가 있다. 이것은 프로세스간 시간경쟁에 따른 결과이다. 만약 이러한 경쟁을 높이게 되면(예를 들면, *nice*나 *renice*로), 전체 시스템의 성능이 떨어질 수 있다. 따라서, 본 시스템에서는 수행 기간이 아주 짧은 프로세스는 놓치더라도 시스템에 결정적인 영향을 주는 프로세스는 감시할 수 있도록 위에서 언급한 것처럼 방어 조치를 취하고 있다.

3.6 분 석

본 시스템에 대하여 CPU 사용량을 검사해본 결과 (그림 7)과 같이 나타났다. (그림 7)에서 가로는 다른 프로세스 수이고 세로는 CPU 사용량(%)이다. 기본적인 시스템에 관련된 프로세스 외에 CPU를 자주 사용하는 프로세스 수가 0개에서 10개 있을 때까지를 검사

해 보았다. 그 결과, CPU 사용량이 처음에는 40여 퍼센트 정도에서 다른 프로세스가 많아짐에 따라 10여 퍼센트 이하까지 감소됨을 알 수 있다. 즉 일반 프로세스와 동등한 입장에서 시간경쟁 하는 것을 알 수 있다. 이 프로그램은 실제로 사용하더라도 눈으로 분간할 수 있는 정도로 시스템 부하가 생기지 않는다는 것을 판단할 수 있다.



(그림 7) 로깅 프로그램 수행에 사용된 CPU 점유 비율

마지막으로, 로깅 시스템에 대한 3단계 침투 단계에 따른 대응방법에 대해 표로 구성해 보았다<표 1> 로그 파일을 보호하기 위해서 하드 링크와 강제적 파일 잠금을 하였고, 로깅을 수행하는 프로세스를 보호하기 위해서 프로세스 ID를 계속해서 변경하도록 하였다. 또한, 일반 사용자가 슈퍼유저 권한을 획득할 수 없도록 제한함으로써 발생할 수 있는 사건을 미리 방비하도록 하였다.

4. 결 론

본 논문에서 구현한 감사 로깅 시스템은 로그인 정보와 명령어 수행 정보를 안전하게 로그파일에 기록한다. ALC에서는 시스템의 안전을 위해 사용자의 슈퍼유저 권한 획득을 제한하며 커널로부터 로깅정보를 수집한다. 이러한 로그파일의 정보는 침입자가 침입한 증거가 되므로 침입자의 주요 목표가 된다. 따라서, 감사 로깅 시스템에서는 침입자의 공격으로부터 대응하기 위한 자체 보호 장치를 두고 있다. 로깅 프로세스를 보호하기 위해서 프로세스 ID를 계속해서 변경하여 침입자가 로깅 프로세스를 강제 종료시킬 수 없도록 하였다. 그리고, 로그 파일의 수정과 삭제가 이루어지지 않도록 보호하기 위하여 강제적 파일 잠금을 하고 여러개의 하드 링크로 생성하였다.

감사 로깅 시스템은 크게 두 가지 문제점을 가지고 있다. 첫째로, 로깅 프로세스가 시간 경쟁(racecondition)에 따른다는 것이다. 즉, 로깅 프로세스에서 모든 실행되는 명령어를 기록할 수는 없다. 둘째로, 로깅 프로세스에서 자신의 프로세스를 보호하기 위해 프로세스 ID를 계속해서 바꾸어 가지만 선형급수로 증가하기 때문에, 침입자가 로깅 프로세스의 ID를 추정하여 프로세스를 종료시키는 명령어를 미리 반복해서 실행시킬 수 있다.

감사 로깅 시스템에서 필요한 사항은 세 가지로 구분할 수 있다. 첫째는 로깅 프로세스가 시간 경쟁에 관계없이 동작될 수 있어야 한다. 즉, 로깅 처리가 커

〈표 1〉 침투 단계에 따는 대응방법

단계	침 투 방 법	대 응 방 법
1단계	일반 사용자의 ID 및 패스워드를 도용한 일반 사용자 로그인	제한하지 않음.
	시스템 관리자의 ID 및 패스워드를 도용한 시스템 관리자 로그인	보안 쉘(secure shell) 프로그램에서 검사하여 연결 해지시킨다.
	시스템 콘솔을 이용한 시스템 관리자 로그인	제한하지 않음.
2단계	일반 사용자의 시스템 관리자 권한 취득	프로세스의 UID와 TTY를 검사하여 UID가 0이고 TTY가 콘솔이 아니면 연결 해지시킨다.
	시스템 관리자의 시스템 내부 침투 공격	로그 파일로 기록을 남긴다.
3단계	감사 로깅 프로세스의 동작을 정지시키려는 시도	감사 로깅 프로세스의 ID를 계속해서 바꿈으로서 프로세스에 대한 공격을 피한다.
	감사 로그 파일을 삭제하려는 시도	하드 링크를 여러개 연결하여 일차적으로 파일을 보호하고 외부 프로세스를 모두 종료시킨다.
	감사 로그 파일을 변조하려는 시도	파일의 상태 정보에 이상이 발견되면 외부 프로세스를 모두 종료시킨다.

널의 수준에서 수행되어야 한다. 둘째는 프로세스 ID나 프로세스 이름을 불규칙적(random)으로 변경하는 방법을 찾아야 한다. 이렇게 함으로써 위의 두 번째 문제점을 해결할 수 있을 것이다. 셋째는 침입자의 침투 유형에 따라 능동적으로 대처하는 프로그램이 되어야 한다는 것이다. 이것은 많은 조사와 시간에 따른 수정이 필요하지만, 앞으로의 숙제라 할 수 있겠다.

Techniques," Computer & Security, Vol.12, No. 4,

참 고 문 헌

[1] Steven M. Bellovin and William R. Cheswick, "Network Firewalls", IEEE Communications Magazine, Sep., pp50-57, 1994.

[2] Boris Kogan, "An Audit Model for Object -Oriented Databases", IEEE 7th Computer Security Applications Conference, Dec., pp.90-99, 1991.

[3] T.F. Lunt, "A Survey of Intrusion Detection Jun., 1993.

[4] Abdelaziz Mounji, Baudouin Le Charlier, Denis Zampuniéris, and Naji Habra, "Distributed Audit Trail Analysis", Proc. of Symposium on Network and Distributed System Security, Feb., pp.102-112, 1995.

[5] Evi Nemeth, Garth Snyder, Scott Seebass, and Trent R. Hein, 'UNIX system Administration Handbook -2e', Prentice Hall, 1995.

[6] J. Picciotto, "The Design of an Effective Auditing Subsystem", IEEE Security and Privacy, Vol.2, pp.13-22, 1987.

[7] Samuel I. Schaen, "Network Auditing: Issues and Recommendations", IEEE 7th Computer Security Applications Conference, pp.66-79, Dec., 1991.

[8] Kenneth F. Seiden and Jeffrey P. Melanson, "The Auditing Facility for a VMM Security Kernel", IEEE Research in Security and Privacy, May., pp.262~277, 1990.

[9] Christopher Wee, "LAFS: A Logging and Auditing File System", IEEE 11th Computer Security Applications Conference, Dec., pp.231~240, 1995.

[10] 윤여웅, 이봉근, 신종태, 소우영, "해킹을 대비한 보안 도구 및 추적 방법", 한국정보처리학회 '97춘계학술발표논문집, pp.1002-1007, 1997.

[11] 임채호, "UNIX 시스템 보안 가이드", 통신정보보호학회지, Vol.1, No.2, pp.86-101, 1991.

김 민 수

e-mai : minsoo@chonnam.chonnam.ac.kr,

1993년 전남대학교 전산통계학과 (학사).

1995년 전남대학교 대학원 전산통계학과 (이학석사).

1995년~현재 전남대학교 시간강사.

1996년~현재 전남대학교 대학원 전산통계학과 박사과정.

관심분야 : 침입탐지, 시스템보안, 통신보안 등.

노 봉 남

e-mail : bongnam@chonnam.chonnam.ac.kr

1978년 전남대학교 수학교육학과 (학사)

1982년 한국과학기술원 전산학과 (공학 석사).

1994년 전북대학교 대학원 전산계학과 (이학박사).

1983년~현재 전남대학교 컴퓨터정보학부 교수

관심분야 : 통신망관리, 정보보안, 컴퓨터와 정보사회 등.