정보시스템 보안을 위한 감사증적모델

김영철 · 남길현
국 방 대 학 원

Audit Trail Model for Information System Security

Yung Chul Kim · Kil Hyun Nam
National Defense College

요 약

본 연구에서는 정보시스템 보안을 위한 기존의 감사증적 메카니즘을 고찰, 분석하여 우리의 실정에 필요한 감사증적모델을 설계하였다. 특히, 우리나라가 표방하는 개방형 시스템, 국가기간전산망 등의 보안측면을 고려하여 UNIX 워크스테이션 환경에 촛점을 맞추었다. 설계된 모델은 시스템접근시도의 실패를 포함한 모든 보안관련 정보를 운영체제수준에서 기록하고 필요한 내용을 쉽게 조회하도록 하였다.

I. 서 론

정보통신기술의 발전과 컴퓨터이용의 증가로 정보화사회가 가속화되면서 정보시스템의 보안문제(security)가 대두되고 이러한 보안과 관련한 감사증적(audit trail)의 중요성이 증대되고 있다. 이는 정보시스템의 유용성에 비해 보유자원의 불법적인 복제·절취·변경등의 컴퓨터범죄와 오류발생의 가능성이 증가하기 때문이다.

정보시스템의 보호체계는 단순히 물리적 수단만으로 해결하려는 문서중심의 정보보호체계와는 다르다. 우선, 접근가능한 정보량이 많고 범위가 광역화되어 있으며 일단 보안사고가 일어나면 파급이 크고 사고추적이 매우 난해하다는 점이 특징이다. 또한, 보안시스템이 아무리 잘 구성되어 있더라도 완벽할 수는 없다.

감사증적은 누가, 무슨 자원(resource)을, 언제, 어떤 순서로 이용했는가를 추적하는 회계측면의 기록성(accountability)에서 출발하여 사용료청구, 통계유지, backup등의 수단으로 이용되었으나 최근에는 보안개념이 추가되면서 정보시스템에 대한 불법적 접근을 어떻게 추적하느냐가 중요한 관심사로 등장하고 있다.

따라서, 정보시스템에는 정보의 불법적 유출을 예방하고 통제원칙의 준수와 불법행위의 추적을 위한 감사능력(auditability)이 확보되어야 하며 정보시스템 관계자의 책임을 명확히 하고 사용자의 행위를 가시화할 수 있는 메카니즘이 요청된다.

본 연구에서는 기존의 감사증적 메카니즘을 고찰하여 주요 특징을 분석함으로써, 개방형 정보시스템에 적합한 감사증적모델을 제안하고자 한다.

Ⅱ. 기존의 감사증적 메카니즘 고찰

1. 감사증적 개요

감사증적 메카니즘은 TCSEC과 ITSEC에서도 기본적인 보안요구사항으로 채택하는 등 정보시스템 보안의 중요한 구성요소가 되고 있으며 감사증적의 목적에는 다음과 같은 사항이 포함되어야 한다. ① 사용자/프로세스가 객체에 대해 접근하는 상황을 검사한다. ② 보안메카니즘을 우회하려는 불법적 시도를 저지하고 기록한다. ③ 중요한 객체의 접근상황을 기록하여 관계자에게 제공한다.

감사증적의 목적을 실현하기 위해서는 불법적 접근시도에 의해 침해되지 않도록 감사기록을 안전하게 관리하고 비밀성(confidentiality), 무결성(integrity) 및 가용성(availability)을 만족하는 메카니즘을 갖추어야 한다. 이러한 메카니즘에는 ① 패스워드의 불법적 사용/변경, 지정된 사용자의 행위, 시스템명령어의 부당한 사용 등을 감시하고 ② 지정된 자원에 대한 접근, 침해, 변경의 실패/성공 등을 감사기록/보관하여 신분/객체의 비밀수준에 의한 선택이 가능하며 ③ 보안침해에 대한 대응조치가 구비되는 한편, ④ covert channel에 대한 감사기능이 요구된다[Glig86]. 또한, 감사증적을 전문으로 취급하는 보안담당자(security officer)가 지정되어야 한다.

2. IBM mainframe환경의 감사증적 메카니즘

◆ GUARDIAN은 IBM CICS(Customer Information Control System)환경에서 보안, 감사 및 통제능력의 제고를 위해 개발된 package이다. 이는 기존의 CICS가 통제보다는 생산성에 치중하고 시스템기록(CICS journal)은 단순히 backup기능만을 제공한다는 보안상의 취약성(vulnerability)을 개선하였다[Moll84].

◆ RACF(Resource Access Control Facility)는 IBM OS/MVS의 확장개념으로 작동하며 자원보호를 위한 접근제어와 접근시도에 대한 감사기록기능을 갖추고 있는 package이다. 이는 시스템자원에 대한 각 사용자의 접근권한을 토대로 사용자의 식별, 권한의 검사 및 감사기록을 수행한다[Berg84].

◆ ACF2(Access Control Facility 2)는 mainframe, 디스크화일, 테이프화일등의 접근제어를 위해 개발되어 은행, 기업등에서 광범위하게 활용되고 있다[Perr84]. 이는 IBM MVS, VS1등의 운영체제와 호환성이 있는 package로서 시스템자원에 대한 비인가된 접근을 검사하며 보호된 명령어에 대한 접근규칙을 제공하여 보안환경에 맞는 data-set을 구성하도록 하는 유연성(flexibility)을 갖추고 있다.

3. UNIX 워크스테이션 환경의 감사증적 메카니즘

◆ Trusted DG/UX는 DoD/NSA의 요구수준을 만족하고 POSIX표준을 유지하며 성능하락을 8%이하의 보증을 목표로 TCSEC의 C2와 B1등급을 만족하는 두종류의 UNIX로 개발되었다[TDO 92]. 모든 보안정책은 TCB(Trusted Computing Base)에 의해 보호되는 한편, 감사기능은 보안정책의 위반사항을 검사, 보고하며 내부/외부의 위협을 기록, 탐지 및 격리시키기 위한 병렬기능과 주체/객체의 표시를 위한 감사 mask를 사용한다. 보안관련 사건(event)은 이러한 mask를 통해 감사되며 사건은 객체에 대한 주체의 접근 또는 보안상태의 변경을 의미한다. 예를 들어 login, 패스워드의 변경 등이 사건이 될 수 있으며 이러한 사건은 type과 원인코드로 정의되어 감사의 대상이 된다.

◆ CMW(Compartmented Mode Workstation)는 미국 DIA의 요구에 의해 첨단수준의 보안기능과 상업성을 목적으로 개발되었으며 TCSEC의 B1등급을 요구하는 CMWRESQ(DIA Document DRS-2600-5502-86)를 수용하여 SUN 2/120의 UNIX 4.2 BSD환경에서 구현된 프로토타입이다[Cumm87]. 감사기능은 사건의 type과 사용자를 기반으로 감사대상을 정하고 정해진 사건이 발생하면 감사자료를 수집하여 저장하고 저장된 감사자료는 보안담당자에게 감사목적으로 제공된다.

4. 특성비교 및 분석

IBM환경의 경우, 시스템자원에 대한 접근책임을 제공하는 기록성(accountability)측면에서는 메카니즘에 따라 다른 특성을 갖고 있으나 시스템에 대한 접근시도, 사용, 보안침해 등의 감사증적을 생성하고 보고하는 감사능력측면에서는 유사한 특징을 보인다. 즉, data-set의 접근, 침해 및 변경을 식별하는 사건기록(event logging)을 이용하여 시스템관리자에게 보안사건을 보고하는 도구(tool)을 제공한다.

◆ RACF는 data-set의 보호를 위해 DSCB(Data Set Control Block)에 bit를 세워 시스템에 정의한다. 이는 방대한 data-set을 생성할 수 있으나 전체적인 보안전략을 세우기 어렵다. RACF의 보안전략은 전체적인 규칙정의가 아니라 요구수준에 따라 변경되는 사용자복귀(user exit)수준에서 구현되며 이러한 사용자 복귀는 사용자의 요구변화에 민감하여 보안상의 취약요소로 작용한다.

◆ ACF2는 모든 자원이 자동보호되고 사용자가 정한 수준에 따라 모든 접근이 감시되며 정해진 변수범위에서 시스템의 개입없이 새로운 보안전략을 구현할 수 있다. 또한, ACF2가 보호하는 자원을 풀기 위해서는 특별한 절차를 요구하는 접근제어기능을 제공하지만 감사기록이 비체계적이고 감사목적에 맞는 보고기능이 미흡하다는 지적이다[Perr84].

UNIX 워크스테이션 환경의 경우 Trusted DG/UX와 CMW는 미국방성표준(DoD standard)을 채택하고 TCSEC의 B1수준을 만족하는 등 보안정책측면에서 매우 유사한 특성을 보이는 반면, 감사레코드의 구성방법에서 다른 면모를 보인다.

◆ Trusted DG/UX는 POSIX 표준에 준용하는 접근제어수단을 제공하며 모든 보안정책은 kernel수준에서 구현된다. 또한, 시스템에서 일어나는 사건의 type과 원인코드를 시스템테이블에 mapping하여 감사레코드를 생성하고 감사증적에 기록한다.

◆ CMW는 보안담당자와 일반사용자를 엄격히 구분하여 kernel수준에서 보안정책을 실행하며 TFM(Trusted Facility Management)을 매개로 보안담당자, 시스템관리자 및 오퍼레이터의 권한을 제한하고 상호간에 견제하도록 한다. 또한, 시스템에서 일어나는 사건을 기반으로 감사레코드를 생성하는 관점은 Trusted DG/UX와 비슷하나 저장과정에서 차이를 보인다. 생성된 감사레코드는 임시기억장치에 기록되고 기록된 감사레코드는 background 모듈에 의해 축약(compression)되어 감사증적화일에 저장된다.

따라서, 위에 열거된 IBM환경의 감사증적 package는 기록성(accountability)측면을 중심으로 독자적 보안정책을 수행하며 UNIX계열은 모든 보안정책이 공인된 평가기준에 의거, kernel수준에서 변경/실행함에 따라 보다 안정된 면모를 보인다. 특히, UNIX계열은 고유의 개방적 속성에 기인하여 특정 단체/기업에 의해 발전되는 수준을 벗어나고 있으며 X/Open, USL, OSF등에 의해 표준안들이 제정되고 개방시스템(open system)이 대두되면서 IBM까지도 OSF에 가입하는 등 UNIX에 접근하는 경향을 보인다.

Ⅲ. 감사증적모델 제안

우리나라도 산·학·연을 중심으로 보안메카니즘에 대한 연구가 추진되고 있으나 감사증적에 대한 연구는 미진한 실정이다. 또한, 기존의 UNIX는 부분적인 감사능력을 갖추고 있으나 회계목적에 치중한 나머지 보안측면에서 많은 취약점이 노출되어 왔다.

따라서, 응용프로그램(application program)수준의 logging개념을 탈피하여 조직환경에 따른 보안목적을 수용하고 범용성을 갖춘 운영체제수준의 추적능력의 확보는 시급한 과제가 되고 있다.

1. 제안목적 및 설계범위

본 연구에서의 감사증적모델은 UNIX 워크스테이션 환경에서의 신뢰성, 안전성, 범용성, 확장성 및 편의성을 갖추고 시스템에서 일어나는 중요한 사건을 기록하는 한편, 기록된 감사자료를 보안담당자에게 효율적으로 제공함을 목적으로 하며 모델의 설계범위를 다음과 같이 제한한다.

◆ 시스템에서 일어나는 사건을 시스템테이블에 정의된 사건과 mapping하여 감사기록을 생성하는 방법을 사용하며 감사기록에는 중요한 시스템자원에 대한 접근실패/성공을 모두 포함하도록 한다.

◆ kernel수준의 감사기록을 생성하며 감사증적화일을 보호하도록 한다.

◆ 운영체제의 변경을 최소화하며 모듈단위로 설계하도록 한다.

◆ 모든 보안정책은 시스템테이블에 보관하며 보안담당자에 의해 관리되도록 한다.

◆ 디스크공간의 효율적 이용을 위해 감사증적화일은 축약된 형태로 보관되도록 하며 보안담당자만이 감사증적화일을 사용하도록 한다.

◆ 감사기록의 조회를 위해 사용자 id, 객체 id, 객체의 보안수준, 사건유형 등의 key를 조합하여 필요한 감사기록만을 관계자에게 제공하도록 한다.

2. 요구특성

감사증적모델의 설계를 위해서는 먼저, 적합한 관리환경과 보안메카니즘이 전제되어야 하고 보안메카니즘은 식별 및 인증, 접근제어, 안전한 경로(trusted path), 시스템 무결성 등을 만족하여야 한다.

◆ 관리환경으로는 UNIX systemV계열의 운영체제를 갖추고 UNIX BSD 4.2에 구축되어 있는 system call중 syscall()과 같은 기능이 제공되어야 한다. syscall()은 system call을 호출하는 모든 프로세스의 제어를 반드시 중재하도록 고안된 시스템 명령어의 일종이다[Picc87]. 또한, 보안담당자는 감사증적을 분석하고 Superuser의 권한을 허용/취소하며 사용자/group id, 패스워드, 보안수준 및 객체에 대한 접근모드를 지정/변경하는 등의 관리기능을 수행할 수 있어야 한다.

◆ 보안메카니즘은 사용자의 식별을 위한 정보와 보안수준 및 권한에 관한 정보를 포함하는 인증자료를 유지하여야 한다. 패스워드는 패스워드테이블(/etc/passwd)에 암호화된 형태로 보관되고 보안정보테이블(/etc/usec_info)에는 패스워드의 유효기간, 사용자의 허가수준, 주체와 객체의 권한 bit string 등이 요구된다.

◆ 접근제어는 DAC, MAC, 객체재사용(object reuse) 및 Labeling을 포함하여야 한다. DAC정책에서는 접근제어리스트(access control list)를 사용하며 MAC정책에서는

객체의 비밀수준(sensitivity)과 주체가 갖는 허가수준(clearance)에 근거하여 객체에 대한 접근을 허용하여야 한다. 객체의 재사용성 문제는 주기억장치의 경우 재배치할 때 잔여정보를 지우고 보조기억장치는 할당될 때 지워야 한다. Labeling은 각 주체와 객체의 비밀수준을 나타내며 비밀수준은 자료에 대한 실제 보안등급(security level)을 표현한다. Labeling 메카니즘은 보안수준을 생성, 유지 및 표시할 수 있어야 하며 프로세스도 Labeling되어야 한다. 프로세스의 보안정보는 프로세스의 보안수준(PDLS: Process Data Sensitivity Label), 접근하려는 객체에 대한 보안정보 및 flag를 포함하여 시스템의 프로세스테이블(/etc/ps_info)에 저장되도록 한다. 또한, 실행화일의 inode(index_node)에는 화일에 관한 type, 실행권한, 디렉토리 엔트리의 갯수 등이 포함되어야 한다.

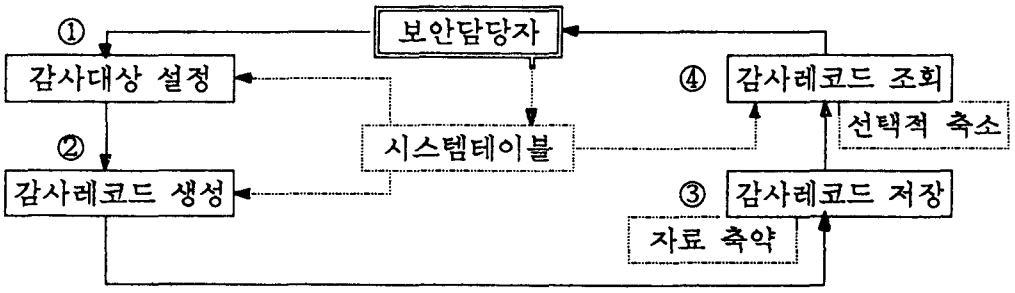
◆ 안전한 경로는 감사기록과 비밀수준의 변경을 위해 갖추어야 하며 이는 불법적인 속임수(spoofing)를 방지하도록 한다.

◆ 운영체제의 정확한 복제(correct copy)를 보증하는 부팅(booting) 메카니즘이 요구된다. 이를 위해 운영체제와 boot device의 식별자가 내장되고 보안담당자는 이러한 식별자의 값을 이용하여 워크스테이션을 부팅하여야 한다. 이는 비인가 사용자가 테이프나 디스켈에 저장된 운영체제로 부팅하는 것을 방지한다.

3. 감사증적모델 설계

3.1 구성체계

본 모델은 기본적으로 ① 보안담당자가 감사대상을 설정하고 ② 설정된 감사대상을 기준으로 사용자의 프로세스활동을 검사하여 감사레코드(audit record)를 kernel수준에서 생성하여 ③ 저장하며 ④ 보안담당자에게 감사레코드를 선택적으로 조회하도록하는 4가지의 단계로 이루어진다. <그림 1>의 시스템테이블은 시스템의 사건정보, 프로세스의 mapping정보, 객체에 대한 사용자의 접근권한을 정의한 보안정보, 프로세스의 권한, 객체의 보안정보 등 감사증적과 관련되는 모든 정보를 제공하며 kernel수준에서 보호된다. 자료축약(data compression)은 감사레코드의 저장공간을 줄이는 기능을 제공한다. 또한, 선택적 수집은 보안담당자가 원하는 감사레코드만을 선택하는 기능을 제공한다.



<그림 1> 감사증적모델의 구성체계

◆ 감사대상의 설정방법은 ① 사용자의 login, 객체에 대한 접근시도 등 감사대상이 되는 사건을 규정하고 ② 규정된 사건과 관계되는 system call, 서브루틴, 프로그램 등의 모든 루틴을 선택하며 ③ 사건이 발생했을 때 생성할 감사레코드의 종류를 결정한 다음 ④ 결정된 모든 사항들은 선택된 루틴별로 재배열하고 각 루틴들이 UNIX

에서 제공하는 고유번호(루틴들의 id)와 mapping이 가능하도록 하는 코드들을 시스템 테이블에 구축한다.

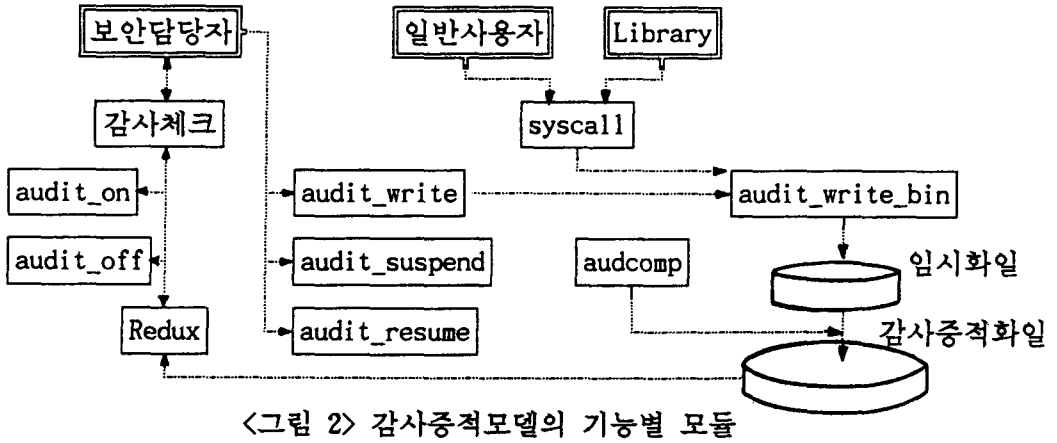
◆ 사용자 프로세스가 호출하는 루틴은 system call, 서브루틴 및 프로그램으로 분류되며 각 루틴에 대한 감사여부를 판정하고 감사레코드를 생성하는 방법은 다음과 같이 3가지로 나누어 진다. ① 프로세스가 호출하는 루틴이 system call인 경우에는 감사여부의 판정 및 감사레코드의 생성기능을 syscall모듈에서 수행하도록 한다. ② 프로세스가 호출하는 system call이 syscall모듈을 경유한 후, 되돌아 오지 않음에 따라 감사레코드를 생성할 수 없는 특수한 system call이면 해당되는 system call자체에서 감사여부를 판정하고 감사레코드의 생성기능을 수행하도록 한다. 이러한 system call에는 execve(), open(), maccess()등이 포함된다. ③ 프로세스가 호출하는 루틴이 충분한 신뢰성을 갖춘 서브루틴/프로그램이면 보안담당자는 해당되는 루틴내부에서 감사레코드의 생성기능을 독자적으로 수행하도록 하여 루틴내부에 존재하는 여러가지 system call이 kernel수준의 감사기능을 우회하도록 한다. 워드프로세서(WP)와 같은 프로그램은 kernel수준의 감사가 불필요하기 때문이다. 한편, 독자적인 감사기능이 내장된 루틴에 대해서는 보안담당자가 특별한 권한(NO_AUDIT)을 부여함으로써 kernel수준의 감사기능을 우회하도록 한다.

◆ 생성된 감사레코드는 임시화일(bin)에 저장되고 임시화일은 background프로그램에 의해 자료축약(data compression)과정을 거쳐 감사증적화일에 최종적으로 저장되도록 한다. 자료축약 알고리즘은 UNIX에서 제공하는 루틴(compact)을 이용한다.

◆ 보안담당자가 감사레코드를 활용하기 위해서는 저장된 레코드에 대한 선택적 축소(selective reduction)방법을 사용한다. 선택적 축소방법은 감사증적화일에 저장된 감사레코드를 uncompress하여 필요한 사항만을 조회하도록 하는 기술이다.

3.2 기능별 모듈

기능별 모듈은 보안담당자의 관리활동과 관련되는 모듈, 감사레코드를 생성·저장하는 모듈, background모듈 등 3가지로 분류된다(<그림 2>참조).



<그림 2> 감사증적모델의 기능별 모듈

보안담당자와 관련된 모듈은 감사체크(audit_check), audit_on, audit_off, Redux, audit_write, audit_suspend, audit_resume 등이 있으며 감사체크모듈과 Redux모듈은 보안담당자와의 인터페이스 및 확장성을 고려하여 프로그램형태를 취하고 나머지는 처

리속도를 고려하여 system call형태를 갖도록 한다. 감사레코드를 생성하고 저장하는 모듈로는 syscall 및 audit_write_bin이 있으며 audit_write_bin모듈은 kernel내에서의 호출을 위해 서브루틴의 형태를 취하도록 한다. 또한, background모듈로는 임시화일(bin)에 저장된 감사레코드를 감사증적화일에 순차적으로 축약시키는 audcomp모듈이 있다. audcomp모듈은 정보시스템이 부팅되면 작동되어 power off될 때까지 활동하는 프로그램(daemon)의 형태를 취하도록 한다.

모든 모듈들은 실행화일로 존재하여 이를 호출하려면 실행권한이 요구되어야 한다. 실행권한은 AUDIT_SET, AUDIT_WRITE, AUDIT_SUSPEND, NO_AUDIT 등이 있으며 보안담당자에 의해 실행화일의 inode에 저장되도록 한다.

3.2.1 감사체크모듈

감사체크모듈은 보안담당자와의 인터페이스기능을 수행하며 호출하는 프로세스는 AUDIT_SET권한을 가져야 하며 다음과 같은 기능을 제공하여야 한다.

◆ 정보시스템의 감사기능을 개시(auditing on)한다. 또한, 감사기능이 작동되면 감사레코드가 저장될 임시화일(bin)을 생성하여 자료축약이 가능하도록 한다.

◆ 정보시스템의 감사기능을 중단(auditing off)한다. 또한, 감사기능이 중단될 때 EOF(End_Of_File)표시를 현행 bin에 기록하고 bin을 close한다.

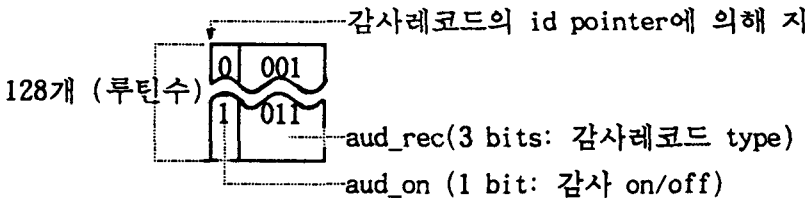
◆ 감사기능의 동작중 인터럽트없이 감사증적 화일의 교체와 감사중인 루틴을 변경한다. 보안담당자가 감사증적화일의 변경을 요청하게 되면 감사체크모듈은 감사증적화일의 filename을 현행 bin에 기록하도록 하여 interrupt없이 감사가 계속되도록 한다.

◆ 현재 감사중인 사건을 조회한다. 보안담당자가 사건조회를 요청하면 감사체크모듈은 Redux모듈을 호출하여 감사자료를 제공하도록 한다.

감사체크모듈에서는 감사될 루틴과 감사레코드 type을 결정하는 AUDIT_FLAGS테이블을 갖는다(<그림 3>참조). AUDIT_FLAGS테이블은 감사레코드의 header에 있는 id field(<그림 4>참조)를 pointer로 하여 지정되며 이러한 id field는 시스템에 정의된 모든 루틴의 고유번호를 의미한다.

본 연구에서는 system call, 서브루틴, 프로그램 등 시스템에 정의된 모든 루틴중 감사대상의 루틴수를 128개의 범위로 제한하여 AUDIT_FLAGS테이블을 구성한다. 예를 들어 AUDIT_FLAGS테이블의 26번째에 있는 aud_on bit가 "1"이고 aud_rec이 "011"이면 <표 1>에 나타난 바와 같이 고유번호(id field)가 26인 rename()은 감사되고 감사레코드는 3번째 type인 audit_sir(보안정보레코드)를 의미하게 된다(<그림 4>참조).

<표 1>은 AUDIT_FLAGS테이블의 aud_on bit와 감사될 루틴들이 어떻게 mapping되는지를 나타내는 예이다. 보안담당자는 시스템여건을 고려하여 이러한 aud_on bit를 적절히 변경하여야 한다.



<그림 3> AUDIT_FLAGS테이블의 구조

3.2.2 audit_on모듈

audit_on모듈은 정보시스템에 감사기능이 작동되도록 하는 한편, 감사증적화일을 교체하거나 감사중인 루틴을 변경하는 기능을 수행한다. 이러한 모듈을 호출하는 프로세스는 AUDIT_SET권한이 있어야 하며 다음과 같은 3개의 인수로 호출되도록 한다.

◆ filename pointer: 감사레코드가 기록될 감사증적화일을 지정한다.

◆ bit mask: 128 bit 배열에 대한 pointer이며 배열의 각 bit(bit가 "1"이면 감사실행)는 감사될 루틴을 mapping한다. 이러한 bit들은 감사체크모듈의 AUDIT_FLAGS테이블에 저장된다.

◆ 감사될 프로세스 id의 하한값(MINPID): 이는 kernel수준의 감사에서 사용되며 시스템 프로세스와 background 프로그램(daemon)은 MINPID보다 낮은 값을 갖는다.

<표 1> 감사될 루틴과 AUDIT_FLAGS 테이블과의 mapping

고유번호	1	2	25	26	128
루틴	halt	fork()	open()	rename()	maccess()
aud_on bit	0	1	0	1	1
aud_rec	001	001	011	011	101

3.2.3 audit_off모듈

audit_off모듈이 호출되면 감사기능이 중단되어 더이상의 감사레코드가 생성되지 않도록 다음과 같은 동작을 수행하도록 한다.

◆ audit_off모듈을 호출하는 프로세스의 AUDIT_SET권한을 검사한 후, AUDIT_FLAGS 테이블의 모든 aud_on bit를 지운다.

◆ 감사증적화일을 close한다.

3.2.4 audit_write모듈

kernel수준의 감사기록이 불필요하거나 충분한 신뢰성을 구비한 서브루틴/프로그램에 대해서는 중복된 감사레코드의 생성을 방지하여야 한다. 이를 위해 보안담당자는 다음과 같은 절차를 수행하여 이러한 서브루틴/프로그램이 audit_write모듈을 호출하도록 한다.

◆ 해당되는 서브루틴/프로그램의 source code에서 감사기록이 필요한 영역을 설정하여 audit_write모듈을 호출하는 명령어를 추가한다.

◆ 설정된 영역의 앞부분에는 kernel수준의 감사를 지연시키는 audit_suspend기능을 추가하고 뒤에는 kernel수준의 감사를 재개시키는 audit_resume기능을 추가한다.

◆ 수정된 서브루틴/프로그램의 실행화일에 AUDIT_WRITE와 AUDIT_SUSPEND권한을 부여한다.

한편, 감사기록이 전혀 불필요한 서브루틴/프로그램에 대해서는 source code의 첫번째 line에 audit_suspend기능을 추가하고 실행화일에 AUDIT_SUSPEND권한을 부여한다. 또한, Mistress DBMS와 같이 독자적인 감사 subsystem을 갖춘 프로그램의 경우에는 보안담당자가 NO_AUDIT권한을 부여하여 불필요한 감사레코드의 생성을 방지한다.

audit_write모듈을 호출하는 프로세스는 AUDIT_WRITE권한을 갖추어야 하며 호출시의 인수로는 감사레코드에 대한 pointer를 필요로 한다. audit_write모듈은 다음과 같은 사항을 기초로 하여 감사레코드에 추가정보를 수록하고 audit_write_bin모듈을 호출

함으로써 감사레코드를 임시화일(bin)에 기록하도록 한다.

◆ 감사레코드의 id field는 kernel constant인 MINAPPIDS보다 크거나 같아야 하며 MINAPPIDS는 응용프로그램수준에서 취할 수 있는 프로세스 id의 하한값이다.

◆ id field는 AUDIT_FLAGS테이블의 index로 사용되며 AUDIT_FLAGS테이블은 감사레코드 type을 제공한다.

◆ 감사레코드는 호출하는 루틴이 제공하는 보안정보와 시스템테이블에서 얻은 정보를 토대로 구성된다. 감사레코드의 구조는 audit_header를 공유하고 <그림 4>과 같이 5가지로 나누어지며 audit_txt와 audit_lbl은 본 모듈에서 구성된다. 또한, 응용프로그램의 침해로 부터 audit_header를 보호하기 위해 length와 id를 제외한 모든 field는 overwrite되어야 한다.

```
struct audit_header                    /* Header 구조          */
     { short id;                       /* 감사될 루틴 번호     */
       short euid;                     /* 유효 사용자 id       */
       short ruid;                     /* 실제 사용자 id       */
       short pid;                      /* 프로세스 id          */
       short par_pid;                  /* parent 프로세스 id   */
       short term_pid;                 /* 단말기 id            */
       short {long tv_sec;
              long tv_usec;} time;     /* 기록된 시간          */
       short length; };                /* variable 레코드길이  */

struct audit_sar                       /* 표준 감사레코드  */
     { struct audit_header aud_head;   /* header           */
       int u_arg[3];                   /* system call 인수 */
       int rval1;                      /* 첫번째 return값  */
       int rval2;                      /* 두번째 return값  */
       char uerror; };                 /* error code       */

struct audit_sir                       /* 보안정보 레코드   */
     { struct audit_header aud_head;   /* header            */
       struct i_node sir;              /* i_node구조        */
       int rval1;                      /* 첫번째 return값   */
       char uerror;                    /* error code        */
       short fdes; };                  /* 화일 설명자       */

struct audit_txt                       /* Text 감사레코드      */
     { struct audit_header aud_head;   /* header               */
       char *text; };          /* 실패된 password/경로/명령어 */

struct audit_lbl                       /* Label 감사레코드  */
     { struct audit_header aud_head;   /* header            */
       int pid;                        /* 프로세스 id       */
       struct i_node lbl1;             /* 첫번째 label      */
       struct i_node lbl2;             /* 두번째 label      */
       char *text; };                  /* 관련 데이타       */
```

<그림 4> 감사레코드의 구조

3.2.5 audit_suspend모듈

audit_suspend모듈을 호출하는 프로세스는 AUDIT_SUSPEND권한을 갖추어야 하며 다음과 같은 기능을 수행하도록 한다.

◆ audit_suspend모듈을 호출하는 프로세스에 대한 kernel수준의 감사를 중단시킨

다. 그러나 audit_write모듈을 호출하는 프로세스가 수행하는 감사는 중단되지 않는다. 이는 독자적인 감사기능과 신뢰성을 갖춘 프로세스를 허용하기 때문이다.

◆ AUDIT_SUSPEND권한이 확인되면 프로세스테이블에 있는 AUD_SUSPEND_FLAG를 set한다. AUD_SUSPEND_FLAG는 syscall모듈에서 검사된다.

3.2.6 audit_resume모듈

호출하는 프로세스는 AUDIT_SUSPEND권한을 갖추어야 하며 audit_resume모듈은 호출하는 프로세스에 대한 감사를 재개하도록 한다. AUDIT_SUSPEND권한이 확인되면 해당되는 AUD_SUSPEND_FLAG를 unset한다.

3.2.7 syscall모듈

syscall모듈은 프로세스가 호출하는 모든 system call을 검사하고 감사여부를 결정하여 필요시 감사레코드를 생성하도록 한다. system call을 호출하는 모든 프로세스의 제어는 반드시 syscall모듈을 경유한다.

◆ syscall모듈은 호출되는 모든 system call을 다음과 같은 순서로 검사하여 감사여부를 결정하도록 한다. ① 이 system call이 감사되어야 하는가? 이는 system call의 고유번호를 index로 하여 AUDIT_FLAGS테이블의 aud_on bit를 참조한다. ② 이 프로세스는 감사되어야 하는가? 이는 감사될 프로세스 id의 하한값을 제공하는 MINPID(audit_on모듈에서 지정)와 비교한다. ③ 감사가 중단된 프로세스인가? 이는 프로세스테이블에 있는 AUD_SUSPEND_FLAG를 검사한다. ④ 이 프로세스는 NO_AUDIT권한을 가지고 있는가?

위의 검사가 모두 통과되면 이 system call은 감사기록을 생성한다는 의미이며 이러한 경우 flag(do_audit)가 "1"로 set되도록 한다.

◆ system call의 검사가 끝나면 정상적인 system call이 호출된다. system call이 수행된 다음, return되었을 때의 flag(do_audit)가 "1"이면 return values와 error code를 감사레코드에 저장하고 감사레코드 type을 결정하여 audit_write_bin모듈을 호출하도록 한다. syscall모듈이 종료되면 제어는 사용자 프로세스로 되돌아 간다.

◆ syscall모듈을 통과하는 일부 system call은 다음과 같은 이유로 독자적인 감사레코드를 생성한다. ① 어떤 system call은 syscall모듈로 되돌아 오지 않는다. 예를 들면 reboot(), execve()등이 그것이다. ② audit_off모듈 등의 system call은 실행 후 감사될 수 없다. 따라서, execve(), open(), close(), namei(), audit_off, audit_suspend, maccess등의 system call은 syscall모듈에서 감사되지 않으며 각 모듈별로 다음과 같은 기능을 추가하여야 한다.

① 감사여부의 결정을 위해 syscall모듈과 같은 4가지 검사를 실행한다.

② 감사레코드를 구성하여 audit_write_bin모듈을 호출한다.

3.2.8 audit_write_bin모듈

감사기능을 수행하는 모든 kernel모듈은 감사레코드를 임시화일(bin)에 기록하기 위해 본 모듈을 호출한다. 본 모듈에서는 먼저 유효한 bin을 찾아서 감사레코드를 기록하지만 bin이 없으면 확보될 때까지 모든 감사를 지연(suspend)시킨다. 보안담당자는 감사자료가 기록될 유효공간을 확보하여야 한다.

3.2.9 audcomp모듈

audcomp모듈은 bin에서 감사레코드를 추출하여 감사증적화일에 압축시키는 기능을 background에서 수행하며 압축알고리즘은 축약할 자료량/중복성을 고려해서 채택하여야 한다. 또한, audit_write_bin모듈이 감사레코드를 bin에 기록하는 동안, audcomp모듈은 bin에 기록된 자료를 순차적으로 읽고 축약하여 보안담당자가 지정한 감사증적화일에 기록하도록 한다.

◆ 시스템이 부팅(booting)되면, audcomp모듈은 background 프로그램(daemon)으로 시작되어 다음과 같은 순서로 동작을 수행하도록 한다. ① 감사세션(audit session)에 대한 축약책임을 담당할 child프로세스(audcomp_c)를 fork한다. 감사세션이 끝나면 child프로세스는 죽게되며 audcomp모듈은 새로운 child프로세스를 fork한다. 감사세션은 정보시스템에 감사기능이 시작(audit on)되어 중단(audit off)될 때까지의 기간을 의미한다. ② audcomp_c는 bin화일을 찾을 때까지 looping을 시작하고 화일을 찾으면 레코드를 읽기 시작한다. ③ 첫 레코드(AUD_SWITCH_FNAME)에 있는 filename을 통해 감사증적화일을 open한다. ④ audcomp_c는 나머지 레코드를 읽고 축약하여 감사증적화일에 기록하는 looping을 계속한다. ⑤ audcomp_c가 end of bin(AUD_SWITCH_BIN)표시를 만나면 현행 bin을 close하고 삭제시킨 후 다음 bin을 open하여 looping을 계속한다. ⑥ audcomp_c가 EOF(End_Of_File)를 만나면 단순히 읽기를 재시도한다.

◆ 감사기능이 중단(audit_off)된 경우, audcomp_c가 EOF를 만나면 bin과 감사증적화일을 close하고 bin을 삭제하고 죽는다. 이 때, audcomp모듈은 새 child프로세스를 fork하고 audcomp_c프로세스가 다시 작동된다.

◆ 감사증적화일이 재지정된 경우 먼저 현행 감사증적화일을 close하고 현행 bin과 관련되는 변수와 루틴의 위치를 제외하고 축약루틴의 상태를 reset한다. 이어서 새로운 감사증적화일을 open하여 현행 bin의 레코드를 읽어서 기록하기 시작한다.

3.2.10 Redux 모듈

Redux모듈은 감사증적화일에서 축약된 감사레코드를 요구수준에 따라 조회하고 uncompress하여 보안담당자에게 제공하는 기능을 수행하도록 한다. 또한, uncompress 작업과 선택적 축소(selective reduction)작업을 병렬로 수행하며 output을 화면에 표시하거나 화일에 저장하도록 한다.

◆ 감사증적화일의 축약된 레코드를 다음과 같은 순서로 uncompress한다. ① Redux 모듈은 child프로세스를 fork한다. child프로세스에는 pipe가 형성되고 감사증적화일의 filename이 전수된다. ② child프로세스는 축약된 감사레코드를 읽고 uncompress하여 pipe에 기록한다.

◆ pipe에 기록된 자료는 다음과 같은 순서로 축소작업이 이루어 진다. ① Redux모듈은 먼저 pipe로부터 audit_header를 읽어 id를 얻는다. 이러한 id를 통해 AUD_FLAGS 테이블의 감사레코드 type을 얻고 하나의 감사레코드를 읽는다. ② 감사레코드의 프로세스 id와 프로세스테이블의 프로세스 id가 일치될 때까지 linked list를 찾는다. 일치하는 엔트리가 있으면 엔트리의 stack에 레코드를 push하고 없으면 새로운 엔트리를 생성한다. 새로운 엔트리에는 사용자 id, 프로세스 id, 현행 working directory를 set하고 stack을 clear하여 레코드를 push한다. ③ push된 레코드를 조사하여 필요한 경우 working directory, 화일정보테이블 등을 linked list의 엔트리에 update한다. ④ 엔트리의 레코드를 조사하여 보안담당자에게 제공할 사건정보이면 레코드를

pop하고 사건레코드를 구성한다. 사건레코드는 시스템테이블을 통해 사람이 알아 볼 수 있는 stream으로 변형시킨다. ⑤ 보안담당자가 지정한 조회 basis를 토대로 변형된 stream을 filter한 다음, 화면에 표시하거나 화일에 저장한다.

Ⅳ. 결 론

본 연구에서는 개방형 시스템과 정부, 군기관 등의 보안측면을 고려하고 선진국에서 개발된 감사증적 메카니즘의 주요기능을 분석하여 UNIX워크스테이션 환경에서 응용할 수 있는 감사증적모델을 제안하였다. 또한, UNIX고유의 개방적 속성에 기인하는 보안상의 취약성 개선과 신뢰성있는 보안감사를 위한 근거제공에 역점을 두었으며 TCSEC의 C2등급을 기본적으로 수용하고 상위등급으로의 조정이 용이하도록 하였다.

제안된 감사증적모델은 보통의 UNIX워크스테이션 환경에 응용될 수 있으며 특히, 행정전산망의 UNIX환경은 kernel수준의 감사증적기능이 요구되고 있다. 또한, 본 연구내용은 보안측면만을 강조하였기 때문에 정보시스템에 대한 성능(performance)과 용량(capacity)측면은 고려되지 않았으며 이러한 문제는 구현과정에서 연구되어야 할 것이다.

참 고 문 헌

[Berg84] B. Berg and H. Leenaars, "Advanced Topics of a Computer Center Audit," Computers & Security, Vol.3, 1984, pp. 171-185.

[Cumm87] P.T. Cummings, D.A. Fullam, M.J. Goldstain, M.J. Gosselin, M.J. Picciotto, J.P.L. Woodward, J. Wynn, "Compartmented Mode Workstation: Results through Prototyping," 1987 IEEE Symposium on Security and Privacy, April 1987, pp. 2-12.

[DRG 91] Datapro Research Group, Information Technology Security Evaluation Criteria(ITSEC), McGraw-Hill, 1991.1.

[GISA89] German Information Security Agency, Criteria for the Evaluation of Trustworthiness of Information Technology System, GISA, 1989.

[Glig86] V.A. Gligor and E.L. Burch, "On the Design and The Implementation of Secure XENIX Workstation," Proc. of IEEE Symposium on Security and Privacy, April 1986, pp. 102-117.

[Moll84] Carol G. Molloy, "Improving CICS Controls with GUARDIAN," AUERBACH, Auerbach Publishers Ins., 1984.

[Perr84] William E. Perry, "Reviewing the Selection of Data Security Software," AUERBACH, Auerbach Publishers Ins., 1984.

[Picc87] J. Picciotto, "The Design of an Effective Auditing Subsystem," 1987 IEEE Symposium on Security and Privacy, April 1987, pp. 13-22.

[TDO 92] Trusted DG/UX Overview, Data General Corporation, 1992.2.