침입 복구 및 대응 시스템을 위한 실시간 파일 무결성 검사

(Real-Time File Integrity Checker for Intrusion Recovery and Response System)

전 상 훈 † 허 진 영 † 최 종 선 †† 최 재 영 †††
(Sanghoon Jeun) (Jinyoung Hur) (Jongsun Choi) (Jaeyoung Choi)

요 약 파일 무결성 검사는 시스템 자원의 안정성 여부를 판단할 수 있는 가장 확실한 방법이지만, 관리자의 경험과 능력에 따라 검사 시간과 효과에 많은 차이가 있을 수 있다. 따라서 가능한 빠른 대응을 필요로 하는 침입 복구 및 대응 시스템에 바로 사용하기에는 적합하지 않으며, 또한 손상 자원의 복구에 필요한 자원의 상태 정보는 수집 가능하지만 침입을 차단하기 위해 필요한 침입 행위 주체에 대한 정보는 수집할 수 없다. 본 논문에서는 위의 두 가지 문제를 해결하기 위해 시스템 호출 감시와 파일 무결성 검사를 연동하여 실시간으로 파일의 무결성을 검사하는 rtIntegrit을 제안한다. rtIntegrit은 SysWatcher라는 시스템 호출 검사 도구를 사용하여 요청된 행위를 항상 감시한다. 만약 지정된 파일에 이상 동작이 나타나면 이의 변화에 대하여 실시간으로 무결성 검사를 수행하도록 하고, 해당 관련된 프로세스의 정보를 수집하고 보고함으로서 침입 차단에 활용하게 한다. 또한 IDMEF 형식의 표준으로 감사 자료를 생성하여, 침입 대응 및 복구 시스템들과 쉽게 연계할 수 있다.

키워드 : 침입 복구, 침입 대응, 침입 감사, 파일 무결성

Abstract File integrity checking is the most reliable method to examine integrity and stability of system resources. It is required to examine the whole data whenever auditing system's integrity, and its process and result depends on administrator's experience and ability. Therefore the existing method is not appropriate to intrusion response and recovery systems, which require a fast response time. Moreover file integrity checking is able to collect information about the damaged resources, without information about the person who generated the action, which would be very useful for intrusion isolation. In this paper, we propose rtIntegrit, which combines system call auditing functions, it is called SysWatcher, with file integrity checking. The rtIntegrit can detect many activities on files or file system in real-time by combining with SysWatcher. The SysWatcher audit file I/O relative system call that is specified on configuration. And it can be easily cooperated with intrusion response and recovery systems since it generates assessment data in the standard IDMEF format.

Key words : Intrusion Recovery, Intrusion Recovery, Intrusion Auditing, File Integrity

1. 서 론

컴퓨터 시스템 및 네트워크의 기술이 발전되고 보급되면서, 그에 따른 역기능으로 해킹, 바이러스 등으로 인한 침해 사고도 증가하고 있다. 그리고 인터넷의 발달로 인하여, 어느 한곳에 피해가 집중되지 않고 네트워크로 연결된 모든 곳을 대상으로 급속하게 퍼져나가 전 세계에 동일한 시기에 유사한 방법으로 발생하는 특징이 있다. 미국 CERT(Computer Emergency Response Team)에서 발표한 시스템 및 네트워크에 대한 취약점 및 침해 사고에 대한 통계에 따르면, 그 피해 정도가 매년 2배 이상 증가하고 있다[1]. 국내 침해사고의 증가 추세를 알아보기 위해 한국정보보호진흥원의 통계를 살펴보면, 2002년 총 침해사고 발생회수가 15,192건인데

· 이 논문은 2002-3년도 한국학술진흥재단의 선도연구자 지원사업의 지원(KRF-2002-D20498)으로 연구되었습니다.
† 비 회 원 : (주)엠엠씨테크놀로지 연구원
shjeun@mmctech.com
jyhur@mmctech.com
†† 비 회 원 : 숭실대학교 컴퓨터학과
jschoi@ss.ssu.ac.kr
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
choi@ssu.ac.kr
논문접수 : 2004년 6월 4일
심사완료 : 2005년 2월 24일

반해 2003년에는 8월까지의 침해사고가 19,903건으로 미국과 비슷한 증가 추이를 보이고 있다[2].

반면 이러한 침해 사고를 줄이기 위한 노력도 지속적으로 이루어지고 있다. 근본적으로 새로운 기술이 개발될 때마다 새로운 취약점이 발생할 수 있으며, 새로운 취약점은 바로 침해사고로 연결될 수 있다. 보안 시스템은 취약점이 노출되지 않게 하거나, 취약점이 노출되어 공격 여부를 검사하는 기능을 제공하지만, 현재의 보안 시스템은 일단 취약점이 노출되어 공격이 시작된 후에는 별다른 대응 수단이 없다. 다만 미리 알려진 공격 방법이라면 시스템 또는 네트워크 관리자가 정해진 대응 절차에 따라 대응 작업을 수행하게 되지만, 그렇지 못한 경우에는 완벽하게 대응하지 못한다[3].

침입 대응은 매우 넓은 범위를 포괄하고 있어서, 이에 관련된 연구들은 방화벽, 침입 탐지 시스템 등으로 구성된 보안 체계위에 이들 시스템 연계할 수 있는 기반 구조를 만들고, 이를 바탕으로 침입으로 판단되는 공격 행위가 더 이상 진행하지 못하도록 한다. 따라서 여러 보안 시스템이 함께 연동할 수 있는 기반 환경을 구성하는데 연구가 진행되어 왔다[4-8]. 국내 침입 탐지 시스템 중 일부도 침입 관련 세션 종료와 같은 부분적인 침입 대응 기능을 가지고 있다[9,10]. 그러나 시스템 무결성 검사, 바이러스 감지 등의 기능들의 연계는 피상적인 수준의 접근만 진행되었다[11].

본 논문은 공격으로 이미 손상된 시스템을 복구하는데 초점을 맞추었다. 시스템의 복구를 위해서는 피해 범위의 측정, 안전한 저장소, 이를 위한 관리 정책 등을 필요로 한다. 우선 피해 범위 측정을 능력을 높이기 위하여 기존에 사용되던 파일 무결성 검사 도구를 개선하여 실시간으로 파일 및 파일 시스템의 변조 유무를 판단하는 도구에 대한 연구를 수행하였다. 현재의 파일 무결성 검사 도구들은 자원의 변경 유무를 판단할 수 있는 자료만 제공하는 일종의 침입 탐지 시스템이다. 파일 무결성 검사는 공격 행위를 비롯한 여러 행위의 진행 여부에 관계없이 수동적으로 검사 도구를 수행시켜야만 감사 자료만 제공한다. 따라서 실제 진행 중인 공격에 대해서는 어떠한 대응도 할 수 없다. 또한 행위의 주체 등에 관련된 감사 자료 또한 제공하지 않으므로, 관리자가 실제 침입에 의한 변경된 것인지 정상적인 방법으로 합당한 권한에 의하여 변경된 것인지 판단하기 어렵다[12,13].

본 논문은 기존의 파일 무결성 검사도구가 가지는 이런 문제점들을 공개 프로젝트로 개발된 Integrit을 기반으로 보완하였다. Integrit은 파일 무결성 검사 도구 중 비교적 크기는 작지만, 필요한 기능을 대부분 가지고 있으며, 공개 프로젝트로 개발되어 변형이 용이하다. 수동적 탐지 방법을 자동화시키고, 실시간으로 감사하도록 하여 진행 중인 공격에 대응할 수 있게 하였으며, 표준화된 감사 자료를 생성하기 위하여 IDMEF(Intrusion Detection Message Exchange Format)를 채택하고, 부족한 행위 주체에 관한 자료도 제공하도록 하였다. 본 논문에서 제안하는 파일 무결성 검사 도구를 rtIntegrit (real-time Integrit)이라고 하였다. rtIntegrit의 실시간 기능 및 부가 정보 제공을 위하여 커널 내부에서 시스템 호출 감시 모듈을 개발하여 연동하였다. 커널에서 시스템 호출 감시하는 모듈을 SysWatcher라고 하였다. SysWatcher는 파일 접근에 관련된 시스템 호출들을 감시하여 이상 행위를 수행하는 프로세스에 관련된 정보를 rtIntegrit에게 제공한다. 그리고 구현된 rtIntegrit과 SysWatcher의 성능 부담을 파악하기 위하여 리눅스 커널 2.4.20에서 LMBench 2.0.4으로 성능을 측정하였다. 실험 결과 SysWatcher의 부하는 크게 증가하지 않았음을 확인하였다.

본 논문은 다음과 같이 구성되었다. 우선 2장 관련 연구에서는 기존의 침입 대응 방법과 무결성 검사도구, IDMEF, 그리고 시스템 호출 감시 방법을 중심으로 정리하였다. 3장에서는 rtIntegrit의 개발 요구사항을 관련 연구에서 정리한 내용을 바탕으로 정리하였으며, 4장에서는 rtIntegrit의 구조와 구현 과정을 다룬다. 5장에서는 rtIntegrit의 성능 측정 결과를 정리하였으며, 마지막 6장에서는 결론 및 향후에 계획을 기술하였다.

2. 관련 연구

2.1 파일 무결성 검사 도구

파일 무결성 검사 도구는 공격이 발생한 이후 시스템의 무결성을 검사하는데 이용되는 도구이다. MD5 같은 알고리즘을 이용하여 파일의 변조 유무를 판단한다. 일반적으로 감시 영역을 설정하고, 초기 상태에서 파일 또는 디렉토리가 새로 생성, 변경 또는 삭제 등이 이루어져 있는지를 파악하며, 부분적으로 접근한 사용자에 대한 정보를 관리자에게 알려줄 수 있다. Tripwire가 가장 대표적인 파일 무결성 검사도구로서 이전에 사용되던 Tripwire의 동작적인 자원 관리 기법과 독자적인 리포팅 방법을 사용하고 있다[12].

Tripwire 이후에 Integrit, AIDE, AFICK, YAFIC 등 공개 프로젝트로 개발되는 다양한 파일 무결성 도구들이 개발되었지만, 기본적인 데이타베이스 관리 방법은 Tripwire와 동일하다. Integrit은 다른 도구들에 비하여 기능이 안정되고, XML로 리포트를 출력할 수 있는 기능을 가지고 있기 때문에 활용도가 높다[13]. 파일 무결성 검사 도구를 침입 대응 및 복구 시스템에 연계하기 위해서는 최소한 검사한 유효성에 관련된 정보를 침입

탐지 시스템과 유효한 형식으로 변환해야 한다[11]. 현재 제안된 표준화된 침입 탐지 형식은 CIDF[7]와 IDWG(Intrusion Detection Working Group)의 IDMEF가 있다. IDMEF가 XML을 사용하며 IETF Working Group으로 진행되어 표준화될 가능성 높기 때문에, 본 논문에서는 IDMEF의 형식으로 개발하였다.

2.2 IDMEF(Intrusion Detection Message Exchange Format)

IDMEF는 IETF의 IDWG(Intrusion Detection Working Group)에서 표준을 정의하고 있다. IETF의 IDWG는 경보 데이타를 전달하기 위하여 IAP와 IDXP 두 개의 프로토콜을 제안하였다. IAP는 IDP의 요구사항을 만족시키기 위한 첫 번째 시도로 개발되었으나, 몇 가지 제약사항으로 인하여 IDXP를 경보데이타를 전달하는 표준 프로토콜로 정의하였다[14]. IDMEF는 IAP나 IDXP가 전달하는 경보데이타의 형식을 나타낸다. IDMEF는 침입 탐지 및 대응 시스템, 관리 시스템 간에 상호 작용을 하는데 필요한 정보를 공유하기 위한 데이타 형식과 정보 교환 절차를 정의하고 있다[15]. IDMEF의 데이타 형식을 구현하기 위해서 MIB (Management Information Base)를 기술하는 SMI (Structure of Management Information) 방안과 XML 문서를 기술하는 DTD 형식의 두 가지 안이 제안되었고, 2000년 2월 IDWG 회의에서 더 높은 확장성과 응용력을 가지고 있는 XML/DTD가 IDMEF의 표준 데이타 형식으로 채택하였다[14].

IDMEF 데이타 모델은 IDMEF-Message 클래스를 최상위 클래스로 하고 Alert, Heartbeat 두 개의 하위 클래스가 정의되어 있다. Alert 클래스는 속성 정보를 표현하기 위한 속성 클래스들로 구성되어 있다. Alert 클래스의 부가정보를 표현하기 위해서 ToolAlert, CorrelationAlert, OverflowAlert 클래스 등이 정의되어 있다.

2.3 시스템 호출 감시

호스트 기반 침입 탐지 시스템에서 가장 많이 이용되는 방법이 시스템 호출 감시 방법이다. 시스템 호출은 응용 프로그램이 시스템의 요청하는 최종 행위이므로, 이것을 통해 공격 행위를 탐지 한다. 최근 운영체제들은 자체의 보안성을 높이기 위하여 일정 수준의 이상의 감사 자료를 제공하는 모듈을 통하여 시스템 호출 감시 기능을 제공한다[16-18]. 대부분의 호스트 기반 침입 탐지 시스템은 운영체제의 감사 모듈이 제공하는 감사 자료를 바탕으로 침입 행위를 탐지한다[19].

감사 기능 이외에 접근 제어 기능 또한 운영체제의 커널 수준에서 기능이 제공되어야 한다. 최근에는 다양한 접근 제어 정책을 쉽게 구현하고 적용할 수 있도록 표준화된 프레임 워크를 제공하기 위한 연구가 진행되었다. 특히 소스가 공개된 Linux의 LSM이 SELinux를 비롯한 여러 보안 운영체제 개발에 많이 활용되고 있다[20]. 그러나 시스템 호출을 감시하는 작업을 구현할 때에는 오버헤드가 크게 발생하지 않도록 하여 시스템의 성능을 저하시키지 않도록 한다[21].

침입 대응 및 복구를 수행하기 위해서는 현 단계 보다 훨씬 높은 탐지율과 낮은 오탐율을 가져야 하며, 특히 파일 복구 등을 수행하기 위해서는 파일의 수정 및 변형에 관련된 정보들이 포함되어야 하지만, 기존의 BSM, LxBSM, Snare 등의 감사 자료 생성 모듈들은 이러한 정보를 제공하지 못한다. 따라서 파일 무결성 검사 도구와 연계하는 방안을 고려하였다.

3. 시스템 분석

Tripwire, Integrit 등의 파일 시스템 무결성 검사 도구는 비교적 간단하게 시스템의 상태를 파악할 수 있는 도구이다. 파일 시스템 무결성 검사 도구는 시스템의 운영 정책에 따라 정해진 정책에 따라서 파일의 무결성을 검사하기 위한 무결성 데이타베이스를 생성하고, 무결성 검사를 할 때에는 무결성 데이타베이스의 정보에 위배되는지 여부를 통하여 판단한다. 만일 위배된다면 그 파일을 사이즈 변화, 변경 시간, 마지막 사용자등의 정보 등을 함께 보고한다[12,13].

그러나 현재의 파일 무결성 검사 도구들은 모두 수동적이다. 침입에 대하여 능동적으로 대처해야하는 실제 침입 대응 시스템이나, 복구 시스템에 운용하기에는 어렵다. 우선 파일 무결성 검사 도구들은 독자적으로 사고가 발생을 감지할 수 없다. 본 논문의 rtIntegrit은 어떤 사건이 발생하는 순간에 검사를 진행하여 판단의 정확성과 동시성을 높이는 것을 목적으로 한다.

rtIntegrit을 침입 대응 시스템에 적합하게 하기 위해서는 다음 문제를 해결해야 한다. 현재 설계되는 대부분의 침입 대응 시스템들은 특정한 침입 탐지 시스템에 의존하지 않고 여러 시스템을 연계하여 운용할 수 있는 공용화된 자료 형식을 요구한다. 그리고 기존의 파일 무결성 검사 도구는 침입 탐지 시스템으로 설계되지 않았으므로, 사건의 발생 유무와 관계없이 파일의 생성 또는 삭제, 변경 등에 관련된 정보만을 제공한다. 따라서 침입 탐지 시스템 대용으로 직접 사용하기 어려우며, 비동기적으로 사건의 판단과 대응 작업이 이루어지게 되어, 사건이 발생한 순간 바로 대응할 수 없다. 본 논문에서 제안하는 rtIntegrit은 이러한 것들을 실시간으로 처리하여 사건의 발생과 동기화시킬 수 있으므로, 신속하게 대응 작업을 진행시킬 수 있다.

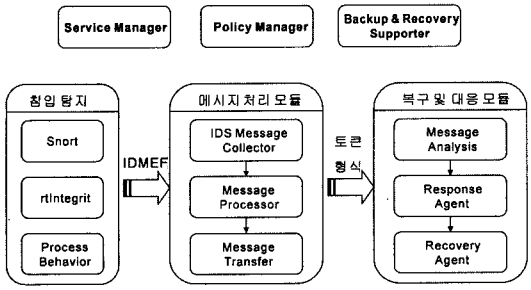

그림 1 침입 복구 및 대응 시스템

그림 1은 본 연구에서 목표로 하는 침입 복구 및 대응 시스템 모델을 나타낸다. 침입 복구 및 대응 시스템 모델은 크게 침입 탐지 시스템, 메시지 처리 및 분석 시스템 모듈, 복구 및 대응 시스템 모듈의 세 가지 주요 하부 시스템으로 구성된다. 침입 탐지 시스템들이 발생한 메시지들은 IDMEF 형식으로 메시지 처리 모듈로 전송된다. 메시지 처리 모듈은 시스템의 운영 정책에 따라서 대응 여부를 판단하게 된다. 운영 정책은 접근 권한을 바탕으로 설정되지 않고, 탐지 결과를 바탕으로 지정한다. 복구 및 대응 모듈은 메시지 처리 모듈이 생성한 정보를 바탕으로 필요한 작업을 진행하게 된다.

침입 복구 및 대응 시스템 모델에서 오탐율을 줄이고 탐지 결과의 정확성을 높이기 위하여 서로 다른 특성을 갖는 여러 침입 탐지 시스템을 연동하여 구성한다[4,11]. 침입 탐지 시스템간의 연동을 위해서는 표준화된 자료 형식을 필요로 하며, 또한 파일 무결성 도구도 이와 같은 침입 대응 시스템의 프레임워크에 적용하려면 적합한 감사 자료 형식을 따라야 한다.

rtIntegrit은 탐지 결과를 IDMEF로 표현한다. 우선 IDMEF에서 탐지 결과를 Alert 클래스를 바탕으로 표현하는데, Alert 클래스는 침입 탐지 시스템이 발생시키는 메시지 개체를 표현할 때 사용하는 클래스이며, Analyzer, CreateTime, DetectTime, AnalyzerTime, Source, Target 등의 결합(Associate) 클래스를 이용하여 메시지의 세부 속성을 표현한다. Source, Target 결합 클래스는 자신에게 속한 결합 클래스를 가진다. Source는 공격자를 표현하는데 사용되고, Target는 공격 대상을 기술하는데 사용된다. 파일 무결성 검사 도구는 주로 파일이 공격 대상이 되므로, Target 클래스로 기술되는 정보는 주로 FileList 클래스에 존재하게 된다.

FileList 클래스는 File 클래스의 집합으로 구성되어 있다. File 클래스는 name, path, create-time, modify-time, access-time, data-size, FileAccess, Linkage, Inode, Checksum 등을 통하여 Target으로 지정된 파일의 마지막 상태를 표현할 수 있도록 설계되었다. 그러나 대응 시스템에서 실시간으로 대응하기 위해서는 마지막 행위자(process)를 알아야 하며, 이를 위하여 File 클래스에 Process 항목을 추가하였다. File 개체의 Process 클래스는 Source, Target 클래스의 속성에 속하는 Process 클래스와는 다르며, 속성에 대한 자세한 내용은 4.2절에서의 SysWatcher 동작 과정을 기술할 때 다룬다. 또한 File 개체에 IDMEF에 정의되지 않은 새 속성을 추가한 것은 행위자를 표현하기 어려운 부분을 보완하면서, Integrit이 생성하는 정보를 손상시키기 않고 전달하기 위한 것이다.

4. rtIntegit의 설계 및 구현

4.1 rtIntegrit

rtIntegrit은 Integrit을 확장하여 설계하였다. Integrit은 다른 파일 무결성 검사 도구와 마찬가지로 사용자 검사 대상이 되는 파일 또는 디렉토리를 지정하여 무결성 검사를 수행하면 초기 데이타베이스가 생성된다. 무결성 검사를 다시 수행하면 새로운 DB가 생성되고, 이를 바탕으로 보고를 발생시킨다. rtIntegrit은 SysWatcher를 이용하여 파일 접근에 관련된 시스템 호출을 감시한다. SysWatcher는 설정 파일에 검사 대상이 되는 파일 및 디렉토리에 대한 접근이 발생하면 FIFO에 파일과 관련된 프로세스 정보를 기록한다.

그림 2는 rtIntegrit의 동작 구조와 기존 Integrit에서 개선된 부분을 보여준다. 기존 Integrit은 설정 파일(Configuration)에서 검사 대상이 되는 디렉토리 및 파일에 대한 정보를 가져와 자신이 구축한 무결성 데이타베이스(knowndb)와 비교하여 무결성 검사를 수행하고, 이에 대한 Report를 출력하면서 동작하였다. Report의 결과는 knowndb와 currentdb를 비교하여 상태를 표현한다. rtIntegrit은 크게 3가지 조건에 따라 무결성 검사를 한다. 1) 기존 Integrit과 마찬가지로 사용자의 요구에 따라서 수행하고, 2) 설정 파일에 지정된 시간에 따라서 주기적으로 검사할 수 있다. 3) 마지막으로 설정 파일에 즉시 검사하도록 지정된 검사 대상은 SysWatcher가 파일 접근에 관련된 시스템 호출을 감시하며 SysWatcher는 해당 파일에 대한 수정작업이 감지되면 즉시 무결성 검사를 수행하도록 신호를 보낸다. 또한 SysWatcher는 설정 파일에 설정된 디렉토리 내에 존재하는 파일에 대한 시스템 호출이 발생했을 때 부분적으로 currentdb를 갱신하고, FIFO에 지정된 파일에 접근한 프로세스 정보를 기록한다. 마지막으로 Report 처리 부분에서 FIFO의 내용을 바탕으로 무결성 탐지 결과와 합쳐서 IDMEF 형식으로 변환하여 결과를 보고한다.

그림 3은 rtIntegrit의 설정 파일이다. 설정 파일은 rtIntegrit의 동작 방식에 따라 구분하였다. database 설정 방법 등은 기존의 Integrit의 설정 방법과 동일하고,

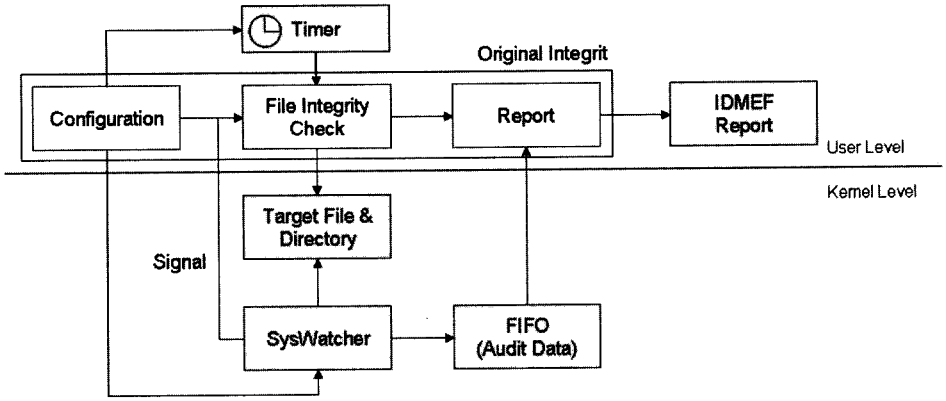

그림 2 rtIntegrit의 구조

```
# root.conf

# database locations
known=/etc/root_known.cdb
current=/etc/root_current.cdb

# rules for checking
root=/
0 30 /root          # 0: SysWatch Logging, 실시간 보고
0 30 /sbin
0 30 /etc
1 120 /bin          # 1: SysWatch Logging
1 120 /usr
2 2400 /home        # 2: SysWatch Logging 하지 않음
```

그림 3 rtIntegrit 설정 파일 예

실시간 및 주기적 감시 작업이 가능하도록 이를 지정하기 위한 속성들을 반영하였다.

검사 대상이 되는 디렉토리 및 파일을 기술할 때 우선순위와 검사 주기를 앞에 표시한다. 디렉토리 각각의 중요도에 따라 우선순위와 주기를 설정한다. 우선순위는 0, 1, 그리고 2로 지정되며 낮을수록 우선순위가 높다. 0인 경우에는 FIFO에 기록하고, 사건이 발생하는 순간 rtIntegrit에 알리고, rtIntegrit은 해당 디레톡리에 즉시 무결성 검사를 수행한다. 1인 경우는 FIFO 기록만 하고, 알리지는 않는다. rtIntegrit은 지정된 주기에 따라서 무결성 검사를 수행하고, 이때 FIFO의 정보와 바인딩하게 된다. FIFO의 정보가 남아 있으므로, 무결성 검사 리포트에는 행위자 정보가 남게 된다. 마지막으로 우선순위가 2인 경우는 가장 일반적인 무결성 검사만 수행하여 리포트하게 된다. 이때는 행위자 정보가 남지 않고, 기존의 Integrit이 제공하는 것과 동일한 정보를 제공한다.

두 번째 인자는 SysWatcher의 이벤트 발생과 상관이 자체 검사 주기를 분 단위로 나타낸다. 중요한 시스템 파일들에 대해서는 실시간으로 검사하여, 그 때마다 데이타베이스를 갱신하고 리포트를 발생시킨다. 그림 3의 설정 파일 예에서는 혹시 모르는 상황을 대비하여 우선순위가 높은 자원의 검사 간격을 짧게 주었다. 중요도가 떨어지는 경우에는 SysWatcher가 참조하지 않으며, 검사 주기를 길게 하여 불필요하게 검사를 많이 하여 자원이 소모되는 것을 방지한다.

4.2 SysWatcher의 구조

그림 4는 SysWatcher의 동작 구조를 나타낸다.

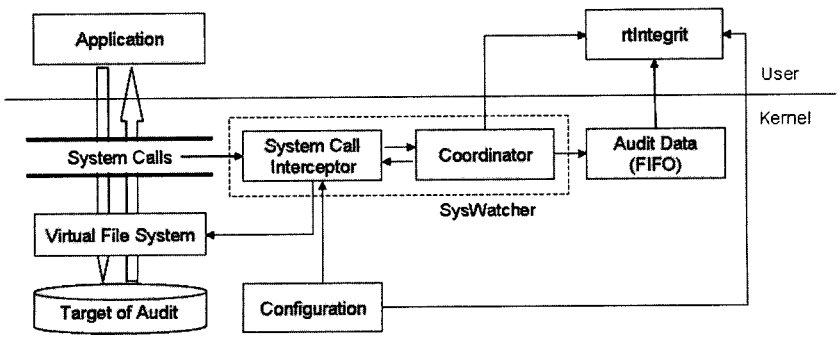

그림 4 SysWatcher의 구조

SysWatcher는 프로세스들의 파일 접근에 관련한 시스템 호출만 집중적으로 감시한다. SysWatcher는 LKM (Linux Kernel Module) 방식으로 작성되어서 동적으로 모듈의 적재 및 해제가 가능하다[21]. SysWatcher는 커널 내부 System Call Table에서 파일에 접근하는 시스템 호출들을 가로채서, 각 프로세스들이 어떤 파일을 접근하는지를 검사한다. rtIntegrit 설정 파일에 설정된 검사 대상 파일에 대한 시스템 호출이 탐지될 경우 즉시 대응하도록 설정된 디렉토리 및 파일에 대해서는 rtIntegrit에 재검사하도록 알리고, FIFO를 통하여 rtIntegrit에 관련 프로세스 정보를 넘겨준다. Coordinator는 설정에 따라 Signal을 사용자 프로세스로 보내고, 프로세스 관련 정보를 FIFO에 저장한다.

Linux에서 파일 접근에 관련된 시스템 호출들의 유형을 정리해 보면, 파일의 생성과 제거, 파일의 내용을 읽는 것과 쓰는 것, 파일의 상태정보를 얻거나 변경하는 것, 그리고 lseek()와 같은 파일 접근에 필요한 정보를 변경하는 것 등으로 나눌 수 있다. 이들 중 실제 감시해당하는 시스템 호출은 파일생성 및 제거, 쓰기, 상태정보를 변경하는 시스템 호출들이다. SysWatcher는 시스템의 오버헤드를 줄이기 위해 파일 변경에 직접적으로 관련이 있는 open, write, unlink, close 시스템 호출만을 감시한다. open 시스템 호출의 인자만으로도 사용자의 행위 유형을 감시할 수 있다. 기본적으로 파일 무결성 검사는 파일의 신규 생성, 변경, 삭제 등을 모니터링한다. 신규 생성은 open 함수의 인자가 create 속성의 포함 여부를 가지고 판단할 수 있다. 변경에 대해서는 open 함수의 인자가 쓰기 속성을 포함하고 있으면, open으로 얻어진 지시자로 호출된 write 함수를 추적함으로써 파악할 수 있다. close 시스템 호출은 open으로 생성된 세션의 경계를 확인하는데 사용된다. unlink 함수는 파일의 삭제를 모니터링할 수 있다. SysWatcher는 지정된 파일 및 디렉토리에 이와 같은 행동이 발생하면 해당 정보를 FIFO에 기록하고, rtIntegrit에게 시그널을 이용하여 사건의 발생을 알린다. rtIntegrit은 해당 지점에 대하여 무결성 검사를 수행하고, 결과를 IDMEF로 결과로 출력한다.

표 1은 SysWatcher에서 감시 대상에 대한 파일 접근 시스템 호출이 발생할 때 FIFO를 통해 rtIntegrit에 알리는 정보이다. syscall에는 시스템 콜의 이름이 포함되고, file에는 접근 대상이 되는 파일 경로, 그리고 pid에는 접근하는 프로세스의 이름이 각각 표현된다. SysWatcher가 이 정보를 FIFO에 기록할 때에는 ':'을 구분자로 사용한다. arg와 env는 시스템 호출을 표현하는데 크게 필요하지 않으므로 생략된다. 필요할 경우 이에 대한 정보도 함께 나타낸다.

표 1 SysWatcher 로그 정보

속성	기능
syscall	시스템 호출 이름
file	접근 대상 파일 패스
pid	프로세스 번호
pname	프로세스 이름
arg	명령 라인 인자
env	환경 변수

5. 실험 및 성능 측정

이 장에서는 구현한 rtIntegrit의 실험과 그에 대한 성능 측정 결과를 기술한다. 실험 과정에서는 rtIntegrit이 정상적인 형식으로 출력하는지의 여부를 보고, 성능 측정 항목에서는 rtIntegrit과 SysWatcher가 시스템에 끼치는 영향을 알아보기 위하여, LmBench를 이용하여 성능을 측정하였다.

5.1 실험 결과

그림 5는 /etc 디렉토리를 실시간 감시 대상 디렉토리로 설정하여 실험하였으며, /etc/hosts라는 파일이 생성되었음을 나타내는 감사 자료이다. Alert 개체가 생성되고, Alert의 Analyzer에는 실험한 시스템의 정보와 Analyzer로 rtIntegrit이 사용되었음을 보여준다. CreateTime은 메시지의 생성 시각 등을 알려주고, 오른쪽의 FileList 개체에 포함된 File 개체에 파일의 이름, 패스, 권한의 상태 정보가 나타나며, Process 개체에는 시스템 호출, 관련 프로세스 번호, 프로그램 이름이 나타난다. 이를 통하여 구체적인 행위자를 표현할 수 있게 되는 것이다. 이를 통하여 침입 대응에 관한 정책을 수립할 때 행위자 별로 대응 방법을 달리 할 수 있게 된다.

5.2 성능 측정

rtIntegrit은 무결성 검사를 진행할 때 시스템에 가장 큰 부담을 주지만, 무결성 검사 방법 자체는 기존의 Integrit과 동일하며, 특정한 사건이 발생하였을 때만 동작하므로 성능을 측정할 필요성이 떨어진다. 그러나 시스템 호출 과정을 계속 감시하는 SysWatcher가 동작하면서 시스템에 영향을 많이 미친다면, rtIntegrit의 활용도는 떨어질 수밖에 없다. 본 실험은 SysWatcher의 부하율을 검사하여 SysWatcher가 시스템의 부담을 크게 주지 않음을 확인한다. 실험에는 Intel Pentium4 2.4C, 512MB 시스템에 Linux Kernel 2.4.20 커널을 사용하였다. 커널의 성능을 측정하기 위하여 LMBench 2.0.4를 사용하였다[22].

SysWatcher는 rtIntegrit의 설정 조건에 관계없이 항상 지정된 시스템 호출을 감시 한다. 따라서 rtIntegrit의 설정에 관계없이 항상 일정한 부하를 발생시키므로, 아무것도 사용하지 않는 일반적인 상태(Normal)와

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE IDMEF-Message PUBLIC "-
//IETF//DTD RFC XXXX IDMEF v1.0//EN" "idmef-
message.dtd">
 <IDMEF-Message version="1.0">
  <Alert>
   <Analyzer analyzerid="rtintegrit3.02"
ostype="Linux" osversion="2.4.18-17hl">
    <Node category="hosts">
     <name>rocalhost</name>
     <Address category="ipv4-addr">
      <address>192.168.0.1</address>
     </Address>
    </Node>
   </Analyzer>
   <CreateTime
ntpstamp="0xc285b679.0x3db3bfb5">
    2003-06-02T20:41:45
   </CreateTime>
   <Source spoofed="no">
    <Node>
     <location>console</location>
     <Address category="ipv4-addr">
      <address>192.168.0.1</address>
     </Address>
    </Node>
   </Source>
   <Target decoy="no">
    <Node>
     <location>console</location>
     <Address category="ipv4-addr">
      <address>192.168.0.1</address>
     </Address>
    </Node>
    <User category="os-device">
     <UserId type="original-user">
      <number>500</number>
     </UserId>
    </User>
   </Target>
```

```
   <FileList>
    <File category="current" fstype="ufs">
     <name>hostt</name>
     <path>/etc/hosts/path>
     <FileAccess>
      <UserId type="user-privs">
       <name>root</name>
       <number>0</number>
      </UserId>
      <permission>read</permission>
      <permission>write</permission>
     </FileAccess>
     <FileAccess>
      <UserId type="group-privs">
       <name>root</name>
       <number>0</number>
      </UserId>
      <permission>read</permission>
     </FileAccess>
     <FileAccess>
      <UserId type="other-privs">
       <name>world</name>
      </UserId>
      <permission>read</permission>
     </FileAccess>
     <Process>
      <syscall>open</syscall>
      <pid>1019</pid>
      <name>vi</name>
     </Process>
    </File>
   </FileList>
  </Alert>
</IDMEF-Message>
```

그림 5 결과 감사 자료

SysWatcher 모듈을 동작시킨 경우(SysWatcher) 두 가지로 구분하여 정리하였다.

표 2는 시스템 호출에 대한 처리 지연 시간의 변화를 보여준다. Open/Close와 Null I/O를 제외하면 지연시간에 변화가 거의 발생하지 않는다. SysWatcher는 실제 File 접근 시스템 호출만 감시하며, SysWatcher가 커널의 다른 시스템 호출에 영향을 미치지 않음을 알 수 있다. 발생하는 지연시간은 SysWatcher가 감시 대상 파일의 접근 과정을 검사하는 과정과 감사 결과를 FIFO에 쓰는 과정에서 발생한다. 가장 빈번하게 참조되는 open()/close() 시스템 호출의 지연시간 증가가 1 마이크로초 이내이다. Exec Proc/Sh Proc은 모두 실행 파일을 불러들이기 위하여 내부에서 파일 IO관련 시스템 호출을 부르기 때문에 SysWatcher의 영향을 약간 받는다.

표 3은 지역 통신 지연 시간을 나타낸다. 전반적으로 SysWatcher를 사용하였을 경우에 시간이 증가하였다. 지역 통신은 소켓을 이용하여 연결 작업을 수행한다. 이때 Open/Close, Write 시스템 호출을 사용하므로 SysWatcher의 영향을 받기 때문에 일부항목에서 지연시간이 약간 증가하였다. TCP를 사용하는 경우 지연시

표 2 시스템 호출 처리 지연 시간 변화 (단위: microseconds)

Linux 2.4.20	Null Call	Null I/O	Stat	Open Close	Select TCP	Sig Inst	Sig Handle	Fork Proc	Exec Proc	Sh proc
Normal	0.46	0.51	1.73	2.38	5.552	0.78	2.65	90.8	320	1722
SysWatcher	0.45	0.73	1.71	3.07	5.547	0.78	2.66	91.3	326	1739

표 3 지역 통신 지연 시간 변화 (단위: microseconds)

Linux 2.4.20	PIPE	AF UNIX	UDP	RPC/UDP	TCP	RPC/TCP	TCP conn
Normal	5.021	7.02	10.3	19.2	12.5	23.8	35.9
SysWatcher	6.580	8.62	10.5	19.2	14.1	25.5	38.3

표 4 문맥 교환 지연 시간(단위: number of process/in cache size, microseconds)

Linux 2.4.20	2P/0K	2P/16K	2P/64K	8P/16K	8P/64K	16P/16K	16P/64K
Normal	1.160	2.12	4.78	2.64	20.6	5.37	24.1
SysWatcher	1.510	2.33	4.88	2.99	20.7	5.73	24.2

간 증가가 조금 크게 나타났고, UDP의 경우 훨씬 적게 영향을 받는 것으로 보인다.

표 4는 문맥 교환에 따른 지연시간 증가에 대한 오버헤드를 보여준다. SysWatcher를 사용하였을 경우 약간의 오버헤드가 발생하는 것으로 보이지만, LMBench를 여러 번 반복하였을 경우에 같은 환경에서도 시간차가 표 3에서처럼 나타났다. 이는 문맥 교환 시간 측정에서 나타나는 허용 오차로 보인다.

실험 결과의 일부 항목에서 SysWatcher를 사용하였을 때 미묘하게 더 좋은 결과가 나타나는 경우가 있는데, 실제로 SysWatcher에 의한 영향을 받지 않는 부분이다. 캐쉬 효과에 따른 부분적인 오차로 보인다. 파일 입출력에 관련된 항목에서는 지연시간의 증가가 발견되었지만 측정시간의 단위(마이크로초)를 감안하면 크게 증가하지 않음을 알 수 있다.

6. 결론 및 향후 계획

기존 파일 무결성 검사 도구의 문제점은 대부분 수동적인 시스템으로 주기적으로 실행되며 관리자의 역량에 많은 부분을 의존하고 있기 때문에 침입이 발생하였을 때 실시간 감사 자료 보고가 불가능하고, 그만큼 침입에 대한 대응이 늦어져서 피해가 커지는 단점이 있다. 또한 침입 이후의 단순 파일 상태 정보만을 제공하기 때문에 손상된 파일 정보를 수집하여 손상 파일의 복구는 가능하지만 침입이 발생하였을 경우에 파일 손상과 관련된 프로세스 정보를 수집할 수 없으며, 실시간 보고가 불가능하기 때문에 침입을 중간에 차단하여 침입으로 인한 파일 손상이 더 이상 진행되는 것을 막을 수 없다.

본 논문에서는 이러한 문제들을 해결하기 위해 rtIntegrit을 제안하였다. 또한 파일 접근 관련 프로세스 정보를 제공하는 SysWatcher를 추가로 구현하였다. SysWatcher는 실시간으로 파일 접근 시스템 호출들을 감시하여, 감시 대상 파일에 대한 시스템 호출이 발생하면 관련 프로세스 정보를 FIFO를 통해 rtIntegrit에 전달한다. rtIntegrit은 FIFO를 통해 받은 정보를 이용하여 파일 무결성 검사를 수행하고 파일 상태 정보와 함께 관련 프로세스 정보를 IDMEF 형식의 감사 자료로 보고한다. 파일에 접근하는 시스템 호출을 감시하는 SysWatcher를 파일 무결성 검사와 연동함으로써 실시간 보고가 가능할 뿐 아니라 파일의 상태 정보와 함께 관련된 프로세스 정보도 수집할 수 있다. 또한 감사 자료를 IDMEF 형식으로 보고하기 때문에, 확장성과 유연성을 확보하여 침입 대응 및 복구 시스템과 쉽게 연계시킬 수 있다.

시스템에 영향을 미치는 성능 측정은 LMBench를 사용하였으며, 실험 결과에서 확인하였듯이 Syswatcher와 연동하여 rtIntegrit을 사용하였을 때의 오버헤드가 시스템에 영향을 줄만큼 크지 않으므로 충분히 시스템에 적용시킬 수 있다. 시스템 호출 감시는 LKM (Loadable Kernel Module)의 형태 또는 커널 소스 패치의 방법을 사용하기 때문에 커널 모드에서의 실행 시간 증가를 주요 성능 평가의 기준으로 사용한다. SysWatcher가 동작하는 동안에도 커널의 서비스 지연시간이 크게 증가하지 않아 전체 시스템 상의 성능 감소는 적은 것으로 나타났다. 그러나 rtIntegrit의 무결성 검사 기능 자체는 Integrit의 것을 그대로 사용하기 때문에 개선의 여지가 많이 남아 있다.

본 연구에서는 사용자가 실시간 감시 대상인 파일과 그렇지 않은 파일을 구분하여 설정하도록 하였다. 그러나 추후에는 침입 유형별로 생성, 변경, 삭제되는 파일들과 시스템 호출들을 분류하여 감시 대상인 파일과 시스템 호출의 범위를 사용자가 필요에 따라 조정할 수 있도록 하여야 한다.

참 고 문 헌

[1] CERT/CC Statistics 1988-2004, http://www.cert.org/stat/cert_stat.html.

[2] 해킹바이러스 통계 및 분석 월보, http://www.certcc.or.kr/statistics/2003/0308_statistics.pdf, 2003. 8.

[3] Kenneth R. van Wyk & Richard Forno, *Incident Response*, O'Reilly & Associates, Inc., 2001.

[4] Dan Schnackenberg, Kelly Djahandari and Dan Sterne, *"Infrastructure for Intrusion Detection and Response,"* Proceedings of the DARPA Information Survivability Conference and Exposition(DISCEX-I) 2000, 2000.

[5] Active Networks Intrusion Detection and Response (AN-IDR), http://www.isso.sparta.com/research/documents/anidr.pdf.

[6] Curtis A. Carver, Jr., Udo W. Pooch, *"An Intrusion Response Taxonomy and its Role in Automatic Intrusion Response,"* Proceedings of the 2000 IEEE Workshop on Information Assur-

ance and Security, 2000.
[7] Common Intrusion Detection Framework, http://www.isi.edu/gost/cidf.
[8] D. Curry and H. Debar, *"Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition,"* http://www.ietf.org/internet-drafts/draft-ietf- idwg-idmef-xml-10.txt, 2003.
[9] NeoGuard ESM, http://www.inzen.com/kor/products/neoguard/intor.asp.
[10] Suhoshin IDS, http://www.securesoft.co.kr/english/product/idc_02.html.
[11] D. Schnackenberg, H. Holliday, R. Smith, K. Djahandari, D. Sterne, "Cooperative Intrusion Traceback and Response Architecture(CITRA)," Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX-II), 2001.
[12] Gene H. Kim, Eugene H. Spafford, *"The Design and Implementation of Tripwire: A File System Integrity Checker,"* COAST Laboratory, Purdue University, 1994.
[13] Integrit, http://integrit.sourceforge.net/.
[14] Intrusion Detection Working Group, http://www.ietf.org/html.chaters/idwg-charter.html.
[15] B. Feinstein, G. Matthews, J. White, *"The Intrusion Detection Exchange Protocol (IDXP),"* http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-07.txt, 2002.
[16] Sun Microsystems, *SunShield Basic Security Module Guide*, Sun Microsystems, 1998.
[17] 전상훈, 최재영, 김세환, 심원태, *"LxBSM: C2 수준의 감사 자료 생성을 위한 리눅스 기반 동적 커널 모듈의 설계 및 구현"*, 정보과학회논문지: 컴퓨팅의 실제, 제10권 제2호, pp.146-155, 2004. 4.
[18] Snare, http://www.intersectalliance.com/projects/Snare/, 2001.
[19] Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji, *"Intrusion Detection using Sequences of System Calls,"* University of New Mexico, 1998.
[20] C. Wright, C. Cowan, J. Morris, S. Smalley, G. Kroah-Hartman, *"Linux Security Modules: General Security Support for the Linux Kernel,"* USENIX Security Symposium, 2002.
[21] Pragmatic/THC, *(nearly) Complete Linux Loadable Kernel Modules*, http://www.thehackerschoice.com/papers/LKM_HACKING.html, 1999.
[22] LMBench, http://www.bitmover.com/lmbench/, 1998.

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

허 진 영

2002년 숭실대학교 컴퓨터학부(학사). 2004년 숭실대학교 컴퓨터학부(석사). 2004년~현재 (주)엠엠씨테크놀로지 연구원 관심분야는 시스템 프로그래밍, 보안 커널, 네트워크, 침입대응

최 종 선

2000년 숭실대학교 컴퓨터학부(학사). 2002년 숭실대학교 컴퓨터학과(석사). 2003년~현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 시스템 소프트웨어, 침입대응, 보안

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, 고성능컴퓨팅, 유비쿼터스컴퓨팅