침입 복구 및 대응 시스템을 위한 실시간 파일 무결성 검사

허진영° 전상훈 최재영
숭실대학교 컴퓨터학과
builder@nownuri.net°, shjeun@shinbiro.com, choi@computing.ssu.ac.kr

Real-time File Integrity Checking for Intrusion Recovery or Response System

Jinyoung Hur, Sanghoon Jeun, Jaeyoung Choi
School of Computing, Soongsil University

요 약

파일 무결성 검사는 시스템 자원의 안정성 여부를 판단할 수 있는 가장 확실한 방법이지만, 감사가 이루어질 때마다 매번 모든 데이터를 검사해야 하며, 관리자의 역량에 많은 부분을 의존하였다. 이는 최대한 빠른 대응을 요하는 침입 복구 시스템에 적합하지 않다. 또한 손상 자원의 복구에 필요한 자원의 상태 정보는 수집 가능하지만, 침입을 차단하기 위해서 침입 행위 주체에 대한 정보는 수집할 수 없다. 위의 두 가지 문제를 해결하기 위해 파일 무결성 검사와 시스템 호출 감사 기법을 연동한다. 시스템 호출 감사를 통해서 자원에 대한 침입 행위 주체의 정보를 수집하고 파일 무결성 검사를 하여, 침입 차단 및 복구를 위한 감사 자료를 수집한다. 또한 보다 효과적으로 침입을 탐지하고 복구하기 위하여 여러 침입 탐지 시스템과 연동하여 침입 복구 시스템을 구성할 수 있도록 탐지 메시지를 IDMEF 형식으로 보고한다.

1. 서 론

침입 복구 시스템은 침입으로 인해 손상된 자원을 이전의 정상상태로 복구하는 시스템이며, 침입 대응 시스템은 손상 자원의 복구를 포함하여 침입 행위의 차단 기능도 가진다. 침입 대응 시스템은 그 과정에 따라서 침입 탐지, 분석, 대응의 세단계로 이루어지며, 침입 탐지 시스템은 침입 탐지 방법에 따라 호스트 기반과 네트워크 기반으로 나뉜다. 호스트 기반 시스템의 경우 파일 무결성 검사와 시스템 호출 감시를 통한 프로세스 행동 감시 방법을 주로 사용한다 [1]. 파일 무결성 검사 도구는 그 검사가 주기적으로 행해지고 검사 시간이 오래 걸리기 때문에 가능한 빠른 대응을 요하는 침입 복구 및 대응 시스템에 적합하지 않다. 또한 침입 이전과 이후의 파일 상태 정보만을 제공하기 때문에 파일 복구에 필요한 정보는 제공하지만, 실제 침입 행위에 대한 대응에 필요한 파일 접근 프로세스에 대한 정보는 제공하지 않는다. 본 논문에서는 이를 해결하기 위해 즉 가능한 빠른 대응과 관련 프로세스 정보 수집이 가능하도록 파일 무결성 검사 도구와 시스템 호출 감시를 연동하여 실시간으로 파일 상태를 감시하며, 관련 프로세스 정보를 수집하고 추후 침입 복구 시스템과의 통합을 위하여 IDMEF 형식의 감사자료 보고 기능을 가지는 침입 탐지 시스템을 설계, 구현한다. 침입 탐지 메시지를 표준 형식을 사용하여 표현한다. 이는 서로 다른 특성을 갖는 여러 탐지 시스템을 통합하여 유연성 및 확장성을 확보할 수 있기 때문에 보다 정확하게 침입 상황을 파악할 수 있다. 본 연구에서는 위의 사항들을 구현하기 위해 기존의 파일 무결성 검사 도구인 Integrit에 IDMEF 형식의 감사자료 보고 기능과 SysWatcher와의 연동을 위한 기능을 추가하여 rtIntegrit을 구현한다. 그리고 시스템 호출 감시를 위해 LKM 형태의 SysWatcher를 구현하여 rtIntegrit과 연동 하였다.

* "이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2003-041-D00498)"

2. 실시간 파일 무결성 검사

2.1 Integrit

Integrit은 공개 파일 무결성 검사 도구로서 TripwireI나 Aide와 비슷하지만 단순한 기능을 제공한다. Integrit은 파일 시스템 내의 지정된 디렉토리에 존재하는 파일들의 상태정보 데이터베이스를 생성한다. 이 데이터 베이스를 안전한 장소에 보관한 후 검사할 때 지정된 디렉토리의 파일들의 상태정보를 재생성한다. 그리고 저장된 데이터베이스와 비교하여 변경, 추가, 제거된 파일들을 찾아낸다. 그후 결과를 Integrit 고유의 텍스트 또는 XML 형태로 보고한다. 동작 구조는 그림 1과 같다. Knowndb는 미리 생성해둔 검사 대상 디렉토리의 파일 상태 정보이며 currentdb는 검사 시점에 새로 생성한 파일 상태 정보이다. 이 둘을 비교하여 검사 결과를 보고한다 [2].

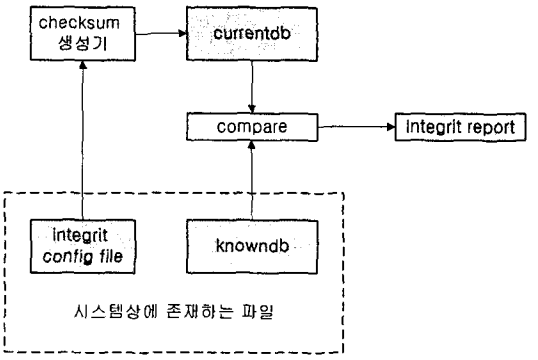

그림 1. Integrit 구조

2.2 IDMEF

IDMEF는 IETF의 Intrusion Detection Working Group에서 제안한 침입 탐지 메시지 표현을 위한 XML DTD로서 침입 탐지, 분석, 복구 시스템 간의 메시지 교환을 위한 표준을 제공

하며, 서로 다른 침입 탐지 시스템들의 메시지가 가지는 내용들을 모두 표현할 수 있도록 자료 구조를 제공한다. 현재 잘 알려진 네트워크 기반 침입 탐지 도구인 Snort에서는 IDMEF 형식으로 결과를 보고하는 출력 플러그인을 제공하고 있으며, 향후 보다 많은 탐지 도구들이 IDMEF 형식의 결과 보고 기능을 제공할 것으로 전망된다. 그러므로 IDMEF 형식으로 탐지 메시지를 변환하여 탐지 보고를 함으로써 여럿의 서로 다른 침입 탐지 시스템들의 침입 탐지 보고를 연동하여 침입을 분석 보다 정확하게 침입을 탐지할 수 있으며 보다 효과적으로 침입에 대응할 수 있다 [3].

2.3 rtIntegrit

rtIntegrit의 동작 구조는 그림 2와 같으며 그림 1의 Integrit의 구조에 SysWatcher의 기능과 IDMEF 형식으로 결과를 보고하는 기능이 추가되었다. Integrit은 주기적으로 직접 실행시켜야 하지만 rtIntegrit에서는 데몬화시켜서 자동으로 일정 주기마다 파일 시스템을 검사한다. 그리고 SysWatcher에 의해 생성된 로그 파일을 읽어서 Integrit의 파일 검사 결과에 파일에 접근한 프로세스의 정보를 추가한다. 그리고 IDMEF 형식으로 보고한다. 또한 중요 파일들에 대해서는 SysWatcher에서 접근을 실시간으로 감시하고, 필요하다면 rtIntegrit에게 시그널을 보내면, rtIntegrit은 검사를 수행하여 결과를 보고한다.

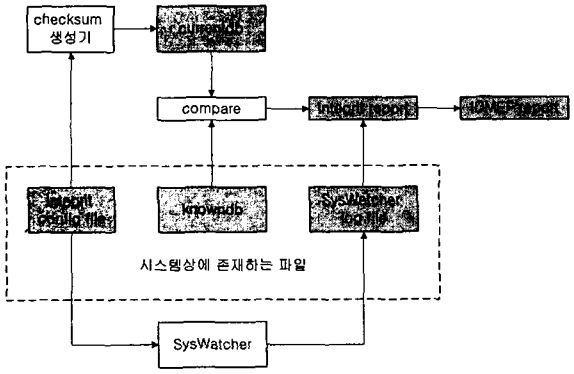

그림 2. rtIntegrit 구조

rtIntegrit의 설정 파일은 그림 3과 같으며 검사 대상이 되는 디렉토리들을 기술하며, 각 대상의 검사 주기와 우선순위를 앞에 표시한다. 디렉토리 각각의 중요도에 따라 우선순위와 주기를 설정했다. 중요한 시스템 파일들에 대해서는 실시간으로 검사하여 가능한 빠르게 침입에 대응할 수 있도록 하며, 우선순위가 낮은 파일들은 주기를 길게 하여 검사시간을 줄일 수 있다.

```
# 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. 우선 순위와 주기를 갖는 rtIntegrit 설정 파일

그림 4에서 SysWatcher는 LKM(Loadable Kernel Module)으로서 프로세스들의 파일 접근에 관련된 시스템 호출들을 감시한다. SysWatcher는 커널 내부 자료 구조 중 System Call Table에서 파일 접근 시스템 호출들에 해당하는 부분을 자신의 함수로 바꾸고, 프로세스에서 시스템 호출을 사용 시 어떤 파일에 대한 접근인지를 검사하여 원래의 시스템 호출로 넘겨 준다 [4]. rtIntegrit config에 설정된 검사 대상 파일에 대한 시스템 호출이 탐지될 경우 SysWatch log 파일에 기록하고 실시간 우선 순위를 갖는 파일이면 rtIntegrit에 시그널을 통하여 알린다.

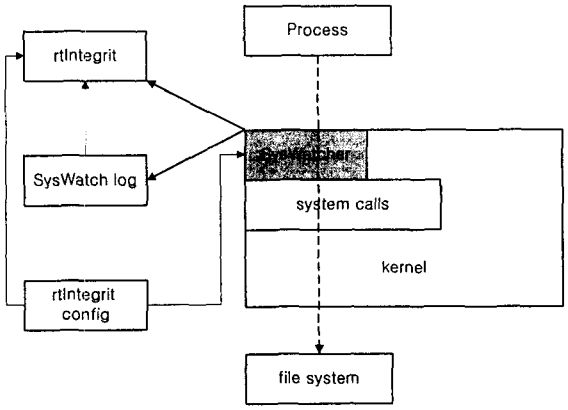

그림 4. SysWatcher 구조

파일 접근에 관련된 시스템 호출들은 표 1과 같으며 감시 대상 파일들에 대한 시스템 호출 발생시에 표 2의 형식으로 로그 파일에 기록한다.

표 1. 파일 접근 시스템 호출 리스트

이름	동작
read	read from a file descriptor
write	write to a file descriptor
open	open a file or device
close	close a file descriptor
creat	create a file or device
link	make a new name for a file
unlink	delete a name
mknod	create a special or ordinary file
chmod	change permissions of a file
lchown	change ownership of a file
stat	get file status
utime	set file access and modification time
rename	change the name or location of a file
mkdir	create a directory
rmdir	remove a directory
fchmod	change access permission mode
fchown	change owner and group of a file
getdents	read directory entries
chown	change ownership of a file

표 2. SysWatcher 로그 형식

time	호출 시간
syscall	시스템 호출 이름
file	접근 대상 파일 패스
name	프로세스 이름
pid	프로세스 번호
path	프로그램 패스
arg	명령 라인 인자
env	환경 변수

```
<IDMEF-Message version="1.0">
    <Alert>
      <Analyzer analyzerid="integrit3.02"
...
<CreateTime ntpstamp="0xc285b679.0x3db3bfb5">
        2003-06-02T20:41:45
      </CreateTime>
...
        <User category="os-device">
          <UserId type="original-user">
            <number>500</number>
          </UserId>
...
      <FileList>
        <File category="current" fstype="ufs">
          <name>a</name>
          <path>/usr/a</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>
            <name>vi</name>
            <pid>762</pid>
            <path>/bin/vi</path>
        </File>
    </FileList>
```

그림 5. 탐지 결과 보고 예

그림 5는 탐지 결과 예로 IDMEF 형식으로 표현되었으며 /usr/a 라는 파일이 pid 762를 갖는 vi 프로세스에 의해 생성되었다는 것을 나타낸다.

3. 결론

기존 파일 무결성 검사 도구로는 단순 파일 상태 정보 만을 수집할 수 있으며 관련된 프로세스에관한 정보는 수집이 불가능하고 시간 또한 오래 걸리기 때문에 실시간 탐지가 불가능했다. 그러나 시스템 호출 감시와 연동함으로써 실시간 탐지가 가능해 졌다. 시스템 호출 감시만으로도 모든 일을 할 수는 있지만 시스템 호출 감시는 커널 모듈로써 시스템 호출을 가로채고 감사자료를 수집한 후에 본래의 시스템 호출로 넘겨주기 때문에 가로채고 넘겨주는 사이에 하는 일이 너무 많으면 시스템 성능에 오버헤드가 크다. 그러므로 SysWatcher에서 하는 일을 최소화하기위하여 우선순위를 두어 중요한 파일들에 대해서만 시스템 호출을 감시하여 로그 정보를 남기고, rtIntegrit 데몬에게 signal을 통하여 알림으로써 보다 효율적인 침입 탐지가 가능해졌다. 침입 복구 및 차단의 침입 대응을 위한 파일 상태 정보와 관련 프로세스 정보를 동시에 수집 가능하며, 추후 복구 및 대응 시스템과의 통합을 위하여 감사자료를 IDMEF 형식의 표준으로 보고하도록 하였다. 본 연구에서는 SysWatcher를 파일 접근 시스템 호출을 감시하는 용도로만 사용하였으나 프로세스들의 시스템 호출 패턴 비교를 통하여 프로세스들의 비정상 행동을 탐지해주는 프로세스 비정상 행동 감시 도구로 확장할 수 있다.

참고문헌

[1] Julia Allen, Alan Christie, William Fithen, John McHugh, Jed Pickel, Ed Stoner, "State of the Practice of Intrusion Detection Technologies" Technical Report CMU/SEI-99-TR-028

[2] http://integrit.sourceforge.net

[3] http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt

[4] Snyder, D. On-line Intrusion Detection Using Sequences of System Calls. Master's thesis, Department of Computer Science, Florida State University 2001

[5] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, 6:151--180, 1998.

[6] A. Somayaji. Operating System Stability and Security through Process Homeostasis. Doctor of Philosophy Computer Science The University of New Mexico July 2002