DOI: 10.3745/KIPSTC.2008.15-C.4.239

Root 권한 프로세스 추적을 통한 침입 탐지 기법

박 장 수[†] · 안 병 철[††]

요 약

보안 침입 사건이 있은 후 해당되는 취약점만을 패치시키는 방식으로만 침입 피해를 줄이는 것은 충분하지 않다. 시스템 내에 취약한 코드가 있더라도 시스템의 내구성을 높여 보다 포괄적으로 침입을 막을 수 있는 방법이 필요하다. 본 논문은 리눅스 시스템에서 관리자를 대신하여 root 권한을 가진 프로세스를 감시하는 강건한 실시간 침입탐지기법을 제시한다. 이 기법은 사용자IP 주소를 프로세스 테이블에 추가하고 root 권한으로 기동되는 모든 프로세스의 IP 주소를 감시한다. 제안한 기법이 버퍼 오버 플로우 취약성에 대해 방어하는 것을 KON 프로그램을 통해 확인한다. 또한 원격으로 시스템을 관리할 수 있는 설정 프로토콜을 제안하며, 이 프로토콜을 통해 관리자 호스트의 IP주소가 침입으로부터 안전하게 보호될 수 있다.

키워드: root 권한 프로세스, 침입탐지, PCB, pass IP 주소, 설정 프로토콜

An Intrusion Detection Method by Tracing Root Privileged Processes

Jangsu Park[†] · Byoungchul Ahn[††]

ABSTRACT

It is not enough to reduce damages of computer systems by just patching vulnerability codes after incidents occur. It is necessary to detect and block intrusions by boosting the durability of systems even if there are vulnerable codes in systems. This paper proposes a robust real-time intrusion detection method by monitoring root privileged processes instead of system administrators in Linux systems. This method saves IP addresses of users in the process table and monitors IP addresses of every root privileged process. The proposed method is verified to protect vulnerable programs against the buffer overflow by using KON program. A configuration protocol is proposed to manage systems remotely and host IP addresses are protected from intrusions safely through this protocol.

Key Words: Root privileged process, Intrusion detection, PCB, pass IP address, Configuration protocol

1. 서 론

웹의 등장으로 인터넷 사용이 편리해 지면서 인터넷 사용자의 수도 폭발적으로 늘어나 누구나 쉽게 인터넷에 연결된 모든 호스트에 접근할 수 있게 되었다. 손쉽게 얻을 수 있는 자동화된 공격 도구들로 인해 인터넷에 연결된 시스템들에 대한 공격이 보편화되었으므로 확인된 침입 사건의 수를 세는 것이 공격의 범위와 영향을 평가하는데 도움이 되질 못하고 있다. 그러한 이유로 CERT에서는 2004년부터 침입 사건에 대한 통계를 발표하지 않고 있다[1].

방화벽을 비롯한 다양한 보안 대책 방안이 연구되었는데 이 중에서 지난 수년 동안 많은 관심을 가진 것은 Intrusion Detection System(IDS)이다. IDS는 침입을 탐지할 뿐만 아니라 탐지된 침입으로부터 시스템을 안전하게 지킬수 있어야 한다. IDS는 호스트 기반의 Host based Intrusion Detection System(HIDS)와 네트워크 기반의 Network based Intrusion Detection System(NIDS)가 있다. 호스트 시스템을 감시하는 HIDS는 높은 정확도를 가지나 모든 시스템에 설치하여야 하므로 관리하기가 어려운 단점이 있다. 그리고, NIDS는 정상적인 메시지를 악의적인 것으로 오판하는 단점이 있다[2]. 특히 최근의 초고속 네트워크에서는 오판률이 높아진다.

일반적으로 HIDS는 NIDS가 찾아내진 못한 것을 발견할 수 있도록 NIDS와 함께 설치된다. 이 논문에서는 새로운 HIDS 기법을 제안한다. 논문의 구성은 2장에서는 리눅스 시스템에서의 침입탐지에 관한 기존의 연구들과 문제점을 살펴보고, 3장에서는 IDIP 시스템을 구현하는 과정을 설명

† 준 회 원 : 영남대학교 컴퓨터공학과 박사과정
†† 종신회원 : 영남대학교 전자정보공학부 교수(교신저자)
논문접수 : 2007년 10월 31일
수 정 일 : 1차 2008년 2월 25일, 2차 2008년 3월 27일
심사완료 : 2008년 4월 28일

한다. 4장 및 5장에서는 su 프로그램 및 버퍼 오버플로우 공격을 사용하여 IDIP 시스템을 시험한 결과를 보여준다. 6장에서는 안전한 pass IP 주소 설정을 위한 설정 프로토콜에 대해 제안한다. 7장에서 결론을 맺는다.

2. 관련 연구 및 문제점

리눅스 운영체제에서 많이 사용되는 IDS는 Snort이다. Snort는 경량 네트워크 침입 탐지 시스템(NIDS)으로 사용할 수 있는 libpcap 기반의 패킷 감시 및 로깅을 하는 솔루션이다[4]. 그러나 이 솔루션 또한 NIDS로서의 한계를 가지고 있다. 그리고 침입자를 식별하기 위해 NPTrace는 지역 root 로그인 세션 및 원격 root 로그인을 구분하는데, 프로세스는 다음과 같은 경우에 root 권한을 가질 수 있다[5].

- 부모 프로세스와 같은 호스트에 있는 모든 자식 프로세스는 루트 권한 수준(Rootable)이다.
- Physically Secure Subset(PSS)에 속하는 호스트에 의해 원격으로 실행된 지역 프로세스는 루트 권한 수준(Rootable)이다.

그러나 이러한 접근은 관리자가 원격으로 지역 호스트를 관리하는 것을 제약하므로 지나치게 제한적이다. NPTrace의 기준은 관리자가 외지에서 지역 호스트로 로그인하는 것을 결코 허락하지 않는다. 그리고 침입자가 telnet이나 ssh를 통해 root 권한을 얻는 것이 아닌 버퍼 오버플로우 공격은 방지할 수 없다. 본 논문에서 제안하는 IP 주소 기반의 IDIP(Intrusion Detection using IP address)는 이 문제들을 해결할 수 있는 기법을 제공한다. 이 모델은 관리자가 원격으로 관리할 수 있는 통로를 제공하며 버퍼 오버플로우 공격을 방어할 수 있다.

침입자가 관리자 권한을 획득하려 할 때는 보통 해당 시스템의 사용자 계정을 먼저 획득한다. 그 다음 시스템에서 보안 취약성을 파악하고 exploit라고 불리는 공격 프로그램을 시스템으로 업로드한 후 관리자 권한을 획득한다. 리눅스 같은 유닉스 계열의 시스템에서는 root 계정이 관리자 권한에 해당한다.

시스템이 바이러스에 의해 감염될 때나 exploit에 의해 공격당할 때는 불가피하게 새로운 프로세스가 생성된다. 이들 경우 프로세스는 바이러스의 메모리 footprint일수도 있고 시스템내 취약한 프로그램의 메모리 footprint일수도 있다. 따라서 관리자는 시스템 동작의 이상 동작을 감지하면 우선 현재 실행되고 있는 프로세스들을 확인한다. 프로세스들 중에 비정상적이라고 판단되는 프로세스가 발견되면 해당 프로세스를 종료시키며 로그 검사 등을 통해 프로세스가 발생한 원인을 분석하게 된다. 경우에 따라 침입자의 IP 주소를 추적할 수 있다.

그러나 관리자가 침입 프로세스를 항상 탐지하는 것은 현실적으로 불가능하다. 실제로 exploit가 실행되어 관리자 권한을 획득하는 것은 아주 짧은 시간에 이루어지므로 관리자가 이상 동작을 감지하지 못한다. 가령 DOS 공격에 의해 시스템 성능이 현저히 떨어지더라도 관리자가 시스템의 상태를 항상 감시하지 않는 한 관리자는 침입 시도를 알 수 없다. 더욱이 리눅스와 같은 운영체제에서는 관리자가 root 권한으로 작업중에도 다른 사람에 의해 root 권한을 다시 얻을 수 있는데, 이것을 관리자가 찾아내기란 사실상 어렵다. 그 이유는 리눅스 커널이 프로세스의 사용자 ID를 관리하지만 실제로 몇 명이 해당 ID를 사용하는지는 확인하지 않기 때문이다.

따라서 본 논문에서는 프로세스 테이블에 사용자의 IP 주소를 저장하여, 이 주소를 통해 원격 사용자들이 실행하는 root 권한 프로세스를 감시한다. 이러한 방법에 의해 버퍼 오버플로우 공격에 대한 방어가 가능한 것을 확인하였다.

3. 제안 모델

IDIP는 관리자를 대신하여 시스템이 스스로 프로세스를 감시하고 침입에 대응하기 위한 방법이다. 프로세스를 생성하고 종료시키는 것은 커널이므로 실시간으로 감시하려면 보안 시스템이 커널과 연동하여야 한다. 커널 부담을 줄이기 위해 침입에 대한 판단은 어플리케이션에서 결정하며, 침입이라고 판단되는 프로세스에 대해서는 강제 종료시켜야 한다. 원거리에서 시스템에 접근하여 프로세스를 실행시킬 때 프로세스가 사용자 IP 주소 정보를 가진다면 IP 주소에 따라 프로세스의 권한을 root 권한 수준(rootable) 및 사용자 권한 수준(non-rootable)으로 나눌 수 있다. 이들 프로세스가 가지는 조건은 다음과 같다.

- root shell을 획득할 수 있는 프로세스는 root 권한 수준이며, root 권한을 획득할 수 없는 프로세스는 사용자 권한 수준이다.
- pass IP 주소를 가진 프로세스는 root 권한 수준이다. pass IP 주소는 원격 터미날로 사용되는 호스트의 IP 주소이다.

root 권한 수준이란 프로세스의 유효 사용자 ID(effective user ID)가 0인 것을 의미한다. 관리자가 외부 터미날 호스트에서 원격으로 IDIP가 설치된 시스템을 관리하고 싶다면 자신이 실행시키는 프로세스, 즉 자신의 IP 주소 정보를 가진 프로세스만 root 권한 수준이 되도록 할 수 있다. 이 경우 pass IP 주소를 터미날 호스트의 IP 주소로 지정하며, 다른 사용자는 IP spoofing 기술을 통해 자신의 IP 주소를 속이지 않는 한 root 권한을 얻을 수 없다. IP spoofing을 방지하기 위해 관리자가 작업하는 동안에만 자신의 IP 주소를 pass IP 주소로 지정하고 그 이외 시간에는 pass IP 주소를 null상태로 만들어 spoofing을 방지한다. 작업시간 동안 침입자가 자신의 IP 주소를 pass IP 주소로 속여서 접속한다면, 같은 IP 주소를 가진 프로세스가 새로 발생하였음을 관

리자에게 경고하여 spoofing 문제를 해결한다.

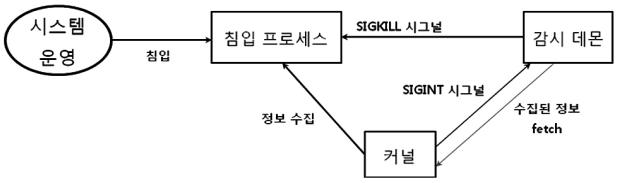
프로세스가 사용자 IP 주소 정보를 가진다면 관리자가 침입자의 IP 주소를 추적하는 시간을 절약할 수 있다. 여러 관리자가 공동 작업을 위해 동시에 호스트에 로그인을 시도하는 경우 여러 개의 pass IP 주소가 설정될 수 있다. 이 경우는 모든 관리자가 자신들의 IP 주소가 침입자에 의해 도용되지 않도록 주의하여야 한다.

4. 침입 탐지 기법의 구현

4.1 동작 방법

새로운 침입 차단 시스템이 프로세스를 처리하는 과정은 (그림 1)과 같다.

① 커널내 프로세스가 생성되는 지점에 새로운 코드를 삽입하여 프로세스에 대한 정보를 모은다.
② 정보가 수집된 후 커널은 새롭게 추가된 프로세스 감시 데몬에 SIGINT 시그널을 보내어 새로운 프로세스가 생성되었음을 알린다.
③ 감시 데몬은 우리가 추가한 시스템 콜을 통해 프로세스에 대한 정보를 읽어들인다.
④ 감시 데몬은 두가지 규칙에 따라 침입 프로세스 여부를 판단한다. 만약 침입 프로세스라고 판단되면 SIGKILL 시그널을 침입 프로세스에 보내어 강제 종료시킨다. 첫번째 규칙은 pass IP 주소가 아닌 IP 주소에서 접속하는 사용자는 로그인을 할수 없도록 하는 것이다. 두번째 규칙은 침입자가 실행시킨 프로세스의 부모 프로세스들을 강제 종료시킨다.



(그림 1) 침입방지 기법의 동작

4.2 사용자 IP 주소 확인

프로세스 정보에 사용자 IP주소를 포함시키기 위해 Process Control Block(PCB)인 task_struct 구조체에 필드를 추가한다. 이 구조체의 필드는 fork 시스템 콜에 의해 자식 프로세스가 생성될 때 복사되며 IP 주소가 상속된다. 따라서 사용자 권한 shell이 exploit에 의해 root 권한 프로세스를 실행하게 되면 IP 주소가 자식 프로세스의 PCB에 전달된다.

또한 이 필드에 저장할 IP 주소를 얻기 위해 accept 시스템 콜에 해당하는 sys_accept 함수에 코드를 추가한다. accept 시스템 콜은 사용자가 telnet 프로그램을 통해 서버에 접속할 때 서버 프로그램에서 호출된다. 즉, 서버에 접속되는 순간 클라이언트의 IP 주소를 획득한다. 획득된 IP 주소는 현재 프로세스의 PCB에 저장된다.

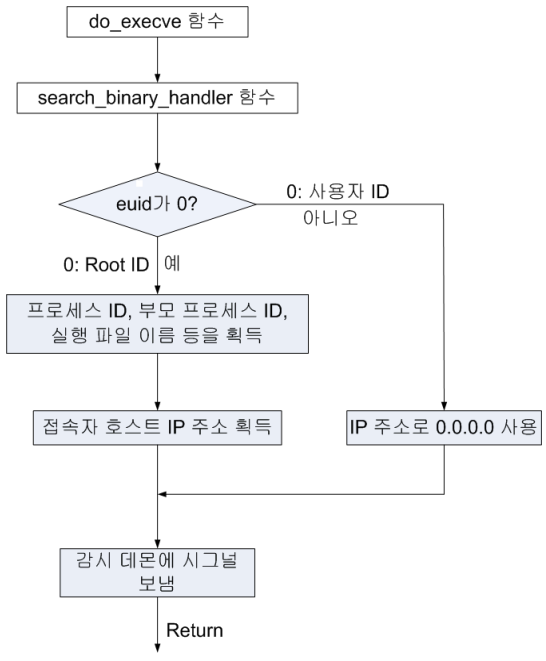
IP 주소는 execve 시스템 콜이 아닌 accept 시스템 콜에서 얻어야 한다. 최근 클라이언트로 많이 사용되는 ssh의 서버 프로그램(sshd)이나 웹 데몬(httpd)은 메모리에 상주한 채로 있다가 접속이 이루어지면 fork 시스템 콜에 의해 자신의 이미지를 만들어 클라이언트와 연결된다. 이때는 execve 시스템 콜이 실행되지 못하므로 accept 시스템 콜에 의해 IP 주소를 확인하여 자식 프로세스에게 전달하는 것이 효과적이다.

4.3 프로세스 체크인(check-in)

프로그램이 실행될때 프로세스를 생성하기 위해 execve 시스템 콜에 해당하는 do_execve 함수가 호출된다. 이 함수는 현재 리눅스에서 주로 사용되는 ELF 형식에 해당하는 실행 파일 핸들러를 찾기 위해 search_binary_handler 함수를 호출한다. 이 함수 호출 후에 프로세스 정보를 확인하기 위한 코드를 추가하였다. 먼저 현재 프로세스의 유효 사용자 ID가 0(root 권한)인지 확인한다. 수집된 프로세스 정보는 (그림 2)와 같이 프로세스 ID, 실행 파일 이름, 유효 사용자 ID, 접속자 호스트 IP 주소 등이다.

root 권한으로 수행되는 프로세스를 감시하므로 IP 주소를 제외한 다른 정보는 유효 사용자 ID가 0인 프로세스에 대해서만 수집하는 것이 효과적이다. 이것은 모든 프로세스의 정보를 수집하는 것에 비해 커널의 부담을 줄인다.

침입자가 exploit를 자신의 계정에 업로드 한 후 crontab 처럼 실행 지연 특성을 가진 프로그램을 사용하여 일정 시간 후에 exploit가 실행되도록 한다면 accept 시스템 콜은 IP 주소를 얻을 수 없다. 이런 경우엔 임시 IP주소를 부여한다. 임시 IP주소는 pass IP가 아닌 모든 값을 사용한다. exploit가



(그림 2) 프로세스 정보를 수집하는 과정

root 권한 프로세스를 실행할 때 임시 IP주소가 프로세스의 PCB에 전달된다. 관리자가 pass IP 주소를 제외한 다른 IP 주소를 임시 IP 주소로 설정하면 임시 IP 주소를 사용하는 root 권한 프로세스는 실행되지 못한다. 물론 이때 Set-User-ID 프로세스는 우회되어야 한다.

이렇게 프로세스의 정보가 확인되면 감시 데몬에게 SIGINT 시그널을 보내서 새로운 프로세스가 실행되었음을 알린다.

4.4 커널 수정

커널이 수집한 프로세스 정보를 감시 데몬이 읽기 위한 시스템 콜을 추가한다. /proc 디렉토리를 읽어서 정보를 얻을수도 있으나 /proc 디렉토리의 가상 파일을 읽기 위해서는 여러 개의 시스템 콜이 사용되므로 실시간으로 프로세스를 감시하기 위해서는 시스템 콜을 이용하는 것이 효과적이다. (그림 3)은 프로세스 정보를 담은 문자열의 예이며, 시스템 콜은 이 정보를 사용자 공간으로 복사한다.

```
PID:[프로세스 ID]
PPID:[부모 프로세스 ID]
COMMAND:[실행 파일 이름]
REMOTEHOST:[접속자 IP 주소]
SUID:[Set-User-ID]
SGID:[Set-Group-ID]
```

(그림 3) 프로세스 정보의 문자열 예

5. 기법의 동작 검증

5.1 su 프로그램에 의한 시험

시스템 관리자가 원격으로 리눅스를 관리하려면 사용자 계정으로 먼저 접속후 su 프로그램에 의해 root 계정으로 다시 로그인해야 한다. IDIP가 설치된 후에는 su를 사용하더라도 (그림 4)에서 처럼 로그인을 할 수 없다.

root 패스워드를 입력후 root shell이 실행될 때 감시 데몬은 접속자의 IP 주소인 192.168.10.3이 pass IP 주소가 아님을 확인하고 shell을 강제 종료시킨다.

```
[superman@localhost superman]$ su
Password:

pid=3447,gid=0,ppid=3446,comm=bash,logname=500,remotehost=192.168.10.3,suid=0,sg
id=0
sending KILL signal
[superman@localhost superman]$
```

(그림 4) su프로그램 시험

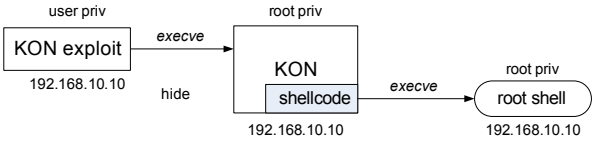
5.2 버퍼 오버플로우에 의한 시험

레드헷 리눅스 9.0에서 간지 문자를 출력할수 있도록 지원하는 프로그램인 KON(Kanji Console Emulator) 0.3.9b는 버퍼 오버플로우 취약성을 가진다[16]. 이 취약 프로그램을 통해 제안 모델을 검증하였다.

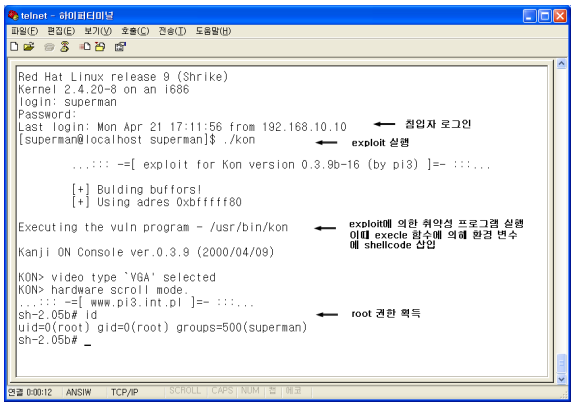
우선 지역 실행만 가능한 KON 프로그램을 원격 실행이 되도록 수정하였다. 이후 이미 잘 알려진 KON exploit를 일반 사용자 계정에 업로드 한 후 호스트 IP 주소가 192.168.10.10인 윈도우 XP 호스트에서 텔넷으로 접속하여 실행하였다.

KON exploit는 많이 알려진 환경변수 영역을 이용한 버퍼 오버플로우 기법을 사용한다. KON exploit는 execve 시스템 콜을 사용하여 Set-User-Id 프로그램인 /usr/bin/kon을 실행시키고, 이때 환경변수 영역에 (그림 5)처럼 shellcode를 삽입한다.

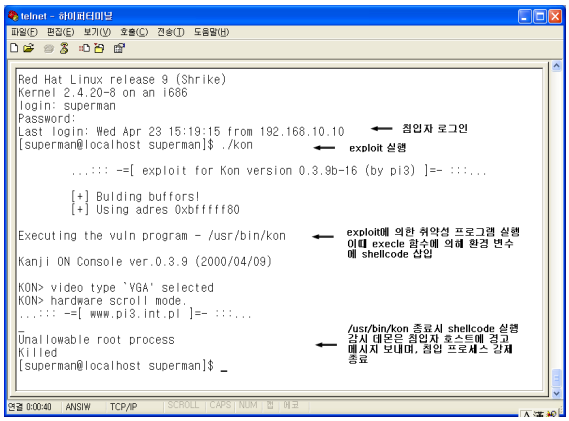
또한 exploit는 KON 프로그램이 종료할 때 참조할 return address를 shellcode가 저장된 주소로 변경하기 위해 주소가 반복되는 긴 문자열을 KON 프로그램 실행시 인수로 전달한다. 이렇게 함으로써 KON 프로그램 종료시 shellcode가 수행되고, shellcode는 execve 시스템 콜에 의해 root shell이 실행되어 침입자가 root 권한을 획득한다. (그림 6)은 IDIP가 설정되지 않았을 때 KON exploit가 실행되



(그림 5) KON exploit에 의해 root shell이 실행되는 과정



(그림 6) IDIP 설정전 KON exploit에 의한 root 권한 획득 화면



(그림 7) IDIP 설정 후 KON exploit에 대한 root 권한 차단 화면

어 root 권한을 획득한 장면이다.
(그림 7) 은 IDIP가 설정된 후 KON exploit가 실행한 shell이 감시 데몬에 의해 강제 종료되는 것을 보여준다. 이때 감시 데몬은 사용자에게 “Unallowable root process” 메세지를 보내는 것에 의해 부적절한 행위에 대해 경고한다.

침입자 IP 주소를 커널이 끝까지 추적할 수 있는 이유는 프로그램을 실행하기 위해 유일하게 사용되는 execve 시스템 콜의 특징과 관련된다. 제안 모델에 의해 침입자가 실행한 exploit 프로세스의 PCB에는 침입자 호스트의 주소가 저장된다. 이후 exploit가 실행한 KON 프로그램과, KON 프로그램의 연장된 코드로 동작하는 shellcode가 실행한 root shell은 모두 execve 시스템 콜에 의한 것이므로 이들 세 프로그램은 모두 같은 PCB를 사용한다. execve 시스템 콜은 PCB에서 새 프로그램의 세그먼트 관련 정보가 들어있는 일부 필드 값들을 변경하지만 나머지 필드 값들은 보존한다. 따라서 침입자 호스트 주소는 root shell이 실행된 후에도 PCB에 남아서 커널에 의해 확인이 가능하다. (그림 5)는 KON exploit의 PCB에 저장된 침입자 IP 주소 192.168.10.10이 KON 프로세스 및 root shell 프로세스에서도 유지됨을 보여준다.

shellcode가 root shell을 실행시키기 위해 fork 시스템 콜과 execve 시스템 콜을 함께 사용하더라도 문제가 되지 않는다. fork 시스템 콜 함수가 자식 프로세스의 PCB를 생성할때는 IDIP가 추가한 필드 값을 포함한 대부분의 부모 프로세스 필드 값을 변경없이 사용하므로 제안 모델은 여전히 방어가 가능하다.

6. pass IP 설정

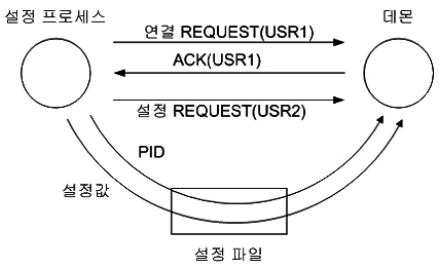
관리자는 IDIP 시스템에 원격 호스트의 주소를 설정하고 원격으로 시스템을 관리할 수 있다. 본 논문에서는 pass IP 주소를 다음과 같이 정의한다.

- pass IP 주소는 IDIP가 설치된 시스템에서 root 권한을 가질 수 있는 원격 호스트의 IP 주소이다.

IDIP 시스템에서 감시 데몬이 동작중에 안전하게 원격 호스트의 IP 주소를 설정할 수 있어야 한다. IDIP 모델은 침입후 차단 방식을 사용하므로 커널내 스케쥴러에 의해 침입 프로세스가 먼저 실행된 후 데몬이 스케쥴링되는 경우를 대상으로 한다. 따라서 침입 프로세스는 1 timeslice 동안 root 권한을 가질 수 있게 되며, 데몬의 설정상태를 변경할 수 있다. 따라서 사용자 설정 프로그램과 데몬 사이에 특별한 프로세스간 통신이 요구되며, 이것을 설정 프로토콜이라고 한다. 프로세스간 통신으로 메시지 큐와 같은 Inter Process Communicaton(IPC) 시스템 콜을 사용할 수도 있으나 IPC는 침입 프로세스가 설정 프로세스로 위장할 수 있는 위험을 안고 있다[7].

설정 프로토콜의 필요 조건은 다음과 같다.

- 침입자 프로세스가 root 권한을 획득하더라도 IDIP 모델에 의해 CPU를 1 timeslice 를 초과하여 점유하지 못한다.
- 원격 접속에 의한 설정은 금지된다. 그래서 콘솔에 의한 설정만 할 수 있으며, 원격 접속에 의한 설정시 터널링 프로토콜들이 함께 사용되어야 한다.

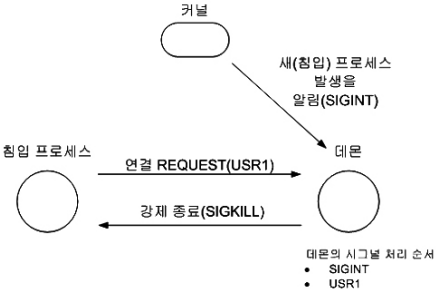


(그림 8) 설정 프로토콜에 의한 처리 절차

설정 프로토콜은 (그림 8)과 같으며 아래와 같이 수행하며, REQUEST, ACK 시그널은 USR1 또는 USR2 같은 기존 리눅스 시그널을 사용할 수 있다.

① 설정 프로그램은 사용자로부터 설정값을 입력 받는다.
② 설정 프로그램은 설정파일에 자신의 프로세스 ID를 넣는다.
③ 설정 프로그램은 데몬에 연결 REQUEST 시그널을 보낸다.
④ 데몬은 설정 프로그램의 프로세스 ID를 읽는다.
⑤ 데몬은 설정 프로세스에게 ACK 시그널을 보낸다. 여기까지가 설정 프로그램과 데몬이 연결되는 과정이다.
⑥ 설정 프로세스는 설정 파일에 사용자 설정값을 저장한다.
⑦ 설정 프로세스는 데몬에게 설정 REQUEST 시그널을 보낸다.
⑧ 위의 과정이 순서대로 이루어지면 데몬은 설정 파일에서 설정 값을 읽어들인다.

윗 과정은 데몬 및 설정 프로세스가 한 번씩 번갈아가며 수행되어야 하므로 침입 프로세스가 설정 프로세스로 위장하여 과정 2까지 수행하더라도 과정 3에서 데몬은 침입 프로세스를 차단한다. 침입 프로세스가 설정 프로세스로 위장했을 경우 커널이 수행하는 작업을 포함한 진행 과정은 (그림 9)에 나와있다.



(그림 9)의 과정은 다음과 같다.

① 설정 프로세스로 위장하기 위한 침입 프로세스가 발생한다.
② 커널은 새 프로세스가 발생하였으므로, 데몬에 SIGINT 시그널을 보낸다.
③ 침입 프로세스는 설정 파일에 자신의 프로세스 ID를 넣은 후 데몬에 연결 REQUEST 시그널을 보낸다.
④ 커널이 데몬을 스케쥴링한다.
⑤ 데몬은 먼저 받은 SIGINT 시그널에 대한 핸들러를 수행하여 침입 프로세스를 차단한다. 따라서 연결 REQUEST 시그널을 위한 핸들러는 수행되지 못한다.

이것을 위해서 침입 프로세스가 2 timeslice 이상 연속적으로 root 권한을 점유하지 못한다는 조건이 필요하다. 그 이유는 침입 프로세스가 root 권한을 오랫동안 얻을 수 있다면 보안 시스템으로서의 기능을 상실하기 때문이다.

7. 결론 및 향후 계획

프로세스 감시를 통한 호스트 기반 실시간 침입 차단을 할 수 있는 방법에 대해 기술하였다. su 프로그램에 의한 시험과 버퍼 오버플로우에 대한 시험을 통해 허락되지 않는 root 권한 프로세스를 통제하는 것이 가능하다는 것을 확인하였다. 시험을 위해 취약성이 알려진 KON 프로그램을 사용하였으며, IDIP가 버퍼 오버플로우 공격에 대해 방어할 수 있음을 확인하였다.

su 프로그램 시험에서 보았듯이 IDIP는 단지 오버 플로우 공격만을 방지하기 위한 것이 아니므로 다양한 형태의 침입 시도에 대해 응용할 수 있다. IDIP는 NPTrace 프로토타입이 가진 원격 관리 문제나 버퍼 오버플로우 취약성에 대한 대안을 제시한다. 또한 IDIP는 Snort와 같은 NIDS와 함께 설치하여 사용할 수 있는 HIDS이다. 이 보안 모델은 기존 커널을 크게 수정하지 않고 구현될 수 있으며 커널에 추가된 코드는 시스템 성능에 거의 영향을 주지 않는다.

pass IP 주소의 안전한 설정을 위해 설정 프로세스와 IDIP 데몬 사이의 설정 프로토콜을 제안하였다. 기존 프로세스간 통신 알고리즘에서 부족한 보안 문제를 새로운 설정 프로토콜에 의해 해결할 수 있음을 확인하였다.

이와 같이 IDIP는 시스템의 내구성을 높여 시스템 내에 설령 취약한 코드가 있더라도 좀 더 포괄적으로 침입을 막을 수 있는 방법을 제공한다.

그러나 원격 접속에 의한 설정시 패킷 암호화를 이용한 터널링, IPSEC 등과 같은 기술이 연계되어야 한다. 그리고, fork bomb, Denial of Service(DOS)와 같은 공격에 대해 제안 모델이 완벽하게 방어를 할 수 없으므로 향 후 보완을 할 수 있는 연구가 필요하다.

참 고 문 헌

[1] http://www.cert.org
[2] Makoto Shimamura, Kenji Kono. Using Attack Information to Reduce False Positives in Network IDS. Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)
[3] Lu. AN ADAPTIVE REAL-TIME INTRUSION DETECTION SYSTEM USING SEQUENCES OF SYSTEM CALL. CCECE 2003 - CCGEI 2003, Montreal, May/mai 2003
[4] Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In LISA'99: USENIX 13th Systems Administration Conference on System Administration
[5] Amit Purohit, Vishnu Navda and Tzi-cker Chiueh. Tracing the Root of "Rootable" Processes. Proceedings of the 20th Annual Computer Security Application Conference(ACSAC'04)
[6] One. Phrack 49. Smashing The Stack For Fun And Profit
[7] D. L. and Ritchie, D.M. 1985, "Interprocess Communication in the Eighth Edition UNIX System", Proceedings of the 1985 summer USENIX Conference, Portland, Oreg., 1985
[8] Grover. Linux Magazine. Buffer Overflow Attacks and Their Countermeasures
[9] Security Team gloomy & The Itch. Instruction pointer schurken.
[10] http://kerneltrap.org/node/644
[11] http://www.redhat.com/archives/fedora-announce-list/2004-June/msg00017.html
[12] Aurobindo Sundaram. An Introdunction to Intrusion Detection
[13] Anon. Linux Security Audit Project. http://lsap.org/
[14] Stevens. Addison Wesley. Advanced Programming in the UNIX Environment
[15] DANIEL P.BOVET & MARCO CESATI O'REILLY. Understanding Linux Kernel Second Edition
[16] http://packetstormsecurity.nl/0306-exploits/

박 장 수
e-mail : admin@kernelman.com
1992년 영남대학교 물리학과(학사)
2005년 아주대학교 정보통신대학원(석사)
2006년~현 재 영남대학교 컴퓨터공학과 박사과정
관심분야 : 정보보호, 실시간 시스템, 센서 네트워크

안 병 철
e-mail : b.ahn@yu.ac.kr
1976년 영남대학교 전자공학과(학사)
1986년 오레곤주립대 전기 및 컴퓨터 공학(석사)
1989년 오레곤주립대 전기 및 컴퓨터 공학(박사)
1976년~1984년 국방과학연구소연구원
1989년~1992년 삼성전자 수석연구원
1992년~현 재 영남대학교 전자정보공학부 교수
관심분야 : 임베디드시스템, 실시간운영체제, 멀티미디어처리