리눅스 보안관리 기법

김경환, 김형구, 이상연
2001.1.16.

차례

---

1. Linux 보안의 필요성과 그 개요

1.1 Linux 보안의 필요성 위로

유닉스가 처음으로 소개된 이후, 유닉스 특유의 강력한 업무수행능력이 인정 받음에 따라 많은 분야에서 사용되어왔다. 특히 오픈 소스인 Linux는 높은 가격대 성능비를 보이며 최근 몇 년 사이에 각광을 받고있다.

오픈 소스 정책에 의해 개발되는 소프트웨어들은 소스 코드가 모든 이에게 공개되기 때문에 버그가 빨리 발견되고, 그 속도만큼이나 개선되는 속도도 빠르다. 소스 코드가 공개되지 않는 소프트웨어들은 사용하면서 발견되는 버그가 소스 코드가 공개되는 소프트웨어 못지않게 많으며, 이러한 소프트웨어는 제작자 이외에는 문제점을 해결할 수 없으므로 그만큼 개선되는 속도가 떨어진다.

이러한 버그는 크래커가 침입하는 통로로서 이용되어 귀중한 자원을 파괴하는 결정적인 역할을 하게 된다. 하지만 소프트웨어의 문제점은 항상 존재하는 것이며, 완벽한 프로그램은 기대하기

가 무척이나 힘들다.

리눅스(Linux) 역시 완벽한 소프트웨어라고 하기 어렵다. 커널 개발 과정에서도 무수히 많은 문제점이 발견/수정되고 있으며 리눅스가 작동하기 위해 필요한 커널 이외의 시스템 소프트웨어와 응용 소프트웨어의 버그를 감안한다면 리눅스 시스템은 항상 문제에 직면해 있는 것이다.

그런데 MS-DOS처럼 혼자 사용하는 운영체제에서 발견되는 버그는 사용자 한 사람 혹은 시스템 한 대에만 영향을 주기 때문에, 피해가 확대되지 않지만, 여러 사람이 동시에 사용하는 리눅스와 같은 운영체제에서는 그 시스템을 사용하는 모든 사용자에게 피해가 확대되며, 더 나아가 네트워크로 연결된 주변의 다른 시스템까지 피해를 줄 수 있다.

프로그래머가 아닌 시스템을 관리하는 사람이 할 수 있는 일은 어떠한 것이 있을까?

1.2 크래커들의 공격 방법 위로

리눅스 시스템의 보안을 위해서는 크래커들이 어떠한 방법으로 시스템을 공격하는지 알아야 한다. 공격방법은 크게 나누어 3가지로 정리할 수 있다.

A. 정보 수집

서비스를 제공하는 시스템과 네트워크에 관한 정보를 수집하는 것은 얼핏 보면 별일이 아니라고 생각하기 쉽다.
비유를 하자면, 도둑이 건물을 털기 위해서 건물의 지도를 보고 침입 계획을 세우는 것과 같다. 꼭 침입을 목적으로 하는 정보 수집만이 아니라 두 시스템 사이에서 오고 가는 중요한 정보를 외부에서 의도적으로 수집해서 비밀을 요구하는 중요한 데이터가 외부로 유출될 수도 있다.

이러한 정보 수집의 방법 중 하나로서 먼저 스니핑(sniffing)을 들 수 있다.

스니핑이란 네트워크 상에서 오고 가는 통신을 가로채 그로부터 정보를 얻어내는 행위를 가리킨다. 이더넷(ethernet)의 특성상 LAN의 모든 시스템은 동일한 네트워크 선(wire)에 연결되어 있기 때문에 각 시스템은 항상 네트워크를 지나다니는 통신을 듣고 있다.

그러나 평상시에는 필터링 과정을 거쳐 자신을 목적지로 하는 통신만을 선택해서 받아들이게 된다. 이러한 필터링은 물리 계층인 이더넷 인터페이스에서 이루어지는데, 이때 이더넷 인터페이스를 "PROMISCUOUS" 모드로 설정하게 되면, 이더넷 인터페이스에 도착하는 통신에 대한 필터링을 하지 않고, 모두 받아들이게 된다.

따라서 네트워크 상에서 오고 가는 모든 통신을 받아들여서 읽을 수 있게 되고, 이로써 스니핑이 이루어지게 된다.

스니핑은 사용자의 username과 패스워드를 도청하는데도 사용된다. 이 경우, 해당 사용자뿐 아니라 동일 시스템을 사용하는 다른 사용자의 개인 정보가 유출되거나 해당 시스템이 큰 피해를 입을 수도 있다.

이것 이외에도 시스템의 주요 서비스로의 정보를 입수하여 특정 공격이 가능한지를 확인하는 용도로도 사용된다.

두 번째로 스캐닝(scanning)이라는 방법을 살펴보자.

스캐닝은 목표 시스템에 각 프로토콜에 관련된 정보를 보내어, 응답을 분석해서 목표 시스템이 제공하는 서비스, 목표 시스템의 운영체제 종류 등 취약점을 알아내는 용도로 사용된다.

기본적으로 서비스에 관한 요청이 들어오면 그에 대해서 응답을 하도록 프로그램 되어 있기 때문에 운영체제, 프로토콜, 서비스 등에 관해 특징적인 응답을 보내게 된다. 악의적인 크래커는 그러한 정보를 이용하여 시스템에 관한 공격 방법을 결정하게 된다.

B. 서비스 방해

서비스 방해 공격은 시스템의 자원을 고갈시킴으로써 정상적인 서비스를 제공하지 못하도록 하는 공격이다. 단지 시스템의 정상적인 운영만을 방해하므로 관리자 권한을 획득하거나 시스템을 파괴시키고, 정보를 유출하는 등의 일은 하지 못하나 네트워크나 시스템 서비스 등을 마비시켜서 특정 서비스를 사용하는 사람들에게 큰 불편을 준다.

이 공격은 요즘 들어 심각하게 거론되고 있다. 인터넷이 상업적인 서비스 와 연결되면서 안정적인 서비스를 인터넷 사업의 중요한 요인의 하나로 생각하게 되었기 때문이다.

C. 시스템 파괴

시스템 파괴는 주로 한 시스템을 대상으로 이루어지는 공격이다. 프로그램이 가지고 있는 버그를 이용하여 관리자 권한을 획득하는 것이다. 악의적인 크래커가 관리자 권한을 악용하게 된다면, 시스템의 정보를 유출하거나 시스템을 파괴하거나 다른 시스템을 파괴하는 용도로 사용하게 된다.

1.3 크래커의 공격 시나리오 위로

(1) 공격 대상 탐색 및 약점 탐지

첫 단계로 특정한 사이트를 목표로 한 게 아니라면, 광범위한 네트워크에 대한 탐색이 우선적으로 행해진다. 가령 210.33.*.* 혹은 *.co.kr과 같은식 으로 넓은 범위로 약점이 존재하는 사이트에 대한 자료를 수집한다.

이 때, 기본적으로 수집하는 정보로는 시스템의 운영체제, 가동중인 네트워크 장비들, 각 호스트가 제공하는 서비스는 기본이고 , 각 시스템에 이미 알려진 보안 문제가 존재하는지, 방화벽(firewall)이 있는지 등이 모두 포함된다.

어떤 네트워크에 어느 정도의 시스템이 연결되어 있는지 알기 위해서는 기본적으로는 ping을 해보는 것 만으로도 충분하다. 또한 DNS조회를 통해서도 가능하다.

시스템의 운영체제 종류와 버젼 등등의 자세한 정보를 알기 위해서는 queso, nmap 등의 도구를 이용한다. 이들 도구는 운영체제 별로 IP 스택의 구현이 다르다는 점을 이용한 "IP stack fingerprinting"이라는 기법으로 운영체제의 종류를 구별할 수 있다.

특정 시스템의 약점을 조사하기 위해서는 시스템의 포트를 스캐닝하는 방법이 주로 이용된다. 사용되는 도구로는 sscan, mscan등이 있는데, 이를 통해 ftpd, telnetd등의 서비스가 제공되고 있는지, 어떤 데몬(daemon)이 사용되고 있는지 등을 알 수 있게 된다. 여기서 알아낸 약점은 다음 단계의 침입에서 우선적으로 사용된다.

만약 대상 네트워크에 방화벽이 존재한다면, 보통 방화벽이 필터링 하지 않는 ICMP 패킷을 이용해서 네트워크 내부의 정보를 알아낼 수 있다. 또한 방화벽 자체의 필터링 규칙을 알아내어 약점을 찾아내는 방법도 있다. 이런 경우에 firewalk, hping 등을 이용하게 된다.

(2) 특정 시스템의 침입

다양한 방법이 존재한다. 크게 나누면 시스템 자체의 헛점을 노리는 방법과 시스템 관리상의 헛점을 노리는 방법의 두 가지로 나눌 수 있는데, 이 두 가지가 뚜렷이 구별되는 것은 아니고 대개 중복되게 된다.

(1)의 단계에서 알아낸 시스템 중에 가령 wu-ftpd의 버퍼 오버플로우 (buffer overflow) 버그가 있는 Linux를 찾아내었다고 가정하자. 크래커는 매우 손쉽게 wu-ftpd의 해킹툴(exploit)을 구할 수 있으므로, 그 원격 해킹툴(remote exploit)을 통해서 관리자 권한의 쉘을 실행시킴으로써 관리자 권한을 획득할 수 있고, 곧바로 passwd 파일을 조작하여 시스템을 장악할 수 있다.

시스템 자체에 그런 헛점이 없더라도, 가령 시스템의 passwd 파일을 구할 수 있다면, crack을 통해서 사용자의 암호를 해독하여 침입 할 수 있다.

일단 침입이 성공한 후에는 rootkit 등을 설치하여, 기존의 시스템 유틸리티를 바꿔 치고, 로그

파일을 조작하거나 모두 지우고 , 다른 시스템들이나 다른 네트워크에 대한 공격의 전진기지로 활용하게 된다.

(3) 주위로의 침입 확대

대부분의 네트워크에서 사용되는 브로드캐스팅(broadcasting) 방식의 이더넷은 네트워크을 지나는 모든 패킷을 관찰 할 수 있으므로, 주변 시스템에 대한 침입이 용이하다.

공격자는 sniff라는 기법을 통해 네트워크 사용자의 계정 이름과 암호를 얻어내어 주변 시스템으로 로긴(login)을 시도하게 된다. 일단 로긴이 성공하면, 원격으로 시스템에 대한 침입을 시도하는 것보다 손쉽게 시스템을 장악할 수 있다.

또한 rlogin, rsh등의 r-service를 시스템에서 제공한다면, 이를 통해 이웃한 시스템으로 로긴해 들어갈 수도 있다.

전문적으로 특정 사이트를 목표로 하는 경우에는 위의 단계를 반복하여 추적을 어렵게 만든 다음 목표로 하는 시스템을 공격하게 된다.

(4) 가상의 시나리오

최근의 주식시장 폭락사태에 열 받은 크래커 K씨는 R증권사의 인터넷 주식거래 사이트를 마비시키기로 결심한다.

R사이트는 방화벽과 각종 보안 장치의 보호를 받으므로 R사이트를 직접 공격하는 것은 어려운 일이라는 것을 깨닫는다.

그래서 간접적으로 공격하는 방법을 택하기로 한다.

일단 K씨는 보안이 취약하기로 소문난 다수의 인터넷 업체와 학교 연구기관 등을 장악하기로 결정한다.

sscan과 rscan 그리고 nmap등을 돌려서 희생자를 물색한다.

드디어 고구마 대학 모 연구실의 컴퓨터 몇 대와 인터넷 곰팡이 판매업체 B가 취약하다는 정보를 수집한다. 이외에도 10여개 이상의 취약한 사이트를 알게 된다.

고구마 대학 네트워크에는 방화벽도 없고, 연구실 컴퓨터는 몇 년 전에 설치된 리눅스이므로 buffer overflow 버그를 이용하여 손쉽게 컴퓨터 한 대를 장악한다. 연구실의 여러 사용자들은 telnet을 사용하고 있었으므로, sniffer 를 돌려 다른 컴퓨터들도 희생명단에 넣게 되었다.

물론 이미 장악한 시스템의 로그는 지우거나 조작하고, rootkit을 설치하는 작업 등은 잊지않고 한다. 즉, 추적을 피하고 재침입을 쉽게 하기 위한 백도어를 만드는 작업이다.

고구마 대학 시스템을 이용하여 연속해서 회사B로 침입을 시도한다.

회사B의 경우에는 방화벽이 있었으나 방화벽 자체의 약점을 이용하여 침입한다. B회사는 방화벽을 믿고 내부 보안은 오히려 허술했다.

이런 식으로 몇 십 개의 시스템을 장악하고 나서, 본격적으로 R사이트에 대한 공격에 들어간다. TFN2k등의 도구를 이용하여 DoS(Denial of Service)공격을 시도한다. 여기저기 네트워크에서 공격을 받게 되어 결국 시스템은 마비되고 더 이상 서비스를 할 수 없게 된다.

2. 각 공격 유형에 대한 대응 방법

2.1 기본적인 설정 위로

(1) init 모드에 따른 기본 설정

init 프로세스는 AT&T 의 System V 유닉스에서 유래된 것으로, 데몬(daemon)이라 불리는 시스템 서비스의 시작과 종료를 관리한다. MS윈도우즈에서의 안전 모드와 같이, 각각 다른 상황에서 다른 시작 과정을 거치도록 도와준다. 각각 다른 모드를 런레벨(runlevel)이라고 한다.

init 프로세스는 다음과 같은 이점이 있다.

- 작업이 간단해진다. : 중앙 집중식의 일관된 관리가 가능하다.
- 맞춤형 설정 : 각각의 런레벨을 상황에 따라 맞추기 용이하다.
- 문제 해결 : 관리자는 여러 가지 런레벨을 사용해서 시스템 시작과 종료에서 일어나는 많은 문제점을 해결할 수 있다.

* 주의

슬랙웨어(Slackware)와 같은 시스템은 BSD init 프로세스를 사용한다. 레드햇(RedHat), 맨드레이크(Mandrake), 수세(SuSE), 칼데라(Caldera)는 System V init 프로세스를 사용한다. 이 글에서는 System V init 프로세스에 관련된 것만 설명하게 된다.

(2) init.d 디렉토리

init.d 디렉토리는 데몬(daemon)을 실행시키는 대부분의 스크립트가 저장되어 있는 디렉토리다. 대부분의 배포본에서 /etc/rc.d/init.d 에 있다.

이 디렉토리에 있는 스크립트를 실행하기 위해서는 "start"나 "stop" 둘 중 하나를 사용해서 시작 및 종료한다. 예를 들어 inetd 데몬을 실행하기 위해서는 :

/etc/rc.d/init.d/inetd start

비슷한 방법으로, inetd 데몬을 종료하려면 :

/etc/rc.d/init.d/inetd stop

을 실행하면 된다.

(3) 런레벨(runlevel)

리눅스 시스템에서 사용되는 런레벨을 살펴보면

Runlevel 0:

시스템이 종료된 상태

Runlevel 1:

Single User mode라고 부르는 상태이다. 시스템은 최소한의 데몬만 작동시킨다. 루트 파일 시스템은 읽기 전용으로 마운트 된다. 이 런레벨은 다른 런레벨로 부팅이 실패할 경우 사용하는 것이 일반적이다. 주로 시스템을 진단하거나 문제점을 해결하기 위해 주로 사용된다.

Runlevel 2:

네트워크 데몬(named, nfs 등)을 제외한 대부분의 서비스가 작동한다. 네트워크 서비스의 문제점이나 공유한 파일시스템을 확인하는데 유용하다.

Runlevel 3:

모든 서비스가 작동한다. 시스템은 정상적인 상태에 있게 된다.

Runlevel 4:

대부분의 리눅스 배포본에는 사용되지 않는다. 슬랙웨어(Slackware)에서는 런레벨 4는 런레벨 3의 상태에 X를 이용한 로그온이 가능하게 된 상태가 된다.

Runlevel 5:

모든 서비스가 작동하고, X를 이용한 로그온이 가능하다.

Runlevel 6:

시스템이 리부트된다.

(4) 런레벨 바꾸기

init 프로세스의 특징이자 제일 혼동되는 부분은 런레벨을 바꾸는 것이다. 시스템은 /etc/inittab에 명시된 런레벨이나 LILO: 프롬프트에서 주어지는 런레벨에 따라 부팅하게 된다.

먼저 기본 런레벨(default runlevel)을 바꾸기 위해서는 /etc/inittab 을 문서 편집기로 열어보면 다음과 같은 라인이 있을 것이다.

id:3:initdefault:

id: 다음에 나오는 숫자가 기본 런레벨이다. 이 숫자는 설치한 과정에 따라서 달라질 수 있다. 이 숫자를 원하는 값으로 바꾸면 된다.

*주의

기본 런레벨을 0이나 6으로 바꾸면 안 된다. 시스템이 부팅과 동시에 종료되거나 리부트되기 때문이다.

그리고 LILO: 프롬프트에서는 다음과 같이 설정하면 된다.

LILO: image runlevel

여기서 image는 커널의 이미지고 보통 linux이며, runlevel은 원하는 런레벨의 숫자를 넣으면 된다.

시스템이 동작하고 있을 때는

init runlevel

여기서 runlevel에는 원하는 런레벨의 숫자를 넣으면 된다.

*주의

시스템이 동작 중에 런레벨을 바꾸는 경우에는 사용하고 있는 서비스가 중단되어도 되는지 확인해야 한다. 예를 들어 런레벨 1로 바꾸면 네트워크 기능이 중단되면서 접속한 사용자들의 서비스를 중단하게 된다.

(5) 런레벨 디렉토리

각 런레벨은 init.d 디렉토리에 있는 스크립트를 가리키는 심벌릭 링크가 저장되어있는 디렉토리다. 이 디렉토리는 보통 아래와 같다.

/etc/rc.d/rc0.d
/etc/rc.d/rc1.d
/etc/rc.d/rc2.d
/etc/rc.d/rc3.d
/etc/rc.d/rc4.d

/etc/rc.d/rc5.d
/etc/rc.d/rc6.d

심벌릭 링크의 이름은 그에 해당하는 서비스가 시작될 것인가 멈출 것인가를 결정한다. 대문자 S로 시작하는 링크는 해당하는 런레벨로 부팅될 때마다 해당하는 서비스를 시작시킨다. 대문자 K로 시작하는 링크는 해당하는 런레벨로 부팅될 때마다 해당하는 서비스를 중단시킨다.

시스템이 런레벨을 바꿀때는 init가 현재 런레벨의 kill list(K로 시작하는 링크들)와 바꿀 런레벨의 start list(S로 시작하는 링크들)을 비교한다. 그에 맞추어서 데몬을 시작하거나 중지시킨다.

특정 런레벨에서 서비스를 제거하려면 단순히 첫 문자를 S에서 K로 바꾸어 주거나 지워버리면 된다. 추가할 때는 init.d 디렉토리의 스크립트에 심벌릭 링크를 만들면 된다. 예를 들어 lpd 를 런러벨 3에 추가하려면

ln -s /etc/rc.d/init.d/lpd /etc/rc.d/rc3.d/S99lpd

하면 된다. S와 lpd 사이의 숫자는 부팅되는 순서인데, 적당히 지정하면 된다. 기본적으로 서버 환경에서 다음 서비스는 제거하도록 하자.

S85gpm -> K85gpm
S90xfs -> K90xfs
S99linuxconf -> K99linuxconf
S60lpd -> K60lpd
S45pcmcia -> K45pcmcia

먼저 서버의 특성상 콘솔에서 X를 이용하여 접근할 필요가 없을 것이라는 판단아래 gpm은 제거한다. 그리고 gpm은 로컬유저가 root권한을 얻을 수 있는 exploit이 발견되었다.

xfs는 X 폰트 서버이다. 콘솔에서 X를 이용할 이유가 없다는 가정하에 제거하였다.

나머지 linuxconf는 원격으로 linuxconf를 이용하여 설정을 하지 않는다고 가정하고, lpd는 프린터를 이용하지 않을 것이라고 가정하고 제거했다. pcmcia기기를 사용하지 않으므로 제거했다.

(5) PATCH

설치 후 제일 먼저 해야 할 일이 패키지를 업데이트 하는 것이다. 소프트웨어는 항상 버그를 지니고 있으며, 악의적인 크래커는 그 약점을 이용해서 시스템에 관리자 권한으로 피해를 주기 때문이다. 패치는 소홀해지기 쉽지만 일단 처음 설치한 후 어느정도만 해 주면 한달 내지는 두달 정도 안심할 수 있다.

RedHat 의 패치는 RedHat의 패치사이트에서 얻을 수 있다.

RedHat의 패치사이트 주소 : http://www.redhat.com/apps/support/updates.html

rpm을 받아서 아래와 같이 설치하면 된다.

rpm -Uvh 패키지이름

(3) inetd.conf

inetd는 internet super-server이다. inetd는 부팅하면서 /etc/rc.local과 함께 실행되는 것이 보통이다. 실행되는 상태에서는 특정한 internet socket를 기다린다. 접속이 이루어지면 원하는 서비스를 /etc/inetd.conf 에서 찾아서 적절한 프로그램과 연결해준다. inetd 하나만 실행되어 있으면 다른 여러 가지 프로그램을 편리하게 띄울 수 있다.

설정은 /etc/inetd.conf 에서 할 수 있으며 # 은 주석을 뜻한다. 각 설정의 필드는 탭이나 스페이스로 띄워주어야 한다. 아래는 설정의 기본적인 내용이다.

service name

socket type
protocol
wait/nowait[.max]
user[.group]
server program
server program arguments

ftp, telnet 을 제외하고는 다른 서비스에 주석을 처리하는 것을 권장한다. telnet도 가급적이면 사용하지 말고 ssh를 사용하는 것을 권한다. 그리고 ftp의 경우에도 tcp_wrapper와 함께 사용하는 것이 좋다.

(4) root로 오는 메일 forwarding

시스템에 문제가 생기거나 침입에 대한 경고 메일은 root 계정으로 온다. 하지만 사용자가 root로 로그인을 하지 않는 경우가 많다. 이런 경우 root로 오는 메일을 관리하는 사람의 메일로 전달해주는 것이 좋다. /etc/mail/aliases 또는 /etc/aliases 를 수정해야 한다.

root: admin@test.test.co.kr

이렇게 수정을 한 후 newaliases를 한 다음 sendmail을 다시 시작해주어야 한다.

가령 sendmail pid가 4039라면,
kill -HUP 4039와 같이 하여 sendmail이 설정 화일을 다시 읽어오게 하거나,
#/etc/rc.d/init.d/sendmail restart로 sendmail service를 다시 실행시킨다.

이제는 root로 오는 메일은 admin@test.test.co.kr 로 오게된다.

(5) kernel variable

리눅스 커널은 보안을 위해서 많은 변수를 지정해두고 있다. 아래는 /etc/sysctl.conf 에서 추가하면 된다.

a. net.ipv4.tcp_syncookies

커널이 CONFIG_SYNCOOKIES 옵션이 추가된 상태로 컴파일 되었을 때 유효하다. 소켓의 SYN 한계가 넘을 때 syncookie를 보낸다. 이 방법은 SYN flooding 공격을 막는 일반적인 방법이다. 기본값은 FALSE이다. 하지만 접속이 많은 시스템에서 접속 수를 제한하는 방법으로 사용되어서는 안된다. 이때 SYN flooding 로그가 남게 된다. 이런 경고가 나타나지 않게하려면 아래 변수를 적절히 조정해야 한다.

net.ipv4.tcp_max_syn_backlog

이 변수는 SYN 플래그가 있는 패킷이 동시에 들어올 때 최대한 대기시킬 수 있는 패킷의 최대값을 뜻한다. 동시 접속수가 128이상이라면, 가령 동시에 200개의 접속이 가능하다면 256정도로 설정하는 것을 고려할 수 있다.

b. net.ipv4.icmp_echo_ignore_all

이 변수는 모든 ICMP ECHO 요청을 무시하는 것이다. 특수한 경우가 아니라면 설정하지 않는 편이 좋다.

c. net.ipv4.icmp_echo_ignore_broadcasts

이 변수는 브로드캐스트 또는 멀티캐스트 ICMP ECHO 요청을 무시한다. 기본으로는 설정되어 있지 않지만, IDC(Internet Data Center)와 같은 환경에서는 필수적이다.

2.2 sudo - root 권한을 지키기 위한 기본 중의 기본 위로

sudo main page : http://www.courtesan.com/sudo/
Licence : BSD style - slightly different

(1) sudo사용의 필요성

UNIX시스템에서 root는 무한대의 권한을 가지고 있다. 따라서 침입자들에게는 root 권한을 얻어내는 것이 최우선의 목표이다.

UNIX시스템에서 root의 권한은, 시스템 관리자, 시스템 커널 자체, 시스템의 서비스 데몬의 대개 세가지 경우로 한정되어있다.

root계정으로 원격으로 로긴하는 것은 대부분의 시스템에서 불가능하도록 설정하는게 보통이지만, su를 쓰면 원격으로 로긴한 계정에서도 root가 될 수 있다.

su대신에 sudo를 쓰면 다음의 장점이 있다.

첫째로 root passwd를 효율적으로 관리할 수 있다. 한 명이 아닌 여러 명의 관리자에 의해 시스템이 운영될 경우에, 여러 사람이 root 패스워드를 공유하면 root 패스워드의 정기적인 변경이 어려워짐은 물론, root 패스워드가 외부로 노출될 확률이 상대적으로 높아진다.

둘째로 root가 시스템 상에서 작업하는 시간이 짧아진다. root shell을 실행시켜서 오래 걸리는 작업을 하는 것은 많은 경우에 매우 큰 위험을 초래한다. 실수로 root filesystem에서 rm -rf와 같은 위험한 커맨드를 실행할 가능성도 있다. 따라서 sudo를 써서 root권한을 사용하는 시간을 최소화하는 것은 매우 큰 장점이다.

세째로 log추적이 쉬워진다. su root를 할 경우가 많아지게 되면, su root가 되어 무슨 작업을 했는지 일일이 확인하는 것이 불가능하다. 심지어 시스템의 일부가 손상되어도 그 사실을 눈치 채지 못할 수도 있다. sudo를 쓰게 되면 command단위의 logging이 가능하므로 정확한 기록을 확인할 수 있다.

네째로 관리의 권한과 책임이 명확해진다. root권한을 여러 사람이 공유할때, 어떤 작업이 누구에 의해 행해졌는가는 매우 불분명해진다. sudo를 써서 작업한다면 누가 어떤 작업을 했느지가 명확해진다.

다섯째 일종의 침입탐지의 기능을 한다. ps나 w command의 결과에서 만약 시스템에 root shell이 실행되어 있다면 분명히 침입을 확신할 수 있다. 왜냐면 그 시스템의 여타의 관리자는 분명히 sudo로 작업을 하므로 root shell을 실행시키지 않기 때문이다. 실제로 관리자가 root shell을 실행시킬 일이 있다 해도 매우 드문 경우이므로 관리자의 주의를 환기시키게된다.

시스템의 침입자는 sudo를 사용할 수가 없다. 왜냐면 sudo는 sudoers라는 파일에 등록된 사용자만 사용할 수 있는데, 침입자가 자신을 sudoers에 등록시키면 존재가 노출 될 수 밖에 없기 때문이다. 따라서 침입자는 무슨 일을 하든 시스템을 조작하기 위해서는 root shell을 사용해야 하고 침입이 당장 노출된다.

지금까지의 이유 이외에도 sudo를 쓴다는 것은 시스템을 제대로 관리하고 있다는 상징성을 가지고 있다. 즉 root권한을 최소화 시키는 것이 보안의 첫걸음 이라는 관점에서 보면, sudo의 사용은 바로 보안의 기본이라고 할 수 있다.

(2) sudo의 사용법

sudo를 사용하려면 당연히 시스템에 sudo가 설치되어 있어야 한다. 여기서는 설치법을 다루지는 않는다. 앞에서 연결해놓은 링크를 따라가면 sudo의 소스를 구할 수 있는데, 설치법은 소스안에 포함되어있다.

sudo가 설치되어 있다면, sudoers 파일에 원하는 사용자를 등록시킨다. 여기서는 /etc/sudoers가 설정파일이라고 가정한다. sudoers 화일의 위치는 설치 시에 지정한 옵션에 따라 달라질 수 있다.

visudo라는 커맨드를 사용하여 편집 가능하다.

만약 visudo를 쓰지 않고 직접 /etc/sudoers를 편집할 경우, 편집하다가 실수하면 sudo를 쓰지 못하게 되므로, 문법체크를 해주는 visudo를 쓰는게 좋다.

$ visudo

```
/etc/sudoers 중에서..
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for the details on how to write a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
stevens         ALL=(ALL) ALL
```

위와 같이 등록시키면 user stevens는 sudo command를 사용할 수 있다. sudoers의 복잡한 문법은 피하고 일단 쉽게 이해할 수 있도록 간단한 예제를 더 들어보자.

dgb boulder = (operator) /bin/ls, (root) /bin/kill, /usr/bin/lprm

이와 같이 설정하면, dgb라는 사용자는 operator의 권한으로 ls를, root권한으로 kill과 lprm을 사용할 수 있다.

사용할 때는 다음과 같이 쓴다.
$ sudo -u operator /bin/ls

sudo command의 일반적인 사용법은 다음과 같다.

$ sudo command [argument]

예제) sudo vi /etc/passwd

password : ******** <== root 패스워드가 아닌 사용자의 패스워드를 쓴다.

root권한으로 /etc/passwd를 편집한다.

2.3 TCP_WRAPPER 위로

(1) 소개

이 패키지(tcp_wrapper)는 시스템으로 들어오는 SYSTAT, FINGER, FTP, TELNET, RLOGIN, RSH, EXEC, TFTP, TALK 등의 여러 인터넷 서비스에 대한 요청(request)을 관찰하고 필터링 할 수 있다. 이 패키지는 4.3BSD-style의 socket과 System V4-style의 TLI를 모두 지원한다. 어디에 속하는 서비스 인지 몰라도 상관없다. 이 패키지는 어떤 설정도 바꿀 필요 없이(inetd.conf 제외) 여러 인터넷 서비스를 제공하는 프로그램들에 대해서 적용 가능하다. 추가적인 특징은 다음과 같다. 어떤 시스템이 어떤 네트워크 데몬에 연결할 수 있는가를 제한할 수 있다.

tcp_wrapper는 원격에서 접속한 사람의 신원을 확인 할 수 있는데, 이를 사용하여 관리자로부터 승인되지 않은 시스템으로부터의 인터넷 서비스 요청을 차단 할 수 있다. 다만 이 패키지를 사용하기 위해서는 inetd.conf를 수정할 수 있기 위한 root권한이 필요하다. 그리고 당연한 말이

지만 네트워크에 연결되어 있어야 한다. 그리고 이미 다 설치되어있을 syslogd이 필요하다. 대개의 시스템에 이것들이 이미 다 준비되어 있다고 가정한다.

(2) wrapper의 작동원리

client program(telnet) - - - - - server(in.telnetd) - - - - - application(remote login)
client program(ftp) - - - - - server(in.ftpd) - - - - - application(file transfer)

위와 같이 telnetd나 ftpd와 같은 프로그램이 telnet또는 ftp와 같은 프로토콜(protocol)의 요청을 받아들여 실제 작업을 수행하는 것이 인터넷 서비스의 원리이다. tcp wrapper는 client와 server사이에 위치하며 어떤 프로토콜에 대해서 어떤 시스템의 어떤 사용자가 요청을 하였는가를 체크 하여 허용되는 경우에는 실제 service에 요청을 넘겨주고, 그렇지 못한 경우에는 이런 정보들을 log로 남긴다. 이런 이유로 해서 tcpd(실제 wrapper)의 사용에는 overhead가 거의 걸리지 않는다.

client(ftp) - - - - - tcp wrapper(tcpd) - - - - - server(in.ftpd)

(3) 결점

tcpd의 경우에는 가장 처음에 오는 client의 요청에만 반응하기 때문에 여러 가지 서비스를 요구하는 NFS서비스에 대해서는 사용이 쉽지 않다. 이 문제는 관련된 서비스에 관한 문서를 참고하도록 한다.

(4) 실행방법

/etc/inetd.conf의 항목에 대해서 첫번째 줄을 두번째 줄로 바꾸는 방식처럼 바꾼다.

tftp dgram udp wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
tftp dgram udp wait root /usr/sbin/tcpd in.tftpd -s /tftpboot

tcpd의 로그는 syslogd에 의해 결정된다. 실제 log가 남는 위치는 /etc/syslog.conf를 참고한다. 메시지는 log파일에 추가되고, console로 보내지거나 아니면 @loghost로 mail형태로 보내진다.

(5) hosts.allow, hosts.deny 수정

이제는 각 서비스 별로 서비스를 허용하고 거부할 시스템 주소들을 정해준다. 이는 hosts.allow , hosts.deny를 수정하면 된다. 파일을 수정하는 방법은 다음과 같다.

% vi hosts.allow

/etc/inetd.conf에서 현재 시스템에서 작동 중인 인터넷 서비스를 확인하고 그 데몬의 이름을 확인한다. 시스템에서 사용하지 않은 데몬이 있을 경우는 hosts.allow 파일에 그 줄(line)의 첫 글자로 # 를 붙인다.그리고, 각 데몬의 이름을 /etc/inetd.conf에서 확인하여 정확히 바꾸어 준다.

다음은 /etc/hosts.allow를 수정하여 서비스별로 접속을 허용할 시스템을 지정한다. 기본적으로 ALL로 모든 시스템의 접속을 허용하고 접속을 거부할 시스템을 EXCEPT 다음에 지정한다. 이때, DNS name보다는 ip address를 사용하기 권장한다.

예)

```
telnetd: ALL \
         EXCEPT  192.168.10. \
                          192.168.12.50 \
                          192.168.12.51 \
                          192.168.11.10
```

hosts.allow의 수정이 끝나면, hosts.deny를 수정하는데, hosts.allow에서 EXCEPT 통해 지정했으므로 다음과 같이 설정한다.

% cat hosts.deny

ALL: ALL : (/usr/sbin/safe_finger -l @%h | /usr/bin/mail -s %d-%h root) &

hosts.deny에서는 hosts.allow에서 허용한 접속 말고는 모두 거부하며 거부된 시스템에 safe_finger를 시도한다. safe_finger는 일종의 finger 클라이언트로서 원격시스템의 finger 서버가 보내는 데이타에 의한 피해를 최소화할 수 있다. safe_finger를 하여 접속을 시도한 시스템의 사용자 신원을 확인할 수 있다.

위 hosts.deny 예제에서는 거부된 접속의 사용자 정보를 얻기 위해 safe_finger를 하고, 얻은 정보를 관리자(root)에게 메일로 통지해준다.

hosts.allow와 hosts.deny의 수정이 끝났으면 /etc 디렉토리에 복사를 한다.

% cp hosts.allow /etc
% cp hosts.deny /etc

(6) inetd.conf의 수정

이제는 inetd.conf 를 수정할 차례이다. 아래 주석이 붙은 wrapper 설정이 안된 것이고, 주석처리 되지 않은 부분이 wrapper 설정이 된 부분이다.

```
#ftp     stream  tcp     nowait  root    /usr/sbin/ftpd          ftpd
#telnet  stream  tcp     nowait  root    /usr/sbin/telnetd       telnetd
#shell   stream  tcp     nowait  root    /usr/sbin/rshd          rshd
#login   stream  tcp     nowait  root    /usr/sbin/rlogind       rlogind
#exec    stream  tcp     nowait  root    /usr/sbin/rexecd        rexecd
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/ftpd
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/rlogind
exec    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/rexecd
```

Wrapper를 통한 서비스만을 수정하는데, 원래의 설정을 그대로 복사한 후, #기호를 붙이고 /usr/sbin/ftpd 등의 각 서비스를 /usr/sbin/tcpd (/usr/sbin/tcpd가 존재하는 경우)로 바꾼다. 그리고, 마지막에 그 전에 있던 서비스를 기입한다. 위의 예를 잘 참고하자.

(7) tcpdchk를 이용한 configuration 확인

wrapper의 설정이 끝나면 tcpdchk를 통해서 제대로 설정이 되었는지를 확인한다.

% tcpdchk

이 때 에러가 나오면 해당 설정을 수정한다.

(8) tcpdmatch를 이용한 configuration 확인

설정이 제대로 되었으면 tcpdmatch 를 이용하여 설정한 시스템에서 접속을 시도할 경우 접속이 허용이 되는지 거부되는지 확인한다.

```
usage: tcpdmatch [-d] [-i inet_conf] daemon[@host] [user@]host
        -d: use allow/deny files in current directory
        -i: location of inetd.conf file
```

```
% tcpdmatch ftpd angel.test.co.kr
client:   hostname angel.test.co.kr
client:   address  192.168.14.53
server:   process  ftpd
```

```
matched:  /etc/hosts.allow line 39
access:   granted

%tcpdmatch telnetd devil.test.co.kr
client:   hostname devil.test.co.kr
client:   address  192.168.12.51
server:   process  telnetd
matched:  /etc/hosts.deny line 17
command:  (/usr/bin/safe_finger -l @%h | /usr/bin/mail -s %d-%h root ) &
access:   denied
```

(9) inetd의 재시동

여기까지 이상이 없이 되었으면 이제는 inetd를 다시 실행시켜 wrapper가 작동하도록 한다. 우선 ps 명령으로 pid를 찾아낸다.

```
% ps -u root | grep inetd
388 ?? I 0:08.56 /usr/sbin/inetd
```

그리고 HUP(hangup) 신호를 전달한다.

```
% kill -HUP 388
```

(10) 설치 확인

devil.test.co.kr 등 접근 거부된 시스템에서 접근했을 때 거부가 되면서 로그가 /var/log/secure에 남아 있는지 확인한다.

2.4 ssh - Network Sniffing의 대비책 위로

(1) Network Sniffing이란?

브로드캐스팅 미디어(broadcasting media)인 이더넷(ethernet)의 경우 한 시스템에서 다른 시스템으로 보내는 모든 패킷은 같은 물리적 네트워크 내의 다른 모든 호스트가 볼 수 있다. 원래 Internet Protocol은 네트워크 내의 모든 시스템들이 믿을 만하다는 전제 하에 설계된 것이기 때문에, 불순한 의도의 사용자 혹은 시스템이 다른 시스템으로 전달 되어야 할 패킷을 가로채는 것에 대비하고 있지 않다.

그러나 인터넷 사용자와 참여하는 호스트들이 점차 늘면서 보안의 위험이 커지고 또한 나쁜 의도를 가진 사용자가 등장함에 따라서, 이더넷 상에서의 통신 약점이 크게 부각되고 있다.

침입자가 어떤 네트워크에 침입 했을 때, 시스템의 정보와 권한을 가로 채는 가장 쉬운 방법은 네트워크를 지나가는 패킷을 가로채는 혹은 들여다보는 것이다.

이더넷을 통해 전달되는 대부분의 패킷은 프로토콜과 내용이 plain text로 되어 있는 경우가 많으므로, 단순히 가로채는 것 만으로도 사용자의 계정, 패스워드, 이메일의 내용, 현재 작업하고 있는 일 등을 모두 알아낼 수 있다. 이와 같은 방법으로 침입을 시도하는 것을 sniffing 이라고 부른다.

(2) sniffing의 위험성

대부분의 침입의 경우, 하나의 시스템이 침입 당했을 경우에 같은 물리적 네트워크 내의 많은 시스템들이 연쇄적으로 침입 당하는 현상을 볼 수 있다.

가령 200대의 시스템이 연결된 네트워크에서 단 하나의 시스템에 약점 혹은 보안 버그가 드러나더라도, 나머지 전체의 시스템들이 모두 위험해지는 상황에 처하게 되는 것은 바로 sniffing이라는 기술이 존재하기 때문이다.

가령 많은 사용자들이 쓰고 있는 telnet service, ftp service등의 경우 로긴(login)시 쓰이는 계정과 패스워드의 패킷 상의 위치가 정해져 있고, 암호화 되지 않는 plain text로 전달되므로 매우 쉽게 가로 채서, 대상 시스템의 계정을 얻을 수 있다. 일단 계정을 얻은 다음에는 내부에서 해킹 툴(exploit)을 이용하여 시스템 관리자 권한을 얻어내는 단계로 진행하게 된다.

그 외에도 E-mail을 훔쳐내거나, NFS로 전송되는 파일을 가로채는 등의 다양한 위험성이 존재한다.

sniffing에 쓰이는 도구로는 Tcpdump, Sniffit, Hunt, DSniff 등이 유닉스 환경에서 쓰이고, MS 윈도우즈에서도 다양한 도구가 존재한다.

(3) 대책

기본적으로 ethernet과 같은 broadcasting network media를 안 쓰는 것이 대책이 된다. Switching Hub를 쓰게 되면, 각 호스트들이 패킷을 broadcasting하지 않게 되므로 패킷을 가로채일 염려가 없어진다. 완전히 보안의 문제가 사라진다는 것을 의미하는 것은 아니지만, 일반적인 단순한 도구로는 sniffing이 어려워진다.

그러나 대부분의 ethernet이 이러한 hub를 쓰지 않는 것은 물론 이거니와, 패킷이 지나가는 모든 물리적 네트워크이 안전한지 확인할 수도 없고, 네트워크상의 패킷의 근본적인 안전성을 확보해야 할 필요성이 있으므로 패킷 자체를 encapsulate혹은 암호화할 필요가 있다.

(3-1) SSH

encrypt되지 않은 telnet, ftp, rsh, rlogin등을 대체하기 위하여 SSH가 존재한다. SSH는 Secure SHell의 약자로서 rsh과 비슷한 인터페이스를 제공하며, 기능적으로는 telnet, ftp, rsh, rlogin, rcp를 대체하도록 되어있다.

SSH는 non-commercial use only의 라이센스로 컴파일 해서 설치 할 수 있다. 버젼1과 버젼2가 있는데, 약간의 라이센스 차이가 있다. SSH는 DES 혹은 RSA의 대칭/비대칭 암호 알고리즘을 적용하여 패킷을 encrypt시키므로 중간에 가로채이더라도, 내용을 볼 수가 없다. telnet을 대체하는 형태로서 전세계에서 광범위하게 쓰이고 있다.

사용방법은 ssh는 rsh(rlogin)과 scp는 rcp와 거의 동일하게 쓸 수 있다. 가령 ping이라는 호스트에 steven이라는 사용자로 login한다면, 다음과 같다.

$ssh ping -l steven

PASSWORD:********

* URL : http://www.ssh.com
* 설치 : Licence문제로 rpm package를 찾기 힘들다.
위 URL에서 소스를 받아서 configure; make; make install로 설치해야 한다.

(3-2) OpenSSH

상업적으로 라이센스가 걸린 SSH를 대체하기 위해, OpenBSD 프로젝트 (http://www.openbsd.org) 에서 내놓은 대안으로 OpenSSH가 있다. SSH와 거의 완벽하게 호환되면서 라이센스는 모든 목적에 대하여 자유롭다. 따라서 원할 경우 리눅스 패키지등에도 포함시킬 수 있다. (사실 이 분야의 주류인 SSH가 그 동안 redhat패키지에 등장하지 않은 이유는 라이센스 때문이라고 볼 수 있다.)

사용방법은 SSH와 거의 동일하다.

* URL : http://www.openssh.com
* 설치 : 리눅스에 rpm으로 설치 시 OpenSSL(OpenSSH와 같은 위치에서 찾을 수 있다. 역시 OpenBSD Project에서 만들었다.)을 rpm으로 설치한다.

예) #rpm -Uvh openssl-0.9.5a-1.i386.rpm

다음으로 openssh를 설치하고, 차례로 openssh-server와 openssh-clients를 설치한다.

예) #rpm -Uvh openssh-2.1.1p4-1.i386.rpm
#rpm -Uvh openssh-server-2.1.1p4-1.i386.rpm
#rpm -Uvh openssh-clients-2.1.1p4-1.i386.rpm

(3-3) scp : ftp의 대안

scp는 SSH와 OpenSSH와 함께 설치되는 프로그램으로서 사용 방법 은 rcp와 동일하고, 계정/패스워드가 노출될 수 있는 ftp를 대신하여 사용할 수 있다. 단 양쪽 호스트에 모두 ssh/scp가 설치되어 있어야 한다.

*사용법

$scp ./testfile steven@somehost:/home/steven/
PASSWORD:******

현재 디렉토리의 testfile이라는 이름의 파일을 somehost라는 호스트 의 steven이라는 사용자의 홈 디렉토리에 옮겨놓는다.

(3-4) host key generation

간혹 host key가 없는 경우에는 host key를 다음과 같이 생성 시켜야 한다.

RSA key generation
#ssh-keygen -f /etc/ssh/ssh_host_key

DSA key generation
#ssh-keygen -d -f /etc/ssh/ssh_host_dsa_key

(4) ssh로 인한 효과

ssh를 씀으로써 ftp, telnet, rlogin, rsh등에서 발생하는 보안의 위험을 거의 다 막을 수 있다. host key로 identification도 가능하므로 host나 ip자체를 속이는 것도 방지할 수 있다. 다만 POP mail service를 할 경우에는 password가 노출될 수 있으므로, SSL을 사용하여야 한다.

2.5 Scan Detect Daemon 위로

(1) scan detect 에 대한 소개

크래커는 공격하려는 네트워크의 시스템들에 대한 정보를 얻기 위해서 일반적으로 스캐닝 도구(scanning tool)를 이용해서 알려진 보안 취약점에 대해서 조사한다.

예전에는 pscan , sscan 스캐너가 많이 쓰였다. 이러한 프로그램들은 web, cgi, 버전 등의 정보를 조사하며, 여러 가지의 TCP 포트(port)에 접속해가면서, 어떤 서비스가 제공되고 있고, 어떤 데몬이 실행 중이며 그 중에서 무엇이 취약한지 검사한다.

따라서, 관리자는 어떤 악의적인 크래커가 자신의 시스템들을 스캐닝 하고 있는지 알아내어 이에 대처해야만 한다. 이러한 것들을 대비하기 위해 스캐닝 탐지 도구(scanning detect tools)가 존재한다.

스캐닝 탐지 도구는 일정 시간동안 열린 포트 등의 정보를 기억하고 있다가 한 시스템에서 연속해서 많은 포트로 접속이 들어오면 스캐닝이라고 판단, 관리자에게 메일을 보내주게 된다.

여기서는 scanlogd 와 scandetd 두 가지 프로그램을 소개한다.

(2) Scan Detect Tools test

Scanlogd

http://www.openwall.com/scanlogd/

1. 설치

   = dependency =

   libpcap-0.5.2.tar.gz
   http://www.tcpdump.org
   ./configure;make;make install
   sudo mkdir /usr/local/include; sudo makedir /usr/local/include/net
   make install-incl ; make install-man

   libnet
   http://www.packetfactory.net/Projects/Libnet/
   ./configure;make;make install

   libnids
   http://www.packetfactory.net/Projects/Libnids/
   ./configure;make;
   chmod u+x mkinstalldirs
   make install

   scanlogd-2.1
   make libnids

   cp scanlogd.8 /usr/local/man/man8
   useradd scanlogd
   => login 못하게 설정한다.
   cp scanlogd /usr/local/sbin

2. 실행

   scanlogd &

3. 실행결과 평가.

   scanlogd 는 일정한 scan 시도를 감지하여 그 결과를 syslog 로 보낸다. syslog의 demon.alert 로 보내므로, /etc/syslog.conf에

   deamon.alert /var/log/alert

   와 같은 항목을 설정하면, scanlogd 가 보내는 메세지 만을 받을 수 있다. 그리고, log audit tool 인 guard 와 scanlogd 를 같이 활용하면 scanning 시도에 대해서 mail 로 결과를 통보 받을 수도 있다.

4. 실제 수행 예.

   atropos라는 이름의 시스템에서 tarzan이라는 컴퓨터로 nmap 프로그램을 돌려 본 결과이다. (tarzan에 scanlogd 가 깔려 있다.) nmap 에는 여러 가지의 port scan 방법이 있다. 가장 보통의 방법은 connect() 함수를 사용하는 방법이고, SYN scan , Stealth scan 등의 방법을 사용하기도 하여 갈수록 추적이 힘들어지고 있다.

   ****
   = 일반적인 scan.
   ****

   [k09@atropos:~]nmap-sT tarzan

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on tarzan.test.co.kr (192.168.12.10):
(The 1515 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
515/tcp    open        printer
946/tcp    open        unknown
1024/tcp   open        kdm

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

tarzan 에서는 다음과 같이 scanning 시도가 기록되었다.

```
tarzan]$ sudo tail /var/log/messages
Sep  6 19:14:18 tarzan scanlogd: 192.168.12.10 to 192.168.12.11 ports 80, 587, 3
74, 734, 473, 862, ..., f??p?uxy, TOS 00, TTL 64 @19:14:18
```

```
*****************
= SYN scanning.
*****************

[k09@atropos:/usr/local/man]sudo nmap -sS tarzan

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on tarzan.test.co.kr (192.168.12.10):
(The 1515 ports scanned but not shown below are in state: closed)
Port       State       Service
1024/tcp   open        kdm
. ( 이하 결과는 생략 ).

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second

[tarzan] tail /var/log/message
Sep  6 19:26:04 tarzan scanlogd: 192.168.12.10:63539 to 192.168.12.11 ports 41,
73, 481, 709, 26208, 1392, 929, ..., fSrpauxy, TOS 00, TTL 45 @19:26:04
```

결과가 잡혔음을 볼 수 있다.

```
*****************
= Stealth scanning
*****************

pos:/usr/local/man]nmap -sF tarzan

.. ( 걸과 생략 )

Sep  6 19:31:52 tarzan scanlogd: 192.168.12.10 to 192.168.12.11 ports 515, 22, 1
7007, 26208, 154, 3049, 812, ..., Fsrpauxy, TOS 00, TTL 44 @19:31:52
```

이것도 감지할 수 있었다.

Scandetd-1.1.3

http://wizard.ae.krakow.pl/~mike/download/scandetd-1.1.3.tar.gz

1. 설치

```
gunzip -c scandetd-1.1.3.tar.gz | tar xvf -
cd scandted-1.1.3
make
make install
```

2. 실행

(아래에서는 rpm package 가 아닌 source를 컴파일 한 것을 가지고 test 한 것이다.)
config.h 에서는 scan 을 어떻게 탐지할 것인가 결정한다. 컴파일 전에 이 값을 변경시키면 탐지 횟수와 강도를 조절할 수 있다. scan 시도가 탐지되었을 경우, root@localhost로 메일을 보내 주고, syslog 의 local2.notice 영역으로 log message를 전달한다. 따라서, root mail을 forwarding 해주어야 하고, /etc/syslog.conf에서 다음을 추가해 주어야 한다.

```
local2.notice     /var/log/scandetd_log
```

다음에 필요한 로그 화일을 만들어준다.
#touch /var/log/scandetd_log

3. 실행결과 평가

역시 nmap 을 가지고 테스트하였다.
****
= 기본 scanning
****

```
[k09@atropos:/usr/local/man]nmap -sT tarzan

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on tarzan.test.co.kr (192.168.12.10):
(The 1515 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
515/tcp    open        printer
946/tcp    open        unknown
1024/tcp   open        kdm

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second

#tail /var/log/scandetd_log

Sep  6 20:28:38 tarzan scandetd: www connection attempt from atropos.test.co.kr
Sep  6 20:28:39 tarzan scandetd: port 681 connection attempt from atropos.test.co.kr
Sep  6 20:28:39 tarzan scandetd: port 135 connection attempt from atropos.test.co.kr
Sep  6 20:28:39 tarzan scandetd: port 402 connection attempt from atropos.test.co.kr
Sep  6 20:28:39 tarzan scandetd: port 1530 connection attempt from atropos.test.co.kr
Sep  6 20:28:39 tarzan scandetd: port 39 connection attempt from atropos.test.co.kr
```

root@localhost 로 다음과 같은 메일 발송.

```
Possible port scanning from atropos.test.co.kr,
I've counted 1009 connections.
```

```
First connection was made to 80 port at Wed Sep  6 20:37:35 2000
Last connection was made to 5236 port at Wed Sep  6 20:37:50 2000

Probably it was SYN scan (0 FIN flags and 1009 SYN flags)


****************
= Stealth scanning
****************

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on tarzan.test.co.kr (192.168.12.10):
(The 1515 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
515/tcp    open        printer
946/tcp    open        unknown
1024/tcp   open        kdm

Nmap run completed -- 1 IP address (1 host up) scanned in 8 seconds
```

역시 detect 하였다.

```
Possible port scanning from atropos.test.co.kr,
I've counted 1401 connections.

First connection was made to 80 port at Wed Sep  6 20:38:56 2000
Last connection was made to 205 port at Wed Sep  6 20:42:19 2000

Probably it was FIN stealth scan (1400 FIN flags and 1 SYN flags)
```

(stealth 검색은 Port 를 random 하게 검사하기 때문에 일정하지가 않다.)

(3) 결론

scanlogd , scandetd 두 개의 scanning tool 을 설치하고, 시험해 보았다. 두 프로그램 모두 일정 정도 이상의 성능을 보이며, 고급의 scanning 기술도 detect 할 수 있는 능력을 보인다. 단, scanlogd 는 mail 발송의 능력이 없어서 guard 등과 같은 log auditing tool 과 같이 사용해야 한다는 점이 아쉬웠다.

두 프로그램 중에서는 scandetd 가 좀더 우수한 능력을 보이는 것으로 생각된다.

2.6 libsafe - Stack Buffer Overflow Attack과 그 대비책 위로

(1) stack buffer overflow attack이란?

buffer overflow attack은 메모리상에 임시저장공간으로 쓰이는 stack영역에 정해진 buffer크기보다 많은 크기로, shell code등을 보 내서 root shell을 실행시키기 위해 시도하는 공격이다.

주로 대상이 되는 process는 suid로 실행되고 있는 데몬이 되며, 프로그램 중 버퍼를 사용하게 되는 strcpy, strcat, sprintf, gets 등의 함수가 공격 대상이 된다.

이런 공격이 존재할 수 있는 이유는 C언어의 특징 때문인데, C언어 가 메모리 스택을 이용하는 변수에 대하여 메모리 경계를 체크하지 않기 때문이다.

(2) 위험성

현재 크래킹이 일어나는 경우 중, DoS(Denial of Service)등 서비스 방해를 목적으로 하는 경우가 아니고, 정보를 빼내거나, 사이트를 조작 하거나 다른 사이트를 공격할 목적의 전진기지로 이용하는 많은 경우에 stack buffer overflow가 이용되고 있다.

최근에는(2000상반기) RedHat Linux, S.u.S.E Linux, FreeBSD등에서 광범위하게 쓰이는 wu-ftpd에서 이 버그에 의한 공격이 이슈가 되었으며, 전통적으로 는 SunOS의 statd버그, qpopper 혹은 imapd등의 서비스들에 대한 공격이 끊임없이 존재해왔다.

따라서 이 버퍼 오버플로우 공격에 대한 위험이 존재하는 한 시스템이 안전하다고 보장할 수 없다.

(3) 대책

가장 기본적인 대책으로는 버그가 발견된 프로그램에 대한 즉각적인 업그레이드를 들 수 있다. 대체로 이러한 버그가 이슈가 되는 프로그램은 바로 당일에 그에 대한 패치가 등장하고 있다. 대체로 큰 문제가 되는 프로그램은 널리 사용되는 프로그램이고 널리 사용되는 프로그램일수록 패치 및 업그레이드가 빨리 이루어지는 경향이 있기 때문이다.

문제가 되는 경우는, 유닉스의 전문가가 아닌 사람이 관리하는 시스템의 경우이다. 많은 경우에 있어서 관리자들은 운영체제 설치 이후에 꾸준한 관리와 프로그램의 업그레이드를 소홀하기 쉽다. 상황을 더욱 악화시키는 것은 대부분의 관리자가 이슈가 되는 보안 문제에 무관심하거나 무지하다는 것이다.

또한 버퍼 오버플로우가 보안의 문제로 연결되는 경우는 거의 프로세스가 root 권한으로 suid 설정이 되어 있는 경우이므로, 디스크 전체에서 suid가 설정된 프로그램을 최소화 시키는 것도 대책이 될 수 있다. 이렇게 하는 것은 버퍼 오버플로우 공격뿐 아니라 다른 이슈가 될 수 있는 공격도 막을 수 있는 효과가 있다.

제2차적인 대책으로 원천적으로 버퍼 오버플로우를 불가능하게 하려는 시도가 있다. 그 시도로서 StackGuard라는 것이 있다. StackGuard는 컴파일 시에 적용되는데, StackGuard를 통해 컴파일된 코드는 버퍼크기를 넘어서는 입력을 차단하는 역할을 한다. 이 경우 문제가 되는 것은 StackGuard는 소스 코드에 대한 컴파일을 요구하므로, 기존의 프로그램이 모두 StackGuard를 통해 재컴파일 되어야 하는 불편이 생긴다.

(4) 또다른 대책, Libsafe

또다른 대책으로 Libsafe가 존재한다. libsafe는 Bell Labs의 연구로 개발되었다.
(http://www.bell-labs.com/org/11356/libsafe.html) libsafe는 library형태로서 문제가 되는 strcpy, scanf, sprintf등의 함수를 대체하는 역할을 한다. 만약 프로그램이 sprintf등을 호출하면, libsafe는 바운더리 체크를 하는 sprintf를 제공하여 버퍼 오버플로우 공격을 당할 여지를 막아주게 된다.

이것을 가능하게 하기위해서 libsafe는 Linux, FreeBSD의 ld.so에서 제공하는 LD_PRELOAD라는 환경변수를 이용한다. 가령 bash라면
export LD_PRELOAD=/lib/libsafe.so.1과 같이 설정 하므로서 이후에 실행되는 모든 실행파일에 대하여 libsafe를 적용하게 된다. csh, tcsh의 경우에는 setenv
LD_PRELOAD /lib/libsafe.so.1 이 된다.

또한 만약 버퍼에 대한 오버플로우가 발생하는 경우에는 /var/log/secure에 다음과 같은 로그가 남게된다.

```
Dec 21 13:57:40 denver libsafe[15704]: Detected an
attempt to write across stack boundary.
Dec 21 13:57:40 denver libsafe[15704]: Terminating
/users/ttsai/work/security.DO_2/test/t91
Dec 21 13:57:40 denver libsafe[15704]: scanf()
```

libsafe의 장점으로는 프로그램을 재컴파일 할 필요가 없이, 설치 후 환경변수의 적용만으로 이용이 가능하다는 것과, 설치가 쉽다는 점, 그리고 Bell Labs의 자체 테스트에 의하면 스택 바운더리 체크로 인한 성능저하가 최소화된다는 점이다.

(4.1) Libsafe의 설치법

* libsafe licence : LGPL

레드햇 리눅스의 rpm패키지로 설치한다는 것을 가정한다. 패키지는 libsafe-1.3-4.i386.rpm이다.

#rpm -Uvh --nodeps libsafe-1.3-4.i386.rpm

위에서 한 가지 이상한 점을 볼 수 있는데, --nodeps라는 의존성을 제거한 옵션을 주는 것이 바로 그것이다.

rpm패키징 상의 실수인지, dependency가 이상하게도 version 1.3-4, 즉 자기 자신에게 설정되어 있어서, 위의 옵션 없이는 설치가 안 된다.

이와 같이 설정한 후, export LD_PRELOAD=/lib/libsafe.so.1를 적당한 시작 스크립트 혹은 rc 스크립트에 설정해 놓는다.

물론 csh의 경우에는 setenv LD_PRELOAD /lib/libsafe.so.1과 같이 넣으면 된다.

LD_PRELOAD대신에 /etc/ld.so.preload라는 파일 안에 /lib/libsafe.so.1을 넣어놓는 방법도 있다.

여기서 한가지 주의 해야 할 점은 이와 같이 설정해놓으면, 실행되는 모든 프로그램이 이 라이브러리를 찾는데,

만약 실제 그 위치에 libsafe.so.1이 없다면, 프로그램이 실행되지 못한다는 것이다. 즉 운영체제 자체가 부팅되지 못하는 불행한 결과까지도 초래할 수 있다.

물론 그런 경우에는 다른 커널로 일단 부팅시켜서 해당되는 /etc/가 포함된 파일시스템을 마운트 시킨 후, 위의 설정을 삭제해서 원상 복구 시킬 수 있다.

(5) Libsafe 그러나 존재하는 문제점

문제는 최근에 이슈가 된 wu-ftpd 2.6.0에 대해서 libsafe를 설정하고 테스트했을 때, 이를 libsafe가 막아주지 못했다는 점에 있다.

ftpd의 소스를 추적한 결과 문제가 되는 부분에 libsafe가 체크 해주지 않는 snprintf()와 vnprintf()가 존재했는데, 과연 이런 이유 때문이었는지 확신할 수 없었다.

결국 libsafe를 설치했다고 해도, buffer overflow attack에 대한 안전지대에 들어갔다고 할 수는 없다. 앞으로 libsafe의 개선이 요청되는 부분이다.

테스트한 것은 libsafe.1.3이었고, 소스는 위에서 밝힌 Bell Labs의 홈페이지였다. 테스트 환경은

Intel x86 Pentium III 600Mhz
RedHat 6.2
wu-ftpd 2.6.0

이었다.

(6) 커널 자체의 수정에 의한 대책

커널 자체를 고쳐서 stack영역에서 프로세스의 코드가 실행되는 것을 불가능하게 함으로서 스택 버퍼 오버플로우를 막는 방법이 최근에 등장하고 있다.

가령 SunMicrosystems의 Solaris8같은 경우에는 /etc/system이라는 커널 configuration 파일에

set noexec_user_stack = 1 이라는 옵션을 주어서 stack영역에서의 프로그램 실행을 막을 수 있다.

리눅스의 경우에는 openwall project의 Linux kernel patch가 있다.
(http://www.openwall.com/linux/)

그곳에서 linux-2.2.17-ow1.tar.gz을 가져왔다고 가정하고 다음과 같이 설치한다.

물론 이 경우는 커널 버전이 2.2.17인 경우이다.

Linux의 버전 2.2.17의 커널 소스가 /usr/src/linux에 있다고 가정한다.

```
cd /usr/src/linux ( <== /usr/src/linux로 이동한다.)
patch -p1 < ~/linux/linux2.2.17-ow1/linux-2.2.17-ow1.diff
          ( <== 커널 2.2.17의 패치를 한다.)
```

kernel configuration에서 Security options를 적절히 설정한다.

kernel을 build하고 리부트 시킨다.

/etc/syslog.conf에 다음을 추가한다.

```
kern.alert                      /var/log/alert
```

(6.1) 커널 수정이 어려운 경우

일부 동적으로 공유된(dynamically shared) 라이브러리의 경우에는 프로그램의 코드가 어쩔수 없이 스택 영역에 들어가서 실행되어야 한다고 한다. 이 경우에는 이 방법을 포기해야 한다.

(7) 결론

현재까지 Stack Buffer Overflow 공격을 막는 가장 최선의 방법은 모든 프로그램을 최신으로 유지하고, suid가 설정된 프로그램을 최소한으로 유지하는 것이다. 또한 앞으로 libsafe등이 개선이 이루어지면 버퍼 오버플로우 공격에 대한 대비가 더욱 쉬워질 수 있다.

2.7 IPsec & VPN 위로

여기서는 간단히 VPN(Virtual Private Network)의 개념, IPsec의 기능과 IPsec을 이용해서 구현할 수 있는 VPN 터널에 관해 설명한다.

(1) IPsec 소개

IPsec(IP Security)은 IP(Internet Protocol)를 이용한 통신에 확장된 보안 서비스를 제공한다. 데이터를 암호화하거나 변조를 방지할 수 있으며 전자 서명을 통해서 데이터의 출처를 확인할 수 있는 등의 기능을 제공한다.

아마도 SSL(Secure Socket Layer)은 IPsec과 비슷하게 보일 것이다. SSL은 IPsec과 마찬가지로 서버와 클라이언트를 인증하며 암호화된 통신을 지원한다. 다른 점은 SSL이 어플리케이션 계층과 TCP/IP 계층사이에서 작동하는데 반해, IPsec은 IP 계층에서 작동한다는 것이다.

즉, SSL을 사용하기 위해서는 어플리케이션이 SSL을 지원하도록 프로그래밍 되어야 하지만,

IPsec은 IP를 확장한 것이기 때문에 IP를 사용하는 어플리케이션이라면 소스 코드의 변화없이 그대로 사용할 수 있다. 이것이 IPsec의 장점이라고 할 수 있다.

IPSec를 이용해서 서브넷(subnet) 사이의 암호화된 통신 터널을 말들 수도 있고(tunnel mode), 두 호스트 사이의 보안성 높은 통신(transport mode)을 지원할 수 있다. 여기서 생긴 보안성이 높은 통신 VPN(Virtual Private Network)이라고 한다. 일반적으로 IPsec은 아래 세가지 방식으로 동작한다.

HOST - HOST
HOST - NETWORK
NETWORK - NETWORK

(2) IPsec 기능

IP는 전송되는 데이터에 관해서 어떠한 방어도 해주지 않는다. 데이터를 보내는 사람, 받는 사람 모두를 보증해주지 않음은 물론이다. 이런 문제점을 해결하기 위해서 Ipsec은 다음 기능을 지원한다.

기밀성

데이터를 암호화하는 역할을 담당한다. 다른 사람이 인터넷을 통해서 원격 로그인할 때 패스워드를 누출시키고 싶지 않을 때 사용한다. 다른 사람이 자신의 통신을 보지 못하게 한다.

무결성

데이터가 전송되는 과정에서 변경되지 않도록 한다. 예를 들어 계좌번호와 금액을 이야기 하는 과정에서, 전송될 때 데이터가 변조되면 안될 것이다.

인증

자신이 보내는 데이터에 사인(서명)함으로써 그것을 받아보는 사람으로 하여금 진짜 자신이 보낸 것임을 확인시킨다.

재동작 방지

한번만 수행되어야 하는 트랜잭션이 실행될 때 허락 없이 반복되는 것을 방지한다. 다른 사람이 그 트랜잭션을 기록해서도 안되고 재반복 해서도 안될 경우가 있다. 만약 공격자가 암호화된 패킷을 해독하여 전송되는 데이터를 다 읽을 수 있다고 해도, 공격자가 가짜 패킷을 만들어서 트랜잭션을 가로채는 것을 방지할 수 있다.

(3) AH, ESP

IPsec은 위에 설명한 기능을 제공하기 위해서 AH와 ESP라는 두가지 프로토콜을 사용한다.

IPsec으로 이루어진 통신은 대칭형 암호화 알고리즘에 의해 통신 내용이 암호화된다. 이때 사용되는 프로토콜이 ESP(Encapsulated security payload)이다. ESP는 IP 헤더를 제외한 부분을 암호화한다. 암호화된 부분을 해독하기 위한 정보를 ESP 헤더에 입력한다. ESP 헤더는 IP 헤더 뒤에 붙이고, ESP 헤더 뒤에는 암호화된 통신 내용이 붙게 된다.

AH(Authentication Header)는 IP 헤더의 해시값을 계산하여 그 값을 IP 헤더 뒤 AH 헤더에 입력한다. 해시값은 데이터(data)를 확인하는 용도로 사용된다. AH는 암호화를 하지 않는다.

(4) IPsec 작동 방식

IPsec이 작용하는 호스트가 원래 데이터의 출처인지 아니면 게이트웨이(gateway)인지에 따라 두 가지 작동 모드로 나뉜다.

- 전송(transport) 모드는 패킷을 생성하는 호스트가 사용하는 방식이다. 주로 호스트와 호스트 사이에서 IPsec을 이용한 암호화된 통신을 할 때 사용된다. AH는 IP

헤더를 해싱하고, ESP 헤더는 IP 뒷부분의 전송 계층(transport layer) 부분을 암호화하여 저장한다.

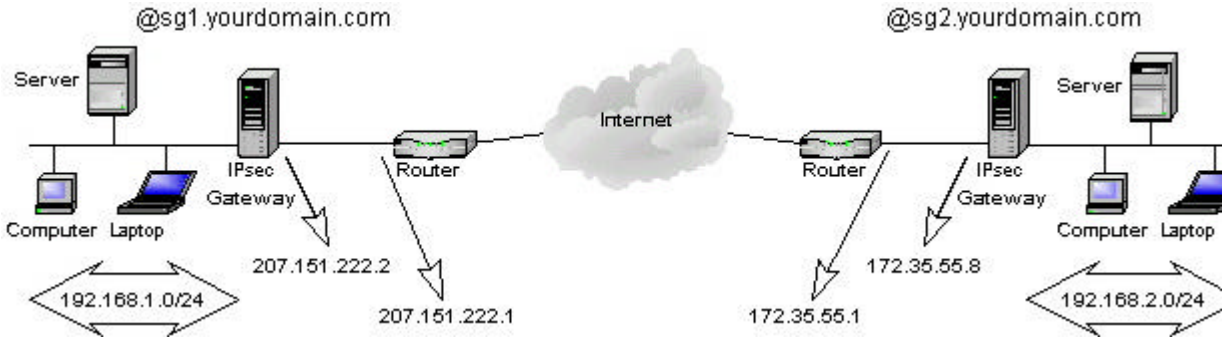
- 터널(tunnel) 모드는 IPsec을 이용한 통신의 한쪽이 게이트웨이(gateway)일 경우 사용된다. 서브넷 사이의 VPN일 경우에 사용된다. 이 경우에는 내부 네트워크 간의 통신은 게이트웨이 간 통신에 캡슐화 된다. AH와 ESP는 VPN 내부의 호스트 사이에 오간 통신 내용을 다루고, IPsec 처리를 해야 할 게이트웨이 주소가 입력된 새로운 IP 헤더가 붙게 된다. 즉 IP-in-IP 통신이 된다.

(5) IPsec의 간단한 작동 예

a. 간단한 subnet-subnet 설정

이 경우는 두 LAN 사이를 VPN으로 연결하는 것이다. 하나는 192.168.1.0/24인 네트워크(네트워크 A라고 하자)이며 다른 하나는 192.168.2.0/24인 네트워크(네트워크 B라고 하자)이다. 네트워크 A와 B 사이의 통신은 각각 207.151.222.2와 172.35.55.8 두 호스트를 IPsec 게이트웨이로 하여 이루어지게 된다.

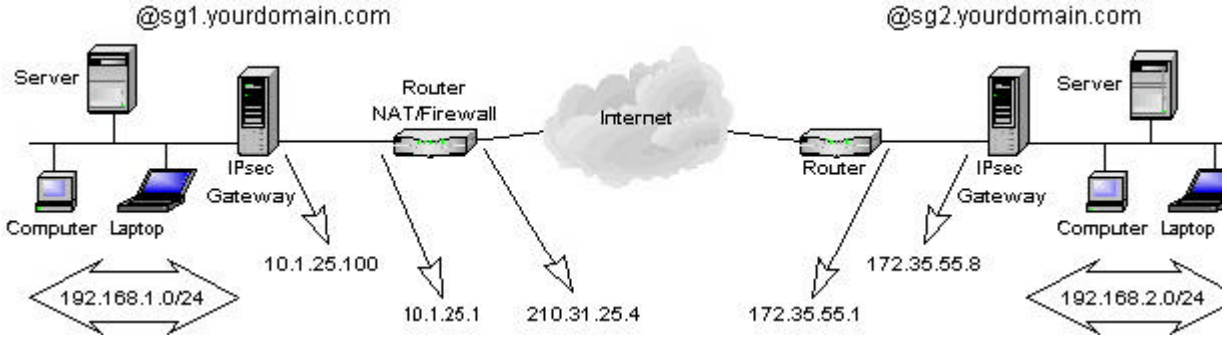
네트워크 A에서 생성된 패킷은 네트워크 A의 IPsec 게이트웨이를 거치면서 암호화되어 인터넷을 지나게 되며, 네트워크 B의 IPsec 게이트웨이에 도착하여 복호화되며 목적지에 도달하게 된다. 즉 네트워크 A와 B사이의 통신은 IPsec 게이트웨이간의 통신으로 비춰지게 되며, 인터넷에서 패킷을 가로채는 크래커는 그 내용을 알 수 없게 된다.



(그림 1)

b. subnet-subnet (내부 네트워크의 게이트웨이)

이 경우는 앞 경우의 네트워크 A의 IPsec 게이트웨이가 내부 네트워크의 구성원일 경우이다. 네트워크 A의 패킷은 10.1.25.100인 IPsec 게이트웨이 패킷으로 전달되며, 10.1.25.1에 도착하면 210.31.25.4라는 주소로 인터넷에 나가게 된다. 이 과정에서 패킷의 IP 헤더가 210.31.25.4로 바뀌게 되므로, IP 헤더의 해시값이 바뀌게 된다. 그러므로 IP 헤더의 해시값을 이용해서 인증을 수행하는 AH를 사용할 수 없고 암호화를 담당하는 ESP만 사용할 수 있다.

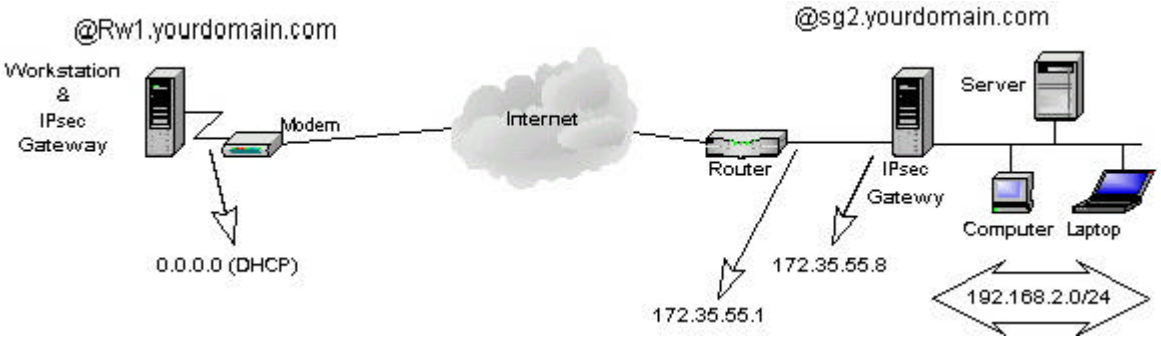


(그림 2)

c. client-subnet 설정

랩탑 또는 이동식 컴퓨터인 경우 IPsec 게이트웨이를 통해 암호화된 터널을 만들어 내부 네트워

크(192.168.2.0/24)에 접근할 수 있다. 이 경우에도 인터넷에서 패킷을 가로채는 크래커는 원격 시스템과 내부 네트워크 게이트웨이(172.35.55.8) 사이의 통신으로 알게 되며 암호화된 내용을 알 수 없다.



(그림 3)

d. 중앙 IPsec 게이트웨이를 터널 허브로 쓰는 경우

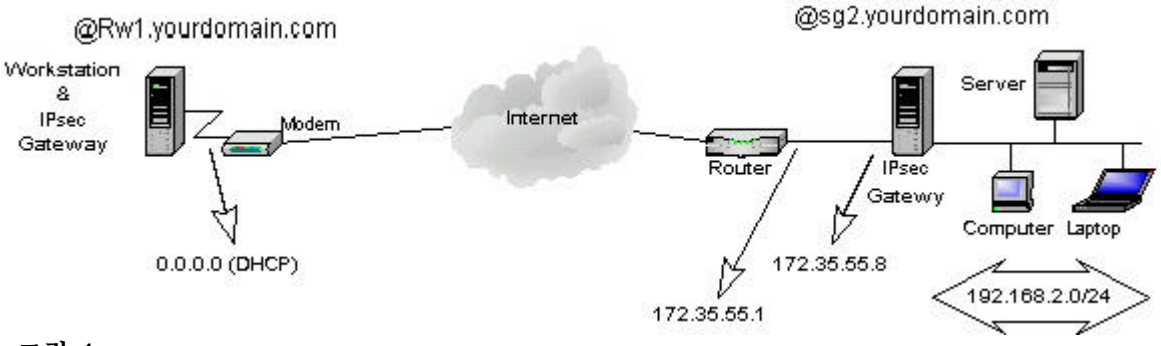
IPsec을 이용해서 VPN을 설정할 경우, 1:1로 연결해야 하는데, 서브넷이 많은 경우 각각 설정해야 할 터널이 많아지므로 중앙 터널 허브를 만드는 것이 효율적이다.

왼쪽에는 192.168.1.0/24(네트워크 A), 192.168.2.0/24(네트워크 B), 192.168.3.0/24(네트워크 C)가 있고 각각은 207.151.222.2(IPsec 게이트웨이 1), 210.105.32.8(IPsec 게이트웨이 2), 104.26.85.7(IPsec 게이트웨이 3)으로 통신을 한다. 오른쪽에는 192.168.0.0/16(네트워크 D)가 있으며 172.35.55.8(IPsec 게이트웨이 4)로 통신을 한다.

네트워크 A, B, C, D 사이의 통신은 중앙 IPsec 게이트웨이로 설정된 게이트웨이 4로 이루어지게 된다. 네트워크 A에서 B로 가는 통신을 예로 들어 보겠다. 네트워크 A에서 생성된 패킷은 게이트웨이 1을 지나면서 암호화 되고 인터넷을 지나 중앙 게이트웨이에 도착한다.

중앙 게이트웨이는 그 패킷이 네트워크 B에 도착해야 하는 패킷임을 확인하고, 게이트웨이 2로 보낸다. 게이트웨이 2에 도착한 패킷은 복호화되고 목적지에 도착하게 된다. 즉 VPN 내부의 통신은 각 네트워크의 IPsec 게이트웨이의 통신으로 비춰지게 되며 인터넷에서 패킷을 가로채는 크래커는 패킷의 목적지와 내용을 알 수 없게 된다.

중앙 허브 방식의 단점이 있다면, 중앙 게이트웨이가 불능이 되면 모든 터널은 작동이 멈춘다. 그리고 중앙 게이트웨이로의 네트워크 전송량이 많아지면, 중앙 게이트웨이의 작업 능력(CPU 성능 등)이 더 많이 필요하게 된다.



<그림 4>

(6) FreeS/WAN

FreeS/WAN 은 리눅스에서 작동되는 IPsec 패키지이며, GPL 라이센스의 무료 배포가 가능한 패키지이다.

홈페이지는 http://www.freeswan.org/ 이다.

2.8 Host IDS

(1) Host IDS 소개

Host IDS

시스템 파일, 디렉토리에 대해 무결성을 검사하여 크래커가 불법적으로, 허가되지 않은 디렉토리에 root 등의 권한을 얻어서 파일을 변경 또는 삭제 하였는지 여부를 검사하거나(File Integrity), log 를 분석하거나, 포트(port)가 열린 상태 등을 검사함으로써 (Audit Tool), 침입 시도가 있는지 또는 침입이 있었는지 여부를 검사한다.

(2) File Integrity

tripwire - 2.2.1

HTTP://www.tripwire.com/downloads/

1. 특 징

File Integrity 중에서는 가장 우수한 프로그램중의 하나라고 알려져 있다. 하지만, 상업적인 목적을 위해서는 이용이 제한되어 있기 때문에, 상용 프로그램에 넣을 수는 없을 것으로 보인다.

> 2. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.
> You may not rent, lease, distribute, sell, assign, pledge, sublicense, loan, timeshare or otherwise use the Software for the commercial benefit of third parties, but you may transfer the Software on a permanent basis, provided you retain no copies and the recipient agrees to the terms of this EULA.

(이상 Tripwire 의 Linux 배포판에서 발췌)
하지만, 가장 좋은 파일 무결성 체크 툴(File Integrity Checking Tools) 로서 비교의 지표가 될 수 있으므로, 설명해 본다.

1. 8 가지의 Integrity checking algorithm 사용. 기본으로 2 가지 사용

   => 2 가지 이상의 Checksum을 동일하게 유지하는 것은 매우 어려움.
   cracker 의 공격 시도를 완전하게 탐지할 수 있다.

2. checksum data를 db 형태로 유지, 직접적인 수정이 힘들고, 여러 가지 작업을 편하게 할 수 있다.

Fcheck - 2.07.512

http://www.securityfocus.com

1. 특 징

   1. md5 와 CRC check 만을 이용해서 무결성을 체크 한다.
      따라서, md5sum 을 그대로 조작하면서 file modify를 할 경우에는 Integrity check 를 제대로 수행할 수 없다.

   2. db 를 이용하지 않고, text file 형태로 checksum 값을 저장하기 때문에, 직접 수정의 위험성이 있다.

      하지만, 몇 가지의 tool 중에서 무결성 체크 툴로는 사용이 매우 쉽고, 설정이 간편해서 cron 데몬(daemon)과 함께 효과적으로 역할을 수행할 수 있을 것으로 보인다.

2. test

   /usr/local/fcheck/ 에 저장된다.
   /usr/local/fcheck/fcheck -ac : /usr/local/fcheck/fcheck.cfg를 토대로 checksum database 를 만든다.
   /usr/local/fcheck/fcheck -a : 지정된 database에 대해서 integrity 검사를 수행한다.

samhain-0.9.4

1. 특 징
   이제까지의 무결성 체크 툴들과 비슷.

2. test
   설치 과정과 문서들이 체계적이다.

   ```
   ./configure;make;make install

   #mkdir /usr/local/var
   #mkdir /var/local/var/log
   #samhain -t init
   ```

   Databae initialize를 하기 시작한다.
   이 때, /usr/local/etc/.samhainrc 의 설정 파일을 참고하면서 checksum을 저장하고, 결과는

   /usr/local/var/log/.samhain_file
   /usr/local/var/log/.samhain_log
   에 저장된다.

   (/usr/local/etc/.samhainrc 를 수정해서 없는 파일들, 디렉토리들은 주석처리해 주어야 한다.)
   file/directory integrity check를 위해서는

   samhain -t check

   를 입력한다.
   예를 들기 위해서, /etc/inetd.conf 와 /etc/sendmail.cf 를 수정하지는 않고, last modified time만 변경시켜 보았다.

   (결과)
   다음과 같이 여러 항목을 check 하여 변경되면 알려 준다.

   ```
   .....
   CRIT   : [2000-09-09T18:59:06+0900] Password file entry is NULL for UID 444
   CRIT   : [2000-09-09T18:59:06+0900] No owner name for file: /etc/inetd.conf
   CRIT   : [2000-09-09T18:59:06+0900] POLICY [ReadOnly] /etc/inetd.conf
   inode ln       mode    ext2flags   owner    group       size                mtime
   ------- -- ---------- ------------ -------- -------- ---------- --------------------
   <   26399  1 -rw-r--r-- ------------             root       3933 [2000-09-07T15:31:16
   >   26399  1 -rw-r--r-- ------------             root       3933 [2000-09-09T18:51:54
   CRIT   : [2000-09-09T18:59:06+0900] POLICY [ReadOnly] /etc/sendmail.cf
   inode ln       mode    ext2flags   owner    group       size                mtime
   ------- -- ---------- ------------ -------- -------- ---------- --------------------
   <   26446  1 -rw-r--r-- ------------     root     root      34181 [2000-02-18T07:51:21
   >   26446  1 -rw-r--r-- ------------     root     root      34181 [2000-09-09T18:54:32
   ```

3. 특징

   checksum algorithm 으로 tiger algorithm 을 사용한다.

(3) Audit Tools

Guard-2.6

http://www.penguin.cz/~ondrej/guard/guard.tar.gz

1. 특 징

   syslogd 는 각종 프로그램에서 나오는 log를 지정된 파일에 쓰도록 syslog.conf에 지정된 대로 수행한다. 하지만, 관리자가 이러한 로그를 일일이 확인하는 것은 귀찮기 때문에 많은 관리자들은 그렇게 하지 않는다.

   Guard는 syslogd 로 오는 모든 log message를 확인하여, 지정된 log message에 대해서 pattern matching을 수행, 원하는 log message 를 지정하면 그것을 mail 로 보내준다. 따라서, 중요한 log를 놓치지 않고 잡아낼 수 있다.

   단, 단점은 syslogd 로 가는 message만 잡아낼 수 있다는 점이다.

2. test

   ```
   # ./install.sh

   this installation file is pre-alpha version!
   it's simple set of commands with default settings and without error checking.
   it may help you install guard on fresh system. do not use it to upgrade
   from older versions. and do not use it if you don't want default
   configuration. when some problems occurs, please read the source code of
   this script to know what has been done and how to reverse it.
   to prove you have read and understood this now write the 26th word from the
   beggining.
   : fresh

   making directory /usr/guard
   adding user guard
   changing shell for user guard to /bin/false
   copying files
   creating logging FIFO
   creating logfile /var/log/security
   assigning permissions
   chown: root.guard: invalid group
   chown: root.guard: invalid group
   chgrp: invalid group name `guard'
   chgrp: invalid group name `guard'
   chgrp: invalid group name `guard'
   chgrp: invalid group name `guard'
   adding line to /etc/syslog.conf

   *** done
   guard is installed in directory /usr/guard

   these steps have to be done manually now:
   1. edit /usr/guard/guard.conf and set all variables that have to be set
   2. put /usr/guard/g to startup scripts
     WARNING: system lockup occurs when syslogd starts writing to FIFO
              and there's no guard process to read from it. always start guard
              before syslogd!
   3. start guard and killall -HUP syslogd
   ```

   다음과 같이 file을 복사해 준다.
   bash# cp g /etc/rc.d/init.d/

bash# ln /etc/rc.d/init.d/g /etc/rc.d/rc3.d/

useradd guard
cd /usr/guard
chown -R guard .
vi guard.conf

=> OutputDevicec 주석처리
관리자의 Mail 주소 달아주고, 설정 변경한다.
UID guard 로 바꾼다.

syslogd 전에 g 를 띄워야 한다. 그래야 pipe 를 이용해서 syslogd 로 message 넘겨줄 수 있다.

rules/generic-linux 에서 pattern match 되는 것에 대해선 /var/log/security에 기록하고, mail 로 report 를 보내 준다.

** error : Mailer 설정에서 에러 나서 메일이 보내지지 않는다.

3. 평 가

rule 을 조정함으로써 메일로 통보 받을 로그 메시지를 조절할 수 있다. 실시간으로 로그의 상태를 메일로 받아볼 수 있다.

logcheck-1.1.1

1. 특징

logcheck 도 지정된 여러 로그 파일에서 log를 검사하여, pattern match 를 수행하여 시스템에 유해하다고 여겨지는 error message를 검출, 지정된 email 로 보내 준다.

하지만, guard 와는 다음과 같은 방법에서 차이점을 보인다.

guard 가 syslog 의 input을 pipe로 받아서 데몬 형식으로 실행되면서, 실시간으로 에러 메시지 체크를 수행하는데 반해서, logcheck는 cron 에서 일정 시간만큼 실행시켜 주어야 한다.

그러면, logcheck는 이제까지 자신이 보았던 log file의 위치를 기억하고 있다가, 새로 진행된 부분만큼만 pattern match를 수행한다. 그래서 결과를 종합적으로 정리해서 email 로 보내 준다.

2. test

#make linux
vi /usr/local/etc/logcheck.sh
=> SYSADMIN (email 주소) 등을 수정한다.
cron 에다 /usr/local/etc/logcheck.sh 를 한시간마다 실행하도록 설정한다.
실행되면, 다음과 같이 log 를 분석해서, email 을 보내 준다.

```
Security Violations
=-=-=-=-=-=-=-=-=-=-
Sep  9 19:58:15 tarzan PAM_pwdb[10348]: authentication failure; k09(uid=502) -> root f
Sep  9 19:58:21 tarzan PAM_pwdb[10350]: authentication failure; k09(uid=502) -> root f

Unusual System Events
=-=-=-=-=-=-=-=-=-=-=-
Sep  9 19:58:15 tarzan PAM_pwdb[10348]: authentication failure; k09(uid=502) -> root f
Sep  9 19:58:21 tarzan PAM_pwdb[10350]: authentication failure; k09(uid=502) -> root f
```

3. 평가

guard 와의 공통점, 차이점을 주로 설명하면,

공통점으로는 문자열 pattern match 를 통한 syslog 로의 메시지를 audit 한다는 점이고,

차이점으로는 guard 가 데몬 형식으로 syslog로 가는 메세지를 pipe 형식으로 검사하는 데 반해서, logcheck는 로그 파일들의 이전 검사 위치를 기억, 그 후부터 검사한다는 데 있다.

즉각 대응력으로 보면 guard 가 우수하고, reporting 능력과 안정성에서는 logcheck 가 우수한 것으로 생각된다.

(4) 전체적인 평가

파일 무결성 체크 툴(File Integrity Tool)과 Audit Tool 에 대해서 살펴보았다.

test 한 툴들은 모두 나름대로의 특징을 가지고 있으며, 침입(Intrusion)의 여부를 탐지하는데 도움이 될 것으로 보인다. 하지만, 침입이 무엇인가 규정하는 데에는 각각의 시스템마다 다르기 때문에, 설치만 해 놓았다고 끝나는 것이 아니다. 각 시스템의 특성을 반영한 설정을 통해서 어떤 행위에 대해서 탐지할 것인가를 정해야 할 것이다.

File Integrity 에 있어서는 Fcheck 가, audit 에 있어서는 guard 가 비교적 좋은 성능을 가지고 있는 것으로 보인다.

2.9 Network IDS 위로

(1) 소개

Network IDS

네트워크 상에서의 패킷을 검사함으로써 침입 시도를 분석하는 방법이 다. 이더넷에서의 스니핑이 가능한 점을 이용해서 한 이더넷 세그먼트(segment) 내에서의 모든 패킷을 검사, pattern matching 등을 이용해서 침입을 탐지한다.

임의의 호스트에 대한 연속적인 포트 검색이나, remote exploit code 의 탐색 등을 통해서 네트워크 상에서의 침입 시도를 분석한다.

(2) Scanning Tools

nmap

1. 소개

프로그램들이 서로 통신을 할 때에는 서로를 인식하기 위한 구분자가 필요하다. 그런데, 하나의 시스템에는 여러 개의 프로그램들이 동시에 실행하고 있기 때문에, 시스템의 주소만으로는 서로를 완전히 구분할 수 없다.

그래서 프로그램들은 포트(port)번호를 하나씩 부여 받게 되고, 그러한 포트 번호로써 상대 시스템의 프로그램과 통신할 수 있게 된다. 그리고, http는 80번, telnet은 23번 등으로 약속된 포트 번호가 있어서, 특정한 프로그램은 특정한 포트 번호를 사용하게 된다. 이를 역으로 이용하면, 열려 있는 포트 번호를 알게 되면 어떠한 프로그램이 실행 중인지 알 수 있다. 이러한 작업을 포트 스캐닝(port scanning)이라고 한다.

nmap은 한 머신 혹은 큰 네트워크에 대해서 포트 스캐닝을 할 때 사용되는 도구이며, 많은 기능과 강력한 성능으로 유명하다. 주요 기능은 다음과 같다.

- TCP connect() scanning,
- TCP SYN (half open) scanning,
- TCP FIN, Xmas, or NULL (stealth) scanning,

- TCP ftp proxy (bounce attack) scanning
- SYN/FIN scanning using IP fragments (bypasses some packet filters),
- ICMP scanning (ping-sweep)
- TCP Ping scanning
- Remote OS Identification by TCP/IP Fingerprinting, and

2. 간단한 사용법 (%는 user 권한, #는 root 권한을 나타낸다)

   예를 들어서 알아보도록 하자.

   %nmap -sT foo.bar.com

   foo.bar.com 에 대해서 일반적인 connect() system call을 이용해서 port scan을 한다.

   #nmap -sS foo.bar.com

   foo.bar.com에 대해서 SYN scan 을 행한다. 이 때에는, 3-way handshake 를 완전히 하지 않고, SYN|ACK만들 듣기 때문에, 상대 host에서는 connection으로 기록되지 않는다.

   #nmap -sF foo.bar.com

   Stealth scan. 여러 가지 방법을 이용하여 방화벽이나 IDS(Intrusion Detection System, 침입 탐지 시스템)에 걸리지 않도록 스캐닝한다.

   #nmap -O foo.bar.com

   TCP/IP fingerprint를 이용해서 운영체제의 종류와 버전을 판별해 낸다.

saint

1. 소개

   saint는 종합적인 네트워크 관리 툴로서, 시스템이나 네트워크에 대해서 가능한 많은 정보를 모은 후, 포트(port) 검사, 프로그램의의 취약점 분석, 네트워크 설정 오류, 보안 문제 등을 분석해 준다. 관리자로서 네트워크 설정이나 보안상의 취약점을 알아보고자 할 때 사용할 수 있다.

   분석 결과가 Web을 통해 나오기 때문에, 분석이 용이하고, 분석 결과가 저장되어서 지난 결과도 손쉽게 찾아볼 수 있다.

2. 간단한 사용법

   프로그램을 설치한 후, root 권한으로

   #./saint

   를 실행한다.

   Web 상에서 target host를 지정한 후 scan을 행하면, 분석 결과가 나온다. 열려있는 포트와 버전 정보, 그리고 버전에 따른 취약점 정보까지 상세하게 알려 준다.

(3) IDS Tools

tcpdump-3.5.2

1. 특징

   Network Monitoring Tools 의 대표격인 프로그램이다. 거의 모든 TCP / UDP / ICMP 등

등의 packet 을 보여준다.

2. test

= dependency
libpcap-0.5.2.tar.gz
http://www.tcpdump.org
./configure;make;make install
sudo mkdir /usr/local/include; sudo makedir /usr/local/include/net
make install-incl ; make install-man

보통 많이 쓰이는 것으로는 "자신의 네트워크 밖에서 들어오는 패킷 중에서 웹 접속이 아닌 모든 패킷 검사" 가 있다. 이를 위해 다음 과 같이 입력한다.

tcpdump src not 192.168 and tcp and not port 80
tcpdump: listening on eth0
17:31:03.762427 cracker.test.co.kr.3312 > atropos.test.co.kr.ssh: . ack 2348842403 win 16268 (DF)

이렇게 하면 네트워크 세그먼트의 웹 접속을 제외한 모든 접속에 대해서 완전한 로그를 얻을 수 있다.

3. 결 론

tcpdump의 장점으로는 모든 패킷을 완전히 보여준다는 점에 있다. 그리고 옵션의 사용에 따라서 사용이 매우 강력해질 수 있으며, 필터링(filtering) 기능이 강력하다는 점을 들 수 있다. 하지만, 결과를 있는 그대로 보여주기 때문에, 보기가 불편하다는 것이 단점이다.

snort-1.6.2.2

✍ 특징

snort는 tcpdump보다 진보된 방식으로 패킷을 다룰 수 있다. 특정 IP로 들어오는 패킷에 대해서만 필터링을 할 수 있고, IP 별로 디렉토리를 생성, 패킷을 정리해서 보여준다.

그리고, 특정 쉘 코드에 대해서 경고 기능이 있어서 알려진 remote exploit 에 대해서 대응할 수 있다.

알려진 공개 소스 네트워크 침입탐지 시스템 중에서는 가장 강력한 툴로 생각된다.

✍ test

./configure;make;make install
make /var/log/snort
http://www.snort.org -> Rules Database 에서 최신의 rule을 다운로드 받는다.
모든 기능들은 Rule file 에 정의되어진 대로 작동한다.

Rule File을 보면, 다음과 같은 사항에 대해서 network level에서 탐지한다.

Backdoors - Signature Based
Denial of Service
Finger Rules
FTP Rules
Overflows
ICMP / PING
Scan / Probe
SMTP
TELNET
Mail Virus

(3) 전체적인 평가

Network IDS의 경우에는 전적으로 패킷 스니퍼(Packet Sniffer)를 사용하여 침입을 조사한다.

따라서, 네트워크 트래픽이 과도하게 많은 네트워크의 경우 네트워크 인터페이스 카드와 컴퓨터 하드웨어 사이에 너무 많은 신호가 생겨서 심한 경우에는 시스템이 다운되는 현상이 발생할 수 있으므로, 가능하면 다른 시스템과 분리해서 사용하는 것이 좋다.

그리고, 이더넷과 같은 브로드캐스팅 네트워크 단위로 검사할 수 있기 때문에 다른 네트워크의 경우에는 또 설치해야 한다. snort 의 경우에는 pattern matching rule 이 많이 있기 때문에, 그 중에서 무엇을 선택할 것인가를 잘 살펴서 선택해야 한다. http://www.snort.org 를 참고하면 될 것이다.

2.10 Firewall - 간단하지만 강력한 보안성 강화 위로

(1) Firewall사용의 필요성

하나의 네트워크 안에서 여러 대의 시스템들을 관리해야 하는 경우에 시스템의 숫자가 늘어나면, 하나하나의 시스템에 대한 관리가 어려워 질 수 있다.

특히 보안의 측면에서는 앞서 나온 모든 문제들을 점검하고, 시스템의 프로그램들을 최신으로 유지하고, 로그를 살피고, 침입 가능성을 항상 염두에 둔다는 것은 매우 어려운 일이다.

firewall을 설치해서 얻을 수 있는 장점은 대략 다음과 같다.

첫째로 불필요하고 불순한 의도를 지닌 모든 접근을 차단시킬 수 있다. firewall은 IP와 포트(Port)별로 통제할 수 있으므로, 네트워크 전체에 대하여 원하는 서비스만 제한적으로 외부에 대해 허용할 수 있다.

가령 내부적으로 telnet 서비스를 이용하고 있더라도, 외부에서는 telnet으로 접근을 못하게 할 수 있다. 아니면 특정 IP에서의 접근만 허용할 수도 있다.

둘째로 내부의 약점을 캡슐화 하여 외부에 공개하지 않는다. 내부적으로 일부 약점이 존재하더라도, 외부에서 그 약점을 이용할 방법을 차단하며 더 나아가 약점의 존재 자체를 드러내지 않는다.

가령 외부에서 포트 스캐닝을 하더라도, firewall에서 차단되므로, 어떤 약점이 존재하는지 알 수 없게 된다.

세째로 네트워크 전체에 대한 접근을 통제하는 일관된 rullset 체계를 만들 수 있다.

각 시스템에 tcp_wrapper를 설치하면, 보안성에서는 비슷한 효과를 얻을 수 있을지 몰라도, 설정이 복잡해지고 체계적으로 관리하기 어려워진다. 가령 네트워크 구성 체계의 변화가 생기는 경우 모든 시스템의 설정을 바꾸어 주어야 하는 불편이 있을 수도 있다.

이에 반해 firewall의 경우에는 firewall의 설정만 바꾸면 되므로, 설정이 쉬워진다.

(2) firewall의 사용법

상업적으로 나온 다수의 firewall이 있지만, 일단 여기서는 리눅스를 이용한 firewall을 소개한다.

(2-1) ipchains

- rpm package
RedHat package에 포함

혹은 다음 사이트에서 구할 수 있다.

ftp://ftp.freshmeat.net/pub/rpms/

- 기능

IP masquerading, packet filtering (firewall)의 기능을 동시에 할 수 있다.

각 패킷에 대하여, ACCEPT, DENY, REJECT, MASQ를 설정할 수 있다. 그 중에서 MASQ는 IP masquerade에 관한 행동을 지정한다.

input packet, forward packet, output packet에 대해 규칙을 지정할 수 있다.

-설정

예제1)

ipchains -I forward -s 192.168.1.0/24 -d 147.47.0.0/255.255.0.0 -j MASQ

source 192.168.1.*의 네트워크에서 147.47.*.*의 네트워크로의 packet forwarding을 규칙의 가장 앞에 추가한다.

192.168.1.*의 네트워크은 IP masquerade로 만들어진 가상 네트워크이다.

예제2)

ipchains -A forward -s 192.168.0.0/24 -d 0.0.0.0/0 -p tcp -j MASQ

source 192.168.0.*의 네트워크에서 임의의 destination으로의 tcp packet forwarding을 허가하며, 규칙의 가장 마지막에 append한다.

예제3)

ipchains -P forward DENY

기본적으로 이 호스트를 지나가는 모든 packet을 차단하고, 규칙에 허용된 패킷 만을 지나가게 한다.

예제4)

ipchains -A forward -p tcp -s 0.0.0.0/0 -d 0.0.0.0/0 telnet -j ACCEPT

이 호스트를 지나가는 모든 telnet packet을 허용한다.

예제5)

ipchains -A input -j DENY

이 호스트로 들어오는 모든 패킷을 차단한다.

!!!주의!!!
원격으로 이 명령을 실행하면, 실행하는 순간부터 모든 입력이 거부된다. ipchains을 수정조차 할 수 없게 되므로 주의한다.

예제6)

현재 설정 보기 ==>

ipchains --list

```
Chain input (policy ACCEPT):
Chain forward (policy DENY):
target prot opt     source          destination      ports
MASQ   tcp  ------  192.168.0.0/24    anywhere    any ->   any
ACCEPT tcp  ------  anywhere          anywhere    any ->   telnet
Chain output (policy ACCEPT):
```

예제7) 모든 설정 지우기

ipchains --flush

예제8) 설정 지우기

ipchains -D forward 1

forward에서 1번째 설정을 지운다.

-테스트

테스트1) ping의 icmp차단

ipchains -A input -p icmp -j DENY

ping 0로 테스트

결과)

```
PING 0 (0.0.0.0) from 127.0.0.1 : 56(84) bytes of data.

--- 0 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
```

테스트2) telnet의 차단

위에서 예제에서 설정한대로 telnet 서비스 차단

telnet 0로 테스트

결과)

```
Trying 0.0.0.0...
telnet: Unable to connect to remote host: Connection refused
```

(3) firewall만으로 안심할 수 있는가?

firewall을 통해서 하나의 네트워크를 묶는 종합적인 보안 체계를 만들 수 있다. firewall은 물론 보안의 궁극이라고 볼 수는 없다. 그보다는 일단 대문을 튼튼히 만드는 일이라고 할 수 있다.

대문이 튼튼해도 담장이 허술하거나, 뒷문이 열려있거나 한다면 대문의 자물쇠를 잠가도 소용없는 일이다. 혹은 대문 자체는 튼튼한데 자물쇠를 열어 놓는다면 이 또한 대문을 쓸모없게 만드는 일이다.

3. 정리 위로

앞에서 우리는 이제까지 리눅스 보안을 위해 필요한 각종 설정들과 도구들에 대해서 알아보았다.

시스템 운영체제가 설치된후 먼저 보안에 관련된 기본적인 설정이 필요하다. 이 부분은 빠뜨리기 쉽지만, 설정이 제대로 되어 있을 경우에는 문제의 원인을 찾는 것이 더 쉬워지므로 크래킹을 사전에 방지하는 역할을 할 수 있다.

관리자는 root권한으로 작업을 해야 하는 경우가 많은데, root계정을 사용하거나 su를 쓰는 것보다 보안성을 높일 수 있는 sudo를 사용하는 것이 좋다.

시스템으로의 원치않는 접근을 차단하기 위한 방법으로 tcp_wrapper를 사용할 수 있다. 이 방법을 통하여 시스템 크래킹을 어렵게 만들고, 보안 문제의 확산을 방지할 수 있다.tcp_wrapper를 사용해서 TCP 기반의 서비스에 접근 제한 기능을 설정할 수 있고, inetd 수퍼 서버를 이용하는 서비스와 libwrap과 함께 컴파일 한 응용 프로그램에 적용할 수 있다.

리눅스를 사용하는 많은 경우에, 시스템 콘솔(console)에서 직접 작업을 하는 경우도 있으나, 원격에서 접속을 하여 작업을 하는 경우도 많다. 이 때, sniffing 공격을 당할 수 있다. ssh를 사용하여 접속 과정을 포함한 통신의 모든 과정을 암호화하여 중요한 정보가 노출될 위험으로부터 벗어날 수 있다.

크래커가 시스템을 공격하는 주요 방법 중 하나가 시스템의 정보를 알아낸 뒤 취약점이 있는 서비스를 공격하는 것이다. scandetd등과 같은 프로그램을 이용하면 크래커의 시스템 정보 수집 시도를 탐지할 수 있다. 스캐닝(scanning)을 통한 크래킹 시도를 조기에 차단할 수 있게 해준다.

크래커의 공격 방법 중 한가지인 buffer overflow를 막기위한 대책으로서 libsafe를 비롯한 여러가지 방법이 존재한다. 근본적인 대책으로 문제가 있는 프로그램의 버젼 업그레이드 또한 필요하다.

telnet이나 rsh, rlogin,ftp 등 이외의 모든 다른 서비스에 대해 암호화된 통신을 제공하기 위해서는 IPsec을 이용해야 한다. IPsec은 IP를 이용하는 모든 서비스를 암호화하는 통신 서비스이고, IPsec을 이용하면 security gateway만 설정하면 지점간 통신을 모두 암호화하고, 나아가 가상 사설망을 구축하여 보안을 요구하는 네트워크를 구현할 수 있다.

크래커가 이미 침입했는지의 여부를 알기 위한 방법으로 host-IDS(호스트 침입 탐지 시스템)와 network-IDS(네트워크 침입 탐지 시스템)가 있다. host-IDS의 경우에는 화일 시스템과 같은 중요 자원의 변경이나 무결성을 검사해서 침입시도가 있었는지 와 침입을 당했는지 여부를 검사하며 network-IDS는 네트워크상의 패킷을 검사해서 침입시도를 분석한다.

네트워크 전체에 대한 원치 않는 접근을 차단할 수 있는 방법으로 ipchains가 있다. ipchains를 이용하면 리눅스 커널이 제공하는 방화벽(firewall) 기능을 이용하여 해당 시스템을 포함한 네트워크 전체에 접근 제한 기능을 사용할 수 있다. 접근 제한을 해놓은 시스템은 그 시스템에 보안 문제가 있더라도 접근 자체가 허용되지 않으므로 피해를 줄일 수 있게 된다. ipchains는 리눅스 커널 자체에 포함된 기능이며, 그 자체만으로도 상업용 방화벽을 대신할 만한 훌륭한 대안이다.

이상으로 간단하게나마 리눅스의 보안을 강화할 수 있는 여러 가지 방법들과 툴들에 대하여 알아보았다.

물론 여기 있는 것들이, 보안의 전부라고 말할 수는 없다. 크래킹의 방법이 날로 다양해지고, 새로운 방법이 등장하고 있는 경향에 따라, 보안의 방법도 다양해지고 있다. 그러나 그러한 새로운 모든 방법들이 결국은 기존의 보안을 기초로 하여 발전하고 있고 그러한 기초 없이는 보안이 이루어 질 수 없음은 명백하다. 또한 결국 방법과 툴의 다양성에도 불구하고 보안의 기본을 이루는 것은 보안을 철저하게 하겠다는 관리자의 기본적인 마음 자세와 태도 그리고 성실성이라는 것도 지적하고 싶다.

이 문서를 통해서 좀 더 안전하고 효과적인 시스템과 네트워크들이 날로 증가하기를 마음속 깊이 바라면서 부족한 문서를 마치려 한다.

이 문서가 만들어지기까지 아낌없는 격려와 성원으로 도와주고 지원해주신 (주)클루닉스와 동료들에게 특별한 감사를 전하고 싶다.

위로