보 안

1999

리눅스코리아(주)

목 차

I. 리눅스 보안

1 들어가며

현대 사회는 극도로 컴퓨터 통신망이 발전하고 복잡해 지고 있다. 이 컴퓨터 망은 점점 더 그 규모가 커지고 있으며, 또한 인터넷을 누구나 값싸게 ISP를 통해서든 다른 방법으로든 사용할 수 있고, 매우 빠르게 네트워킹 소프트웨어가 발전 및 개발되어가고 있는 상황에서 보안이라는 것은 점점더 중요한 이슈가 되고 있다. 현재 보안은 컴퓨터 네트워킹에 있어서 기본적으로 해결되어야하는 문제이다. 왜냐하면 컴퓨터 네트워킹은 근본적으로 보안에 취약하기 때문이다. 여러분이 A지점에서 B지점으로 데이타를 보낼 때는 그 중간 경로에 존재하는 수많은 컴퓨터들을 통해 데이타가 보내어진다. 악의가 있는 크래커에 의해 여러분의 데이터가 검출될 수 있으며, 조작될 수 도 있다. 패스워드가 오고간다면 여러분만이 알고 있는 개인 정보가 유출될 수 있는 열쇠가 크래커에게 쥐여 질지도 모른다.

2 그러면 무엇을 보호해야 하는가?

여러분의 시스템을 안전하게 크래커의 침입으로부터 보호하기 위해서 어떤 조취를 취하기 전에 여러분은 어떤 종류의 위험을 막을 것인가, 위험은 어느 정도 감수할 것인가, 여러분의 시스템은 얼마나 보안에 취약해도 상관없을 것인지를 결정해야한다. 시스템을 분석하여 보호대상을 결정하고, 그렇게 해야할 가치 및 이유를 생각하고 여러분의 데이타 및 다른 정보들을 누가 책임질 것인지를 결정해야 한다. 침입자가 여러분 소유인 파일을 읽거나 쓸 수 있으면 안되며 피해를 입힐 수 있는 명령어에 대한 실행권한이 있으면 안될 것이다. 중요한 데이타를 잃을 수도 있고 여러분의 회사가 지금까지 이루어 놓은 작업이 하루 아침에 물거품이 될 수도 있다. 여러분의 계정이나 시스템에 접근할 수 있는 사용자가 여러분을 아연실색하게 할 수 있다는 것을 기억해야 한다. 중요한 데이타를 잃고 나서 복구하거나 새로 만드는 시간이 얼마나 될 지 미리 생각해 보는 것이 좋다. 처음부터 신경 써서 보안 설정을 해 두는 것이 중요하며 , 데이타를 복구하는 데에서는 보안 설정을 하는 시간보다 10배 이상 든다는 것을 기억하자. 최근에 중요한 데이타를 점검한 적이 있는 지(그렇지 않다면 지금 부터라도 정기적으로 점검하라) 확인해 보고, 데이타 복구를 위한 백업 전략을 잘 수립하는 것도 중요하다.

3 어느 정도 안전하게 내 시스템을 지킬 수 있을까?

한 마디로 이 세상에 완벽한 보안 시스템은 없다. 다만 아직 알려지지 않은 것 뿐이다. 물론 알려진다면 즉시 대처하여 구멍을 떼워야 할 것이다(이점이 리눅스가 좋은 점이다. 구멍이 발견되면 즉시 해결방법이 나오던지 새롭게 패치된 소프트웨어가 나온다). 다만 여러분은 보안상의 헛점이 없도록, 더 정확하게 말한다면 헛점이 보이지 않도록 최선을 다해야 한다. 되도록이면 침입자가 시스템을 넘보지 못하도록 시스템 크래킹을 어렵게 만들어야 한다. 당

연한 얘기가 되겠지만, 보안을 강화하면 할 수록 시스템을 사용하는 데 제약이 따른다는 것을 알아야 할 것이다. 왜냐하면 보안을 강화한다는 것이 시스템 접근을 어렵게 만드는 것이 때문이다. 적당한 선에서 잘 조절해야 한다. 어느 정도가 적당한 것인지 여러분이 보안 정책을 잘 수립해야 한다. 보안 정책에 대한 것은 다음 RFC에 잘 설명되어 있다. 다음으로 가보라.

http://blitzen.canberra.edu.au/RFC/rfc/rfc2196.html

기본적으로 여러분의 시스템을 사용하는 사용자들이 이해하기 쉬운 일반적이며 간단한 보안 정책을 수립하는 것이 좋으며 다음과 같은 정책을 기본으로 하라. 허용되지않은 것은 금지하라. 위의 말은 사용자에게 어떤 서비스를 제공할 것이 아니라면 허용된 채로 두지 말고 금지하라는 말이다. 여러분이 모뎀과 PPP를 통한 인터넷 사용자이거나 소규모의 사이트를 운영하더라도 안심해서는 안된다. 침입자는 큰 사이트 만을 노리는 것은 아니다. 여러분의 소규모 사이트를 침입하여 다른 큰 사이트를 침입할 발판으로 삼을 수도 있는 것이다. 충분히 여러분 사이트의 보안의 헛점을 틈타 들어와 다른 사이트로 침투할 수 있다.

보안 문제를 여러 영역으로 나누어 생각해 보기 전에 시스템 관리자로서 알아야 할 사항을 다음에 적는다.

첫째, 시스템을 감시하라. /var/log/messages와 같은 시스템 일지를 체크하라.
둘째, 현재 시스템에서 서비스하는 데 사용되는 서버용 소프트웨어(이를 테면 아파치 웹서버)와 관련된 버그를 주시하고 판올림되는 것을 자주 확인하라. 지금까지 왜 보안이 중요하며, 무엇을 보호할 것인가 그리고 특별히 염두에 두어야 할 것들을 다루었다. 이제는 보안 문제를 여러 영역으로 나누어 알아 보자.
셋째, 주기적으로 암호를 바꾸되 되도록 예측할 수 없게 하라.
넷째, 슈퍼유저로서의 사용을 최대한 줄여라
다섯째, 중요한 디렉토리와 파일에 대한 접근 권한을 엄격하게 제한하라
여섯째, 중요한 자료는 모두 백업을 해 두라.

4 물리적 보안

처음으로 고려해야 할 것은 컴퓨터의 물리적 보안이다. 이것은 말 그대로 물리적 원인에 의한 컴퓨터 보안 문제를 다룬 것이다. 다시 말하면 악의를 가진 어떤 자가 컴퓨터를 직접 건드릴 때 발생하는 문제에 대처하기 위함이다.

이것은 여러분이 처한 상황에 따라 그 대처 방안이 다를 수 있다. 집에서 쓴다면 아이들이나 기타 같이 거하는 다른 이들로 부터 보호해야 할 것이다. 회사에서 사용하고 있고, 내 컴퓨터를 보호할 필요가 있다면 자리를 비울 때 컴퓨터를 잠궈두거나, 방의 문을 잠근다든가 등의 조치를 취할 필요가 있다.

요즘 나오는 신형 케이스들은 물리적 보안을 위한 여러가지 사항들이 고려되어 나온다. 잠금 기능이 있는데 본체 전면 열쇠를 통하여 케이스를 열지 못하게 하여 하드웨어 도난 방지를 막을 수 있다. 플로피나 다른 하드웨어 장비로 부팅할 수 없게끔 되어 있을 수 도 있다. 어떤 PC들은 케이스를 뜯어야만 열수 있는 경우도 있다. 열쇠를 잘 따는 자가 있다면 무용지물이지만 그래도 유용한 보안 장치가 된다.

바이오스의 패스워드 기능을 이용할 수 있다. 물론 여러분은 패스워드를 잊지 않아야 한다. 이 기능을 이용하면 바이오스 설정 사항을 변경시키는 것을 막을 수 있을 뿐만 아니라 시스템이 부팅할 때 암호를 모르면 부팅할 수 없게 할 수 없다. 이것도 케이스를 열 수 있으면 바이오스 를 리셋하여 피해를 줄 수 있으므로 완벽한 보안은 될 수 없지만 약간의 보안은 제공한다.

LILO같은 부트로더의 패스워드 기능을 이용할 수 도 있다. 패스워드를 알아야 부팅할 수 있다. 따라서 패스워드를 모르는 나 이외의 사람은 컴퓨터를 부팅할 수 없다. 하지만 플로피로 부팅하여 부트로더를 없애버린다면 소용없는 일이다.

부팅된 후에는 여러분의 콘솔이나 X Window 화면을 xlock이나 vlock을 이용하여 잠궈둘 수 있다. 이것들이 실행된 후에는 패스워드를 입력하여야만 화면 잠금을 풀 수 있다. 여러분이 작업된 내용을 알 수 없고 변경할 수 없지만 리셋하면 그 만이므로 이것도 완전하지는 않다. xlock이나 vlock에 대해 더 알기 원하면 매뉴얼 페이지를 참고하기 바란다.

가능한한 위의 모든 것을 다 동원하여 보안에 신경을 쓰는 것이 좋을 것이다.

그러면 물리적 보안 뚤렸다는 것을 어떻게 알 수 있을까? 먼저 여러분의 컴퓨터가 언제 재부팅되었는가를 조사하여 확인할 수 있을 것이다. 리눅스는 안정적이고 튼튼한 OS이다. 리눅스를 서버로 쓴다면 하드웨어의 교체나 OS의 판올림을 하고자 할 때 재부팅을 할 것이다.

여러분이 재부팅하지 않았는데도 재부팅되었다면 물론 누군가가 재부팅을 한 것이다. 시스템에 침입하는 방법중에 재부팅시키거나 전원을 꺼야 하는 것이 필요한 일이 많다. 보안 경고등이 켜진 것으로 알고 경계해라. 또한 컴퓨터 주변에 어떤 변화는 없었는지 점검해야 한다.

5 지역 보안

지역 보안이라는 것은 지역 사용자 즉 여러분의 시스템에 계정을 갖고 있는 자들의 공격에 대한 보안을 말하는 것이다.

연락처를 알지 못하는 자에게는 계정을 주지 않는 것이 좋다. 침입자들이 가장 먼저 시도하는 것은 지역 사용자들의 권한을 얻는 것이다. 일단 지역사용자들의 권한을 얻으면 슈퍼유

저권한을 얻기 위해 시스템이 제공하는 서비스의 헛점이나 버그등을 이용한다. 지역 사용자들에 대한 보안이 뛰어나면 침입자들에 또하나의 넘어야 할 보안 장벽을 주는 셈이된다.

지역사용자를 만들 때 주의해야 할 점이 있다. 그들에게 작업을 위한 최소한만의 권한만을 부여하라. 사용자들의 로그인 시간 및 로그인 장소를 알아야 하며 오래 동안 접속하지 않거나 연락처를 모르거나 접속할 필요가 없는 사용자의 계정은 삭제하는 것이 좋다.

시스템에 대한 모든 권한을 갖고 있는 계정이 있다. 이 계정으로 로그인 하면 시스템을 자기 마음대로 할 수 있다. 이러한 계정을 슈퍼유저(root) 계정이라 한다. 그러므로 일단 슈퍼유저 계정이 되면 이후하는 모든 작업에 상당한 주의를 기울여야 하는 것은 말할 것도 없다.

시스템을 관리하는 자도 일반 사용자 계정을 만들어서 사용하고 필요할 때만 슈퍼유저로 로그인해서 시스템을 관리하는 습관을 들여야 한다. 슈퍼유저 계정으로 들어가서 작업할 때 특별히 주의할 사항들이 있다.

1. 파일을 변경시크는 명령어(cp, rm, mv같은 명령어)와 와일드 카드를 함께 써서 어떤 작업을 할 때는 각별한 주의가 필요하다. 예를 들어 다음과 같은 명령을 수행할 때 지워서는 안되는 파일도 지워지는 것은 아닌지 확인해 보라.

```
rm foo*.bak
```

이때는 ls foo*.bak해서 지우면 안되는 파일이 있는 지 확인할 필요가 있다. 다른 방법으로는 정말 그러한 작업을 할 것인지 확인하는 옵션을 덧붙이는 것이 좋다. 쉘의 RC파일(.bashrc같은 파일)에 에일리어싱(aliasing)을 주어서 사용하는 것이 좋을 것이다. 다음처럼 말이다.

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

2. /etc/securetty 파일은 루트가 접속할 수 있는 터미널들을 적은 파일이다. 말그대로 안전한 터미날을 의미한다. 레드햇을 사용한다면 tty1부터 tty8까지가 기본적으로 적혀 있을 것이다. 이외의 터미날에는 루트로 원격 로긴할 수는 없다. 원격로긴을 굳이하여서 루트 권한으로 무슨 일을 하려면 먼저 일반 사용자 계정으로 로긴하고 su를 하여 사용하는 것이 좋다.

3. 루트로는 rlogin/rsh/rexec 종류의 명령들을 사용하지 마라. 이것들은 "공격" 방법의 대상이며, 루트로 쓸 때 매우 위험하다. /root 디렉토리에 .rhosts 파일은 절대로 만들지 말기 바란다.

4. 루트로 로긴하였을 때의 커맨드 패스 (command path)는 매우 중요하다. PATH 환경 변수에 의해 나타내어지는 커맨드 패스는 쉘이 실행할 프로그램들을 찾는 디렉토리를 명시하는 것이다. 커맨드 패스에는 절대 현재의 디렉토리를 포함해서는 안된다.

루트로서 작업을 할 때에는 언제나 느긋하고 신중하게 행동하고 많은 것들에 영향을 줄 수 있으므로 충분히 생각한 후에 명령어를 입력하라.

또한 패스에 있는 디렉토리는 소유주이외에는 쓰기 권한을 주면 안된다. 쓰기 권한 을 주어서 침입자가 커맨드 패스 중 하나에 이진파일을 넣을 수 있고 심지어는 시스템 실행 파일을 지울 수도 있다. 침입자가 넣은 이진파일이 실수로 실행되기라도 한다면 시스템은 먹통이 될 것이 뻔하다. 왜냐하면 침입자가 넣어두는 이진 파일은 시스템을 망치기 위해 실수로 누군가가 실행하길 바라는 이진파일이기 때문이다.

8.6 패스워드 보안과 암호화 (encryption)
여러분 대부분은 패스워드 사용에 익숙해 있을 것이다. 패스워드를 다른 사람이 쉽게 추측할 수 없도록 하는 것은 매우 중요한 일이다. 만약에 패스워드가 나 이외의 다른 사람이 알아내었다면 어떤 일이 생길 지 한번 상상을 해보라. 요즘의 리눅스 배포본들은 쉽게 추측할 수 있는 패스워드는 설정할 수 없도록 해주는 'passwd' 프로그램을 포함하고 있다. 예로, 영어 사전에 나오는 단어를 사용한다든지 하면 경고 메시지를 낸다. 암호는 영문자와 숫자를 조합하는 식으로 써서 추측이 어렵도록 하는 것이 좋다.

이제 암호화에 대한 것을 살펴 보자.

8.6.1 PGP와 공개 열쇠식 암호기법 (Public Key Cryptography)

PGP 등에 사용되고 있는 공개식 열쇠 암호기법은 두개의 열쇠를 사용한다. 하나의 열쇠로 암호화하고 다른 열쇠로는 암호를 복호한다. 이는 동일한 하나의 열쇠로 암호화와 복호화를 둘 다 처리하는 전통적 암호기법과는 다르다. 암호가 하나 뿐인 경우에는 암호화하는 쪽과 복호화하는 쪽의 양편이 모두 암호를 가지고 있어야 서로 비밀 정보를 주고 받을 수 있다. 또한 무슨 수로든 보안을 유지하면서 한 쪽에서 다른 상대방으로 열쇠가 전달되어야 한다.

기존의 방법에서는 하나의 암호를 사용한다. 그러나 공개 열쇠식 암호법은 공개용 열쇠와 비밀 열쇠라는 두 개의 열쇠를 사용한다. 공개용 열쇠로 암호화된 문서는 그 짝을 이루는 비밀 열쇠로만 열람할 수 있다. 그 반대도 마찬가지이다. 공개용 열쇠를 통하여 비밀용 열쇠를 추측하거나 비밀용 열쇠로 공개용 열쇠를 추측하는 것은 거의 불가능하다. 따라서 기존의 하나의 열쇠를 사용하는 방법에 비해 훨씬 보안에 강하다.

각 개인의 공개 열쇠는 누구나 암호화에 쓸 수 있도록 배포되고 이에 상응하는 복호화에 사용될 개인의 비밀 열쇠는 개인이 보관한다. 따라서 자기가 배포한 공개 열쇠로 암호화되어 자기에서 전달된 문서는 비밀 문서를 갖고 있는 나만이 열람할 수 있다. 공개 열쇠를 준 사

람들에게 편지를 보낼 때는 비밀 열쇠로 암호화하여 보내면 수신측에서 공개 열쇠를 이용하여 복호할 수 있다.

리눅스는 PGP (Pretty Good Privacy)를 지원해 준다. 2.6.2와 5.0이 잘 움직인다고 알려져 있다. PGP에 대한 기본 안내문과 사용법을 알고 싶으면 미국 정부는 강력한 암호 기법을 군용 무기로 취급하고 있고, 이것을 전자적 매체를 통해서 송출하는 것을 "수출 제한 조치"로 금하고 있으므로, 여러분 국가에 맞는 버전을 사용하도록 하라.

http//mercury.chem.pitt.edu/~angel/LinuxFocus/English/November1997/article7.html에

리눅스에서 PGP를 설치하는 자세한 설명서가 있다. 새로운 버전의 리눅스에는 패치를 구해서 붙여야 되는데,

ftp://sunsite.unc.edu/pub/Linux/apps/crypto

에서 구할 수 있다. http://www.rsa.com/rsalabs/newfaq/에 있는 RSA FAQ에서 좀 더 정보를 얻을 수 있다. 여기에서 "디피-헬램 (Diffie-Hellamn)", "공용 열쇠 암호기법 (public-key cryptography)", "전자 인증 (Digital Certificates)" 등의 용어에 대한 정보를 얻을 수 있을 것이다.

6 SSL, S-HTTP, HTTPS 그리고 S/MIME

SSL, 즉 시큐어 소켓 레이어(Secure Sockets Layer)은 인터넷 상에서의 보안을 위해서 넷스케이프 사에서 개발한 것이며 클라이언트/서버 인증용으로 쓰인다. SSL은 트랜스포트 (transprot) 레이어에서 작동되며 많은 종류의 데이터들을 사용자의 눈에 보이지 않도록 배경 작업상에서 암호화하는 안전하며 암호화된 통신로(channel)를 만들어 준다. SSL의 예제는 넷스케이프 커뮤니케이터로 시큐어 사이트의 파일을 열어 볼 때 쉽게 볼 수 있으며 넷스케이프 사의 데이터 인크립션을 비롯한 커뮤니케이터를 이용한 보안 통신(secure communication)의 기초로 쓰인다.

http://www.consensus.com/security/ssl-talk-faq.html 에서 추가 정보를 얻을 수 있다. 넷스케이프 회사의 다른 종류의 보안 기술은 http//home.netscape.com/info/security-doc.html에 가면 있다.

S-HTTP는 인터넷 상에서 보안을 담당하는 또 다른 종류의 프로토콜이다. 다중 열쇠 관리 기법(multiple key management mechanisms)을 지원하며, 데이터를 주고받는 두 사람이 사용할 암호 연산 기법 (cryptographic algorithm)의 일치를 옵션 교섭을 통해서 지원하는 동시에, 기밀성 (confidentiality), 인증 (authenticity), 데이터의 무결성, 송신 사실 증명 기능 등을 공급해 준다. S-HTTP는 사용 허가된 특별한 소프트웨어로만 사용이 되도록 제한되어 있으며, 암호화될 대상 데이터를 부분 부분적으로 잘라서 (블록) 암호화해 준다.

S/MIME (Secure Multipurpose Internet Mail Extension)은 전자우편이나 인터넷상의 메시지를 암호화하기 위한 인크립션 기준이다. RSA에서 개발한 공개 기준이니 만큼, 언젠가는 리눅스에서도 볼 수 있었으면 한다.

S/MIME에 대한 추가 정보는

http://home.netscape.com/assist/security/smime/overview.html

에서 구할 수 있다.

7. 리눅스 X-커넬 IPSEC 기술법

CIPE를 포함한 다른 형식의 데이터 인크립션을 포함해서, 리눅스용의 IPSEC 사용 기술법이 있다. IPSEC은 IETF가 만들었으며, 암호 기술적 보안 조건을 만들어 주는 통신용의 인증, 보전성 (integrity), 엑세스 관리, 기밀성 등을 지원해 주는 제품이다. IPSEC에 대한 정보와 인터넷 드래프트 파일은

http://www.ietf.org/html.charters/ipsec-charter.html

에서 구할 수 있다. 여기에서는 열쇠 관리 기법을 쓰는 다른 프로토콜에 대한 링크와 IPSEC 메일링 리스트, 그리고 메일링 리스트의 아카이브 등을 찾을 수 있다. 애리조나 대학에서 개발하고 있는 "x-커넬"이라는 리눅스 적용 기술은 네트워크 프로토콜을 사용하는 오브젝트-베이스 프레임워크이다. http://www.cs.arizona.edu/xkernel/hpcc-blue/linux.html에서 구할 수 있다. x-커넬은 메시지를 커넬 차원에서 통과시킴으로서 보다 빠른 적용이 되도록 해 준다. 다른 류의 암호화 기법은 수출 제한 조치 때문에 기본적으로 배포본에 포함되지 않는다.

8 시큐어 쉘 SSH와 스텔넷

SSH와 스텔넷은 원격 시스템으로 접속을 하면서 암호화된 커넥션을 만들기 위한 프로그램이다. SSH는 rlogin, rsh, 그리고 rcp를 보다 믿을 수 있는 것으로 대체해 주는 프로그램 뭉치다. 두 호스트간의 통신 암호화와 사용자 인증을 위해서 공개 열쇠 암호 기법을 사용한다. 이것은 중계인식 공격(man-in-the-middle attack: session hijacking)과 DNS 스푸핑도 방지하면서, 원격 호스트에 로그인하거나 호스트끼리 데이터를 복사하는 경우에 사용될 수 있다. 연결 시에 데이터 압축을 실행하며, 호스트간의 X11 통신을 보안화해 준다. SSH 홈페이지 http://www.cs.hut.fi/ssh/에서 구할 수 있다.

또한 윈도우스 웍크스테이션에서 여러분의 리눅스 SSH로 SSH를 보내는 데 사용될 수 있다. 윈도우스 클라이언트로 만든 무료제품이 많은데, http://guardian.htu.tuwien.ac.at/therapy/ssh/ 등이고, 데이터펠로우스사에서 만드는 유료 제

품은 http://www.datafellows.com에 있다. SSLeay는 시큐어 텔넷, 아파치 모듈, 여러 가지 데이터베이스, DES와 IDEA 그리고 불로우피쉬 (Blowfish) 등의 다종의 알고리듬을 포함한 넷스케이프의 SSL의 무료판이다.

텔넷식 통신상에서 암호화를 해 주는 텔넷 교체품이 이 라이브러리를 사용해서 만들어져 있다. 스텔넷은 SSH와는 달리 넷스케이프가 만든 SSL (Secure Sockets Layer)를 사용한다. http://www.psy.uq.oz.au/~ftp/Crypto/에 있는 SSLeay FAQ를 읽어보면 시큐어 텔넷과 시큐어 FTP에 대한 것을 찾을 수 있다.

9 PAM - 장착식 인증 모듈 (Pluggable Authentication Modules)

새로운 버전의 레드햇에는 "PAM"이라는 통일된 인증 방식이 들어있다. PAM은 사용자 여러분이 이진파일을 다시 컴파일할 필요가 없이 인증법, 제한 사항, 지역 인증법의 캡슐 처리 (encapsulate) 방법 등을 쉽게 바꾸게 해 준다. PAM의 캡슐화 방법은 이 파일의 내용 밖의 문제이지만, PAM의 웹 사이트

http://www.kernel.org/pub/linux/libs/pam/index.html

에 가서 꼭 보기를 권한다.

PAM으로 할 수 있는 일 가운데 몇 가지 만 들어보면 아래와 같다.

- 패스워드에 비 (非) DES 암호화 방법을 쓴다. (패스워드를 부르트 포스 공격을 써서 풀어내는 것이 어렵게 된다)
- 사용자들이 쓸 수 있는 (프로세스 수, 메모리의 양 등의) 자원을 제한하는 방법을 써서 서비스 거부식 공격 (Denial of Service DoS)을 못하도록 한다.
- 패스워드를 쉐도우 패스워드로 감추는 것을 쉽게 할 수 있도록 한다.
- 특정한 사용자가 특정한 시간에 특정한 장소에서만 로그인할 수 있도록 제한 조정하는 것이 가능하다.

시스템을 설치하고 조정하기 시작한 지 몇 시간 안으로, 공격 시도 시점에서 막을 수 있다. 예를 들면, 마침표가 붙은 rhosts 파일을 시스템 전체용으로 사용자 홈 디렉토리에서 사용하는 것을 막기 위해서 다음을 /etc/pam.d/login에 PAM을 사용해서 넣을 수 있다.

```
#
# Disable rsh/rlogin/rexec for users
#
login auth required pam_rhosts_auth.so no_rhosts
```

10 암호기술이 적용된 IP 인캡슐레이션 (Cryptographic IP Encapsulation :CIPE)

이 소프트웨어의 일차적 목적은 인터넷 등의 개방형 패킷 네트워크를 가로 질러가는 서브네트워크를 (가짜 메시지 주입, 트래픽 분석 등의 행위로부터) 보호하기 위한 방법을 제공하는 것이다.

CIPE는 데이터를 네트워크 수준에서 암호화한다. 네트워크의 호스트 사이에서 돌아다니는 패킷이 암호화된다. 암호화 엔진은 패킷들을 주고받는 드라이버 근처에 위치한다. 이것은 소켓 수준에서 데이터를 연결함으로 암호화를 하는 SSH와는 다른 것이다.

CIPE는 가상의 개인 네트워크를 구성하기 위해서 터널링에 사용될 수 있다. 저 수준(Low-level)에서의 암호화는 애플리케이션 소프트웨어를 수정할 필요가 없이 VPN에 연결되어 있는 두 네트워크 사이에서 투명하게 작동되도록 만들어 질 수 있는 이점이 있다.

IPSEC 기준은 다른 일도 하지만 암호화된 VPN을 만들기 위해서 사용될 수 있는 프로토콜의 집합을 정의한다. 반대적으로, 많은 옵션을 가지고 있는 IPSEC은 상대적으로는 헤비급이면서 복잡하며, 주어진 프로토콜 전부를 사용하는 경우는 아직은 드물면서도 열쇠 관리 등의 몇 문제는 아직 완벽히 해결되어 있지 않다. CIPE는 좀 더 간단한 방식을 사용하는데, 초기 설정 시에 파라미터 형식으로 (정말로 사용하고자 하는 인크립션 방식을 선택하는 등) 많은 조건에 대한 정해진 선택을 할 수 있다. 이것은 탄력적인 운영을 제한하기는 하지만, 간단한 (그리고, 이 이유로, 쉽게 디버그를 할 수 있는 등으로) 능률적인 설정을 가능하게 해 준다.

정보를 더 원하면 다음을 참조하라.

http://www.inka.de/~bigred/devel/cipe.html

다른 크립토그라피의 경우와 마찬가지로 이것도, 수출 제한 조치 때문에, 커넬과 함께 배포되지 않는다.

11 커브로스.(Kerberos)

커브로스는 MIT의 아테나 프로젝트 아래에서 개발된 인증 방식이다. 사용자가 접속해 들어오면, 커브로스는 패스워드를 사용해서 사용자를 인증하고, 네트워크 상에 흩어져서 존재하는 서버와 호스트들에게 이 사용자의 신분을 증명해주는 방법을 제공한다.

이 인증법은 리모트 로그인 (rhost) 프로그램 등에 의해서 패스워드 없이 사용자가 다른 호스트로 (.rhost 파일을 대신해서) 접속을 할 수 있도록 해준다. 이 인증법은 또한, 보내는 사람 (발송인)이 가짜가 아닌 것을 보증하는 동시에, 메일이 정확한 사람 (수취인)에게 전달

이 되도록 보증하기 위해서, 메일 시스템에 의해 사용될 수도 있다.

커브로스와 딸려 있는 많은 프로그램을 사용하는 궁극적인 효과는, 사용자가 시스템을 속여서 다른 사람인 척 "스푸핑"을 할 수 있는 능력을 거의 없애버리는 데 있다.

커브로스에 대한 추가 정보는

http://www.veritas.com/common/f/97042301.htm

에서 찾을 수 있고, 코드는

http://nii.isi.edu/info/kerberos/

에 있다.

12 쉐도우 패스워드

쉐도우 패스워드는 암호화된 패스워드 정보를 일반 사용자들로부터 비밀로 하기 위한 한 가지 방법이다. 대개 암호화된 패스워드는 /etc/passwd 파일에 누구나 읽을 수 있을 수 있도록 저장되어 있다. 이 파일을 패스워드를 추측해내는 프로그램에 돌려서 패스워드를 알아내려고 할 수 있다.

쉐도우 패스워드는 패스워드에 대한 정보를 특별 권한이 있는 사용자들만 읽을 수 있는 /etc/shadow 파일에 저장한다. 쉐도우 패스워드를 사용하려면, 패스워드 정보에 접근할 필요가 있는 모든 유틸리티들이 쉐도우 패스워드를 지원하도록 다시 컴파일되었는지 확인해야 한다. (위의) PAM은 실행 프로그램들을 리컴파일할 필요 없이 단지 쉐도우 모듈을 장착시킴으로써 쉐도우 패스워드를 쓸 수 있도록 해준다.

필요하다면 Shadow-Password-HOWTO 파일을 읽으면 된다. 이것은

http://sunsite.unc.edu/LDP/HOWTO/Shadow-Password-HOWTO.html

인데, 지금은 약간 오래되었고, PAM을 지원하는 배포본에는 필요가 없다.

13 크랙(Crack)과 존 더 립퍼 (John the Ripper)

무슨 이유가 있어서 Passwd 프로그램을 실행할 때 사용할 때 "쉽게 추측할 수 없도록 만든다"는 패스워드 규칙을 집행하지 못한다면, 여러분 스스로가 패스워드 격파 프로그램을 실행시켜서 실제의 사용자들이 안전한 패스워드를 쓰고 있는지 확인하는 것도 좋은 것이다. 패스워드 깨기 프로그램은 간단한 방식으로 작동한다. 사전에 있는 모든 단어와 그 변화형을 패스워드로 시도한다. 단어 하나 하나를 암호화하면서 암호화된 패스워드와 비교하는 것

이다. 만약에 일치하는 단어를 찾게되면, 암호를 알아낸 것이다. 많은 패스워드 크랙 프로그램들이 있다. 그 중에서 알고 넘어가야 하는 두 개가 바로 "크랙"과 "존 더 립퍼", (http://www.false.com/security/john/index.html)다. CPU 시간을 엄청나게 소비할 것이지만, 이 프로그램을 여러분이 먼저 사용해 봄으로서 혹시나 공격자가 이런 프로그램을 사용해서 시스템에 침입할 가능성이 있는지를 알아내는 동시에, 약한 패스워드를 가진 사용자들을 찾아내서 공지해 줄 수 있을 수도 것이다. 공격자가 여러분의 passwd(유닉스에서는 /etc/passwd) 파일을 얻으려면 우선은 다른 개구멍을 이용해 먼저 들어와 있어야 하겠지만, 이런 개구멍 허점들이 여러분이 생각하는 것보다 훨씬 흔하다는 점(즉, passwd 파일을 구하는 것이 어렵지 않다는 점)에 주의해야 한다.

/usr/doc/cracklib-2.7
/usr/doc/cracklib-2.7/HISTORY
/usr/doc/cracklib-2.7/LICENCE
/usr/doc/cracklib-2.7/MANIFEST
/usr/doc/cracklib-2.7/POSTER
/usr/doc/cracklib-2.7/README
/usr/include/crack.h
/usr/lib/libcrack.so
/usr/lib/libcrack.so.2.7
/usr/lib/libcrack.so.2
/usr/lib/cracklib_dict.hwm
/usr/lib/cracklib_dict.pwd
/usr/lib/cracklib_dict.pwi
/usr/sbin/create-cracklib-dict
/usr/share/gimp/patterns/cracked.pat
/usr/share/wallpapers/cracking_blueberry.jpg
/lib/security/pam_cracklib.so

14 CFS와 TCFS - 암호화 파일 시스템과 투명 암호화 파일 시스템

CSF는 전체 파일시스템을 암호화하고 사용자들이 암호화된 파일을 이 암호화된 파일시스템에 저장할 수 있도록 해주는 방법이다. 이것은 지역 컴퓨터에서 작동 중인 NFS 서버를 사용한다. rpm 파일을

http://www.replay.com/redhat/

에서 구할 수 있고, 그 작동방식에 대한 정보는

ftp://ftp.research.att.com/dist/mab/

에 더 있다.

TCSF는 CFS보다 좀 더 완성도를 높여서 (암호화/복호화 작업을 백그라운드에서 투명하게 실행함으로서) 암호화된 파일시스템을 쓰고 있는 사용자 입장에서는 암호화/복호화 작업이 눈에 보이지 않도록 한 것이다.

http://edu-gw.dia.unisa.it/tcfs/ 에서 더 많은 정보를 구할 수 있다.

15 X11, SVGA와 디스플레이 보안

15.1 X11

디스플레이의 보안은 중요하다. 공격자가 여러분 모르게 입력되는 패스워드를 가로채거나, 여러분의 스크린 상에서 읽고 있는 파일이나 정보를 읽거나, 수퍼유저의 권한을 얻기 위해 보안 개구멍을 이용하기까지하는 일들을 막기 위해서다. 도청자(sniffer)들이 여러분과 원격 시스템 사이의 상호작용을 모두 볼 수 있도록 허락하는 셈이라 할 수 있는, 네트워크 상에서의 원격 X 응용 프로그램 수행도 위험 천만한 일이다.

X는 많은 엑세스 통제 장치를 가지고 있다. 가장 간단한 것은 호스트에 기반한 것이다. 여러분의 디스플레이어 접근할 수 있는 호스트들을 xhost를 사용해서 지정할 수 있다. 안전한 방법은 아니다. 누군가가 여러분의 컴퓨터에 근접할 수 있다면, "xhost +그들의 컴퓨터"라는 명령어를 사용해서 쉽게 들어올 수 있다. 아울러 신임되지 않은 기계의 접속을 허락하면, 그쪽의 누구라도 여러분의 디스플레이를 침탈할 수 있다.

로그인을 위해서 xdm(x display manager)을 쓴다면, 더 나은 엑세스 방법인 MIT-MAGIC-COOKIE-1을 구할 수 있다. 128 비트의 쿠키 (cookie)가 만들어져서 .Xauthority 파일에 저장된다. 원격 컴퓨터에서 여러분의 디스플레이에 접근하는 것을 허용할 필요가 있다면, 그 컴퓨터로부터의 접근만을 제공하기 위해 xauth 명령과 여러분의 .Xauthority 파일에 들어있는 정보를 쓸 수 있다. http://sunsite.unc.edu/LDP/HOWTO/mini/Remote-X-Apps.html에 있는 Remote-X-Apps mini-howto를 보도록 하라.

보안 유지되는 X의 접속을 만들기 위해서 ssh(위의 ssh 참조)를 쓸 수 있다. 앤드 유저의 시점에서는 투명하게 작동되면서도, 암호화되지 않은 자료가 네트워크 상에 떠다니지 않도록 하는 방법이 되는 장점이 있기 때문.

X 보안에 대해 더 많은 정보가 필요하면 Xsecurity의 매뉴얼 페이지를 보기 바란다. 보다 안전한 방법은 xdm을 써서 콘솔에 로그인 하도록 하고, ssh를 써서 X 프로그램을 원격 수

행하려는 원격 사이트들로 가는 것이다.

15.2 SVGA

SVGAlib 프로그램들은 리눅스 컴퓨터에 있는 모든 비디오 하드웨어에 접근할 수 있도록 SUID가 root로 정해져 있다. 이것은 매우 위험한 것이다. 만일 이 프로그램들이 깨지면, 쓸 수 있는 콘솔을 살리기 위해서 다시 부팅 시켜야 한다. 여러분이 실행시키고 있는 SVGA 프로그램들이 진품인지, 그리고 최소 수준이나마 믿을 수 있는 것들인지 확인하라. 더 나은 방법은 SVGA 프로그램들을 아예 수행시키지 않는 것이다.

15.3 GGI (Generic Graphics Interface project)

리눅스 GGI 계획은 여러 가지의 리눅스 비디오 인터페이스 문제들을 해결하고자 노력하고 있다. GGI는 비디오 코드 일부분을 리눅스 커넬안으로 옮겨 실행하는 방식으로 비디오 시스템에 대한 엑세스를 관리할 것이다. 이것은 GGI가 정해놓은 양호 상태로 언제라도 여러분의 콘솔을 복구해 줄 수 있다는 것을 의미한다. 또한 여러분 콘솔에서 트로이 목마식의 로그인 프로그램이 돌지 않도록 하기 위해서, 보안 경계 열쇠(관리)도 허락될 것이다.

더 많은 정보를 얻기 위해서 http://synergy.caltech.edu/~ggi/ 에 가보라.

16 파일과 파일시스템 보안

시스템의 온라인 접속 전, 몇 분 동안의 준비와 계획은 여러분의 시스템과 데이터를 보호하는 것에 큰 도움을 줄 수 있다.

- SUID/SGID를 사용자의 홈 디렉토리에서 쓰게 할 이유가 전혀 없다. 루트가 아닌 다른 사용자들이 적을 수 있는 파티션에는 /etc/fstab에 "nosuid" 옵션을 쓰도록 한다. 어차피 필요 없어야 하는 프로그램의 실행을 금지하며, 블록 디바이스의 형성을 못하도록 /var를 포함해서, 사용자의 홈 파티션에는 "nodev"와 "noexec"을 쓰도록 한다.
- 만약 NFS를 써서 어떤 파일시스템을 네트워크로 송출한다면, /etc/exports를 최대한도 제한하도록 조정하도록 한다. 이것은 와일드카드를 못쓰게 하는 것과, 루트 쓰기 엑세스 (root write access)를 허락하지 않는 것과, 가능하면 읽기 전용만 마운트하는 것을 의미한다.
- 사용자의 파일 생성 umask를 가능한 제한된 값으로 조정한다. 자주 쓰이는 값은 022, 033 그리고 가장 제한적인 077이며 /etc/profile에 적는다.
- 기본 값인 "무제한"이 아닌 다른 값으로 파일시스템의 값을 제한한다. 자원 제한 PAM 모듈과 /etc/pam.d/limits.comf를 사용함으로서 사용자 각각의 제한치를 조정할 수 있다. 예를 들면, "users" 그룹을 위한 제한은 다음과 같을 수 있다.

```
@users      hard  core    0
@users      hard  nproc   50
@users      hard  rss     5000
```

이 경우는 코어 파일의 생성을 금하며, 프로세스의 수를 50으로 제한하며, 사용자 한사람 당 메모리 사용을 5 메가로 제한함을 말한다.

- /var/log/wtmp와 /var/run/utmp 파일들은 시스템 모든 사용자의 접속 기록을 가지고 있다. 이것은 사용자가 (혹은 잠재적 침입자가) 언제, 어디서 시스템에 들어왔는가를 단정하는데 사용될 수 있기 때문에 이 파일들의 보전성은 철저히 유지되어야만 된다. 평상 시스템 작동에 영향을 주는 경우가 없는 한 644 허가권을 가지고 있어야 한다.
- 보호되어야만 하는 파일들을 사고로 지우거나 덧쓰는 경우가 없도록 하기 위해서 이뮤타블 비트(immutable bit: 불변의 비트)를 사용할 수 있다. 또한 이 방법은 파일에 /etc/passwd나 /etc/shadow를 지우는 방법을 쓰는 공격의 원료가 되는, 심볼릭 링크를 만드는 것을 방지한다. 이뮤타블 비트에 대한 추가 정보는 chattr(1)의 man 페이지를 참조하도록 하자.
- SUID와 SGID는 잠재적인 보안 위험 요소이며 철저하게 감시되어야 만한다. 이 프로그램들은 이 들을 쓰는 사용자에게 특별 권한을 주기 때문에, 보안에 불안 요소를 주는 프로그램들이 설치되는 일이 없도록 해야 한다. 크랙커들이 좋아하는 트릭중의 하나는 SUID "루트" 프로그램을 침탈하고 그 후에 원래의 문제점이 고쳐진 후에라도 SUID 프로그램을 통해 뒷문으로 사용해서 들어오는 것이다. 여러분 시스템에 있는 모든 SUDI/SGID를 찾아내서, 그것들이 무엇인지 추적함으로서 잠재적인 침입자를 의미할 수 있는 어떠한 변화라도 알 수 있도록 한다. 다음의 명령어를 사용하면 시스템에 있는 모든 SUID/SGID 프로그램을 찾아낼 수 있다.

```
root#  find / -type f \( -perm -04000 -o -perm -02000 \)
```

chmod(1)을 사용하면 의심쩍은 프로그램의 SUID나 SGID 허가권을 제한적으로 지울 수 있고, 나중에 확실하게 필요함이 느껴지면 다시 바꿔 줄 수 있다.

- 만약 크랙커가 여러분 시스템에 사용권을 얻게되고 특히 시스템 파일이나 월드 라이타블(World-writable)파일들을 변경할 수 있게 되면 심각한 보안 개구멍이 존재하게 된다. 덧붙여서 크랙커들이 마음대로 파일을 덧붙이거나 지울 수가 있게 되므로 월드 라이타블 디렉토리도 위험한 것이다. 월드 라이타블 파일 모두를 찾기 위해서는 다음의 명령어를 사용한다.

```
root# find / -perm -2 -print
```

그리고 이 파일들이 왜 '쓰기 가능'으로 설정되어 있는지 반드시 알도록 한다. 정상적인 운영에 있어서, /dev의 일부와 심볼릭 링크를 포함한 여러 파일들은 라이타블로 되어 있을 것이다.

- 무소속의 파일들 또한 침입자가 시스템에 들어왔다는 징후일 수 있다. 주인이 없거나 그룹에 소속되어 있지 않은 파일들은 다음의 명령어를 쓰면 찾아낼 수 있다.

```
root# find / -nouser -o -nogroup -print
```

- 리모트 호스트 (.rhosts) 파일들은 절대로 있으면 안되는 것이기 때문에, 이것들을 찾는 것은 시스템 관리자 임무의 일부가 되어야만 한다. 주지할 것은 크랙커가 여러분 네트워크에 침투하기 위해서는 단 한 개의 불안전한 계정이 필요할 뿐이라는 것이다. 시스템의 모든 리모트 호스트 파일들은 다음의 명령어로 찾을 수 있다.

```
root# find /home -name .rhosts -print
```

- 마지막으로 시스템 파일의 허가권을 바꾸기 전에, 무엇을 하고 있는 가를 정확히 이해하도록 한다. 단순한 작동의 이유만으로 파일의 허가권을 바꾸는 일이 없도록 해야 한다. 허가권을 바꾸기 전에 파일이 왜 이러한 허가권을 가지고 있는지 알도록 해야 한다.

16.1 umask 조정

umask 명령어는 시스템 파일이 만들어질 때의 허가권 기본 값을 정하기 위해서 사용된다. umask에는 정하려는 파일 모드의 십진 전수 (Octal Complement)를 사용한다. 만약 허가권 기본 값을 정하지 않은 상태에서 파일이 형성된 게 된다면, 사용자가 모르는 사이에 허가권을 가지면 안되는 누군가에게 읽기 쓰기 허가권을 주게 될 수가 있다. 일반적으로 umask 값은 022, 027, 그리고 제일 제한적인 077 등이 있다.

umask는 일반적으로 /etc/profile에서 조정되고, 시스템의 모든 사용자에게 적용된다. 예로서 다음과 같은 값을 가질 수 있다.

```
# Set the user's default umask
umask 033
```

루트의 umask 값은 077로 해서 다른 사용자가 chmod(1)를 써서 뚜렷이 명시하며 바꿔주지 않는 한 읽고 쓰고 실행할 수 없도록 만든다 레드햇을 쓴다면 레드햇의 사용자와 그룹 ID 구성 방법 (User Private Groups)을 따른다는 가정 하에 umask는 002라도 좋다. 기본 구성이 한 그룹 당 한 사용자로 되어있기 때문이다.

16.2 파일 허가권 (File Permissions)

시스템 관리를 할 권리가 없는 사용자나 그룹이 시스템 파일을 임의로 편집하는 일이 없도록 하는 것은 중요하다. 유닉스는 파일과 파일에 대한 엑세스 관리를 owner, group, 그리고 other라는 세 가지 특성으로 구분한다. 언제나 정확히 하나의 소유자 (owner)가 존재하며, 그룹의 멤버 수는 일정하지 않으며, 나머지 사용자들은 other가 된다.

소유권 (Ownership) - 어떤 사용자나 그룹이 노드와 상위 노드의 허가권에 대한 조정을

할 수 있는 권한을 말한다.

허가권 (permission) - 특정 종류의 엑세스가 가능하도록 정해주거나 변경될 수 있는 비트다. 디렉토리에 대한 허가권은 파일에 대한 허가권과는 다른 의미를 가질 수가 있다.

읽기 허가권 (read) -

· 파일의 내용을 볼 수 있는 것이 가능하다.
· 디렉토리를 읽는 것이 가능하다.

쓰기 허가권 (write) -

· 파일에 만들거나 변경을 하는 것이 가능하다.
· 디렉토리에 있는 파일을 지우거나 움직이는 것이 가능하다.

실행 허가권(Execute): -

· 이진 프로그램 (binary)이나 쉘 스크립트를 실행할 수 있다.
· 읽기 허가권이 있을 때, 디렉토리를 탐색하는 것이 가능하다.

SAVE 텍스트 어트리뷰트: (디렉토리)

스틱키 비트 (sticky bit)도 디렉토리에 관해서 적용될 때는 다른 뜻을 가지게 된다. 디렉토리에 스틱키 비트가 붙을 때에는 사용자는 설령 사용자가 디렉토리에 일반적인 쓰기 허가권이 있더라도 소유권이 있거나 확실하게 쓰기 허가권이 허락된 파일만 지울 수 있게 된다. 이것은 /tmp 따위의 월드 라이타블이면서도 일반 사용자가 무조건 파일을 지우면 좋지 않을 디렉토리를 위해 쓰여진다. 스틱키 비트는 긴 디렉토리 리스팅 (ls -l)에서 "t"로 표시된다.

SUID 어트리뷰트 (파일용)

이것은 파일의 set-user-id 허가권을 정의할 때 사용된다. 소유자 허가권에 set-user-id 엑세스 모드가 붙으면 --그리고 파일이 실행 가능한 파일이라면-- 이 파일을 실행하는 프로세스는 프로세스를 만든 사용자가 사용할 수 있는 시스템 리소스를 쓸 수 있는 권한이 부여된다. 이것은 "버퍼 오버플로우 (buffer overflow: 이하 버퍼 범람)"을 사용하는 많은 침탈법의 재료로 쓰여진다.

SGID 어트리뷰트 (파일용)

그룹 허가권에 붙은 경우에는 이 비트가 "set-group-id"를 관리하게 된다. 이것은 그룹이 영향을 받는다는 점을 제외한다면 SUID와 같은 역할을 하는 것이다. 영향을 받으려면 역시 파일은 실행 가능하도록 정의되어야 한다.

SGID 어트리뷰트 (디렉토리용)

만약 SGID를 디렉토리에 사용하면 ("chmod g+s 디렉토리"를 씀), 그 디렉토리 안의 파일들은 디렉토리 소유 그룹의 값을 기본 그룹 값으로 가지게 된다.

여러분 - 파일의 소유자 (owner)

그룹 - 여러분이 가입되어 있는 그룹 (group)

나머지 모든 이 - 파일의 소유자나 파일을 소유한 그룹에 속하지 않은 나머지 사용자 (other)

파일의 보기

```
-rw-r--r--  1 kevin  users         114 Aug 28  1997 .zlogin
1번 비트  (-) 디렉토리인가?     (아니다)
2번 비트  (r) 소유자에 읽기권? (있다.  케빈이 읽을 수 있다)
3번 비트  (w) 소유자가 쓰기권? (없다.  케빈이 읽을 수 있다)
```

4번 비트 (-) 소유자에 실행권? (없다)
5번 비트 (r) 그룹에 읽기권? (있다, users라는 그룹)
6번 비트 (-) 그룹에 쓰기권? (없다)
7번 비트 (-) 그룹에 실행권? (없다)
8번 비트 (r) 모든 이에 읽기권? (있다, 모든 이가 읽을 수 있다)
9번 비트 (-) 모든 이 쓰기권? (없다)
10번 비트 (-) 모든 이에 실행권? (없다)

필요한 만큼만의 최소한의 허가권을 부여한 보기를 적어놓았다. 더 큰 허가권을 주는 것은 가능하지만, 설명된 작업용으로의 최소한도를 적은 것임을 밝혀둔다.

-r-------- 소유자의 읽기 허가권이 파일에 있다.
--w------- 소유자가 파일을 변경하거나 지울 수 있다.
---x------ 소유자가 파일 (프로그램)을 실행할 수 있지만, 읽기권도 있어야 실행되는 쉘 스크립트는 실행하지 못한다.
---s------ 실세의 사용자 ID가 소유자라면 실행할 수 있다. (setuid 참조)
-------s-- 실세의 사용자 ID가 그룹이라면 실행할 수 있다. (setgid 참조)
-rw------T "최근 바뀐 시간 (last modified time)" 정보가 갱신되지 않는다. 스왑 파일 등에 사용된다.
---t------ 상관없음 (전에는 스틱키 비트였음)

디렉토리의 보기

drwxr-xr-x 3 kevin users 512 Sep 19 13:47 .public_html/
1번 비트 (d) 디렉토리인가? (그렇다, 많은 파일을 가지고 있다)
2번 비트 (r) 소유자의 읽기권? (있다, 케빈)
3번 비트 (w) 소유자의 쓰기권? (있다, 케빈)
4번 비트 (x) 소유자의 실행권? (있다, 케빈)
5번 비트 (r) 그룹의 읽기권? (있다, users 그룹)
6번 비트 (-) 그룹의 쓰기권? (없다)
7번 비트 (x) 그룹의 실행권? (있다, users 그룹)
8번 비트 (r) 다른 이의 읽기권? (있다, 아무나 읽을 수 있다)
9번 비트 (-) 다른 이의 쓰기권? (없다)
10번 비트 (x) 다른 이의 실행권? (있다, 아무나 실행할 수 있다)

최소의 허가권을 준 사용 보기이다. 여기에 설명되어 있는 것 보다 허가권을 더 주는 것은 가능하지만, 아래에 설명하는 정도는 최소한도로 필요하다.

dr-------- 내용은 보여질 수 있지만, 파일 어트리뷰트는 읽을 수 없게된다.
d--x------ 디렉토리는 실행 패스 (path)에 넣어져서 사용될 수 있다.
dr-x------ 파일 어트리뷰트는 이제 소유자에 의해서 읽혀질 수 있다.

d-wx------ 디렉토리 현 위치에 있지 않아도 파일은 만들어지고 지워질 수 있다.
d------x-t 쓰기 엑세스를 가진 다른 사용자들이 파일을 함부로 지우는 것을 막는다. /tmp 디렉토리에 사용된다.
d---s--s-- 아무런 작용을 하지 않는다. (SUID와 SGID 참조)

(보통 /etc 안에 있는) 시스템 설정 파일 (system configuration files)들은 640 모드이면서 동시에 루트 소유로 되어 있다. 여러분 사이트의 보안 필요에 따라서 바꾸면 된다. 시스템 파일은 절대로 다른 어떤 그룹이나 누구라도 쓸 수 있도록 하면 안된다. /etc/shadow를 포함한 시스템 파일의 일부는 루트만이 읽기 허가권을 가져야 하고, /etc안의 디렉토리들은 다른 이들이 읽지 못하도록 해야 한다.

SUID 쉘 스크립트.
SUID 쉘 스크립트는 심각한 보안 위험요소이며, 그런 이유 때문에 커넬이 받아들이지 않도록 되어있다. 여러분이 얼마나 쉘 스크립트가 안전하다고 생각을 하던 간에, 이것은 크랙커에게 루트 쉘을 주는 침탈 도구가 될 수 있다.

16.3 트립와이어를 (tripwire: 지뢰선) 사용한 완결성의 검사

트립와이어 등의 완결성 (Integrity) 유지용 검사 도구를 사용하는 것은 지역 사용자에 의한 (그리고 네트워크를 통한) 공격을 탐지해내는 매우 좋은 방법이다. 트립와이어는 중요한 이진 파일들과 설정 파일들의 체섬(checksum) 값을 검출해서 이전에 만들어 놓은 데이터베이스와 비교한다. 파일에 변화가 있으면 표시가 날 것이다. 트립와이어를 쓰면, 플로피로트립와이어를 설치하고, 쓰기 방지 탭을 사용해서 쓰는 것이 좋다. 이렇게 해 놓으면 침입자는 트립와이어에 손을 대거나 데이터베이스를 바꾸지 못하게 된다. 일단 트립와이어를 한 번 설치했으면, 일상적인 보안 관리 임무의 일부분으로 실행하는 것이 좋다. 트립와이어를 매일 밤 플로피에서 돌리고 아침에 메일로 결과를 받도록 다음과 같이 크론탭을 설정할 수도 있다.

```
# set mailto
MAILTO=kevin
# run tripwire
15 05 * * * root /usr/local/adm/tcheck/tripwire
```

이와 같이 하면 매일 아침 5:15am에 리포트를 보내 줄 것이다. 트립와이어는 침입자를 여러분이 눈치채기 훨씬 전에 알려주는 귀중한 존재가 될 수 있다. 하지만 일반적 시스템 안에서는 많은 파일이 항시 바뀌므로 변한 것이 여러분의 일 때문인가, 아니면 크랙커의 행동인가를 파악하는 것에 신중을 기하도록 한다.

16.4 트로이의 목마

트로이의 목마는 호머의 문학 작품에 나오는 전설적인 책략에서 비롯된 이름이다. 그럴듯해

보이는 어떤 프로그램이나 이진 파일을 업로드해 놓고, 다른 사람들이 그것을 다운 받아서 루트로서 돌리도록 한다. 그런 후에 사용자가 신경을 주지 않는 틈을 타서 시스템을 깨고 들어오는 것이다. 방금 받아온 이진 파일이 관리자가 애초에 기대했던 어떤 일을 한다고 생각하는 사이에 --정말로 그런 척 하기도 한다-- 한편으로는 보안을 깨고 들어오는 것이다.

여러분의 컴퓨터에 무슨 프로그램을 설치하였는지는 잘 살피도록 해야 한다. 레드햇은 RPM 파일에 대해서 MD5 체섬과 PGP 시그니춰를 제공하므로, 설치하고 있는 프로그램이 진짜인지 확인할 수 있다. 다른 배포본도 비슷한 방법을 쓴다. 소스도 있거나 잘 알려진 것이 아닌 한, 어떤 이진 파일도 루트로서 실행되어서는 안 된다! 일반 대중이 정밀한 조사를 할 수 있도록 소스를 공개할 크랙커는 없다시피 하므로, 귀찮을 수도 있지만, 프로그램의 소스를 정품 배포 장소에서 가져왔는지 확인하도록 하라. 프로그램이 루트에서 실행될 상황이라면 여러분이나 믿을 만한 누군가가 소스를 훑어보고 확인하도록 해야 한다.

17 커넬 보안

이것은 보안에 관련된 커넬 조정 옵션과, 이 것이 무엇을 하는 지에 대한 설명과, 어떻게 사용하는 지에 대한 설명이다.

커넬이 여러분 컴퓨터 네트워크 업무를 관리하므로, 커넬을 매우 안전하도록 관리하는 것과 커넬 자체의 보안이 깨지지 않도록 하는 것은 중요한 것이다. 최신 네트워킹 공격법의 일부를 방지하기 위해서는 커넬의 버전을 최신의 것으로 관리해야 한다. 새로운 커넬은

ftp://ftp.kernel.org

에서 찾을 수 있다.

17.1 커넬 컴파일 옵션

- IP: 소스에서 라우트되는 프레임을 떨구어 버리는 방법:
  CONFIG_IP_NOSR.

  이 옵션은 켜져야 한다. 소스 라우티드 프레임들은 (Source routed frames) 그 패킷 안에 완벽한 목적지 패스 (Path) 정보를 가지고 있다. 이것은 패킷을 처리하는 라우터가 패킷의 내용을 검사할 필요 없이 그대로 통과 처리한다는 것을 의미한다. 결국은 잠재적인 침탈 수단이 될 수 있는 데이터가 시스템에 들어오는 것으로 발전될 수 있다.

- IP: 방화벽 처리.
  CONFIG_IP_FIREWALL.
  만약 기계를 방화벽으로 사용하거나, 마스커레이드용으로 쓰거나, 아니면 누군가가 다이얼-업 웍크스테이션에 PPP 다이얼-업 인터페이스를 통해서 들어오는 것을 막기 위해서

필요한 옵션이다.

- IP: 포워딩/게이트웨이화: CONFIG_IP_FORWARD.
  IP 포워딩을 켜놓으면 리눅스 상자는 결과적으로 라우터가 되는 것이다. 만약 이 기계가 네트워크 상에 있으면, 이 기계는 데이터를 어떤 하나의 네트워크에서 또 다른 하나의 네트워크로 중매인으로서 송달해 주는 작업을 할 것이고, 이 경우에는 수신자가 이러한 경우를 막으려고 설치해 놓은 방화벽을 뚫어버리는 결과를 부를 수 있다. 데이터를 한 네트워크에서 다른 곳으로 전달할 수가 있고, 그리고 어쩌면 이것을 막기 위해서 그곳에 세워놓은 방화벽을 뚫어버릴 수 도 있다. 보통의 다이얼-업 사용자는 이 옵션을 끄고 사용하는 것이 좋고, 다른 사용자들은 이 옵션을 씀으로서 발생되는 결과를 주의 깊게 살펴보면서 사용해야 할 것이다. 방화벽 기계인 경우에는 방화벽 소프트웨어의 기능과 잘 섞어서 이 옵션을 사용하는 것이 좋다. 다음의 명령어를 사용하면 IP 포워딩을 다이내믹하게 켜고 끌 수 있고,

```
root#  echo 1 > /proc/sys/net/ipv4/ip_forward
```

  다음의 명령으로 꺼버린다.

```
root#  echo 0 > /proc/sys/net/ipv4/ip_forward
```

  /proc에 있는 많은 다른 파일들을 포함해서 이 파일은 길이가 0으로 보여질 것이지만, 실제로는 안 그렇다. 새롭게 소개된 커넬 기능이므로, 커넬 버전 2.0.33 이상의 것을 사용하도록 하라.

- IP: 방화벽 패킷일지 쓰기 (CONFIG_IP_FIREWALL_VERBOSE)
  이 옵션은 발신인, 수신인, 포트번호 등 방화벽이 받은 패킷의 정보를 보여준다.

- IP: 항상 디프레그먼트 실행 (CONFIG_IP_ALWAYS_DEFRAG)
  보통 이 옵션은 꺼져있지만, 방화벽이나 마스커레이딩 호스트를 만든다면, 켜는 것이 좋을 것이다. 데이터가, 한 호스트에서 다른 호스트로 보내질 때, 항시 한 뭉치의 패킷으로 보내지는 것이 아니라 여러 조각으로 나뉘어져서 보내진다. 여기에서 문제점은 포트 번호가 제 1 번 조각에 만 적혀져 있다는 것이다. 이것은 누군가가 나머지 조각들 중에 있으면 안되는 정보를 집어넣을 수 있다는 것이다.

- IP: syn 쿠키 (CONFIG_SYN_COOKIES)
  SYN 공격은, 결과적으로 리부트를 유도하게끔, 모든 사용 가능한 자원을 소비하게 한다는 식의 서비스 거부 공격법 (DoS)의 하나이다. 켜놓을 이유가 없는 옵션이다.

- IP: 방화벽 패킷 넷링크 디바이스 (CONFIG_IP_FIREWALL_NETLINK)
  사용자공간 프로그램 패킷의 첫 128 바이트를 분석해서 이것의 유효 합법성에 따라서 패킷을 받거나 혹은 거부해야 하는 지를 결정하게 해주는 정말로 깔끔한 옵션이다.

17.2 커넬 디바이스들

리눅스에는 보안에 도움이 되는 몇 개의 블록 디바이스와 문자 디바이스가 있다. /dev/random과 /dev/urandom의 두 디바이스는 랜덤 데이터를 언제라도 수용할 수 있도록 커넬에서 제공된다.

/dev/random과 /dev/urandom 둘은 안전한(secure) 난수 발생 기능이 필수적인 PGP 열쇠의 제작, SSH 수하 도전용 (challenge), 그리고 기타 프로그램들 등에 사용될 수 있을 만큼 충분히 안전해야 한다. 공격자가 이 두 기능에서 발생된 숫자들의 조합을 미리 알고 있다고 해서 그 다음의 나올 숫자를 알아내는 일이 가능해서는 안된다. 이 둘로부터 생성되는 숫자들이 진정한 의미로서의 난수가 되도록 보장하는 많은 노력들이 쏟아 부어지고 있다.

차이점은 /dev/random은 난수 바이트들로 만들어지며, 난수 바이트들이 만들어 쌓이는 동안은 대기 상태가 된다는 것이다. 일부 시스템에서는 새로운 사용자 생성 입력이 시스템에 등록되는 시간이 오래 걸릴 수 있고, 그 동안은 사용이 막혀 있을 수 있다는 것을 말하고 싶다. /dev/random을 쓰기 전에는 심사숙고하기를 바란다. 아마도 제일 좋은 방법은 형성이 되고 있는 사이에 "OK 충분합니다" 하는 메시지가 나올 때까지 사용자들이 키보드를 두들기게 끔 하는 것일 것이다

/dev/random은 인터럽트 사이의 시간을 재서 만드는 등의 질이 높은 엔트로피이다. 이것도 랜덤 데이터가 충분할 때까지 막고 있게 된다.

/dev/urandom은 비슷하지만, 엔트로피가 낮을 때의 경우 암호학 기법상 강하다고 할 수 있는 헤쉬 값을 만들어 준다. 비록 이것은 /dev/random으로 만들어지는 값에 비하면 상대적으로는 덜 안전하지만 대부분의 프로그램용으로는 충분하다. 다음과 같은 방법 등으로 디바이스로부터 읽어낼 수 있다.

```
root#  head -c 6 /dev/urandom | uuencode -
```

이것은 패스워드를 만들기에 적절할 6자의 난수를 콘솔로 출력할 것이다. 연산법에 대한 설명은 /usr/src/linux/drivers/char/random.c에 있다.

18 네트워크 보안

사람들이 더 많은 시간을 컴퓨터 접속에 보내면서, 네트워크 보안은 더욱 더 중요해 지고 있다. 네트워크 보안을 도와줄 도구들은 많으며, 갈수록 많은 것들이 리눅스 배포본에 실려 배포되고 있다.

18.1 패킷 스니퍼

침입자가 네트워크의 더 많은 시스템으로 침투하기 위해서 가장 흔하게 쓰는 방법 중의 하나가 이미 깨어진 호스트에서 패킷 스니퍼를 실행하는 것이다. 이 "스니퍼"는 이터넷 포트를 감청하면서 지나가는 패킷 흐름에서 "Password", "Login", "su" 같은 것이 들리면 그 이후의 내용을 녹음해 둔다. 이 방법을 쓰면, 공격자는 침투하려고 시도조차 않았던 시스템으로까지 들어가는 패스워드를 얻게 된다. (암호화가 안된 채로) 평문으로 전송되는 패스워드는 이 공격에 매우 약한 것이다.

예: 호스트 A의 보안이 깨졌다. 공격자는 (여기에) 스니퍼를 설치한다. (잠시 후,) 어떤 관리자가 호스트 C에서 호스트 B로 들어가려는 접속 로그인을 스니퍼가 감지한다. (이제 패킷이 녹음이 되고 있다) 관리자가 B로 로그인을 하는 순간 이 관리자의 개인 패스워드는 녹음이 된다. 잠시 후 관리자가 어떤 문제를 해결하기 위해 'su'를 사용한다. 이제 호스트 B의 루트 패스워드까지 얻게 되었다. 잠시 후에, 관리자가 누군가가 자기 계정에서 다른 사이트에 있는 호스트 Z로 텔넷을 하도록 해 두면 공격자는 이제 호스트 Z로 로그인할 패스워드까지 갖게 된다. 오늘날에는 공격자가 패킷 스니퍼를 쓰기 위해 시스템의 보안을 깨고 침입할 필요조차 없어져 버렸다. 공격자는 랩탑이나 PC를 건물 안으로 들고 들어와서 네트워크를 감청하면 그만인 것이다. ssh나 다른 암호화된 패스워드 방식을 사용하면 이 공격을 방해할 수 있다. pop 계정용의 ATOP 등이 이 공격을 방지한다(유선을 통해 평문 패스워드를 전송하는 방법들이 다 그렇듯이, 보통의 pop 로그인은 스니퍼에 대단히 취약하다).

18.2 시스템 서비스와 tcp_wrapper

어떤 서비스를 제공할 필요가 있는가를 선별하는 것은 네트워크에 리눅스 시스템을 올려놓는 순간부터 해야 할 일이다. 제공할 필요가 없는 서비스를 아예 해체해버리면 걱정거리가 하나 줄고, 공격자가 개구멍을 찾을 대상을 하나 줄여 버리는 것이 되는 것이므로 그렇다. 리눅스 시스템에서 서비스를 꺼버리는 방법은 많이 있다. /etc/inetd.conf 파일을 보면 inetd가 현재 어떤 서비스를 제공하고 있는지 알아볼 수 있다. 필요 없는 서비스는 모두 주석문(remark) 처리를 해서 막아버리고(#을 줄 처음에 써주면 된다), inetd 프로세스에 SIGHUP 신호를 보내도록 하라(killall -HUP inetd). 아울러 /etc/services 파일에서도 서비스를 주석문 처리를 하거나 삭제할 수 있다. 이것은 지역 사용자들도 또한 서비스를 못쓰게 된다는 뜻이다.

예: 만약 여러분이 ftp를 삭제 후, 이 기계에서부터 원격 사이트로 ftp를 사용하려 하면 "서비스 모름: unknown service" 메시지가 나오면서 안 받아줄 것이다) 보안성이 늘어나는 것은 아니므로 꼭 서비스를 없애 버릴 가치는 없다. 만약 지역사용자가 여러분이 주석문 처리를 해서 꺼버린 ftp를 쓰고 싶어한다면, 그는 간단히 자신의 클라이언트를 사용하면서 공용 ftp 포트를 써서 여전히 일을 할 수 있을 것이다. 켜놓는 것이 좋을 서비스들은 ftp, telnet, mail(pop-3 나 imap), identd, time등이 있다.

어떤 패키지를 쓸 일이 없으리라는 것을 알고 있다면, 그 패키지를 완전히 삭제할 수도 있다. 레드햇 배포본에서는 rpm -e 명령으로 한 패키지 전체를 지울 수 있다. 데비안에서는 dpkg로 같은 작업을 할 수 있을 것이다.

덧붙여서, (rlogin이 쓰는) login과 (rcp가 쓰는) shell 그리고 (rsh가 쓰는) exec를 /etc/inetd.conf에서 시작되는 것을 막는 것을 포함해서, /rsh/rlogin/rcp 도구를 꺼버리는 것이 정말로 필요하다. 이들 프로토콜은 극단적으로 허술하며, 예전부터 침탈의 한 근원이 되어왔다. /etc/rc.d/rcN.d를 봐서(여기서 N은 여러분 시스템의 런 레벨이다. 런 레벨이란 시스템의 운영 단계라고 생각하면 될 것이다) 디렉토리에서 실행되는 서버들 가운데 불필요한 것들이 있는가 확인하라. /etc/rc.d/rcN.d안에 있는 파일들은 실제는 /etc/rc.d/init.d 디렉토리로 심볼릭 링크 되어 있다. init.d에 있는 파일들의 이름을 바꿔버리면, /etc/rc.d/rcN.d 안의 모든 심볼릭 링크를 꺼버리는 효과를 가져온다. 만약 특정 런 레벨에 맞추어서 서비스를 꺼주고 싶으면, 이에 상응하는 파일을 소문자 (lower-case)로 이름을 바꿔주면 된다. BSD 형식의 rc 파일들을 갖고 있다면 /etc/rc*을 검사해서 필요 없는 프로그램들을 볼 수 있다.

대부분의 리눅스 배포본에는 모든 tcp 서비스들을 "보호해주는(wrapping)" tcp wrapper가 들어있다. tcp_wrapper(tcpd)는 실제 서버를 실행 할 수 있는 것이 아니고, 대신 inetd가 불러오는 방법으로 실행된다. 그러면 tcpd는 서비스를 요청하는 호스트를 검사해서, 서버를 실행시키거나 그 호스트로부터의 접근을 거부한다. tcpd를 이용해서 tcp 서비스로의 접근을 제한할 수 있는 것이다. /etc/hosts.allow 파일을 만들고, 여러분 컴퓨터의 서비스에 접근할 필요가 있는 호스트들만을 추가하도록 한다.

여러분이 집에서 모뎀을 쓰는 다이얼-업 사용자라면, 필자는 모든 서비스에 대한 접근을 거부하도록 권한다. tcpd는 서비스에 접근하려다가 실패한 시도들을 기록하므로, 공격을 받고 있다는 것을 알려줄 수도 있다. 새로운 서비스를 추가로 설치하게 되면, 반드시 tcp wrapper의 TCP 기반으로 그 서비스를 추가토록 하는 것이 좋다. 예를 들면, 가정의 모뎀 사용자(dial-up user)는 외부인이 자신의 기계에 연결하는 것을 막으면서도, 메일을 받도록 인터넷에 네트워크 연결을 할 수 있다. 이렇게 만들려면 /etc/hosts.allow에 다음을 추가한다.

```
ALL: 127.
```

물론 /etc/hosts.deny에도

```
ALL: ALL
```

이렇게 해 놓으면 외부에서 들어오는 연결은 막으면서도, 내부에서 인터넷으로 나가는 연결은 할 수 있게 된다.

18.3 DNS 정보의 확인

여러분 네트워크의 모든 호스트에 대한 DNS 정보를 최신판으로 유지하는 것으로도 보안이

강화할 수 있다. 만약 불법 호스트가 여러분 네트워크에 연결되는 상황이 벌어지면, DNS 엔트리가 없을 것이므로, 침입을 알아챌 수가 있게 된다. 많은 서비스들은 유효한 DNS 엔트리가 없는 호스트는 접속을 거부하는 식으로 조정할 수 있게 되어있다.

18.4 identd

identd는 주로 inetd에서 수행되는 작은 프로그램이다. 어느 사용자가 어떤 tcp 서비스를 수행시키는지 추적하고, 요구하는 누구에게든 추적 결과를 보고한다. 많은 사람들이 identd의 유용성을 오해하고, 이것을 꺼버리거나 외부 사이트로부터 오는 요청을 거부하도록 막아둔다. identd는 원격 사이트에 도움을 주기 위해서 있는 것이 아니다. 여러분이 원격 identd로 얻은 자료가 옳은지 알 방법은 없다. identd 요청에는 아무런 인증 절차가 없기 때문이다.

그렇다면 왜 identd를 수행시켜야 할까? identd가 여러분을 도와주기 때문이고, 추적 시에는 검문소의 역할을 하기 때문이다. 여러분의 identd가 변조되지 않았다면 tcp 서비스를 쓰고 있는 사람들의 사용자 이름이나 uid를 identd가 원격 사이트에 말해주고 있는 것을 알 것이다. 만에 하나 원격 사이트의 관리자가 여러분에게 와서 여러분 컴퓨터의 어느 사용자가 자기의 사이트로 침입하려고 했다고 말한다면, 여러분은 손쉽게 그 사용자에 대해서 행동을 취할 수 있다. identd를 실행시키고 있지 않았다면, 누가 그 때 있었는지 알아내기 위해서 수많은 기록들을 살펴보아야 하고, 이런 경우 일반적으로 그 사용자를 추적하기 위해서 훨씬 긴 시간이 걸리게 된다.

대부분의 배포판에 들어있는 identd는 많은 사람들이 생각하는 것보다 더 다양한 설정이 가능하다. 특정한 사용자용으로 identd가 작동하지 않도록 할 수 있고 (이 사용자들은 .noident 파일을 만들면 된다), 모든 identd 요청을 기록하도록 할 수 있으며 사용자 이름 대신 uid나 NO-USER를 표기하도록 할 수도 있다.

18.5 SATAN, ISS, 그리고 다른 네트워크 스캐너 프로그램들

포트와 서비스를 대상으로 컴퓨터들과 네트워크에 대한 검사 (scan)를 수행하는 많은 소프트웨어 패키지들이 있다. SATAN과 ISS는 그 가운데 비교적 잘 알려진 프로그램이다. 이 소프트웨어들은 표적 컴퓨터의 (혹은 한 네트워크 상의 모든 표적 컴퓨터들의) 가능한 모든 포트에 연결하려고 시도하며, 어떤 서비스가 그 곳에서 수행되고 있는지 찾아내고자 한다. 이 정보를 바탕으로 표적 컴퓨터가 어떤 침탈법에 취약한지 찾을 수 있다. SATAN(Security Administrators Tool for Analyzing Networks)는 웹 인터페이스를 가진 포트 검사 프로그램이다. 컴퓨터 한 대나 하나의 네트워크에 대한 검사 강도는 강, 중, 약으로 설정할 수 있다. SATAN을 구해서 여러분의 컴퓨터나 네트워크를 조사해서 발견되는 문제를 고치는 것이 좋다. SATAN을 선사이트나 유명한 FTP, 웹 사이트에서 구할 때 주의해야 한다. 인터넷에 SATAN을 가장한 트로이 목마가 있었기 때문이다.

http://www.trouble.org/~zen/satan/satan.html

ISS (Internet Security Scanner)는 또 다른 포트형 검사 프로그램이다. SATAN 보다 빠르며, 따라서 대규모의 네트워크를 검사하기에 더 적합할 수 있다. 하지만 SATAN이 더 많은 정보를 제공하는 경향이 있다.

아바커스-센츄리 (Abacus-Sentry)는 www.psionic.com에서 구할 수 있는 상용 포트 스캐너다. 홈 페이지에서 더 정보를 구할 수 있다. SATAN이나 ISS 등의 소프트웨어가 여러분의 컴퓨터를 탐지하고 있다는 것을 경보해주도록 만들어진 도구들이 몇 가지 있다. 하지만 tcp_wrapper를 잘 활용하고 기록 파일들을 정기적으로 살펴보기만 해도, 그런 탐색을 알아차릴 수 있다. 가장 낮은 수준으로 설정해 두더라도 SATAN은 보통의 레드햇 시스템 로그 파일에 발자국을 남긴다.

18.6 샌드메일, qmail 과 MTA

여러분이 제공할 수 있는 가장 중요한 서비스들 가운데 하나가 메일 서버이다. 불행하게도 메일 서버는 공격에 가장 취약한 서비스 중의 하나인데, 그 까닭은 수행해야 하는 작업의 숫자와 필요로 하는 권한이 많기 때문이다.

sendmail을 쓰고 있다면, 최신 버전을 사용하는 것이 매우 중요하다. sendmail은 길고도 긴 침탈의 역사가 있다. 가장 최근의 버전을 항상 사용하도록 유의하라. http://www.sendmail.org를 자주 방문하라

매주 sendmail 버전을 업그레이드하기에 지쳤다면, qmail로 바꿔보는 것도 고려해볼 만 하다. qmail은 처음부터 보안을 염두에 두고 설계되었다. 이 프로그램은 빠르고 안정적이고 안전하다. http://www.qmail.org로 가보라

18.7 서비스 거부식 공격 (Denial of Service attacks: 이하 DoS)

DoS 공격은 시스템 자원의 일부를 매우 바쁘게 만듦으로서, 합법적인 요청에 답하지 못하게 만들거나, 정식 사용자의 시스템 접근을 거부하게 만드는 것이다. 이런 공격은 근년에 들어 크게 증가해왔다. 최근의 공격방법 중 잘 알려진 것들을 아래에 적었다. 새로운 공격방법들이 항상 나타나고 있으므로 여기 소개된 것들은 그저 몇 가지 사례에 불과하다는 것을 명심해야 한다. 더 새로운 정보를 얻으려면 리눅스 보안 리스트와 벅트랙 (bugtraq) 리스트와 아카이브를 읽도록 하라.

- SYN 범람(flooding) - SYN 범람은 네트워크를 통한 서비스 거부 공격이다. 이 방법은 TCP 연결 방식 중에 있는 "허점"을 이용한다. (2.0.30 이후의) 새로운 리눅스 커넬들은 SYN 범람 공격 방지하기 위한 조정 옵션들을 가지고 있다. 커넬 보호 옵션을 보려면 커넬 보안 항목을 보도록 하라.

- 펜티움 "FOOF" 버그 - 인텔의 정품 펜티움 프로세서에 일련의 어셈블리 코드를 보낼 경우 컴퓨터가 재시동을 한다는 것이 최근에 발견되었다. 이것은 어떤 운영체제인가에

관계없이 (모조품과 펜티움 프로, 펜티움2를 제외한) 펜티움 프로세서를 사용하는 모든 컴퓨터에 영향을 미친다. 2.0.32 이상의 리눅스 커넬에는 이 버그로 인해 컴퓨터가 서는 것을 막는 우회법이 포함되어 있다. 2.0.33 커넬은 좀 더 개선된 커넬 수정안을 가지고 있고, 2.0.32보다 좋은 것으로 인식되고 있다. 펜티움을 사용하고 있다면, 지금 업그레이드를 해야 한다.

- Ping 범람 - Ping 범람은 간단한 부르트 포스 DoS 공격의 일종이다. 공격자는 ICMP 패킷 하나를 "홍수처럼" 여러분의 컴퓨터에 보낸다. 공격자가 이 짓을 여러분의 컴퓨터 보다 연결 속도가 빠른 컴퓨터에서 한다면, 여러분의 컴퓨터는 네트워크로 아무 것도 전송할 수 없게 될 것이다. 이 공격법의 변종 중 하나인 "스머핑"은 범인을 찾아내는 것이 더 어렵도록 ICMP 패킷들의 발신인을 여러분 컴퓨터의 주소로 위장해서 다른 호스트에 보낸다. "스머프" 공격에 대해서는

  http://www.quadrunner.com/~chuegen/smurf.txt

  에서 더 정보를 얻을 수 있다. Ping 범람 공격을 받고 있다면, 어디에서 패킷이 오는지 혹은 오는 것처럼 보이는지 알아내기 위해서 tcpdump 같은 도구를 쓰도록 하고, 여러분의 ISP에게 이 사실을 연락하도록 하라. Ping 범람은 라우터 수준에서 차단하거나 방화벽을 쓰는 것이 가장 쉽다.

- 죽음의 핑(Ping o' death)
  죽음의 핑 공격은 커넬 데이터 구조가 수용할 수 있는 것보다 크게 만들어진 IMCP 에코를 요청하는 (IMCP ECHO REQUEST) 패킷이 원인이다. (65510 바이트의) 커다란 "핑"을 시스템에 보내면 시스템이 서버리거나 죽어버리기 때문에, "죽음의 핑"이라고 불리게 되었다. 이 문제는 오래 전에 이미 해결책이 나와있으니 크게 걱정할 필요는 없다.

- 티어드랍(눈물방울) / 뉴 티어
  최근 침탈법의 하나인데 리눅스와 윈도우스의 IP 프래그멘테이션 코드에 존재하는 버그를 쓴다. 2.0.33 버전의 커넬에서부터 고쳐지기 시작했고, 고칠 때 커넬 컴파일-타임 옵션을 선택할 필요는 없다. 리눅스는 "뉴 티어" 침탈법에는 영향을 받지 않는다. 대부분의 침탈법 코드와 이 것들이 어떻게 움직이는 지에 대한 깊은 설명이 필요하면 http://www.rootshell.com에서 서치 엔진을 써서 구할 수 있다.

18.8 NFS (네트워크 파일 시스템) 보안

NFS는 매우 널리 쓰이는 파일 공유 프로토콜이다. NFS를 이용하면 커넬에서 nfs 파일시스템을 지원해 주는 만약 리눅스가 아닌 경우에는 다른 클리이언트가 지원해 주는 다른 컴퓨터들로 nfsd와 mountd를 실행하는 서버가 파일시스템을 "수출" 할 수 있게 해 준다. Mountd는 /etc/mtab에 마운트된 파일시스템을 관리하면서 내용을 보여줄 수 있다. 사용자들에게 홈 디렉토리를 제공하기 위해서 NFS를 많은 사이트이 사용하고 있으며, 이렇게 함으로써 사용자들이 어느 컴퓨터에서 로그인 하였던 간에 사용자들은 홈 파일들을 가질 수

있게 된다. 파일시스템을 공유할 때 사용할 수 있는 얼마 안되는 "보안" 설정이 몇 가지 있다. 여러분은 원격 컴퓨터의 루트 사용자(uid=0)를 nobody 사용자로 대응시켜서, 공유된 파일시스템 전체 접근 권한을 갖는 것을 거부하도록 nsfd를 설정해야 한다. 그러나 개인 사용자는 각자의 (혹은 최소한 같은 uid의) 파일에 대한 접근권이 있기 때문에, 원격지의 수퍼유저는 자기 계정으로의 로그인이나 su 사용이 가능하며, 자기 파일들에 대해서 완전한 접근권을 가질 수 있다. 이렇게 하는 것은 원격 파일시스템을 마운트할 권한을 가진 공격자에게는 사소한 장애물밖에 되지 못한다.

NFS를 꼭 써야한다면, 꼭 공유해야만 하는 컴퓨터들로만 전송 되도록 조정하라. 루트 디렉토리 전부를 공유해서는 절대로 안되며, 필요한 디렉토리만 공유해야 한다. NFS에 대한 더 자세한 정보가 필요하면 NFS 하우투를 보도록 하라.

18.9 NIS (네트워크 정보 서비스) (예전의 YP)

네트워크 정보 서비스(Network Information service, 예전의 YP)는 그룹의 컴퓨터들에 정보를 배포하는 한 가지 방식이다. NIS 주 서버는 정보표를 소유하며 그것들을 NIS 대응(map) 파일들로 변환한다. 이 대응 파일들이 네트워크를 통해 제공됨으로써 NIS 클라이언트 컴퓨터들은 로그인과 패스워드, 홈 디렉토리와 쉘에 대한 정보 (즉 보통의 /etc/passwd 파일에 들어있는 모든 정보)를 얻을 수 있게 된다. NIS를 이용하면 사용자들은 패스워드를 한 번만 바꾸면 그 NIS 영역에 들어있는 모든 컴퓨터에 (정보가 갱신되도록) 할 수 있다.

NIS는 안전한 것이 아니다. 원래부터 안전을 염두에 두고 만든 것이 아니었다. 단지 간편하고 쓸모 있는 작업 역할로 만든 것뿐이다. (네트워크 상 어디에 있건) 여러분의 NIS 도메인의 이름을 알아맞힐 수 있는 사람은 여러분의 passwd 파일 복사본을 얻을 수 있고, 여러분의 사용자 패스워드를 깨기 위해 Crack과 John the ripper를 쓸 수 있게 된다. NIS를 속여서 (spoof) 온갖 지저분한 일을 하게 할 수도 있다.

꼭 NIS를 써야 겠다면, 이런 위험들을 감수해야 한다. NIC+라고 불리는 NIC보다 안전한 대체품이 있다. NIC HOWTO를 읽어보기 바란다

http://sunsite.unc.edu/mdw/HOWTO/NIS-HOWTO.html

18.10 방화벽

방화벽(firewall)은 여러분의 지역 네트워크 안팎으로 어떤 정보가 출입할 것인가를 허가 통제하는 한 가지 방법이다. 전형적으로 방화벽 호스트는 인터넷과 지역 랜에 연결시키고, 랜에서 인터넷으로의 엑세스는 방화벽을 통해서만 가능하도록 하는 것이다. 이렇게 하면 방화벽에서 인터넷과 여러분의 사이를 오가는 정보를 제어할 수 있다.

방화벽을 설정하는 수많은 유형과 방법들이 있다. 리눅스 컴퓨터는 상당히 저렴하면서도

훌륭한 방화벽이 될 수 있다. 방화벽 코드는 컴파일을 통해 2.0 이상의 커넬에 바로 삽입될 수 있다. 사용자 공간 도구인 ipfwadm(레드햇 6.0에서는 ipchains라는 툴로 바뀌었다)을 쓰면, 어떤 종류의 네트워크 트래픽을 허락할 것인가를 손쉽게 바꿀 수 있다. 또한 일정한 유형의 네트워크 트래픽은 일지에 적도록 (log) 조정할 수 있다.

방화벽은 네트워크 보안화에 있어서 매우 중요하고도 유용한 기술이다. 하지만 방화벽이 있으니까 그 뒤의 네트워크에 있는 컴퓨터들의 보안은 필요 없다고 생각해서는 절대로 안 된다. 이렇게 생각하는 것은 치명적인 실수다. 방화벽과 리눅스에 대한 정보를 더 얻고 싶으면 선사이트에 가서 최근에 만들어진 방화벽과 하우투를 읽어보도록 하라. Firewall HOWTO 추가의 정보가 IP-마스커레이드 미니 하우투 파일에도 있다.

19 우선적 보안 대책

좋다. 시스템에 대한 전반적인 검사를 하고, 가능한 한 안전하게 만들었다고 판단을 했으면, 이제는 접속을 할 순서다. 침입이 생길 경우 빠르게 침입자를 무능력화 시키고, 원상복구를 하려면 준비를 해야 할 것이 몇 가지 있다.

19.1 완벽한 백업을 만들 것

백업 방법과 저장을 이 파일의 주제에서 벗어나는 일이지만 백업과 보안에 관하여서 몇 마디는 적어놓겠다. 만약에 하나의 파티션에 650 mb 이하의 데이터를 가지고 있다면 CD-R을 쓰는 것이 좋다. (내용을 변경 당하는 걱정이 없고, 오랜 시간 동안 저장 할 수 있기 때문이다) 내용을 변경 당하는 경우가 없도록, 테이프와 다른 재사용이 가능한 것들은 백업이 끝나는 대로 쓰기 방지를 하고 확인을 하는 것이 좋다. 백업은 안전한 제 2의 장소에 저장하도록 하라. 제대로 된 백업은 시스템을 다시 복구하는 시발점이 될 것이기 때문이다.

19.2 좋은 백업 스케줄을 고르기

6개의 테이프를 1 주기용 한 묶음으로 쓰는 것이 관리하기에 편하다. 4개의 테이프를 주중에 사용하고, 한 개를 격주로 짝수 금요일에, 남은 한 개를 격주로 홀수 금요일에 사용하는 것을 방법이다. 매일 부분 백업(incremental backup)을 만들고, 전체 백업을 적절한 (격주로 사용되는) 금요일 용 테이프에 만든다. 만약 특별히 중요한 변경을 했거나, 어떤 중요한 데이터를 추가했을 경우에는 특별히 백업을 만드는 것이 좋을 것이다.

19.3 RPM 데이터베이스의 백업

침입의 경우에는 RPM 데이터베이스를 트립와이어처럼 사용할 수 있겠지만, 이 경우에는 데이터베이스까지도 변경되지 않도록 확인을 하고 써야 할 것이다. RPM 데이터베이스를 플로피에 카피하고, 이 것을 항상 제 2의 장소에 보관한다. 데비안 배포본도 비슷한 것을

가지고 있다. 특별한 것은, /var/lib/rpm/fileindex.rpm과 /var/lib/rpm/packages.rpm은 디스켓 한 장에 들어가지 않을 수 있다. 압축되면 각각 한 장씩에 들어갈 것이다. 만약 시스템이 침입을 당하면 다음의 명령어를 사용할 수 있다.

```
root#  rpm -Va
```

이것을 사용해서 시스템의 각 파일을 확인한다. 이 명령어는 다소 말이 많은 셈이니 (짧은 결과가 되도록 하는) 옵션들을 보기 위해서, RPM 매뉴얼 페이지를 보도록 하자.

이 방법을 쓰면 새로운 RPM을 더할 때마다 RPM 데이터베이스를 다시 복사해 두어야 한다. 장단점은 여러분이 결정해 써야 할 것이다.

19.4 시스템 사용 정보 (account data)의 조사

시스로그 (시스템 일지: syslog)의 정보가 침해를 받지 않도록 하는 것은 중요한 것이다. /var/log 파일을 제한된 사용자들만이 읽고 쓸 수 있도록 하는 것은 좋은 시작이다. "auth" 항에 특히 주의를 두면서, 일지에 어떤 것들이 쓰여지고 있는 지를 감시하는 것이 좋다. 예를 들어서, 여러 번의 접속 실패는 침입 시도를 의미하는 것일 수 있다. 로그 파일 (log file)이 어디에 있는 지는 여러분 배포본의 종류에 따라서 다르다. 레드햇과 같이 "리눅스 파일 시스템 규격 (Linux Filesystem Standard)을 따르는 배포본이라면, /var/log을 보면서 메시지를 확인하고, mail.log과 다른 것들을 보는 것이 좋다.

/etc/syslog.conf를 보면 여러분이 쓰는 배포본이 어디에 무엇에 대한 일지(log)를 만드는지 볼 수 있다. 이 파일은 syslogd (시스템 일지 데몬)에게 어떤 내용의 일지를 어떤 장소에 적어 놓아야 한다는 것을 알려주는 것이다. 시간이 날 때 검사할 있도록 -- 여러분의 일지 보관 스크립트나 데몬을 조정해서 -- 일지들을 오래 보관하도록 하는 것도 좋다. 최근의 레드햇 배포본의 "logrotate" 패키지를 한 번 보라. 다른 배포본도 비슷할 것이 있을 가능성이 있다.

만약 여러분의 로그 파일이 변경된 것처럼 보이면, 언제 변경이 시작되었는지, 어떤 내용이 변경되었는지 결정을 내리도록 노력해 보라. 일지가 오랜 시간 동안 비어 있는가? (만약 있다면) 백업 테이프에 있을 변경되지 않은 일지를 뒤져보는 것도 좋은 생각이다. 보통은 침입자가 침입 흔적을 없애기 위해서 바꿔 버리지만, 그래도 이상한 일들을 검사하기 위해서 일지를 확인하는 것이 좋다. 어쩌면 침입자가 침입을 시도하는 것이나, 루트 계정을 얻으려고 프로그램의 침탈을 시도하는 것을 알아챌 수 있다. 침입자가 채 바꾸기 전에, 일지를 볼 수도 있는 것이다.

"su"를 써서 사용자를 바꾸려는 시도나, 접속 시도나, 다른 사용자 계정 정보 등의 다른 일지 데이터 등에서 "auth"항은 별개로 하는 것이 좋다. 가능하다면, 가장 중요한 데이터는 보안이 철저한 시스템으로 복사본을 보내도록 syslog을 조정하라. 이 방법은 /login/sy/ftp/etc 시도를 지움으로서 흔적을 지우려는 침입자를 막을 것이다. syslog.conf의 매뉴얼 페이지를

보고, "@" 옵션을 참조하도록 한다.

마지막으로, 아무도 읽지 않는 다면, 일지(日誌)라는 것은 무용지물이다. 가끔 시간을 내서 일지를 읽도록 하고, 평일 평상시의 작업이 일지에는 어떻게 보이는가를 알아두는 것이 좋다. 이것을 알아두는 것이 이상한 일을 알아채는 데 도움이 된다.

19.5 새로운 시스템 업데이트의 설치

대부분의 리눅스 사용자는 CD-ROM에서 설치를 한다. 보안 구멍 막이 작업은 그 주기가 빠르므로, 새로이 (고쳐진) 프로그램은 항상 나오고 있다. 네트워크에 기계를 연결하기에 앞서서, 배포본의 (ftp.redhat.com 등의) ftp에 가서 업데이트된 팩키지를 받아서 새로 설치를 먼저 하는 것이 좋다. 이런 팩키지는 중요한 보안 개선책을 자주 담고 있으므로, 설치를 반드시 하는 것이 좋을 것이다.

19.6 침입 도중이나 후에 할 일들

드디어 여기에 적혀 있는 (혹은 다른 곳의) 조언을 따른 결과로 침입을 감지해 내었다면?

첫 번째로 할 일은 냉정을 유지하는 것이다. 성급한 행동은 공격자가 저지를 수 있는 것보다 더 큰 해를 끼칠 수 있다.

19.6.1 보안 공격 진행 중일 때

진행 중인 보안 공격을 알아차리는 것은 긴장되는 일일 수 있다. 여러분이 어떻게 대응하는가에 따라 중요한 결과를 가져올 수 있다. 공격이 물리적인 것이라면, 이상은 누군가 여러분의 집이나 사무실, 연구실에 침입한 것을 감지한 것일 것이다. 경찰에 알려야 한다. 연구실 환경이라면 누군가가 케이스를 열려고 하거나 컴퓨터를 재부팅하려고 하는 것을 볼 수도 있다. 여러분의 권한과 절차에 따라, 여러분은 그들에게 중지하도록 요구하거나 지역 보안 책임자에게 연락할 수 있다.

지역 사용자가 보안을 훼손하고자 하는 것을 감지했을 경우, 가장 먼저 해야 할 일은 그 사용자가 실제 본인인지 확인하는 것이다. 그 사용자가 어디에서 로그인 하려고 하고 있는지 확인해 보도록 하라. 그 곳이 평상시에 로그인해서 들어오는 곳인가? 그렇지 않은가? 다음에는 컴퓨터를 통하지 않은 직접적인 연락을 취해보도록 하라. 예를 들어 전화를 걸거나 그 사용자의 집 혹은 사무실로 직접 가서 이야기를 나눌 수 있다. 만일 그 사용자가 자기의 행위를 시인한다면, 그의 행위에 대해서 설명하도록 요구할 수 있고 그런 행위를 중지하라고 말할 수도 있다. 그가 부인하거나, 여러분이 말하는 사건에 대해서 모른다면 좀 더 조사를 해야 한다. 비슷한 사건들을 알아보고 고발이나 비난을 하기 전에 많은 정보를 확보하도록 하라.

네트워크를 통한 침투를 감지했다면, 처음 할 일은 (할 수 있다면) 네트워크 연결을 끊는 것이다. 침입자가 모뎀으로 접속했다면 모뎀 선을 뽑아버리도록 하고, 인터넷을 통해 접속했다면 인터넷 선을 뽑아라. 이렇게 하면 침입자가 더 큰 피해를 입히는 것을 막을 수 있고, 침입자는 아마 자신이 들통났다고 생각하기보다는 네트워크에 문제가 생긴 모양이라고 여길 것이다. 여러분이 네트워크 연결을 끊을 수 없다면 (접속이 빈번한 사이트이거나, 컴퓨터에 대한 물리적 관리권한이 없다면), 차선책은 침입자의 사이트로부터 접속해 들어오는 것을 막기 위해 tcp_wrapper나 ipfwadm 같은 프로그램을 사용하는 것이다. 침입자의 사이트에서 들어오는 모든 사람들의 접근을 거부할 수 없는 입장이라면, 사용자들의 계정을 폐쇄하여야 한다. 하나의 계정을 폐쇄하는 것은 쉬운 일이 아니라는 점에 주의하라. .rhosts 파일과 FTP를 통한 접근, 매우 많은 뒷문 (backdoor)을 염두에 두어야 한다.

위의 조치들 (네트워크 절단, 공격자의 사이트로부터 오는 접근 시도 거부, 그리고/혹은 그들의 계정 폐쇄) 가운데 한 가지를 하고 나면, 공격자의 모든 사용자 프로세스를 죽이고 그들을 로그오프 시켜야 한다. 공격자는 다시 들어오려고 시도할 것이므로, 다음 몇 분 동안은 여러분의 사이트를 자세히 감시해야 한다. 공격자는 아마도 다른 계정을 쓸 것이고, 다른 네트워크 주소를 쓸 수도 있다.

19.6.2 보안 훼손이 이미 일어난 경우

이미 일어난 사고를 뒤늦게 겨우 알아차렸거나, (바라기로는) 감지된 공격자를 여러분의 시스템에서 잠가서 쫓아내 버렸다.. 이제는 무엇을 해야 할까?

19.6.2.1 개구멍 막아내기

공격자가 여러분의 시스템에 들어오기 위해 사용한 방법이 무엇인지 알 수 있다면, 그 구멍을 막도록 해야 한다. 예를 들어서, 어쩌면 침입자가 들어오기 바로 직전에 FTP 사용이 여러 번 보이는 것을 보았다고 하자. 이런 경우에는 FTP 서비스를 중지하고 개정 버전이나 알려진 교정 사항 목록이 있는지 찾아봐야 한다. 로그 파일들을 확인해 보고, 여러분의 보안 리스트와 페이지 사이트에 가서, 고쳐야 하는 새롭거나 잘 알려지고 있는 침탈법이 올려져 있는지 목록을 살펴보도록 하라. 칼데라 보안 수정안은

http://www.caldera.com/tech-ref/security/

에서 구할 수 있다. 레드햇은 아직 보안 수정안 목록을 버그 수정안에서 구분하지 않고 있지만, 배포본의 수정안 정보는 http://www.redhat.com/errata에서 구할 수 있다. 또한 한 제작자가 보안 수정안을 내면, 다른 대부분의 제작자 또한 내놓을 수 있다. 공격자를 완전히 차단하지 않으면, 그는 대개 다시 돌아온다. 여러분의 컴퓨터뿐 아니라, 여러분의 네트워크 안의 어딘 가로 말이다. 공격자가 패킷 스니퍼를 작동시키고 있었다면, 그는 지역 내의 다른 컴퓨터로 접근할 수 있다.

19.6.2.2 피해 평가

첫 번째 할 일은 피해를 평가하는 것이다. 무엇이 훼손되었는가? 트립와이어 같은 완전성 검사 프로그램을 사용하고 있다면, tripwire를 실행시켜서 알아볼 수 있다. 이런 프로그램이 없다면, 모든 중요한 자료들을 일일이 살펴보아야 한다. 리눅스 시스템은 갈수록 설치하기 쉬워지고 있으므로, 중요한 설정 파일들을 따로 저장해 두고 디스크를 아예 지워버린 다음 리눅스를 처음부터 다시 설치한 뒤, 백업으로부터 사용자 파일과 설정 파일들을 복구하는 것을 고려해 볼 수도 있다. 이렇게 하면 깨끗한 시스템을 새로 갖게 된다. 만약 보안이 깨진 컴퓨터의 백업을 만든다면 -- 침입자가 심어둔 트로이의 목마일 수 있으므로 -- 어떠한 이진 파일이라도 주의해서 보아야 할 것이다.

19.6.2.3 백업, 백업, 그리고 또 백업

정기적으로 백업을 해두는 습관은 보안 문제에 있어서는 정말 중요하다. 시스템의 보안이 깨졌을 때, 필요한 자료를 백업으로부터 복구할 수 있다. 물론 공격자에게도 가치 있는 자료는, 훔쳐서 자기의 사본을 만들어 둔 다음에 파괴하겠지만, 최소한 자료를 도난 당할지언정 잃지는 않는 것이다. 변조된 파일을 백업된 것으로 복구하기 전에, 이전의 여러 백업본들을 인해 보아야 한다. 침입자가 파일을 오래 전에 망쳐놓았다면, 이미 변조된 파일들만 잔뜩 백업해 놓았을 수도 있기 때문이다. 물론 백업본들에 대해서도 보안 문제가 있다. 백업본들을 안전한 장소에 두었는지 확인하여야 하고, 누가 거기 접근할 수 있는지 알고 있어야 한다. (공격자가 백업본을 얻을 수 있다면, 여러분이 모르는 사이에 여러분의 모든 자료에 접근할 수 있게 되는 것이다.)

19.6.2.4 침입자 추적

침입자를 몰아내고, 시스템을 복구했다고 해서 모든 일이 끝난 것은 아니다. 대개 침입자들은 잡히지 않지만, 그래도 공격 사건을 보고해야 한다. 공격자가 여러분 시스템을 공격하던 사이트의 관리자에게 그 사건을 알려주어야 한다. 연락처는 "whois"나 internic 데이터베이스를 이용해서 찾아볼 수 있다. 상관된 모든 일지 내용과 날짜 및 시간을 첨부해서 저쪽 관리자에게 email을 보내는 것도 좋다. 침입자에 대해서 어떤 특이한 점을 발견했다면 그것도 함께 알려주도록 하라. email을 보낸 뒤에 (하고 싶다면) 전화를 써서 연락을 계속하도록 하라. 저쪽 관리자가 그 공격자를 찾아냈다면, 그 관리자가 다시 공격자가 들어온 사이트의 관리자에게 말하고 뭐 그렇다.

뛰어난 해커는 대개 많은 건너뛰기용 중간 시스템들을 사용한다. 이 시스템들 중의 어떤 (혹은 여러분의) 장소에서는 침입 당했다는 사실조차 모를 수도 있다. 크래커의 원래 시스템까지 쫓아가는 것은 어려운 일이다. 여러분이 이야기하게 되는 관리자들에게 공손하게 대하는 것은 그들로부터 도움을 얻어내는데 좋다. 여러분이 관계되는 (CERT나 이와 비슷한) 모든 보안 조직들에도 알려주어야 한다.

II IP 방화벽

여러분이 시스템의 보안 설정을 할 때 특정 서비스 별로 다른 보안 정책을 세울 수 있다. 그러나 보안 장치는 하나만 사용하기 보다는 여러 가지 보안 장치를 겹쳐서 사용하여야 한다. 여기서 다루고자 하는 IP 방화벽 기능은 최고의 저수준에서 아예 패킷 자체를 막아 버리는 강력한 보안 방법이다. 우리가 다룰 방화벽 명령어는 ipchains이다. 다음 홈페이지를 가면 더 많은 정보를 얻을 수 있다.

http://www.rustcorp.com/

1 기본 개념

▶ 패킷이란?
네트웍을 통하는 모든 것은 패킷의 형태를 띤다. 각 패킷의 앞 부분에는 패킷이 어디로 향하고 있는지, 어디서 왔는지, 어떤 종류의 패킷인지 그리고 그 밖의 관리 상 필요한 세부 사항이 적혀있다. 이 부분을 패킷의 '헤더(header)'라 부른다. 나머지 부분에는 전송하고자 하는 실제 자료가 들어있으며 '몸체(body)'라 부른다.

▶ 패킷필터란?
패킷 필터란 지나가는 패킷의 헤더를 보고 패킷의 운명을 결정짓는 소프트웨어이다. 패킷을 마치 패킷을 안받은 것처럼 버리기 위해 부인(deny)할 수 있고 패킷이 지나가도록 허용하거나 패킷을 거절(reject)할 수 있다.

▶ 어떤 패킷을 필터링 할 수 있나?
들어오는 패킷, 나가는 패킷, 전달되는 패킷을 필터링할 수 있다. 이 세가지의 입/출력에는 입/출력을 관장하는 규칙이라는 게 있다. 이러한 규칙은 여러개가 존재할 수 있다. 입/출력과 그에 해당하는 규칙 목록이 연결되어 있다는 의미에서 '사슬'이라고 한다고 생각하면 좋을 것이다. 기본적으로 입력 사슬, 출력 사슬, 전달 사슬 세 개의 사슬이 존재한다. 이 사실은 삭제가 불가하다. 이외에 사용자가 정의할 수 있는 사슬도 있다.

▶필터링 목표에는 어떤 것이 있나?
우리는 입력 사슬, 출력 사슬, 전달 사슬의 패킷이 어떤 조건(규칙)을 만족할 때 무엇을 할 것인가(패킷을 받아들일 것인가, 아니면 버릴 것인가등의) 목표를 정할 수 있다. 여러분은 시스템 관리자로서 사소한 것부터 복잡하고 큰 문제까지 보안에 대해서 많은 생각을 해야 한다. 그리고 결정을 내려야 한다. ipchains에서 지원하는 필터링 목표로는

1. 허가(Accept)
2. 부인(Deny)
3. 거절(Reject)
4. 반환(Return)

5. 가장(Masquerading)
6. 방향전환(Redirect)

이다. '허가'는 패킷을 허용한다는 것이다. '부인'과 '거절'은 조건(규칙)에 의거하여 필터를 통과할 수 없는 패킷을 버리는 행위에 있어서는 같다. 하지만 '거절' 정책에서는 "ICMP Destination Unreachable"이라는 목적지에 도착할 수 없다는 메시지를 패킷을 보내온 곳에 보내 준다. 이렇게 하면 발신지 측에서 영문도 모른 채 반응을 기다리게 하는 것에서 구제해 줄 수 있다. '반환'이라는 것은 현재 조건을 만족하면 다음 조건을 따지지 말고 끝내라는 것이다. '가장'이라는 것은 패킷의 발신지를 다른 발신지로 속이기 위해 사용한다. '방향전환'이라는 것은 말 그대로 패킷을 다른 곳으로 전환한다는 것이다.

2 패킷 필터링을 하기 위해 준비해야 할 것들은?

▶ 커널 컴파일
커널에 어떤 기능을 넣지 않으면 ipchains를 사용할 때 커널에서 지원하지 않는다는 메시지를 받을 것이다. 이 때는 커널을 다시 컴파일 해 주어야 한다. 최소한 다음과 같은 부분이 설정되어 컴파일되어 있어야 한다. 배포판마다 다르겠지만, 거의 모두 이 기능은 포함시킨 채로 배포할 것이다. 파워 리눅스 6.0을 사용한다면 커널을 컴파일할 필요는 없다.

```
Networking options ----->
        [*] Network firewalls
        ....
        [*] IP: firewalling
        [*] IP: masquerading
        ....
        [*] IP: optimize as router not host
```

네트웍 방화벽(Network firewalls) 기능과 그리고 IP 방화벽(IP: firewalling)은 필수적으로 커널 기능에 설정해 두어야 한다. 나중에 아주 유용한 IP 매스커레이딩을 필요로하는 경우가 있으므로 IP 매스크러에딩(IP: masquerading)도 켜두면 좋다. 마지막으로 여러분의 리눅스 박스가 하나의 독립적인 호스트라기보다는 라우터로서의 역할을 더 많이 한다고 생각할 때는 마지막 옵션을 설정하여 라우터로서 더 최적화되도록 컴파일한다.

3 IP 패킷 필터링 조건(규칙)

패킷 헤더에는 발신지의 주소, 도착 목적지의 주소, 포트(Port) 번호, 그리고 TCP/UDP 등의 패킷 형태 등 많은 정보가 저장되어 있다. 패킷 헤더를 조사하여 허가/거부 여부를 결정한다. 패킷 필터링에 사용되는 조건들을 분류하여 설명하면 다음과 같다.

발신지/목적지 주소(-s/-d)
패킷이 어디로부터 오는지, 그리고 어디로 가는지 점검하여 결정한다.

사용하는 프로토콜의 종류(-p)
패킷에는 접속 지향의 TCP, 접속 비지향의 UDP, 그리고 특별한 영역의 ICMP가 있다. 각 서비스는 보통 이 3가지 중 하나를 사용하는데, 프로토콜 자체에서 필터링해버리는 방법이 있다.

발신지/목적지 포트 번호
TCP/UDP 헤더(header) 부분에는 포트 번호가 적혀 있다. TCP/IP 네트웍은 서비스를 바로 고유한 포트 번호로 구별한다. 포트란 패킷이 다다르는 목적지 주소의 어느 위치에 정박할 것인가를 나타낸다고 생각하면 된다. 같은 호스트에 도착한 패킷이라도 23번 포트를 향해 온 패킷은 텔넷 서비스이며, 25번 포트를 향해 온 것은 메일 서비스이다. 웹서비스는 80번 포트를 일반적으로 사용한다. 따라서 특정 포트를 거부하면 그 서비스는 사용할 수 없다. 포트 번호와 서비스간의 관계는 /etc/services를 참고하면 알 수 있다.

TCP 표식(flag)(-y)
연결 지향 방식의 서비스에서는 접속과정을 위한 TCP 패킷 내부의 ACK/SYN 비트라는 것이 있다. 이것을 필터링해 버리면 외부에서의 연결 지향 서비스에 대한 접속이 불가능해진다. ACK/SYN 비트는 나중에 '서비스 거부'에서 많이 사용하고 있어 주의를 기울여야 하는 것 중 하나이다.

ICMP 메시지 유형
ICMP 패킷에는 ICMP 메시지의 종류에 대한 정보가 들어 있다. 예를 들어, 어떤 호스트가 살아 있는지 여부를 확인해 보는 ping을 잘 알 것이다. 여기서 발신되는 반향 요청(Echo Request) 패킷을 거부해 버리면 외부 침입자는 도대체 리눅스 박스가 살아 있는지, 아닌지 여부를 직접 달려와서 모니터를 쳐다 보지 않는 이상 알 도리가 없다.

4 방화벽 관리도구 : ipchains

ipchains는 필터링 조건을 결정해 나가는 도구이다. 명령어 사용법이 복잡하나 사슬, 조건, 전달인수, 기타 옵션 네가지로 크게 나누어 정복해 나간다면 어려울 것 없다. 사용법은 대략 다음과 같다.

지금 이해가 가지 않아도 상관 없다. 이 글을 읽다보면 이해할 수 있을 것이다.

4.1 명령어 사용법

o ipchains 명령 사슬 조건 목표
  이때 명령어로는 -A(append), -D(delete), -C(check)이 온다.

ex) 발신지가 모든 호스트이고(-s 0.0.0.0/0) 수신지가 192.168.1.1인(-d 192.168.1.1) 조건을 만족하는 입력(input) 패킷에 대하여 '부인'(-j DENY)하는 필터링 정책을 추가한다(-A).

```
ipchains -A    input    -s 0.0.0.0/0 -d 192.168.1.1       -j DENY
         명령  사슬                  조건                  목표
```

4.2 사슬의 종류

다음 네 개의 사슬로 나눌 수 있다.

1. 입력 사슬(input)
2. 출력 사슬(output)
3. 전달 사슬(forward)
4. 사용자 정의 사슬

입력/출력/전달 사슬은 내장(built-in)사슬이다.

4.3 사슬을 다루는 명령

전체 사슬을 다루는 명령
우선 사슬을 선택한 후 각 사슬에 맞게 명령을 사용하면 된다. 명령을 분류해 보자.

o -N <사슬이름>
  새 사슬을 만든다.

o -X <사슬이름>
  빈 사슬을 지운다.

o -P <기본목표>
  policy. 내장 사슬에 대해 기본 목표를 설정/변경한다. <기본목표>값으로는 '허가/부인/거절'이 올 수 있다. 패킷 필터링 방화벽이 적절한 규칙을 발견하지 못했을 때 참조한다.

o -F
  flush. 규칙을 모두 지운다. IP 방화벽 설정은 다시 하고자 할 때 맨 처음에 적어 준다.

o -L
  list. 규칙을 화면에 표시한다.

사슬의 규칙을 관리하는 명령

o -A <사슬>, -I <사슬>, -D <사슬> -R <사슬>
Apend, Insert, Delete, Replace. 각각 규칙을 추가하고 특정 위치에 삽입하고 삭제하며 기존 사슬속의 특정 규칙을 새 규칙으로 바꾼다.

o -M [-L|-S]
Masquerading. 매스커레이딩을 설정한다. -L을 같이 적어주면 현재 매스커레이드도니 접속 규칙을 나열하며 -S를 적어주면 매스커레이딩 설정할 타임아웃값을 부가로 적어 주어야 한다.

4.4 필터링 조건 표현

▶ 발신,도착 IP주소(-s,-d)

-s <주소>[/넷매스크 또는 숫자 표현식 매스크] [포트번호,...]
-d <주소>[/넷매스크 또는 숫자 표현식 매스크] [포트번호,...]

각각 발신지/도착지를 명시할 때 사용한다. <주소>에는 호스트명, IP 주소, 네트웍명이 올 수 있다. 넷매스크는 255.255.255.0 처럼 적는다. 또는 상위 24비트가 1이고 나머지 비트가 0이라는 의밍에서 24라고도 적는다. 포트 번호는 서비스를 나타낸다. /etc/services를 참고하여 telnet, rlogin같은 서비스에 대한 포트 번호를 써 주면된다. 서비스 명칭을 그대로 써 주어도 상관없다. 범위로도 지정할 수 있다. 1023:65535 처럼 지정하면 된다. 포트로 명기할 때 콜론(:) 왼쪽 포트 번호를 생략하면 0을 오른쪽 포트 번호를 생략하면 65535를 의미한다.

▶ 역규칙 명시하기(! <프로토콜|포트>)

'-s', '-d' 플래그를 포함하는 많은 플래그 앞에 'not'을 뜻하는 '!'를 붙이면 주어진 주소나 포트(즉 서비스)와 같지 않는 것을 표현한다. 즉 '그 이외의 것'을 의미한다. 예를 들어 '-s ! localhost'는 localhost에서 오지 않는 모든 패킷과 일치한다. ! www 은 www 이외의 모든 서비스와 일치한다.

그렇다면 여러분은 다음 두 조건의 차이를 구별 할 수 있을 줄로 안다.

-p TCP -d ! 192.168.1.1 www

-p TCP -d 192.168.1.1 ! www

첫번째 것은 192.168.1.1 을 제외한 모든 머신의 WWW 포트를 사용하는 TCP 패킷을 가리키는 반면 두번째 것은 192.168.1.1 머신으로 오는 WWW 포트 이외의 모든 TCP 접속을 나타낸다.

▶ 프로토콜 명시하기(-p <프로토콜>)

패킷 형태를 선택할 때 사용한다. 보통 TCP, UDP, ICMP를 지칭한다.

▶ ICMP 타입과 코드 명시하기

ICMP 역시 부가 옵션을 가지고 있다. 하지만 ICMP 에 있어 포트란 없기 때문에 그 의미는 전혀 다르다.

'-s' 옵션 다음에 ICMP 이름(이름을 알아보려면 'ipchains -h icmp'라고 명령 한다)을 적거나 또는 숫자로 된 ICMP 유형과 코드로 적을 수 있다. ICMP 유형은 '-s' 옵션 뒤에 나오고 코드는 '-d' 옵션 뒤에 나올 수 있다.

ICMP 이름은 상당히 길다(ipchains -h icmp 해 보면 알 수 있다). 따라서 다른 것과 구별할 수 있을 정도로만 적어 주면 된다. 다 적어줄 필요는 없다.

가장 흔한 ICMP 패킷을 아래 정리하였다.

번호	이름	필요로 하는 곳
0	echo-reply	ping
3	destination-unreachable	모든 TCP/UDP 자료 교환
5	redirect	라우팅 데몬을 사용하지 않을 때의 라우팅
8	echo-request	ping
11	time-exceeded	traceroute

▶ 인터페이스 명시하기(-i)

인터페이스 이름으로는 eth0, ppp0등이 온다. +로 끝나면 그 문자열로 시작하는 모든 인터페이스 이름을 지칭한다. 예를 들어 eth+라고 하면 eth0, eth1...을 나타내는 것이다. 여기에서 역규칙을 명시할 수 있다. 즉 ! eth+라고 하면 eth0, eth1...가 아닌 인터페이스를 나타내는 것이다.

▶ TCP SYN 패킷만 명시하기(-y)

TCP 접속이 이루어지기 위해서는 처음에 '접속을 원한다(SYNC)', '승인한다(ACK)', '고맙다(FIN)'의 세가지의 패킷이 양방간에 오고 가야 한다. 때로는 TCP 접속을 한 방향으로만 허용할 필요가 있다. 예를 들어 외부 WWW 서버에 대한 접속을 허용하면서도 지금 서버상에 작동 중인 서버에는 접속하지 못하도록 했으면 할 때가 있다. 자연스런 접근 방식은 서버로부터 오는 TCP 패킷을 봉쇄하는 것이다. 하지만 불행하게도 TCP 접속은 양방향으로 패킷이 오갈 수 있어야만 한다.

이에 대한 해결책으로서 접속을 요청하는 패킷만 봉쇄하는 방법을 사용한다. 이 패킷을 SYN 패킷이라 부른다.(기술적인 설명으로는 SYN 플래그가 설정되어 있고 FIN, ACK 플래그는 설정되지 않은 패킷을 가리킨다) 이 패킷을 허용하지 않음으로써 접속 요청을 막을

수 있다.

'-y' 플래그를 이런 용도로 사용한다. 오로지 TCP 프로토콜에만 해당한다. 예를 들어 192.168.1.1 로부터 오는 TCP 접속을 표현할 때는 다음과 같다.

```
-p TCP -s 192.168.1.1 -y
```

4.5 조각(Fragments) 처리하기

때때로 하나의 패킷이 한 번에 한 회선을 통과하기에는 너무 큰 경우가 발생한다. 이 때는 패킷이 '조각'으로 나뉘어 여러 개의 패킷으로 전송된다. 받는 쪽에서는 이 조각을 모아 하나의 패킷으로 재구성한다.

조각과 관련된 문제는 다음과 같다. 앞서 나열한 몇 가지 명시 방법 (특히 발신지 포트, 목적지 포트, ICMP 유형, ICMP 코드 또는 TCP SYN 플래그)은 커널이 패킷의 시작 부분을 보고 판별하는데 이 정보가 오로지 첫번째 조각에만 존재한다는 사실로부터 유래한다.

여러분의 머신이 외부 네트웍으로 나가는 유일한 연결 창구라면 커널 컴파일시 'IP: always defragment'를 Y 로 설정하고 컴파일함으로써 리눅스 머신을 지나가는 모든 조각을 모으도록 할 수 있다. 이렇게 함으로써 문제를 산뜻하게 처리할 수 있다.

그렇지 않을 때는 조각이 필터링 규칙에 의해 어떻게 처리되는지 이해하는 일이 중요하다. 우리가 갖고 있지 못한 정보를 요구하는 어떤 필터링 규칙도 일치하지 않을 것이다. 이는 첫번째 조각이 마치 다른 패킷과 마찬가지로 처리됨을 뜻한다. 하지만 두번째 이후부터는 문제가 발생한다. 따라서 '-p TCP -s 192.168.1.1 www' 라는 규칙(발신 포트가 'www'인 것)은 첫번째 조각을 제외하고 나머지 조각에 대하여 일치하지 않는다. 그 반대 규칙인 '-p TCP -s 192.168.1.1 ! www' 도 마찬가지다.

그렇지만 '-f' 플래그를 사용하여 두번째 이후의 조각에 대하여 규칙을 명시할 수 있다. 이 조각들에 대하여 TCP, UDP 포트, ICMP 유형, ICMP 코드 또는 TCP SYN 플래그를 명시하는 것은 분명히 적절지 않다.

'!'를 '-f' 앞에 붙임으로써 두번째 이후의 조각이 아닌 것에 대한 규칙을 명시하는 것이 가능하다.

일반적으로 두번째 이후의 조각은 그냥 놔두는 것이 안전하다. 왜냐하면 필터링 규칙이 첫번째 조각에 영향을 줄 것이고 목적지 호스트에서 제대로 조각이 모이지 못하게 할 것이기 때문이다. 하지만 이런 점을 이용하여 조각들을 보내게 되면 머신이 다운되는 버그가 발견된 적 있다. 여러분의 판단에 맡긴다.

네트웍 관리자가 주의할 점 : 잘못 형성된 패킷(TCP, UDP, ICMP 패킷이 포트나 ICMP 코

드, 유형을 방화벽 코드가 읽기에는 너무 작은 경우) 또한 조각으로 간주한다. 8 번째 위치에 있는 TCP 조각만이 명시적으로 방화벽 코드에 의해 버려진다.(이 때 syslog 에 에러 메시지가 나타난다)

예를 들어 다음 규칙은 192.168.1.1로 가는 모든 조각을 버린다.

```
# ipchains -A output -f -D 192.168.1.1 -j DENY
#
```

4.6 필터링의 부차적인 효과

지금까지 규칙을 사용하여 패킷을 잡아내는 모든 방법을 배웠다. 만약 패킷이 규칙에 부합하면 다음과 같은 일이 일어난다:

1. 규칙에 대한 바이트 카운터가 패킷의 크기(헤더오 모든 자료)만큼 증가한다.
2. 규칙에 대한 패킷 카운터가 증가한다.
3. 규칙에서 요구하는 경우 패킷을 기록한다.
4. 규칙에서 요구하는 경우 패킷의 서비스 유형(Type Of Service) 필드를 변경한다.
5. 규칙에서 요구하는 경우 패킷을 표시한다.(2.0 커널 시리즈에서는 안됨)
6. 규칙 목표를 조사하여 패킷에 대하여 어떤 일을 할 것인지 결정한다.

4.7 목표 명시하기

'target' 은 커널이 규칙에 맞는 패킷을 어떻게 처리할 것인지 말해주는 것이다. ipchains는 '-j' (어디어디로 점프한다고 생각하라)을 써서 목표을 명시한다.

가장 간단한 경우는 아무런 목표가 없는 경우이다. 이런 규칙(보통 '회계(accounting) 규칙'이라 부른다)은 특정 유형의 패킷에 대하여 갯수를 셀 때 사용된다. 규칙에 맞든 틀리든 커널은 사슬 속의 다음 규칙으로 향한다. 예를 들어 192.168.1.1로부터 오는 패킷의 갯수를 세고자 할 때는 다음과 같이 한다:

```
# ipchains -A input -s 192.168.1.1
```

('ipchains -L -v'를 사용하여 각 규칙에 연관된 바이트, 패킷 카운터를 볼 수 있다.)

6 개의 특별한 목표가 있다. 처음 3 가지는 'ACCEPT', 'REJECT', 'DENY' 로서 매우 간단하다. ACCEPT는 패킷이 통과하도록 허용한다. DENY는 마치 패킷을 받지 않은 것처럼 버린다. REJECT는 패킷을 버리지만 패킷의 발신지에 ICMP 답신을 보내어 목적지에 도착할 수 없음을 통보한다.(ICMP 패킷이 아닌 경우)

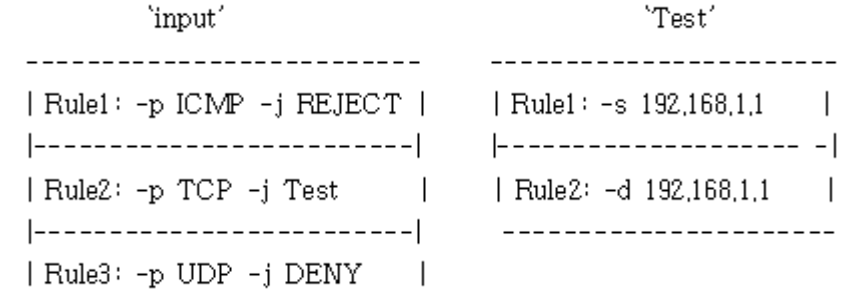
그 다음은 'MASQ'로서 커널로 하여금 패킷을 매스커레이드하게 한다. 제대로 작동하기 위해서는 IP 매스커레이딩이 가능하도록 커널이 컴파일되어 있어야 한다. 이 목표는 'forward' 사슬을 통과하는 패킷에만 유효하다.

또 다른 중요한 특별 목표로는 커널로 하여금 패킷이 향하고 어디로 향하고 있었든 상관없이 지역 포트로 패킷의 방향으로 변경해버리는 'REDIRECT'가 있다. TCP, UDP와 같은 프로토콜에서만 가능하다. 선택적으로 포트(이름 또는 번호)를 '-j REDIRECT' 다음에 적음으로써 어떤 특정 포트로 향하고 있던 패킷을 다른 포트로 방향전환시킬 수 있다. 이 목표는 'input' 사슬을 통과하는 패킷에 유효하다.

특별한 목표의 마지막은 'RETURN'으로서 즉시 사슬의 마지막 항목을 떠나도록 한다.

이외의 다른 목표는 사용자 정의 사슬(``전체 사슬에 대한 동작'' 참고)이다. 패킷은 사슬의 규칙을 통과하기 시작한다. 만약 사슬에서 패킷의 운명이 결정되지 않았고 사슬 통과를 마쳤다면 현재 진행 중이었던 사슬의 바로 다음 규칙에서 패킷 통과 작업이 재개된다.

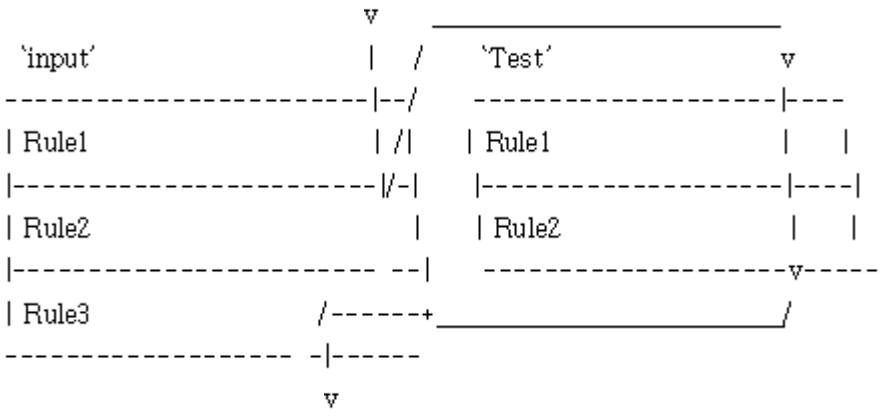
다음 2 개의 사슬을 생각해 보자. 하나는 내장 사슬인 'input'이며 하나는 사용자 정의 사슬인 'Test'이다.



192.168.1.1로부터 와서 1.2.3.4로 가는 TCP 패킷을 생각해본다. 패킷이 입력 사슬로 들어가서 규칙 1 번에 적용해본다. 해당되지 않는다. 규칙 2 번이 맞고 그 목표는 'Test'이다. 따라서 조사할 다음 규칙은 'Test'의 시작 부분이다. Test의 규칙 1 은 목표를 명시하고 있지 않으므로 그 다음 규칙 2 를 적용한다. 맞지 않으므로 사슬의 끝에 도달하게 된다. 여기서 우리는 마치 규칙 2 를 실행하고 나서 규칙 3 을 실행하게 된 것처럼 원래의

'input' 사슬로 돌아간다. 규칙 3 역시 적용되지 않는다.

따라서 패킷의 경로는 다음과 같다:



4.8 패킷 기록하기

규칙에 맞을 때 가질 수 있는 부수 효과이다. 일치하는 패킷에 대하여 '-l' 플래그를 가지고 기록할 수 있다. 일상적인 패킷을 기록하기 보다는 예외적인 상황을 발견하고자 할 때 사용하게 될 것이다. ('man klogd' 또는 'man dmesg' 참고)

4.9 서비스 유형 처리하기

IP 헤더에서 자주 사용되지는 않는 4 개의 비트를 서비스 유형(Type of Service, TOS) 비트라고 부른다. 이 비트들은 패킷의 처리 방식에 영향을 준다. 4 개의 비트는

- "최소 지연(Minimum Delay)"
- "최대 전송률(Maximum Throughput)"
- "최대 안정성(Maximum Reliability)"
- "최소 경로(Minimum Cost)"

이다.

한 번에 하나만 설정할 수 있다.

가장 일반적인 사용 방법은 telnet 과 ftp 제어 접속에 대해서는 "최소 지연(Minimum Delay)"을 사용하고 FTP 자료에 대해서는 "최대 전송률(Maximum Throughput)"을 사용하는 것이다. 다음과 같이 하면 된다.

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

'-t' 플래그는 2 개의 전달 변수를 16 진수로 받아들인다. 이렇게 함으로써 복잡한 TOS 비트 조작이 가능하다. 첫번째 매스크는 패킷의 현재 TOS 값에 AND되고 두번째 매스크는 XOR 된다. 너무 복잡하게 여겨진다면 다음 도표를 사용하라.

TOS 이름	값	전형적인 용도
Minimum Delay	0x01 0x10	ftp, telnet
Maximum Throughput	0x01 0x08	ftp-data
Maximum Reliability	0x01 0x04	snmp
Minimum Cost	0x01 0x02	nntp

4.10 기타사항

ipchains 의 유용한 기능 중 하나는 연관된 규칙을 묶어 사슬 속에 넣을 수 있다는 것이다. 내장 사슬('input', 'output', 'forward')과 내장 목표('MASQ', 'REDIRECT', 'ACCEPT', 'DENY', 'REJECT', 'RETURN')과 이름 충돌만 없다면 이 사슬을 어떻게 부르든 상관없다. ipchain의 저자는 대문자 이름을 피하길 권하고 있다. 사슬 이름은 8 자까지 가능하다.

5 사슬 다루기

5.1 새로운 사슬 만들기

새로운 사슬을 만들어보자.

```
# ipchains -N test
```

매우 간단하다. 이제부터 규칙을 추가할 수 있다.

5.2 사슬 지우기

사슬 지우기 또한 간단하다.

```
# ipchains -X test
```

5.3 사슬 비우기

`-F' 명령을 사용하면 사슬로부터 모든 규칙을 간단히 비울 수 있다.

```
# ipchains -F forward
```

사슬을 명시하지 않으면 모든 사슬 내용이 비워진다.

5.4 사슬 내용 보기

`-L' 명령을 사용하여 사슬 속에 든 모든 규칙을 나열할 수 있다.

```
# ipchains -L input
Chain input (refcnt = 1): (policy ACCEPT)
target        prot   opt      source             destination            ports
ACCEPT        icmp -----   anywhere           anywhere               any
# ipchains -L test
Chain test (refcnt = 0):
target        prot   opt      source             destination            ports
DENY          icmp -----   localnet/24        anywhere               any
#
```

`test' 항목에 표시된 "refcnt"란 `test'를 목표로 갖고 있는 규칙의 개수를 나타낸다. 사슬이 지워기 전에 이 값이 0 이어야 한다(그리고 사슬 자체도 빈 상태여야 한다).

사슬 이름이 생략되면 빈 것이라 할 지라도 모든 사슬이 표시된다. 사슬을 지우는데는 몇 가지 제한 사항이 있다. 일단 사슬이 빈 상태여야 한다.(``사슬 비우기'' 참고) 그리고 사슬이 다른 규칙의 목표가 되어서는 안된다. 3 가지 내장 사슬은 지울 수 없다.

사슬 이름이 생략되면 빈 것이라 할 지라도 모든 사슬이 표시된다.

`-L' 명령과 사용할 수 있는 3 개의 옵션이 있다. `-n'(숫자 표현) 옵션은 ipchains로 하여금 IP 주소 살펴보기를 하지 않게 하기 때문에 DNS 가 제대로 설정되지 않아 상당한 지연이 생기거나 DNS 요청을 필터링한 경우에 유용하다. 포트에 대해서는 포트 이름이 아닌 숫자로 표시되도록 한다.

`-v' 옵션을 사용하면 패킷, 바이트 카운터, TOS 매스크, 인터페이스 그리고 패킷 표식과 같은 규칙의 세부 사항을 볼 수 있다. 옵션을 붙이지 않으면 이 값들은 표시되지 않는다. 예를 보자:

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
pkts bytes target     prot     opt    tosa  tosx ifname mark source destination    ports
10    840  ACCEPT   icmp -----  0xFF 0x00 lo           anywhere anywhere  any
```

패킷, 바이트 카운터의 값이 1000, 1,000,000, 1,000,000,000 에 대하여 'K', 'M', 'G'라는 접미사를 사용하여 표시된다는 것을 눈여겨 보자.

'-x' 옵션을 사용하면 그 값이 매우 크다 할 지라도 완전한 숫자로 표시해준다.

6 카운터 재설정하기(0으로 만들기)

카운터 값을 0 으로 만들고 싶을 때가 있다. '-Z'(제로 카운터) 옵션을 사용 하면 된다. 예를 보자.

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
pkts bytes target     prot     opt    tosa  tosx ifname mark  source    destination    ports
10    840  ACCEPT icmp -----    0xFF 0x00  lo          anywhere anywhere       any

# ipchains -Z input
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
pkts bytes target     prot opt     tosa tosx  ifname mark  source    destination ports
 0    0    ACCEPT icmp ----- 0xFF 0x00  lo           anywhere anywhere  any
#
```

이런 접근 방식의 문제점은 종종 재설정 바로 직전에 카운터 값을 알아야 할 필요가 있다는 것이다. 위의 예에서 어떤 패킷이 '-L'과 '-Z' 명령 중간에 지나가는 경우가 발생한다. 이런 이유 때문에 값을 읽으면서 동시에 값을 재설정하기 위해 '-L'과 '-Z'를 같이 사용한다. 하지만 이 방법으로는 하나의 사슬에 대하여 명령할 수 없으므로 한 번에 모든 사슬을 보면서 동시에 0 으로 만들어야 한다.

```
# ipchains -L -v -Z
Chain input (policy ACCEPT):
pkts bytes target    prot    opt    tosa  tosx ifname mark source   destination  ports
10    840  ACCEPT icmp ----- 0xFF 0x00 lo           anywhere anywhere  any
```

```
Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
0  0 DENY icmp ----- 0xFF 0x00  ppp0 localnet/24  anywhere  any
# ipchains -L -v
Chain input (policy ACCEPT):
pkts bytes target   prot   opt   tosa  tosx  ifname mark source  destination ports
10   840  ACCEPT icmp ----- 0xFF 0x00  lo          anywhere anywhere any

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
0     0 DENY icmp ----- 0xFF 0x00  ppp0  localnet/24  anywhere     any
#
```

7 매스러케이딩(Masquerading)에 관련된 동작

IP 매스커레이딩과 관련하여 써먹을 수 있는 몇 가지 매개변수가 있다. 이들을 위한 별도의 도구를 만들 이유가 없다고 생각했기에(이 생각은 앞으로 바뀔 수 있다) 'ipchains' 에 그 기능을 포함시켰다.

IP 매스커레이드 명령은 '-M' 이다. '-L' 과 같이 쓰면 현재 매스커레이드되어 있는 접속 현황을 볼 수 있고 '-S'를 사용하여 매스커레이드 매개변수를 정할 수 있다.

'-L' 명령에 '-n'을 붙이면 호스트 이름과 포트 이름 대신 모두 숫자로 보여주며 '-v' 옵션을 주면 매스커레이드된 접속 상황에 대하여 순차 번호(sequence #)의 델타(delta)값을 보여준다.

'-S' 명령 뒤에는 3 개의 타임아웃 값을 초 단위로 적어야 한다. 하나는 TCP 세션, 또 하나는 FIN 패킷 이후의 TCP 세션, 그리고 하나는 UDP 패킷에 대한 값이다. 값을 변경하고 싶지 않을 때는 간단히 '0'을 적는다.

기본값은 '/usr/include/net/ip_masq.h'에 적혀 있으며 현재는 각각 15 분, 2 분, 5 분이다.

일반적으로 변경하는 값은 첫번째 값으로서 FTP 를 위해서이다.

8 패킷 점검하기

때로는 방화벽 사슬을 디버깅하기 위해 어떤 패킷이 머신에 들어오면 어떤 일이 발생하는지 알고 싶을 때가 있다. 'ipchains'의 '-C' 명령을 사용하면 된다. 커널이 실제 패킷을 검사

할 때 사용하는 동일한 루틴을 사용한다.

'-C' 인수 다음에 패킷을 테스트해 볼 사슬 이름을 적는다. 커널은 언제나 'input', 'output', 'forward' 사슬에서 시작하지만 테스팅을 목적으로 할 때는 어떤 사슬에서든 시작할 수 있다.

'패킷'에 대한 세부 설명은 방화벽 규칙을 명시할 때 사용했던문법 그대로 사용한다. 특히 프로토콜('-p'), 발신지 주소('-s'), 목적지 주소('-d'), 인터페이스('-i') 옵션을 꼭 명시해야 한다. 프로토콜이 TCP 또는 UDP 인 경우 발신지 주소 하나와 목적지 포트를 명시해야 하며 ICMP 프로토콜인 경우에는 ICMP 유형과 코드를 꼭 명시해야 한다. (조각 규칙을 가리키기 위해 '-f' 플래그를 사용한 경우가 아닐 때 그러하다. 만약 '-f' 옵션이 사용된 경우에는 발신지 주소, 목적지 포트 옵션은 쓸 수 없다)

프로토콜이 TCP 라면(그리고 '-f' 플래그가 없는 상태) '-y' 플래그를 사용하여 패킷에 SYN 비트가 설정되어 있음을 표현할 수 있다.

다음은 192.168.1.1 의 60000 포트에서 192.168.1.2 의 www 포트로 향하는 TCP SYN 패킷이 'input' 사슬에 들어가서 어떤 결과를 내놓는지 테스트하는 예이다(전형적인 WWW 접속 시작 과정의 예이다).

```
# ipchains -C input -p tcp -y -s 192.168.1.1 60000 -d 192.168.1.2 www
  packet accepted
#
```

9 한 번에 여러 규칙 만들기와 사건 감시하기

때로는 하나의 명령으로도 여러 규칙에 영향을 미칠 수 있다. 다음과 같은 두 가지 방법이 있다. 우선 DNS 를 통하여 여러 개의 IP 주소로 해석되는 호스트 이름을 명시하게 되면 'ipchains'는 마치 여러분이 각 IP 주소마다 한 번씩 똑같은 명령을 내린 것처럼 처리한다.

따라서 만약 'www.foo.com' 이라는 호스트 이름이 3 개의 IP 주소를 가리키고 'www.bar.com' 호스트 이름이 2 개의 IP 주소를 가리키게 된다고 하고 'ipchains -A input -j reject -s www.bar.com -d www.foo.com' 라 명령하게 되면 마치 6 개의 규칙을 'input' 사슬에 넣은 것과 같다.

'ipchains'가 여러 행동을 취하게 하는 다른 방법은 양방향 플래그('-b')를 사용하는 방법이다. 이 플래그를 사용하면 'ipchains'는 명령을 두 번 내린 것처럼 동작한다. 두번째 명령은 첫번째 것과 '-s', '-d'가 뒤바뀐 명령이다. 따라서 192.168.1.1로 향하거나 그로부터 들어오는 모든 패킷 전달을 막기 위해서는 다음과 같이 할 수 있다.

```
# ipchains -b -A forward -j reject -s 192.168.1.1
```

#

개인적으로는 `-b' 옵션을 좋아하지 않는다. 이게 편하다고 생각하는 사람은 ``ipchains-save 사용하기'' 섹션을 참고하라.

-b 옵션은 삽입(`-I'), 삭제(`-D') (규칙 번호를 적는 경우 제외), 추가(`-A'), 점검(`-C') 명령과 함께 사용할 수 있다.

유용한 플래그로는 여러분의 명령에 따라 'ipchains'가 어떤 일을 하고 있는지 보여주도록 하는 `-v'(장황하게)가 있다. 다수의 규칙에 영향을 미칠 것이라 생각하는 명령에서 쓸모있다. 예를 들어 192.168.1.1과 192.168.1.2 사이의 조각들의 상태를 점검하는 경우를 보겠다.

```
# ipchains -v -b -C input -p tcp -f -s 192.168.1.1 -d 192.168.1.2 -i lo
tcp opt  ---f- tos 0xFF 0x00 via lo 192.168.1.1 -> 192.168.1.2 * -> * packet accepted
tcp opt  ---f- tos 0xFF 0x00 via lo 192.168.1.2 -> 192.168.1.1 * -> * packet accepted
```

10 쓸만한 예제

전화접속 PPP 접속(`-i ppp0')을 하고 있다고 가정하자. 전화접속을 할 때마다 뉴스를 긁어오며(`-p TCP -s news.virtual.net.au nntp') 메일을 가져온다(`-p TCP -s mail.virtual.net.au pop-3'). 리눅스 박스를 정기적으로 갱신하기 위해 데비안 FTP 갱신 방법을 사용한다. (`-p TCP -s ftp.debian.org ftp-data') 그 동안 ISP의 프록시를 통해 웹 서핑을 즐긴다(`-p TCP -d proxy.virtual.net.au 8080'). 하지만 Dilbert Archive 사이트에 보이는 doubleclick.net 으로부터 광고를 싫어한다(`-p TCP -y -d 199.95.207.0/24' & `-p TCP -y -s 199.95.208.0/24').

접속 중에 다른 사람들이 내 머신으로 ftp해 들어오는 것은 게의치 않는다(`-p TCP -d $LOCALIP ftp'). 하지만 내 네트웍 외부의 사람들이 내 IP 주소를 가장하여 사용하길 원치는 않는다(`-s 192.168.1.0/24').

내부 네트웍에 다른 머신이 없기 때문에 설정은 매우 쉽게 이뤄진다.

지역 프로세스 어떤 것도(예를 들어 네스케이프, Gzilla 등등) doubleclick.net에 접속하지 못하도록 하고 싶다.

```
# ipchains -A input -d 199.95.207.0/24 -j REJECT
# ipchains -A input -d 199.95.208.0/24 -j REJECT
#
```

지역적으로 생성되어 'input' 사슬을 통과하는 패킷에 대하여 인터페이스를 'lo'로 설정하기

위해 '-i lo'를 명시할 수도 있다.

이제 나는 외부로 나가는 패킷에 대하여 우선권을 조정하려 한다(들어오는 패킷에 대하여 할 수 있는 일이라곤 별로 없다). 적지 않은 규칙을 가지고 있기 때문에 이 모두를 'ppp-out'이라는 이름의 사슬에 넣어두는 것이 좋다고 생각한다.

```
# ipchains -N ppp-out
# ipchains -A output -i ppp0 -j ppp-out
#
```

웹, 텔넷에 대해서는 최소 지연을 원한다.

```
# ipchains -A ppp-out -p TCP -d proxy,virtual,net,au 8080 -t 0x00 0x10
# ipchains -A ppp-out -p TCP -d 0,0,0,0 telnet -t 0x00 0x10
#
```

ftp 자료와 nntp, pop-3 에 대해서는 우선권을 낮춘다.

```
# ipchains -A ppp-out -p TCP -d 0,0,0,0/0 ftp-data -t 0x00 0x02
# ipchains -A ppp-out -p TCP -d 0,0,0,0/0 nntp -t 0x00 0x02
# ipchains -A ppp-out -p TCP -d 0,0,0,0/0 pop-3 -t 0x00 0x02
#
```

ppp0 인터페이스를 통해 들어오는 패킷을 제한한 몇 가지 규칙이 있다. 'ppp-in' 이라는 사슬을 만들어보자.

```
# ipchains -N ppp-in
# ipchains -A input -i ppp0 -j ppp-in
#
```

ppp0를 통해 192,168,1,* 이라는 발신 주소를 갖고 들어오는 패킷은 모두 기록하고 처리한다.

```
# ipchains -A ppp-in -s 192,168,1,0/24 -l -j DENY
#
```

DNS(모든 요청을 '203,29,16,1'으로 전달하기 때문에 DNS TCP에 대해서는 응답을 허가한다), ftp, 반환하는(return) ftp-data 접속을 허가한다(return ftp-data란 1023 포트 위로 나가는 ftp-data를 말한다).

```
# ipchains -A ppp-in -p TCP -s 203,29,16,1 -d $LOCALIP dns -j ACCEPT
```

```
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 1024: -j
ACCEPT
# ipchains -A ppp-in -p TCP -d $LOCALIP ftp -j ACCEPT
```

마지막으로 그 밖에 지역적으로 생성된 패킷은 모두 허가한다.

```
# ipchains -A input -i lo -j ACCEPT
```

'input' 사슬에 대한 기본 정책은 DENY이다. 따라서 명시된 것 이외의 모든 것은 거부한다.

```
# ipchains -P input DENY
```

실제로 이 순서대로 사슬 설정을 하지는 않는다. 왜냐하면 설정 중에 패킷들이 지나갈 수 있기 때문이다. 가장 안전한 방법은 우선 정책을 DENY로 한 후 규칙을 삽입해나가는 것이다. 당연히 규칙에서 DNS 찾아보기를 필요로 하는 경우에는 문제가 된다.

11 'ipchains-save' 사용하기

여러분이 원하는 대로 방화벽 사슬을 설정해 놓은 후, 다음 번에 어떻게 했는지 기억하며 끙끙거리는 일은 정말 고통스럽다.

바로 이 문제를 위해 'ipchains-save'라는 스크립트가 현재의 사슬 설정을 읽어들여 파일로 저장하는 일을 해준다. 당분간은 'ipchains-restore'가 어떤 일을 하는지에 대해서만 말해두겠다.

```
$ ipchains-save > my_firewall
Saving 'input'.
Saving 'output'.
Saving 'forward'.
Saving 'ppp-in'.


Saving 'ppp-out'.
$
```

12 'ipchains-restore' 사용하기

`ipchains-save'를 사용하여 저장한 사슬을 복구하는 스크립트가 `ipchains-restore' 이다. 두 개의 옵션을 가지고 있다. `-v' 는 어떤 규칙이 추가되고 있는지 보여준다. `-f' 옵션은 앞서 설명한 대로 사슬이 존재하는 경우 사용자 정의 사슬의 내용을 일단 비우도록 한다.

만약 스크립트의 입력 내용에 사용자 정의 사슬이 있다면 `ipchains-restore'는 사슬이 이미 존재하고 있는지 점검한다. 존재하는 경우 일단 사슬의 내용을 모두 비울 것인지(규칙을 모두 지우는 일) 아니면 그냥 이 부분은 아무런 일도 하지 않고 넘어갈 것인지 질문을 받는다. `-f' 옵션을 명령행에 주면 질문 과정은 없다. 무조건 사슬의 내용이 일단 비워진다.

스크립트를 실행하기 위해서는 root 여야 한다; 규칙을 복구시켜놓기 위해 `ipchains'를 사용하기 때문이다.

예를 보자.

```
# ipchains-restore < my_firewall
Restoring `input'.
Restoring `output'.
Restoring `forward'.
Restoring `ppp-in'.
Chain `ppp-in' already exists. Skip or flush? [S/f]? s
Skipping `ppp-in'.
Restoring `ppp-out'.
Chain `ppp-out' already exists. Skip or flush? [S/f]? f
Flushing `ppp-out'.
#
```

III IP 매스커레이딩

1. 매스커레이딩이 왜 필요한가?

다음과 같은 상황을 생각해 보자. 지금 현재 여러분의 사무실에 3대의 컴퓨터가 있다. 그리고 그 컴퓨터는 모두 이더넷 카드로 연결되어 서로 통신이 가능하다. 그런데 사무실에 할당된 공적 IP 주소는 하나 밖에 없다. 가장 성능이 좋은 컴퓨터에 이 공적 IP를 할당하고 외부와 연결되어 있다. 다른 나머지 2대의 PC는 외부와 연결되어 있지 않다. 세대의 PC의 내부적인 주소는 192.168.1.1부터 192.168.1.3까지라고 가정한다.

공적 IP 주소를 210.216.30.111이라고 하자. 외부와의 연결은 이 IP 주소를 가진 컴퓨터에서만 가능하다. 이 PC에서는 웹/텔넷/FTP 등이 가능하다. 즉 인터넷 사용이 가능하다.

이런 시나리오를 생각해 볼 수 있다. 공적인 주소 210.216.30.111를 가진 컴퓨터, 즉 내부 주소로는 192.168.1.1인 컴퓨터 뿐만 아니라 나머지 다른 2대의 PC도 인터넷을 사용할 수 없을까? 외부 세계의 어느 사이트에서 나머지 다른 2대의 PC에 전자메일을 보낼 순 없을까? 해결방안을 곧 제시하겠지만, 지금 상황에서는 불가능하다. 왜냐하면 공적인 주소를 갖고 있지 않으므로 외부세계에서는 패킷을 어디로 보내야 할 지를 알 도리가 없다.

2 해결책

해결책을 제시한다. 외부와 연결되어 있고 210.216.30.111라는 IP 주소를 가진 컴퓨터에서는 패킷이 들어오고 나가는 것이 가능하다. 공적 주소를 가지고 있기 때문이다. 그러면 나머지 두 대의 PC도 나가고 들어오게 하려면? 210.216.30.111를 통해서 나가고 들어오게 하면 된다. 즉 나머지 두 대의 PC에서 나가는 패킷이 마치 210.216.30.111을 통해서 나가는 것처럼 가장하면 될 것이다. 들어오는 것에 대해서는 내부의 어느 PC로 가야 할지 결정하여 그에 맞는 컴퓨터로 보내 주면 된다.

리눅스에서는 이것에 대한 해결책을 제시하고 있다. 외부와의 연결 통로에 존재하는 컴퓨터에 리눅스를 설치하고 IP 매스커레이딩 기능을 넣고 적절히 세팅해주면 끝이다.

3 서버의 준비사항

· 방화벽 역할을 하는 리눅스 머신의 커널이 패킷을 전달하도록 해 준다. 기본은 전달 금지이다. 다음처럼 하면된다.

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

· 마스커레이딩 설정

ipchains이라는 명령어로 마스커레이딩 기능을 구현할 수 있다. 이것은 내부 네트웍에서 나가는 패킷을 마치 방화벽 주소 210.216.30.111에서 나가는 것처럼 속여 주고, 또한 외부에서 들어오는 패킷이 실제 내부 네트웍 중 어디로 가야 하는지 판별해 주는 역할을 한다. 다음 처럼 한다.

```
ipchains -F forward
ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
```

· 커널 모듈 적재

특정 서비스를 내부 네트웍에서 제대로 사용하기 위해서는 리눅스 머신의 /lib/modules/<커널버전>/ipv4 디렉토리에서 사용하고자 하는 서비스에 해당하는 모듈을 리눅스 머신에 적재해 주어야 한다. ip_masq로 시작하는 오브젝트 파일을 적재하면 된다. 예를 들어 ftp 서비스를 사용하려면 다음 명령으로 ip_masq_ftp.o라는 모듈을 적재한다.

```
# /sbin/insmod /lib/modules/<커널버전>/ipv4/ip_masq_ftp.o
```

4. 클라이언트의 준비사항

· /etc/resolv.conf에 서버 머신과 동일한 DNS 서버의 IP 주소를 적어 준다.

· 리눅스 방화벽 컴퓨터와 연결된 컴퓨터들은 모두 방화벽을 거쳐서 나가므로 기본 게이트웨이를 방화벽이 설치된 컴퓨터로 설정한다.

```
route add default gw 192.168.1.1
```

이제 사무실의 모든 컴퓨터가 자유롭게 인터넷을 사용할 수 있다. 인터넷으로 나가고 인터넷에서 들어오는 모든 패킷은 192.168.1.1의 내부 주소를 갖는 리눅스 머신을 해서 나가고 들어온다. 또한, 리눅스 머신에는 엄청난 부하가 걸린다는 것을 명심하자.