한글 FreeBSD 핸드북

Korea FreeBSD Users Group – http://www.kr.FreeBSD.org/

Network Servers: 네트워크 서버

이 영 옥 <rick@rickinc.com> 역.

2004 년 8 월 7 일.
문서갱신번호 1.1

본 문서는 FreeBSD Documentation Project 에서 발행하는 FreeBSD Handbook 을 번역한 것으로 2004 년 8 월 7 일자 수정본을 기준으로 합니다. FreeBSD Handbook 은 지속적으로 갱신되기 때문에, 본 문서의 발간 이후에 갱신된 내용이 무엇인지 알고자 한다면, CVS Repository 를 조회하여 해당일 이후에 갱신된 내용만을 쉽게 찾아볼 수 있습니다. 또한 표지에 표시되는 날짜와 문서갱신번호는 이 문서의 갱신상태를 나타냅니다. 날짜는 번역의 대상이 되는 오리지날 핸드북의 최종 수정일자를 표시하며, 문서갱신번호는 문서에 수정이 있을 때 증가하므로 사용자가 이 문서의 갱신상태를 쉽게 파악할 수 있도록 합니다. 본 문서는 이영옥이 관리, 배포하므로 내용에 오탈자 및 오역이 발견될 경우엔 역자에게 알려주어 수정될 수 있도록 해 주십시오.

23 장 네트워크 서버

23.1 개요

이번 장은 유닉스 시스템에서 자주 사용되는 네트워크 서비스 중 몇 가지를 다룬다. 여러 가지 다른 종류의 네트워크 서비스를 어떻게 설치, 설정, 테스트하고 유지하는지 설명한다. 이번 장 전반에 설정 파일 예제도 포함되어 있다.

이번 장을 읽고 다음과 같은 사항을 알 수 있다:

- inetd 데몬은 어떻게 관리하는가
- 유저 계정을 공유하기 위한 네트워크 정보 서버는 어떻게 설치하는가
- DHCP를 사용하여 자동 네트워크 설정은 어떻게 하는가
- 도메인 네임서버는 어떻게 설치하는가
- 아파치 웹 서버는 어떻게 설치하는가
- 파일 전송 프로토콜(FTP) 서버는 어떻게 설치하는가
- 삼바(Samba)를 사용하여 윈도우 클라이언트를 위한 파일과 프린터 서버는 어떻게 설치하는가
- NTP 프로토콜로 시간과 날짜를 동기화 하고 타임 서버는 어떻게 설치하는가

이번 장을 읽기 전에 다음 사항을 알고 있어야 된다:

- /etc/rc 스크립트의 기본적인 이해
- 기본적인 네트워크 용어 이해
- 부가적인 소프트웨어 설치(4장)

23.2 inetd "슈퍼-서버"

23.2.1 개요

inetd(8)은 여러 서비스의 연결을 관리하기 때문에 "인터넷 슈퍼-서버"라고 한다. 네트워크 서비스를 제공하는 프로그램들은 일반적으로 데몬으로 알려져 있다. inetd 는 다른 데몬을

관리하는 서버다. inetd 로 연결이 요청되면 어떤 데몬으로 연결해야 되는지 결정해서 해당 데몬을 실행하고 소켓을 생성한다. stand-alone 모드에서 각 데몬을 개별적으로 운용하는 것과 비교하여 하나의 inetd 인스턴스로 시스템의 전체 로드를 줄인다.

주로 inetd 는 다른 데몬을 실행하는데 사용되지만 chargen, auth 와 daytime 과 같은 여러 개의 평범한 프로토콜은 직접 관리한다.

이 섹션은 inetd 의 기본 설정부터 명령어 라인 옵션과 설정 파일 /etc/inetd.conf 까지 다룬다.

23.2.2 설정

inetd 는 /etc/rc.conf 시스템으로 초기화된다. 기본적으로 *inetd_enable* 옵션은 "NO"로 설정되어 있지만 sysinstall 로 보안 프로필을 중간으로 설정하면 활성화된다.

/etc/rc.conf 에 다음과 같이 설정하면 부팅할 때 inetd 를 활성화하거나 비활성 한다.

```
inetd_enable="YES"
또는
inetd_enable="NO"
```

게다가 *inetd_flags* 옵션으로 다른 명령어라인 옵션도 inetd 에 적용할 수 있다.

23.2.3 명령어 라인 옵션

inetd 개요:

inetd [-d] [-l] [-w] [-W] [-c maximum] [-C rate] [-a address | hostname] [-p filename] [-R rate] [configuration file]

-d

디버깅 켜기.

-l

성공적인 연결을 로그로 남긴다.

-w

외부 서비스에 TCP Wrapping 켜기(기본적으로).

-W

inetd 에 내장된 내부 서비스에 TCP Wrapping 켜기(기본적으로).

-c maximum

기본적으로 동시에 실행되는 각 서비스의 최대 값을 지정한다; 기본값은 무제한. *max-child* 매개변수를 사용하여 서비스 별로 제한할 수 있다.

-C rate

1 분 동안 하나의 IP 주소로부터 요청될 수 있는 서비스의 기본 최대값을 지정; 기본값은 무제한. *max-connections-per-ip-per-minute* 매개변수를 사용하여 서비스 별로 제한할 수 있다.

-R rate

1 분 동안 서비스가 발생할 수 있는 최대값 지정; 기본값은 256. 값 0 은 무제한으로 허용한다.

-a

바인드 할 특정 IP 주소 하나를 지정한다. 그렇지 않고 IPv4 나 IPv6 주소가 사용되는 호스트 이름과 일치할 경우 호스트 이름을 사용할 수 있다. 보통 호스트 이름은 inetd 가 jail(8) 내부에서 운용될 때 지정하고 이 경우 호스트 이름은 jail(8) 환경과 일치한다.

호스트 이름을 지정하여 사용하고 IPv4 와 IPv6 바인딩이 둘 다 요구될 때, /etc/inetd.conf 의 각 서비스를 위해 각 바인딩의 적절한 프로토콜 타입 엔트리가 하나 필요하다. 예를 들어 TCP 기반 서비스는 *tcp4* 와 *tcp6* 프로토콜 엔트리가 하나씩 필요하다.

-p

프로세스 ID 를 저장할 파일지정

이들 옵션은 /etc/rc.conf 의 *inetd_flags* 옵션으로 inetd 에 적용할 수 있다. 기본적으로

inetd_flags 는 inetd 의 외부와 내부 서비스에 TCP Wrapping 을 작동시키는 "-wW"를 설정한다. 경험이 없는 유저들을 위해 이들 매개변수는 수정할 필요가 없거나 /etc/rc.conf 에 입력되어 있다.

> Note: 외부 서비스는 연결이 되면 실행되는 inetd 의 외부 데몬이다. 한편 내부 서비스는 inetd 가 자체적으로 처리한다.

23.2.4 inetd.conf

inetd 의 설정은 /etc/inetd.conf 파일로 제어된다.

/etc/inetd.conf 를 수정하면 다음과 같이 inetd 프로세스에게 HangUP 신호를 보내서 inetd 가 강재로 설정을 다시 읽도록 할 수 있다:

예제 23-1. inetd 에 HangUP 신호 보내기

```
# kill -HUP `cat /var/run/inetd.pid`
```

설정 파일의 각 라인은 각 데몬을 지정한다. 파일에서 주석은 "#" 표시가 앞에 있다. /etc/inetd.conf 의 포맷은 다음과 같다:

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute]]
user[:group][/login-class]
server-program
server-program-arguments
```

IPv4 를 사용하는 ftpd 데몬 예제는 다음과 같다:

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

service-name

이것은 특정 데몬의 서비스 이름이다. /etc/services 에 나열된 서비스와 일치해야

된다. 이것으로 inetd 가 어떤 포트를 준비해야 되는지 결정한다. 새로운 서비스를 생성한다면 /etc/services 에 먼저 등록해야 된다.

socket-type

stream, *dgram*, *raw* 또는 *seqpacket* 중 하나. *stream* 은 연결 기반의 TCP 데몬에만 사용되지만 *dgram* 은 UDP 전송 프로토콜을 이용하는 데몬에 사용된다.

protocol

다음 중 하나가 된다:

프로토콜	설명
tcp, tcp4	TCP IPv4
udp, udp4	UDP IPv4
tcp6 TCP	IPv6
udp6 UDP	IPv6
tcp46	TCP IPv4 와 v6
udp46	UDP IPv4 와 v6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute]]

wait/*nowait* 은 inetd 가 실행한 데몬이 자체적인 소켓을 제어할 수 있는지 없는지를 보여 준다. *dgram* 소켓 타입은 *wait* 옵션을 사용해야 되지만 보통 멀티 쓰레드인 stream 소켓 데몬은 *nowait* 을 사용해야 된다. *wait* 은 보통 하나의 데몬이 여러 소켓을 사용하지만 *nowait* 은 새로운 각 소켓에 자식 데몬을 생성한다.

inetd 가 생성하는 자식 데몬의 최대값은 *max-child* 옵션을 사용하여 설정할 수 있다. 특정 데몬의 인스턴스를 10 으로 제한한다면 *nowait* 뒤에 */10* 이 필요하다.

게다가 한 곳에서 특정 데몬으로의 최대 연결을 제한하는 다른 옵션을 max-child 에 줄 수 있다. 이것은 *max-connections-per-ip-per-minute* 로 제한한다. 이곳의 값 10 은 특정 IP 주소에서 특정 서비스로 분 당 연결 시도를 10 회로 제한한다. 이 방법은 고의적이거나 고의적이지 않은 리소스 소비와 머신에 대한 서비스 거부 공격을(DOS) 방지하는데 유용하다.

이 필드에서 *wait* 이나 *nowait* 은 필수다. *max-child* 와 *max-connections-per-ip-per-minute* 은 옵션이다.

max-child 와 *max-connections-per-ip-per-minute* 제한이 없는 스트림 타입 멀티 쓰레드 데몬은 간단히 *nowait* 이다.

최대 제한이 10 인 같은 데몬은: *nowait /10* 이다.

게다가 IP 주소별로 분당 연결을 20 개로 제한하고 최대 10 개의 자식 프로세스로 제한한 데몬은: *nowait/10/20* 이다.

이들 옵션은 **fingerd** 데몬의 기본 설정으로 다음과 같이 이용되고 있다:

```
finger stream   tcp        nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

user

user 는 특정 데몬을 운영하는 유저 이름이다. 대부분 데몬은 root 유저로 실행된다. 보안을 위해 몇몇 서버를 daemon 유저나 최소한의 권한을 가진 nobody 유저로 실행하는 것을 볼 수 있다.

server-program

연결 요청을 받았을 때 데몬을 실행하는 전체 경로다. 데몬이 **inetd** 에 의해 내부적으로 서비스된다면 *internal* 를 사용해야 한다.

server-program-arguments

이것은 데몬을 실행할 때 *argv[0]*으로 시작하는 인자를 지정하여 *server-program* 과 결합하여 작동한다. **mydaemon -d** 가 명령어 라인이라면 *mydaemon -d* 는 *server-program-arguments* 의 값이 된다. 역시 데몬이 내부 서비스라면 *internal* 을 여기에 사용한다.

23.2.5 보안

설치할 때 보안 프로필을 선택했느냐에 따라서 많은 **inetd** 데몬이 기본적으로 활성화될 것이다. 필요 없다면 특정 데몬은 비활성 한다. 데몬의 시작 부분에 "#"를 입력하고 hangup 신호를 inetd 에 보낸다. **fingerd** 와 같은 어떤 데몬은 공격에 대한 너무 많은

정보가 제공되었기 때문에 어떤 경우든 필요 없을 것이다.

어떤 데몬은 오랫동안 보안에 대해 신경 쓰지 않았거나 연결 시도에 대해 타임아웃이 없다. 따라서 공격자는 특정 데몬에 눈치채지 못하게 연결을 시도하여 공격해서 이용 가능한 자원을 소모할 수 있다. 특정 데몬에 *ip-per-munute* 와 *max-child* 한계를 두는 것은 좋은 생각이다.

기본적으로 TCP wrapping 은 켜 있다. inetd 로 운용되는 다양한 데몬에 TCP 제한을 적용하는 정보는 hosts_access(5) 매뉴얼 페이지를 참고한다.

23.2.6 내부 서비스 데몬들

daytime, time, echo, discard, chargen 와 auth 는 모두 inetd 의 내부 서비스를 제공 받는다.

auth 서비스는 네트워크 서비스 인증을 제공하고 특정 레벨까지 설정할 수 있다.

좀더 자세한 정보는 inetd(8) 매뉴얼 페이지를 참고한다.

23.3 NFS

FreeBSD 는 다양한 다른 파일시스템 중 NFS 로 유명한 네트워크 파일시스템을 지원한다. NFS 는 시스템이 디렉터리와 파일 시스템을 네트워크를 통하여 공유하도록 한다. NFS 로 유저와 프로그램은 대부분의 원격 시스템의 파일을 로컬 파일처럼 사용할 수 있다.

NFS 가 제공하는 중요한 장점 몇 가지는 다음과 같다:

- 일반적으로 사용되는 데이터를 머신 한대에 저장해서 네트워크로 접근할 수 있기 때문에 로컬 워크스테이션은 더 적은 디스크 공간을 사용한다.
- 네트워크에 있는 모든 머신에 유저의 홈 디렉터리를 생성할 필요 없다. NFS 서버에 홈 디렉터리를 설정해서 네트워크로 사용할 수 있다.
- 플로피 디스크, CDROM 드라이버와 ZIP 드라이브 같은 스토리지 장치는 다른

머신에서 네트워크로 사용할 수 있다. 네트워크를 통해 여러 개의 이동용 미디어 드라이브를 줄일 수 있다.

23.3.1 NFS 는 어떻게 동작하는가

NFS 는 최소한 두 개의 메인 파트로 구성된다: 서버와 하나 이상의 클라이언트. 클라이언트는 서버 머신에 저장되어 있는 데이터를 원격에서 접속한다. 이러한 기능이 정확이 작동하기 위해 몇 가지 프로세스를 설정하고 실행해야 된다:

> Note: FreeBSD 5.X 에서 portmap 유틸리티는 rpcbind 유틸리티로 대체되었다. 그래서 FreeBSD 5.X 유저는 다음 예제들의 모든 portmap 인스턴스를 rpcbind 로 모두 변경한다.

서버에서는 다음 데몬들이 동작해야 된다:

데몬	설 명
nfsd	NFS 클라이언트로부터의 NFS 서비스 요청을 처리하는 데몬
mountd	nfsd(8)이 요청해서 실행되는 NFS 마운트 데몬
portmap	portmapper 데몬은 NFS 서버가 사용하는 포트를 NFS 클라이언트가 감지하도록 한다.

클라이언트도 nfsiod 로 알려진 데몬을 실행할 수 있다. nfsiod 데몬은 NFS 서버의 요청을 서비스한다. 이것은 부가적이고 성능을 향상시키지만 일반적이고 정확한 운용에는 필요 없다. 더 많은 정보는 nfsiod(8) 매뉴얼 페이지를 본다.

23.3.2 NFS 설정

NFS 설정은 비교적 직선적이다. /etc/rc.conf 파일을 약간 수정해서 시스템이 시작할 때 필요한 사항을 모두 실행할 수 있다.

NFS 서버의 /etc/rc.conf 파일에 다음 옵션을 설정한다:

```
portmap_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

NFS 서버가 활성화되면 **mountd** 가 자동으로 실행된다.

클라이언트의 /etc/rc.conf 에는 다음 옵션이 필요하다:

```
nfs_client_enable="YES"
```

NFS 로 공유할 파일시스템을 /etc/exports 파일에 지정한다. /etc/exports 의 각 라인에 공유되는 파일시스템과 어떤 머신이 이 파일 시스템에 접근하는지 정의한다. 이렇게 해서 어떤 머신이 파일시스템에 접근하는지 접근 옵션도 지정할 수 있다. 이 파일에 사용할 수 있는 다양한 옵션이 있지만 여기서는 몇 가지만 언급한다. exports(5) 매뉴얼 페이지에서 다른 옵션을 발견할 수 있다.

여기 몇 개의 /etc/exports 엔트리 예제가 있다:

설정은 여러분의 환경과 네트워크 상황에 따라 다르겠지만 다음 예제는 파일시스템을 어떻게 공유하는지 보여준다. 예를 들어 서버와 같은 도메인(각각을 위한 도메인 이름의 부족으로)을 가지고 있거나 /etc/hosts 파일에 엔트리가 있는 3 대의 예제 머신에 /cdrom 디렉터리를 공유한다. *-ro* 플래그는 파일시스템을 읽기 전용으로 공유한다. 이 플래그 때문에 원격 시스템은 공유된 이 파일시스템의 어떤 것도 변경할 수 없다.

```
/cdrom -ro host1 host2 host3
```

다음 라인은 IP 주소로 3 대의 호스트에 /home 을 공유한다. DNS 서버 설정이 없는 사설 네트워크에 있다면 유용한 설정이다. 부가적으로 /etc/hosts 파일에 내부 호스트 이름을 설정할 수 있다; 더 많은 정보는 hosts(5)를 본다. *-alldirs* 플래그는 서브 디렉터리를 마운트 포인트로 사용할 수 있게 한다. 다시 말해서 서브 디렉터리는 마운트 할 수 없지만 클라이언트가 필요로 하는 디렉터리만 마운트 하도록 한다.

```
/home   -alldirs   10.0.0.2 10.0.0.3 10.0.0.4
```

다음 라인은 /a 를 공유하기 때문에 다른 도메인의 클라이언트 2 대가 파일시스템에 접근할

것이다. *-maproot=root* 플래그는 원격 시스템의 root 유저가 root 로 공유된 파일시스템에 적성을 할 수 있게 한다. *-maproot=root* 플래그가 없다면 유저가 원격 시스템의 root 권한을 가지고 있더라도 공유된 파일시스템을 수정할 수 없다.

```
/a   -maproot=root   host.example.com box.example.org
```

클라이언트가 공유된 파일시스템에 접근하려면 클라이언트는 퍼미션을 가지고 있어야 한다. /etc/exports 파일에 원하는 클라이언트를 지정한다.

/etc/exports 에서 각 라인은 파일시스템 하나를 호스트 하나에 공유하는 정보를 나타낸다. 원격 호스트는 파일시스템 별로 한번씩 지정할 수 있고 하나의 기본 엔트리를 가질 것이다. 예를 들어 /usr 이 하나의 파일시스템이라고 가정하면 다음의 /etc/exports 는 유효하지 않다:

```
/usr/src    client
/usr/ports client
```

하나의 파일시스템 /usr 이 같은 호스트 client 에 두 라인으로 공유되어 있다. 이 상황에 맞는 정확한 포맷은 다음과 같다:

```
/usr/src /usr/ports   client
```

파일시스템에 포함되어 있는 디렉터리들은 원하는 호스트에 공유하려면 모두 한 라인에 지정한다. 클라이언트가 지정되지 않은 라인은 싱글 호스트로 취급된다. 이것은 파일시스템 공유를 제한하지만 대부분의 사람들에게 문제는 없다.

다음 예제는 /usr 과 /exports 가 로컬 파일시스템인 유효한 공유 리스트다:

```
# src 와 ports 를 client01 과 client 02 에 공유하지만 client01 만 root 권한을 갖는다.
/usr/src /usr/ports -maproot=root      client01
/usr/src /usr/ports                    client02
# client 머신은 root 권한을 가지고 있어서 /export 의 어느 곳이든 마운트 할 수 있다.
# 모든 사람들이 /exports/obj 를 읽기 권한으로 마운트 할 수 있다.
/exports -alldirs -maproot=root        client01 client02
/exports/obj -ro
```

/etc/exports 를 수정할 때마다 mountd 를 재 시작해서 변경된 내용을 반영해야 된다. 이것은 mountd 프로세스에게 HUP 신호를 보내면 된다:

```
# kill -HUP `cat /var/run/mountd.pid`
```

그렇지 않고 재 부팅할 필요가 없더라도 재 부팅하게 되면 FreeBSD 의 모든 설정을 정확히 반영한다. 다음 명령을 root 에서 실행하면 모든 것이 시작된다.

NFS 서버에서 아래 명령을 실행한다:

```
# portmap
# nfsd -u -t -n 4
# mountd -r
```

NFS 클라이언트에서 다음 명령을 실행한다:

```
# nfsiod -n 4
```

이제 실재로 원격 파일시스템을 마운트 할 수 있도록 모든 준비가 되었다. 이 예제에서 서버 이름은 *server* 고 클라이언트의 이름은 *client* 다. 원격 파일시스템을 일시적으로 마운트 하거나 설정을 테스트만 하려면 클라이언트에서 root 로 다음 명령을 실행한다:

```
# mount server:/home /mnt
```

이 명령은 서버의 /home 디렉터리를 클라이언트의 /mnt 에 마운트 한다. 모든 설정이 정확하다면 클라이언트의 /mnt 에서 서버에 있는 모든 파일을 볼 수 있다.

컴퓨터가 부팅할 때마다 원격 파일시스템을 자동으로 마운트하려면 /etc/fstab 파일에 파일시스템을 추가한다. 여기 예제가 있다:

```
server:/home    /mnt    nfs rw  0   0
```

fstab(5) 매뉴얼 페이지에서 이용할 수 있는 모든 옵션을 찾을 수 있다.

23.3.3 실용적인 사용

NFS 는 실제로 많이 사용된다. 이들 중 매우 보편적인 것을 아래에 적었다:

- 몇 대의 머신에 CDROM을 공유하도록 설정하거나 다른 미디어(플로피나 테잎)를 이들 머신에 공유한다. 이 방법은 저렴하고 가끔 여러 대의 머신에 소프트웨어를 설치할 때 편리한 방법이다.

- 대형 네트워크에서 모든 유저의 홈 디렉터리를 저장할 중앙 NFS 서버를 설정하면 아주 편리할 것이다. 이런 홈 디렉터리는 네트워크로 공유할 수 있기 때문에 유저가 어떤 워크스테이션에 로그인 하더라도 항상 같은 홈 디렉터리를 사용할 수 있다.

- 몇 대의 머신에 /usr/ports/distfiles 디렉터리를 공유할 수 있다. 여러 대의 머신에서 포트를 설치해야 된다면 이 방법으로 각각의 머신에 소스를 다운로드 할 필요 없이 빠르게 설치할 수 있다.

23.3.4 amd 로 자동 마운트

amd(8)은(자동 마운트 데몬) 파일이나 디렉터리가 있는 파일시스템에 접근할 때마다 원격 파일시스템을 자동으로 마운트 한다. 파일시스템이 일정 시간 동안 사용되지 않으면 **amd** 는 자동으로 언 마운트 한다. 단순히 **amd** 의 기능을 사용하거나 영구적으로 마운트 하려면 보통 /etc/fstab 에 나열하면 된다.

adm 는 NFS 서버의 /host 와 /net 디렉터리에서 동작한다. 파일이 이들 디렉터리 중 한곳에 접근하면 **amd** 는 적절한 원격 마운트 위치를 찾아서 자동으로 마운트한다. /net 은 IP 주소로 공유 파일시스템을 마운트 하는데 사용되지만 /host 은 원격 호스트 이름으로 마운트할 때 사용된다.

/host/foobar/usr 에 있는 파일에 접근하려면 **amd** 에게 공유된 /usr 을 호스트 foobar 에 마운트 하도록 한다.

예제 23-2. amd 로 공유 디렉터리 마운트

showmount 명령으로 원격 호스트의 이용할 수 있는 마운트를 볼 수 있다. 예를 들어 호스트 이름 foobar 의 마운트를 보기 위해 다음 명령을 사용할 수 있다:

```
% showmount -e foobar
Exports list on foobar:
/usr                                10.10.10.0
/a                                  10.10.10.0
% cd /host/foobar/usr
```

예제에서 보았듯이 showmount 는 /usr 를 공유된 것으로 보여 준다. /host/foobar/usr 로 디렉터리를 변경하면 **amd** 는 호스트 이름 foobar 를 해석하고 자동으로 원하는 공유 디렉터리를 마운트 한다.

amd 는 다음 라인을 /etc/rc.conf 에 추가하여 시작 스크립트로 시작할 수 있다:

```
amd_enable="YES"
```

추가적으로 사용자 플래그를 *amd_flags* 옵션으로 amd 에 적용할 수 있다. 기본적으로 *amd_flags* 는 다음과 같이 설정된다:

```
amd_flags="-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map"
```

/etc/amd.map 파일은 공유 디렉터리가 마운트 될 때의 기본 옵션을 정의한다. /etc/mad.conf 파일은 **amd** 의 몇 가지 고급 기능을 정의한다.

더 많은 정보는 amd(8)과 amd.conf(5) 매뉴얼 페이지를 참고한다.

23.3.5 다른 시스템과 통합 문제

ISA PC 시스템의 특정 이더넷 카드는 특히 NFS 와 관련하여 심각한 네트워크 문제를 유발하는 한계를 가지고 있다. 애매하게도 FreeBSD 에 명확히 나타나지 않지만 FreeBSD 시스템도 영향을 받는다.

(FreeBSD)PC 시스템이 Silicon Graphics 나 Sun Microsystems 이 만든 고성능 워크스테이션과 네트워크로 연결되었을 때 이 문제가 거의 발생한다. NFS 마운트 동작은 양호하고 몇 가지 동작도 성공적이지만 다른 시스템이 계속해서 요청을 하더라도 갑자기 서버가 클라이언트에게 응답을 하지 않는다. 이 문제는 클라이언트가 FreeBSD

시스템이거나 워크스테이션일 때 클라이언트에 발생한다. 이 문제가 발생하면 클라이언트를 안전하게 셧 다운할 수 있는 방법이 없다. NFS 문제가 해결되지 않기 때문에 유일한 방법은 클라이언트를 재 부팅해야 된다. "정확한" 해결책은 FreeBSD 시스템에 고성능 이더넷 카드를 붙이면 되지만 이런 에러가 발생하지 않게 하는 간단한 조치가 있다. FreeBSD 시스템이 서버라면 클라이언트에서 마운트 할 때 *-w=1024* 옵션을 준다. FreeBSD 시스템이 클라이언트라면 NFS 파일 시스템을 옵션 *-r=1024*로 마운트 한다. 자동으로 마운트 하도록 이들 옵션을 클라이언트 fstab 의 4 번째 필드에 지정하여 사용하거나 직접 마운트 할 때 마운트 명령에 *-o* 매개변수를 사용한다.

NFS 서버와 클라이언트가 다른 네트워크에 있을 때 다른 문제가 있다. 이런 경우라면 필요한 UDP 정보를 라우터가 라우팅 해야되고 그렇지 않으면 사용하지 못한다.

다음 예제에서 fastws 은 고성능 워크스테이션 호스트(인터페이스) 이름이고 freebox 는 저급 이더넷 카드를 가진 FreeBSD 시스템의 호스트(인터페이스) 이름이다. 또한 /sharedfs 는 NFS 파일시스템(exports(5)를 본다)으로 공유되고 /project 는 클라이언트에서 공유 디렉터리를 마운트하는 마운트 포인트이다. 이와 같은 경우 *hard*나 *soft* 그리고 *bg*와 같은 추가적인 옵션을 어플리케이션이 요구할 것이다.

FreeBSD 시스템이(freebox) 클라이언트인 freebox 의 /etc/fstab 예제는 다음과 같다:

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

freebox 에서 직접 마운트 명령을 사용하려면 다음 명령을 실행한다:

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

FreeBSD 시스템이 서버인 fastws 의 /etc/fstab 의 예제는 아래와 같다:

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

fastws 에서 직접 마운트한다면 다음 명령을 사용한다:

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

거의 모든 16 비트 이더넷 카드는 읽기와 쓰기 크기에 위의 제한이 없이 동작된다.

문제가 발생하면 어떤 일이 발생하고 왜 해결할 수 없는지 설명하면, NFS 는 일반적으로 8k 의 "블록" 크기로 동작한다(더 작은 크기의 조각으로 동작할 수 있지만). 최대 이더넷 패킷이 대략 1500 바이트기 때문에 NFS "블록"은 상위 계층 코드에서 하나의 유닛이 되지만, 여러 개의 이더넷 패킷으로 나누어서 받은 후 재 조합해서 일반적으로 인정되는 유닛이 된다. 고성능 워크스테이션은 NFS 유닛을 구성하는 패킷을 최대한 표준에 맞추어 패킷 간격을 조밀하게 하나씩 내보낸다. 저급 카드가 있는 호스트에 이들 패킷이 전송된 후 모든 패킷이 다시 조합되어 인정되기 전에 나중에 보낸 패킷이 같은 유닛의 이전 패킷을 오버런한다. 이 결과 워크스테이션은 타임아웃 되어 다시 시도하지만 8K 유닛 전체를 다시 시도하게 되고 이 절차는 계속 반복된다.

유닛 크기를 이더넷 패킷 크기 이하로 제한하여 완전히 받은 이더넷 페킷이 교착 상태를 피하고 각각 승인될 수 있다.

오버런은 고성능 워크스테이션이 PC 시스템으로 데이터를 보내면 계속 발생할 수 있지만, 더 좋은 카드에서도 NFS 는 이런 오버런을 보장하지 않는다. 오버런이 발생하면 영향을 받은 유닛은 재 전송되어 받은 후 재 조합해서 인정된다.

23.4 NIS/YP

23.4.1 NIS 는 무엇인가?

네트워크 정보 서비스를 나타내는 NIS 는 유닉스(원래 SunOS) 시스템을 중앙에서 관리하기 위해 Sun Microsystems 에서 개발하였다. 이제 산업 표준이 되어 주요 주요 유닉스 시스템들이 NIS 를 지원한다(Solaris, HP-UX, AIX, Linux, NetBSD, OpenBSD, FreeBSD 등).

NIS 는 원래 옐로우 페이지로 알려졌지만 상표 문제로 Sun 이 이름을 바꾸었다. 이 예전 용어는(yp) 아직도 종종 사용되고 있다.

NIS 는 NIS 도메인의 머신들이 일반적인 설정 파일을 공유하도록 하는 RPC 기반 클라이언트/서버 시스템이다. 따라서 시스템 관리자는 최소한의 데이터 설정으로 NIS 클라이언트 시스템을 설정하고 한 곳에서 설정 데이터를 추가, 삭제 수정할 수 있다.

이것은 Windows NT 도메인 시스템과 비슷하다; 내부적인 수행은 다르지만 기본 기능은

동일하다.

23.4.2 알고 있어야 할 용어/프로세스

FreeBSD 를 NIS 서버나 NIS 클라이언트로 만들 때 수행해야 되는 여러 가지 중요한 유저 프로세스와 용어가 있다:

용어	설명
NIS 도메인 이름	NIS 마스터 서버와 모든 클라이언트는(슬레이브 서버를 포함하여) NIS 도메인 이름을 가지고 있다. Windows NT 도메인 이름과 비슷하고 NIS 도메인 이름은 DNS 와 어떤 연관도 없다.
portmap	RPC(NIS 가 사용하는 네트워크 프로토콜인 원격 프로시져 콜)를 활성화하기 위해 사용해야 된다. portmap 이 실행 중이지 않으면 NIS 서버를 실행할 수 없거나 NIS 클라이언트로 사용할 수 없다.
ypbind	NIS 클라이언트를 NIS 서버에 연결시킨다. ypbind 는 시스템에서 NIS 도메인 이름을 받아서 RPC 를 사용하여 서버에 연결한다. 그리고 ypbind 는 NIS 환경에서 클라이언트-서버 통신의 핵심이다; 클라이언트 머신에서 ypbind 가 정지했다면 NIS 서버에 접근할 수 없다.
ypserv	오직 NIS 서버에서만 실행된다; 이것이 NIS 서버 프로세스다. ypserv(8)이 죽는다면 서버는 더 이상 NIS 요청에 응답할 수 없다(다행히 이 장애를 극복할 수 있는 슬레이브 서버가 있다). 지금까지 사용하던 서버가 죽었다면 다른 서버로 다시 연결하지 않는 NIS 의 몇 가지 문제가 있다. 가끔 이 문제를 해결할 수 있는 유일한 방법은 서버 프로세스(또는 서버 전체를) 또는 클라이언트에서 ypbind 프로세스를 재 시작하는 것이다.
rpc.yppasswdd	NIS 마스터 서버에서만 실행해야 되는 다른 프로세스; NIS 클라이언트에서 유저가 NIS 패스워드를 변경할 수 있게 하는 데몬이다. 이 데몬이 실행되지 않는다면 유저는 NIS 마스터 서버에 로그인하여 클라이언트의 패스워드를 변경해야 된다.

23.4.3 어떻게 동작하는가?

NIS 환경에 3 종류의 호스트가 있다: 마스터 서버, 슬레이브 서버와 클라이언트. 서버는 호스트 설정 정보를 중앙에 저장하는 저장소처럼 동작한다. 마스터 서버는 신뢰할 수

있도록 이 정보를 가지고 있지만 슬레이브 서버는 이 정보를 여분으로 미러한다. 클라이언트는 서버로부터 이 정보를 제공받아 동작한다.

이 방법을 이용하여 다양한 파일 정보를 공유할 수 있다. master.passwd, group 과 hosts 파일이 일반적으로 NIS 를 통해 공유된다. 보통 이런 파일에서 찾을 수 있는 정보가 필요할 때마다 클라이언트는 NIS 서버에 요청하지 않고 직접 바운드(연결)한다.

23.4.3.1 머신 종류

- *NIS 마스터 서버*. Windows NT 주 도메인 컨트롤러와 유사한 이 서버는 모든 NIS 클라이언트가 사용하는 파일을 관리한다. 마스터 서버에 있는 passwd, group 그리고 다양한 다른 파일은 NIS 클라이언트가 사용한다.

  Note: 머신 한대가 하나 이상의 NIS 도메인의 마스터 서버가 될 수 있다. 그러나 여기서는 소규모의 NIS 환경을 다루기 때문에 이 교제에서는 다루지 않는다.

- *NIS 슬레이브 서버.* NT 백업 도메인 컨트롤러와 비슷한 NIS 슬레이브 서버는 NIS 마스터의 데이터 파일 복사본을 관리한다. NIS 슬레이브 서버는 중요한 환경을 백업으로 제공하고 마스터 서버의 로드를 분산하는데 도움이 된다: NIS 클라이언트는 항상 처음으로 응답하는 NIS 서버에 연결되고 여기에 슬레이브 서버의 응답도 포함된다.

- *NIS 클라이언트.* 대부분의 Windows NT 워크스테이션처럼 NIS 클라이언트도 로그인 하기 위해 NIS 서버 인증이 필요하다(또는 Windows NT 워크스테이션의 경우에서는 Windows NT 도메인 컨트롤러의 인증).

23.4.4 NIS/YP 사용

이 섹션은 샘플 NIS 환경 설정을 설명한다.

Note: 이번 섹션은 FreeBSD 3.3 또는 이후의 버전을 사용한다고 가정한다. 여기서 설명하는 내용은 3.0 이후의 FreeBSD 버전에서 동작하겠지만 확실하지는 않다.

23.4.4.1 계획

여러분이 대학의 작은 연구실의 관리자라고 가정한다. 15 대의 FreeBSD 머신으로 이루어진 이 연구실은 현재 중앙 관리가 되지 않고 있다; 각 머신은 각자의 /etc/passwd 와 /etc/master.passwd 를 가지고 있다. 이 파일들은 수동으로 서로 동기화한다; 현재 연구실에 유저를 추가하려면 15 대의 머신에서 adduser 를 실행해야 된다. 이렇게 변경해야 되기 때문에 연구실의 머신 두 대를 서버로 사용하여 NIS 를 도입하기로 결정했다.

따라서 연구실의 설정은 다음과 비슷하다:

머신 이름	IP 주소	머신 역할
ellington	10.0.0.2	NIS 마스터
coltrane	10.0.0.3	NIS 슬레이브
basie	10.0.0.4	교원전용 워크스테이션
bird	10.0.0.5	클라이언트 머신
cli[1-11]	10.0.0.[6-17]	다른 클라이언트 머신들

처음으로 NIS 를 설정할 계획이라면 어떻게 진행할 것인지 생각해 보는 것이 좋다. NIS 를 생성하기 위해 필요한 사항이 얼마 되지 않기 때문에 네트워크 크기에 대해서는 걱정하지 않는다.

[NIS 환경을 구축할 때 주의 사항]

1. NIS 도메인 네임 선택

이 이름은 사용하려는 “도메인 네임”이 아니다. 엄밀히 말해서 "NIS 도메인 네임" 이다. 클라이언트가 서버에게 정보를 요청하기 위해 브로드케스트할 때 자신이 속하는 NIS 도메인 네임도 포함되어 있다. 따라서 하나의 네트워크에 여러 대의 서버가 있을 때 어떤 서버가 요청을 처리할지 결정할 수 있다. 관련이 있는 호스트를 그룹화 할 수 있는 이름을 NIS 도메인네임으로 생각해본다.

어떤 단체들은 NIS 도메인 이름으로 인터넷 도메인을 선택한다. 네트워크 문제를 디버깅할 때 혼란스러울 수 있기 때문에 권장하지 않는다. NIS 도메인 이름은 여러분의 네트워크에서 유일해야 되고 머신 그룹을 설명하는 이름이 유용하다. 예를 들어 학교에서 예술부는 "acme-art" NIS 도메인 일 것이다. 이 예제에서는 *test-domain* 을 선택했다고 가정한다.

그러나 어떤 운영체제(특히 SunOS)는 NIS 도메인 이름을 인터넷 도메인 이름처럼

사용한다. 네트워크에서 하나 이상의 머신이 이러한 제한을 가지고 있다면 NIS 도메인 이름으로 인터넷 도메인 이름(DNS)을 사용해야 된다.

2. 물리적인 서버 요구 사항

NIS 서버로 사용할 머신을 선택할 때 몇 가지 유의사항이 있다. NIS 대한 한가지 불행한 것은 서버에 관한 클라이언트의 의존성 레벨이다. 클라이언트가 NIS 도메인 서버에 연결할 수 없다면 가끔 머신을 사용할 수 없다. 유저와 그룹 정보의 부족으로 대부분의 시스템이 일시적으로 정지된다. 그래서 자주 재 부팅 하거나 개발용으로 사용하던 머신은 선택에서 배제해야 된다. 트래픽이 아주 심하지 않은 네트워크라면 다른 서비스에 사용하는 머신에도 NIS 서버를 올릴 수 있지만 NIS 서버를 사용할 수 없게 되면 모든 NIS 클라이언트도 사용하지 못한다는 것을 생각해야 된다.

23.4.4.2 NIS 서버

모든 NIS 정보는 규칙적으로 복사되어 NIS 마스터 서버라고 부르는 하나의 머신에 저장된다. 정보를 저장하는데 사용되는 데이터베이스는 NIS 맵이라고 한다. FreeBSD 에서 이들 맵은 /var/yp[domainname]에 저장되고 [domainname]은 NIS 도메인 이름이다. 하나의 NIS 서버에서 여러 개의 도메인을 한번에 지원할 수 있기 때문에 지원되는 각 도메인 별로 디렉터리를 가질 수 있다. 따라서 각 도메인은 자신만의 독립적인 맵 세트를 가진다.

NIS 마스터와 슬레이브 서버는 ypserv 데몬으로 모든 NIS 요청을 제어한다. ypserv 는 NIS 클라이언트의 요청에 응답하고, 요청된 도메인과 맵 이름을 적절한 데이터베이스 파일 경로로 해석하여 데이터베이스에서 클라이언트로 데이터를 다시 돌려보낸다.

[NIS 서버 설정]

1. NIS 마스터 서버 설정

NIS 마스터 서버를 설정하는 것은 상당히 직관적이고 목적에 따라 다르다. FreeBSD 는 NIS 의 모든 것을 지원한다. 필요한 모든 것은 /etc/rc.conf 에 다음 라인을 추가하면 나머지는 FreeBSD 가 처리한다.

① nisdomainname="test-domain"

이 라인은 네트워크 설정에 따라 NIS 도메인 이름을 test-domain 으로 설정한다(재부팅 후).

② nis_server_enable="YES"

이 라인은 네트워크가 다시 로드 될 때 FreeBSD 가 NIS 서버 프로세스를 시작한다.

③ nis_yppasswdd_enable="YES"

이것은 위에서 언급한, 클라이언트 머신에서 유저가 그들의 NIS 패스워드를 변경할 수 있게 하는 rpc.yppasswdd 데몬을 활성화한다.

Note: 여러분의 NIS 설정에 따라 더 많은 엔트리를 추가해야 된다. 아래의 NIS 클라이언트가 되기도 하는 NIS 서버에 대한 섹션에서 좀더 자세히 다룬다.

이제 슈퍼 유저에서 /etc/netstart 명령을 실행하면 된다. 이 명령은 /etc/rc.conf 에 지정된 값으로 필요한 모든 것을 설정한다.

2. NIS 맵 초기화

NIS 맵은 /var/yp 디렉터리에 저장된 데이터베이스 파일이다. 이들 파일은 /etc/master.passwd 파일을 제외하고 NIS 마스터의 /etc 디렉터리의 설정파일로부터 생성된다. root 와 다른 관리용 계정의 패스워드를 NIS 도메인에서 모든 서버로 전파시키기지 않는 것이 보안상 안전하므로 NIS 맵을 초기화하기 전에 다음 내용을 따라야 한다:

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
```

```
# vi master.passwd
```

NIS 클라이언트로 전파시키기를 원하지 않는 계정(예를 들어 root 와 UID 0(슈퍼 유저)인 다른 계정)과 모든 전체 시스템 계정(bin, tty, kemem, games 등)을 삭제한다.

> **Note:** /var/yp/master.passwd 는 그룹이나 다른 사람들이 읽지 못하도록(모드 600) chmod 명령을 사용한다.

끝나고 나면 NIS 맵을 초기화한다! FreeBSD 는 초기화하기 위해 ypinit 라는 스크립트를 가지고 있다(더 많은 정보는 매뉴얼 페이지를 참고한다). 이 스크립트는 대부분의 유닉스 운영체제에서 사용할 수 있지만 모든 유닉스에서 사용하지는 못한다. Digital 유닉스/Compaq Tru64 유닉스는 ypsetup 이라고 한다. NIS 마스터 맵을 생성하는 것이므로 ypinit 에 옵션 *-m* 을 붙인다. NIS 맵을 생성하기 위해 위의 단계를 이미 수행한 걸로 간주하고 다음을 실행한다:

```
ellington# ypinit -m test-domain
Server Type: MASTER Domain: test-domain
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
Please continue to add any slave servers, one per line. When you are
done with the list, type a <control D>.
master server   :  ellington
next host to add:  coltrane
next host to add:  ^D
The current list of NIS servers looks like this:
ellington
coltrane
Is this correct?  [y/n: y] y

[..output from map generation..]
```

```
NIS Map update completed.
ellington has been setup as an YP master server without any errors.
```

ypinit 는 /var/yp/Makefile.dist 로부터 /var/yp/Makefile 를 생성한다. 이 파일은 오직 FreeBSD 머신으로 싱글 NIS 서버 환경을 운용한다고 간주한다. *test-domain* 은 슬레이브 서버도 가지고 있기 때문에 /var/yp/Makefile 를 수정해야 된다:

```
ellington# vi /var/yp/Makefile
```

다음 라인이 주석 처리 되어 있지 않는다면 주석 처리해야 된다

```
NOPUSH = "True"
```

3. NIS 슬레이브 서버 설정

NIS 슬레이브 서버 설정은 마스터 서버 설정보다 간단하다. 슬레이브 서버에 로그인해서 이전처럼 /etc/rc.conf 를 수정한다. 유일한 한가지 차이점은 ypinit 를 실행할 때 *-s* 옵션을 사용하는 것이다. *-s* 옵션은 NIS 마스터의 이름이 필요하기 때문에 명령어 라인은 다음과 같다:

```
coltrane# ypinit -s ellington test-domain

Server Type: SLAVE Domain: test-domain Master: ellington

Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n]  n

Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
There will be no further questions. The remainder of the procedure
should take a few minutes, to copy the databases from ellington.
Transferring netgroup...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byuser...
```

```
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byhost...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
```

```
ypxfr: Exiting: Map successfully transferred

coltrane has been setup as an YP slave server without any errors.
Don't forget to update map ypservers on ellington.
```

이제 /var/yp/test-domain 이라는 디렉터리가 생성되었다. NIS 마스터 서버의 맵이 이 디렉터리에 복사되어 있다. 그리고 최근에 변경된 사항도 업데이트한다. 다음 라인들을 슬레이브 서버의 /etc/crontab 에 추가하여 항상 업데이트 할 수 있다:

```
20      *       *       *       *       root    /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root    /usr/libexec/ypxfr passwd.byuid
```

이 두 라인은 마스터 서버의 맵과 슬레이브 서버의 맵을 동기화한다. 패스워드 정보는 서버 시스템에 아주 중요하기 때문에 마스터 서버는 NIS 맵이 변경되었다면 슬레이브 서버와 통신을 시도한다. 이런 엔트리가 필수적이지 안지만 강제로 업데이트하는 것도 좋은 생각이다. 이 방법은 맵 업데이트가 항상 완벽히 수행되지 않는 사용량이 많은 네트워크에서 더욱 중요하다.

이제 슬레이브 서버에서 NIS 서버를 다시 시작하는 /etc/netstart 명령을 수행한다.

23.4.4.3 NIS 클라이언트

NIS 클라이언트는 ypbind 데몬을 사용하여 특정 NIS 서버와 바인딩(결합)이라는 연결을 한다. ypbind 는 시스템의 기본 도메인(domainname 명령으로 설정된)을 체크하고 로컬 네트워크에서 RPC 요청 브로드케스트를 시작한다. 이들 요청에는 ypbind 가 바인드 하려는 도메인 이름을 지정한다. 요청하고 있는 도메인을 지원하도록 설정된 서버가 브로드케스트 패킷을 수신하여 응답하면 ypbind 는 서버의 주소를 기록한다. 여러 대의 서버(예를 들어 하나의 마스터와 여러 대의 슬레이브)가 응답하면 ypbind 는 첫 번째 서버의 주소를 사용한다. 그래서 클라이언트는 모든 NIS 요청을 이 서버로 지정한다. ypbind 는 종종 서버가 운용 중인지 "ping"을 날려본다. 특정 시간 동안 ping 에 대한 응답이 없다면 ypbind 는 응답이 없음을 표시하고 다른 서버를 향해 브로드케스팅을 시작한다.

23.4.4.3.1 NIS 클라이언트 설정

FreeBSD 머신을 NIS 클라이언트로 설정하는 것은 아주 직관적이다.

① /etc/rc.conf에 NIS 도메인 이름을 설정하고 네트워크가 시작될 때 ypbind가 시작되도록 다음 라인을 추가한다:

```
nisdomainname="test-domain"
nis_client_enable="YES"
```

② NIS 서버에서 가능한 모든 패스워드 엔트리를 받아오기 위해 /etc/master.passwd 파일의 모든 유저를 삭제하고 vipw로 다음 라인을 파일의 마지막에 추가한다:

Note: 다음 라인으로 NIS 서버의 패스워드 맵에 계정이 있는 모든 사람들에게 권한이 주어진다. 다음 라인을 변경하여 NIS 클라이언트를 설정하는 여러 가지 방법이 있다. 더 자세한 정보는 아래의 netgroup 섹션을 본다. 더욱 자세한 사항은 O'Reilly 의 *Managing NFS and NIS* 를 본다.

```
+::::::::::
```

Note: /etc/master.passwd 에 최소한 한 개의 로컬 계정이 있고 이 계정은 wheel 그룹이어야 되다. NIS 에 문제가 있다면 이 계정으로 원격 머신에 로그인해서 root 가 된 후 수정한다.

③ NIS 서버에서 가능한 모든 그룹 엔트리를 받아오려면 /etc/group 파일에 다음 라인을 추가한다:

```
+:*::
```

이 단계를 끝낸 후 ypcat passwd 를 실행하여 NIS 서버의 passwd 맵을 참고할 수 있을 것이다.

23.4.5 NIS 보안

보통 원격 유저가 여러분의 도메인 이름을 알고 있다면 이 유저도 ypserv(8)에 RPC 를 사용하여 여러분의 NIS 맵을 갱신할 수 있다. 인증되지 않은 트랜젝션을 막기 위해

ypserv(8)은 지정된 호스트만 접근할 수 있도록 제한하는 securenets 기능을 지원한다. 시작할 때 ypserv(8)은 /var/yp/securenets 에서 securenets 정보를 읽는다.

> Note: 이 경로 변경은 *-p* 옵션으로 지정된 경로에 의존된다. 이 파일에는 네트워크 설정과 공백으로 나누어진 네트워크 마스크로 이루어진 엔트리가 포함되어 있다. "#"으로 시작하는 라인은 주석이고 샘플 securenets 파일은 다음과 비슷할 것이다:

```
# allow connections from local host -- mandatory
127.0.0.1       255.255.255.255
# allow connections from any host
# on the 192.168.128.0 network
192.168.128.0 255.255.255.0
# allow connections from any host
# between 10.0.0.0 to 10.0.15.255
# this includes the machines in the testlab
10.0.0.0        255.255.240.0
```

이 룰에 맞는 주소로부터 요청을 받는다면 ypserv(8)는 보통 요청을 처리한다. 주소가 룰에 맞지 않는다면 이 요청은 무시되고 경고 메시지를 로그로 남긴다. /var/yp/securenets 파일이 없으면 ypserv 는 어떤 호스트의 연결이든 허용한다.

그리고 ypserv 프로그램은 Wietse Venema 의 tcpwrapper 패키지를 지원한다. 따라서 관리자는 접근 제어에 /var/yp/securenets 대신 tcpwrapper 설정 파일을 사용할 수 있다.

> Note: 이 두 가지의 접근 제어 메커니즘은 이미 지정된 포트 테스트와 같은 몇 가지 보안을 제공하지만 "IP spoofing" 공격에 취약하다. 모든 NIS 관련 트래픽은 방화벽에서 막아야 한다.
>
> /var/yp/securenets 를 사용하는 서버는 오래된 TCP/IP 를 사용하는 정당한 NIS 클라이언트를 지원하지 못할 것이다. 이들 중 어떤 것은 브로드케스트 하거나 브로드케스트 주소를 계산할 때 서브 넷 마스크 감지에 실패하면 모든 호스트 비트를 0 으로 만든다. 이 문제 중 몇 가지는 클라이언트 설정을 변경해서 수정할 수 있지만 다른 문제는 클라이언트 시스템을 사용하지 못하거나 /var/yp/securenets 를 포기해야 될 것이다.

서버에서 오래된 TCP/IP 로 /var/ypsecurenets 를 사용하는 것은 아주 적절하지 못하고 거대한 네트워크에서 NIS 기능을 상실하는 원인이 된다.

tcpwrapper 패키지로 NIS 서버를 숨길 수 있다. 그러나 클라이언트 프로그램이 타임아웃이 걸릴 만큼 지연될 수 있고 특히 사용량이 많은 네트워크나 느린 NIS 서버에서 심각하다. 하나 이상의 클라이언트 시스템이 이런 증상을 가지고 있다면 클라이언트 시스템이 NIS 슬레이브 서버로 쿼리를 보내도록 바꾸고 이들을 강제로 바인드 한다.

23.4.6 로그인에서 특정 유저 제외시키기

연구실에 basie 라는 교원 전용의 워크스테이션 있다고 가정한다. 이 머신을 NIS 도메인에서 제외할 계획은 없고 마스터 NIS 서버의 passwd 파일에 교워와 학생의 계정이 모두 있다. 어떻게 하면 좋을 것인가?

NIS 데이터베이스에 계정이 있더라도 머신에 로그인할 수 없게 하는 방법이 있다. 이 방법은 클라이언트 머신의 /etc/master.passwd 파일 끝에 로그인을 허용하지 않는 유저의 이름 *-username* 을 추가한다. vipw 는 /etc/master.passwd 변경을 체크해주고 수정이 끝나면 자동으로 패스워드 데이터베이스를 다시 빌드하므로 vipw 를 사용하는 것이 좋다. 예를 들어 basie 에 로그인하는 유저 bill 이 로그인하지 못하게 하려면 다음 절차를 따른다:

```
basie# vipw
[add -bill to the end, exit]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[password]:0:0::0:0:The super-user:/root:/bin/csh
toor:[password]:0:0::0:0:The other super-user:/root:/bin/sh
daemon:*:1:1::0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5::0:0:System &:/:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,,:/:/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/:/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/:/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8::0:0:News Subsystem:/:/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/share/man:/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/:/sbin/nologin
uucp:*:66:66::0:0:UUCP pseudo-
user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67::0:0:X-10 daemon:/usr/local/xten:/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/sbin/nologin
+:::::::::
-bill

basie#
```

23.4.7 넷 그룹(Net Group) 사용

매우 적은 수의 유저나 머신에 특별한 룰이 필요하다면 이전 섹션에서 보여 준 방법으로 정확히 작동할 것이다. 거대한 네트워크에서 민감한 머신에 로그인하는 특정 유저를 거부하도록 설정하는 것을 잊어버리거나 각 머신을 따로 수정해야 된다면 NIS 의 최대 장점인 중앙 관리를 못하게 되는 것이다.

NIS 개발자들은 이 문제를 *넷 그룹*이라고 부르는 방법으로 해결하였다. 이들의 목적과 의미는 유닉스 파일시스템이 사용하는 일반 그룹과 비교할 수 있다. 가장 큰 차이점은 숫자 id 가 필요 없고 유저 계정과 다른 넷 그룹 포함하여 넷 그룹을 정의할 수 있다.

넷 그룹은 수백 명의 유저와 머신으로 거대하고 복잡한 네트워크를 제어하기 위해 개발되었다. 따라서 이런 상황이라면 좋은 해결책이 된다. 그러나 다른 한편으로 이 복잡성은 아주 간단한 예제로 넷 그룹을 설명하는 것을 거의 불가능하게 만든다. 이 섹션의 나머지에 사용되는 예제로 이 문제를 설명한다.

여러분이 NIS 를 완벽하게 설명하여 다수의 사람들이 관심을 보였다고 가정한다. 다음에 할 작업은 캠퍼스의 다른 머신을 커버하도록 NIS 도메인을 확장하는 것이다. 다음 두 개의 테이블은 새로운 유저와 머신 이름을 포함하여 상황을 간략한 소개한다.

유저 이름	설명
alpha, beta	IT 학과의 일반적인 직원들
charli, delta	IT 학과의 새로운 견습생
echo, foxtrott, golf,..	일반적인 직원들
able, baker,....	현재 인턴들

머신 이름	설명
war, death, famine, pollution	가장 중요한 서버들. IT 직원들만 이들 머신에 로그인할 수 있다.
pride, greed, envy, wrath, lust, sloth	덜 중요한 서버들. IT 학과의 전원이 이들 머신에 로그인할 수 있다.
one, two, three, four,..	일반적인 워크스테이션. 오직 실제 직원들만 이들 머신에 로그인할 수 있다.
trashcan	중요한 데이터가 없는 아주 오래된 머신. 인턴들도 이 머신들에 로그인할 수 있다.

각 유저 별로 따로 제한한다면, 시스템에 로그인을 허용하지 않는 유저들은 각 시스템의 passwd 에 *-user* 로 추가한다. 하나의 엔트리라도 빼먹는다면 문제가 발생할 수 있다. 처음 설정할 때는 정확히 설정할 수 있겠지만 매일 매일 운용 중 새로운 유저를 위한 라인을 빼먹을 수 있다.

넷 그룹으로 이 상황을 제어하면 여러 가지 이점을 제공한다. 각 유저를 따로 제어할

필요가 없다; 유저를 하나 이상의 넷 그룹에 할당하고 넷 그룹 맴버 모두를 로그인 허용 또는 금지할 수 있다. 새로운 머신을 추가했다면 넷 그룹 로그인 제한만 정의하면 된다. 새로운 유저를 추가했다면 유저를 하나 이상의 넷 그룹에 추가하면 된다. 이러한 변경은 각자에게 독립적이다; 더 이상 유저와 머신이 연관되지 않는다." 여러분의 NIS 설정이 세심하게 계획되었다면 머신에 접근을 허용하거나 거부하기 위해 하나의 중앙 설정파일만 수정하면 된다.

첫 번째 단계는 NIS 맵 넷 그룹을 초기화한다. FreeBSD 의 ypinit(8)가 기본적으로 이 맵을 생성하지 않지만 맵이 생성된 후 ypinit 의 NIS 는 맵을 지원한다. 빈 맵을 생성하려면 단순히 다음 명령을 입력한다.

ellington# **vi /var/yp/netgroup**

그리고 목록을 추가한다. 우리의 예제에는 최소한 4 개의 넷 그룹이 필요하다: IT 직원, IT 견습생, 일반 근로자와 인턴들.

```
IT_EMP  (,alpha,test-domain)     (,beta,test-domain)
IT_APP  (,charlie,test-domain)  (,delta,test-domain)
USERS   (,echo,test-domain)      (,foxtrott,test-domain) ₩
        (,golf,test-domain)
INTERNS (,able,test-domain)      (,baker,test-domain)
```

IT_EMP, *IT_APP* 등은 넷 그룹의 이름이다. 각 중괄호 둘러싸서 하나 이상의 유저 계정을 그룹에 추가한다. 그룹 내의 3 개의 필드는 다음과 같다:

① 다음 항목이 유효한 호스트의 이름. 호스트 이름을 지정하지 않았다면 이 엔트리는 모든 호스트에 유효하다. 호스트 이름을 지정했다면 어둠과 공포 그리고 완벽한 혼돈의 세계로 들어가는 것이다.

② 이 넷 그룹에 포함된 계정 이름.

③ 계정의 NIS 도메인. 여러분이 하나 이상의 NIS 도메인으로 불행한 사람이라면 다른 NIS 도메인에서 여러분의 넷 그룹으로 계정을 받아(import)올 수 있다.

각각의 이들 필드는 와일드카드(*)를 포함할 수 있다. 자세한 사항은 netgroup(5)를 본다.

Note: NIS 도메인에 다른 운영체제도 같이 사용하고 있다면 넷 그룹 이름에 8 개 이상의 문자를 사용하지 않는다. 이름은 대 소문자를 구분한다; 넷 그룹 이름에 대문자를 사용하면 유저와 머신 그리고 넷 그룹 이름을 구분하기 쉽다.

어떤 NIS 클라이언트(FreeBSD 가 아닌)는 수많은 엔트리의 넷 그룹을 제어할 수 없다. 예를 들어 넷 그룹이 15 개 이상의 엔트리를 가지고 있으면 오래된 버전의 어떤 SunOS 가 문제를 유발한다. 15 유저 이하의 서브 넷 그룹 여러 개를 생성하여 이런 제한을 피할 수 있고 실제 넷 그룹이 서브 넷 그룹을 포함한다:

```
BIGGRP1  (,joe1,domain)  (,joe2,domain)  (,joe3,domain) [...]
BIGGRP2  (,joe16,domain)  (,joe17,domain) [...]
BIGGRP3  (,joe31,domain)  (,joe32,domain)
BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
```

넷 그룹 하나에 225 이상의 유저가 필요하면 이 과정을 반복한다.

새로운 NIS 맵을 활성화하고 배포하는 것은 쉽다:

```
ellington# cd /var/yp
ellington# make
```

다음 명령은 3 개의 NIS 맵 netgroup, netgroup.byhost 와 netgroup.byuser 를 생성한다. 새로운 NIS 맵을 이용할 수 있는지 ypcat(1)로 체크한다:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

첫 번째 명령의 결과는 /var/yp/netgroup 의 내용과 비슷해야 된다. 두 번째 명령은 호스트에 특정 넷 그룹을 지정하지 않았다면 출력 결과를 생성하지 않는다. 세 번째 명령은 넷 그룹의 유저 리스트를 가져오는데 사용할 수 있다.

클라이언트 설정은 상당히 간단하다. *war* 서버를 설정하려면 vipw(8)만 실행하고 다음 라인을 아래와 같이 변경한다.

```
+::::::::::

+@IT_EMP::::::::::
```

이제 넷 그룹 *IT_EMP* 에서 유저가 정의한 데이터만 *war* 의 패스워드 데이터베이스에 읽혀서 이들 유저만 로그인할 수 있다.

불행히 이 제한은 쉘의 ~ 기능 그리고 유저 이름과 유저 ID 사이를 변환하는 모든 루틴에 적용된다. 다시 말해 cd ~*user* 은 동작하지 않고 ls -l 은 유저 이름 대신 숫자 id 를 보여주며 find . -user joe -print 는 "No such user"와 같은 결과를 출력하고 실패한다. 이것을 해결하려면 서버에 로그인이 허락된 유저 엔트리를 제외한 모든 유저 엔트리를 읽어 들여야 된다.

이 문제는 /etc/master.passwd 에 다른 라인을 추가해서 해결할 수 있다. 이 라인에는 다음 내용이 포함되어 있어야 된다:

```
+::::::::::/sbin/nologin
```

이 의미는 "모든 엔트리를 읽어오지만 엔트리에서 /sbin/nologin 쉘은 대체하지 않는다". /etc/master.passwd 의 값을 기본값으로 해서 패스워드 엔트리의 모든 필드를 대체할 수 있다.

> **주의:** *+::::::::::/sbin/nologin* 은 *+@IT_EMP::::::::::* 뒤에 둔다. 그렇지 않으면 NIS 에서 읽어온 모든 유저 계정의 로그인 쉘은 /sbin/nologin 이 된다.

이것을 변경한 후 새로운 직원이 IT 학부에 들어왔다면 NIS 맵 하나만 변경하면 된다. 좀더 덜 중요한 서버에서 /etc/master.passwd 의 오래된 *+::::::::::* 를 다음과 같이 변경하여 비슷한 결과를 얻을 수 있다:

```
+@IT_EMP::::::::::
+@IT_APP::::::::::
+::::::::::/sbin/nologin
```

일반적인 워크스테이션의 라인은 다음과 같을 것이다:

```
+@IT_EMP:::::::::
+@USERS:::::::::
+:::::::::/sbin/nologin
```

그리고 몇 주 후 정책이 변경될 때까지 모든 것이 정상이다: IT 학부에서 인턴 고용을 시작한다. IT 학과의 인턴들은 일반 워크스테이션과 덜 중요한 서버를 사용할 수 있다; 그리고 IT 수습생들은 매인 서버에 로그인할 수 있다. 새로운 넷 그룹 IT_INTERN 을 추가하고 이 넷 그룹에 새로운 IT 인턴들을 추가하여 모든 머신의 설정을 변경하기 시작한다. 이전에 언급했듯이 "중앙 관리에서 에러는 전체적인 혼란을 초례 한다".

다른 넷 그룹으로부터 새로운 넷 그룹을 생성하는 NIS 의 기능으로 이런 상황을 방지할 수 있다. 한가지 가능성은 역할별로 넷 그룹을 생성하는 것이다. 예를 들면 중요한 서버의 로그인 제한을 정의한 *BIGSRV* 라는 넷 그룹, 덜 중요한 서버는 *MALLSRV* 라는 다른 넷 그룹 그리고 보통 워크스테이션은 *USERBOX* 라고 부르는 세 번째 넷 그룹을 생성할 수 있다. 이들 넷 그룹 각각은 이들 머신에 로그인을 허용하는 넷 그룹을 포함한다. NIS 맵 넷 그룹의 새로운 엔트리는 다음과 비슷할 것이다:

```
BIGSRV      IT_EMP   IT_APP
SMALLSRV    IT_EMP   IT_APP   ITINTERN
USERBOX     IT_EMP   ITINTERN USERS
```

이러한 방법의 로그인 제한은 동일한 제한이 필요한 머신 그룹을 정의할 수 있을 때 편리할 것이다. 불행히 이것은 룰이 아니고 예외다. 대부분 머신 별로 로그인 제한을 정의할 수 있는 능력이 필요하기 때문이다.

머신에 특별한 넷 그룹을 정의하는 것은 위의 개요를 변경하는 정책과 또 다른 문제를 유발할 수 있다. 이 시나리오에서 각 머신의 /etc/master.passwd 는 "+"로 시작하는 두 라인을 포함한다. 이들 중 첫 번째는 이 머신에 로그인할 수 있는 계정으로 넷 그룹을 추가하고 두 번째는 /sbin/nologin 쉘을 가진 다른 모든 계정이다. 넷 그룹 이름처럼 머신 이름도 모두 대문자로 사용하는 것이 좋은 생각이다. 다시 말해 이 라인은 다음과 비슷할 것이다:

```
+@BOXNAME:::::::::
+:::::::::/sbin/nologin
```

모든 머신에서 이 태스크를 끝내고 다시 로컬의 /etc/master.passwd 를 수정할 필요는 없다. 더 자세한 변경은 NIS 맵을 수정해서 제어할 수 있다. 여기 몇 개의 아이디어를 추가한 이 시나리오에 맞는 넷 그룹 맵 예제가 있다.

```
# Define groups of users first
IT_EMP      (,alpha,test-domain)      (,beta,test-domain)
IT_APP      (,charlie,test-domain)    (,delta,test-domain)
DEPT1       (,echo,test-domain)       (,foxtrott,test-domain)
DEPT2       (,golf,test-domain)       (,hotel,test-domain)
DEPT3       (,india,test-domain)      (,juliet,test-domain)
ITINTERN    (,kilo,test-domain)       (,lima,test-domain)
D_INTERNS (,able,test-domain)         (,baker,test-domain)
#
# Now, define some groups based on roles
USERS       DEPT1     DEPT2       DEPT3
BIGSRV      IT_EMP    IT_APP
SMALLSRV    IT_EMP    IT_APP      ITINTERN
USERBOX     IT_EMP    ITINTERN    USERS
#
# And a groups for a special tasks
# Allow echo and golf to access our anti-virus-machine
SECURITY    IT_EMP    (,echo,test-domain)   (,golf,test-domain)
#
# machine-based netgroups
# Our main servers
WAR         BIGSRV
FAMINE      BIGSRV
# User india needs access to this server
POLLUTION   BIGSRV    (,india,test-domain)
#
# This one is really important and needs more access restrictions
DEATH       IT_EMP
#
# The anti-virus-machine mentioned above
ONE         SECURITY
#
```

```
# Restrict a machine to a single user
TWO          (,hotel,test-domain)
# [...more groups to follow]
```

유저 계정을 관리하기 위해 어떤 데이터베이스를 사용한다면 데이터베이스의 리포트 툴로 맵의 첫 번째 부분을 생성할 수 있을 것이다. 이 방법으로 새로운 유저는 자동으로 머신에 접근할 수 있다.

> **경고:** 항상 머신 기반 넷 그룹을 사용하라고 권고할 수 없다. 20 대 또는 수백 대의 머신을 학생들 연구실에 배치한다면 적당한 제한에서 NIS 맵 크기를 유지하기 위해 머신 별 넷 그룹 대신 역할별 넷 그룹을 사용해야 된다.

23.4.8 기억해야 될 중요 사항

NIS 환경에서 따로 해야 될 일이 아직 존재한다.

- 연구실에 유저를 추가할 때 마스터 NIS 서버에만 추가하고 NIS 맵을 다시 빌드해야 된다. 이것을 잊는다면 새로운 유저는 NIS 마스터를 제외하고 어떤 곳에도 로그인할 수 없다. 예를 들어 실험실에 "jsmith"라는 새로운 유저를 추가해야 된다면 다음 명령을 입력한다:

  ```
  # pw useradd jsmith
  # cd /var/yp
  # make test-domain
  ```

  pw useradd jsmith 대신 adduser jsmith 를 실행할 수 있다.

- 관리용 계정을 NIS 맵에서 삭제한다. 관리자 계정에 접근할 필요가 없는 유저가 있는 머신에 관리자 계정과 패스워드가 전파되는 것을 원치 않을 것이다.

- NIS 마스터와 슬레이브의 보안을 유지하고 다운 시간을 최소화한다. 해커나 간단히 이들 머신의 전원을 끌 수 있는 사람이 있다면 이들은 효과적으로 수많은 사람들을 연구실에 로그인하지 못하게 할 수 있다.

이것이 중앙 관리 시스템 관리자의 문제다. NIS 서버를 보호하지 않는다면 수많은 유저의

야유를 받게 될 것이다.

23.4.9 NIS v1 호환

FreeBSD 의 **ypserv** 는 NIS v1 클라이언트를 부분적으로 지원한다. FreeBSD 의 NIS 는 오직 NIS v2 프로토콜만 사용하지만 v1 프로토콜을 포함한 다른 오래된 시스템을 지원한다. **ypbind** 데몬은 이들 시스템에 실제로 필요 없더라도 NIS v1 서버와 연결할 수 있도록 한다(그리고 이들이 v2 서버에서 응답을 받은 후에도 브로드케스팅을 지속할 것이다). 보통 클라이언트의 요청이 지원되더라도 이 버전의 ypserv 는 v1 맵 전송 요청을 지원하지 않는다; 따라서 오직 v1 프로토콜만 지원하는 오래된 NIS 서버가 있는 환경에 마스터나 슬레이브로 사용할 수 없다. 다행히 이런 서버는 요즘 사용하지 않을 것이다.

23.4.10 NIS 클라이언트가 되기도 하는 NIS 서버

서버 머신이 NIS 클라이언트도 되는 멀티 서버 도메인에서 ypserv 를 실행할 때 주의한다. 이들 서버가 바인드 요청 브로드케스트를 하도록 두지 않고 강제로 자신들에게 바인드 하도록 하는 것도 좋은 생각이고 가능할 것이다. 서버끼리 의존관계가 있어서 서버 하나가 다운되고 다른 서버들이 이 서버와 연관되어 있다면 이상한 실패 모드가 발생할 수 있다. 결국 모든 클라이언트가 타임아웃 되어 다른 서버와 바인드 하려고 하지만 상당히 지연된다. 그리고 서버들이 다시 각자 바인드 하기 때문에 서비스는 계속 사용할 수 없을 것이다.

-S 플래그로 ypbind 를 실행해서 호스트를 특정 서버에 강제로 바인드 할 수 있다. NIS 서버를 재 부팅할 때마다 직접 입력하지 않으려면 /etc/rc.conf 에 다음 라인을 입력할 수 있다:

```
nis_client_enable="YES"       # run client stuff as well
nis_client_flags="-S NIS domain,server"
```

더 많은 정보는 ypbind(8)을 본다.

23.4.11 패스워드 포맷

사람들이 NIS 를 실행하려고 할 때 가장 보편적인 문제는 패스워드 포맷 호환성이다.

여러분의 NIS 서버가 DES 암호화 패스워드를 사용한다면 DES 를 사용하는 클라이언트만 지원한다. 예를 들어 네트워크에 Solaris NIS 클라이언트가 있다면 대부분 DES 암호화 패스워드를 사용해야 된다.

서버와 클라이언트가 어떤 포맷을 사용하는지 체크하려면 /etc/login.conf 를 본다. 호스트가 DES 암호화 패스워드를 사용하도록 하려면 *default* 클래스는 다음과 같은 엔트리를 포함할 것이다:

```
default:₩
    :passwd_format=des:₩
    :copyright=/etc/COPYRIGHT:₩
    [Further entries elided]
```

passwd_format 문자열에 포함할 수 있는 다른 값은 *blf* 와 *md5* 가 있다(각각 Blowfish 와 MD5 암호화 패스워드).

/etc/login.conf 를 변경하려면 root 로 다음 명령을 실행해서 로그인 기능 데이터베이스를 다시 빌드해야 된다:

```
# cap_mkdb /etc/login.conf
```

> Note: /etc/master.passwd 의 패스워드 포맷은 로그인 기능 데이터베이스가 다시 빌드 된 후 유저가 자신들의 패스워드를 변경할 때까지 업데이트 되지 않는다.

패스워드가 여러분이 선택한 포맷으로 암호화되었는지 확인하려면 /etc/auth.conf 의 *crypt_default* 에 선택한 순서대로 패스워드 포맷이 있는지 확인한다. 패스워드 포맷을 선택하려면 리스트의 첫 번째에 원하는 포맷을 둔다. 예를 들어 DES 암호화 패스워드를 사용할 때 엔트리는 다음과 같다:

```
crypt_default   =    des blf md5
```

NIS 서버와 클라이언트 기반의 각 FreeBSD 에서 위의 단계를 따랐다면 여러분의 네트워크에서 어떤 패스워드 포맷을 사용하는지 서로 일치해야 된다. NIS 클라이언트에서 인증 문제가 발생한다면 패스워드 포맷 문제부터 분석을 시작한다. 다른 네트워크에 NIS 서버를 두려면 저 사양에서 일반적인 표준이기 때문에 DES 를 모든 시스템이 사용해야 될 것이다.

23.5 DHCP

23.5.1 DHCP 란 무엇인가?

동적 호스트 구성 프로토콜인 DHCP 는 시스템이 네트워크에 연결해서 네트워크와 통신에 필요한 정보를 받아올 수 있다는 의미다. FreeBSD 는 ISC(Internet Software Consortium) DHCP 를 실행하기 때문에 여기서 설명하는 것들은 ISC 배포본을 사용한다.

23.5.2 이 섹션은 무엇에 대해 설명하는가

이 섹션은 클라이언트단과 서버 단의 ISC DHCP 시스템 컴포넌트를 설명한다. 클라이언트단 프로그램 dhclient는 FreeBSD에 통합되었고 서버부분은 net/isc-dhcp3-server 포트에서 사용할 수 있다. dhclient(8), dhcp-options(5), dhclient.conf(5) 매뉴얼 페이지와 아래의 추가적인 레퍼런스에서 유용한 정보를 제공한다.

23.5.3 어떻게 동작하는가

DHCP 클라이언트 dhclient 가 클라이언트 머신에서 실행되면 설정 정보를 요청하기 위해 브로드케스팅을 시작한다. 기본적으로 이들은 UDP 포트 68 번을 사용하여 요청한다. 그러면 서버는 UDP 67 로 클라이언트 IP 주소, 넷 마스크, 라우터 그리고 DNS 서버 같은 관련된 네트워크 정보를 되돌려준다. 이러한 모든 정보는 DHCP 형식으로 뿌려지고 일정시간 동안만(DHCP 서버 관리자가 설정한) 유효하다. 이런 식으로 더 이상 네트워크에 연결되어 있지 않은 클라이언트 IP 주소는 자동으로 회수된다.

DHCP 클라이언트는 서버로부터 여러 가지 정보를 받을 수 있다. 완벽한 리스트는 dhcp-option(5)에서 찾을 수 있을 것이다.

23.5.4 FreeBSD 통합

FreeBSD 는 ISC DHCP 클라이언트인 dhclient 와 완벽하게 통합되어있다. DHCP 클라이언트 지원은 인스톨러와 기본 시스템에서 제공하여 DHCP 서버가 사용되는 네트워크에서 자세한 네트워크 설정에 대한 지식이 필요 없게 된다. dhclient 는 FreeBSD 3.2 부터 모든 FreeBSD 에 포함되어 있다.

DHCP 는 **sysinstall** 에서 지원된다. **sysinstall** 로 네트워크 인터페이스를 설정할 때 첫 번째 질문이 "**Do you want to try DHCP configuration of this interface?**"이고, 긍정적인 대답(**YES**)은 dhclient 를 실행하여 성공하면 자동으로 네트워크 설정 정보를 채운다.

시스템이 시작될 때 DHCP 를 사용하려면 다음 두 가지를 설정해야 된다:

- bpf 장치가 커널에 컴파일 되어 있어야 한다. 컴파일은 커널 설정파일에 *device bpf* (FreeBSd 4.X에서는 *pseudo-device bpf*)을 추가하고 커널을 다시 빌드 한다. 커널 빌드에 대한 더 많은 정보는 8장을 본다.

  bpf 장치는 FreeBSD GENERIC 커널에 포함되어 있으므로 사용자 커널을 가지고 있지 않다면 DHCP 작동을 위해 추가할 필요는 없다.

  Note: 특히 보안에 민감하다면 bpf 도 패킷 스니퍼가 정확하게 동작하도록 하는 장치라는 것을 인식한다(root 로 실행해야 되더라도). 보안에 민감하다면 DHCP 를 정말 사용할 때까지 bpf 를 커널에 추가하면 안 된다.

- 다음 라인을 포함하도록 /etc/rc.conf를 수정한다:

  ```
  ifconfig_fxp0="DHCP"
  ```

  Note: 11 장에서 설명하였듯이 동적으로 설정하기를 원하는 인터페이스로 *fxp0* 를 변경한다.

  dhclient 를 다른 위치에서 사용하거나 추가적인 플래그를 dhclient 에 적용한다면 다음 라인을 추가한다(필요하다면 수정한다):

  ```
  dhcp_program="/sbin/dhclient"
  dhcp_flags=""
  ```

DHCP 서버 **dhcpd**는 포트 컬렉션의 net/isc-dhcp3-server 포트에 포함되어 있다. 이 포트에는 ISC DHCP 서버와 관련 문서가 있다.

23.5.5 파일

- /etc/dhclient.conf

  dhclient 는 설정파일 /etc/dhclient.conf 가 필요하다. 보통 이 파일에는 설명만 있으며 일반적인 설정에 기본값이 적당하다. 이 설정 파일은 dhclient.conf(5) 매뉴얼 페이지에 설명되어 있다.

- /sbin/dhclient

  dhclient 는 고정적으로 /sbin 안에 링크되어 있다. dhclient(8) 매뉴얼 페이지에서 dhclient 에 대한 더 많은 정보를 제공한다.

- /sbin/dhclient-script

  dhclient-script 는 FreeBSD 에 맞는 DHCP 클라이언트 설정 스크립트다. dhclient-script(8)에 설명이 있지만 특별히 유저가 수정할 필요는 없다.

- /var/db/dhclient.leases

  DHCP 클라이언트는 이 파일에 필요한 데이터베이스를 로그로 기록한다. dhclient.leases(5)에 좀더 자세한 설명이 있다.

23.5.6 더 많은 정보

DHCP 프로토콜은 RFC2131 에(http://www.freesoft.org/CIE/RFC/2131/) 완벽하게 설명되어 있다. 관련 정보는 dhcp.org 에(http://www.dhcp.org/) 게시되어 있다.

23.5.7 DHCP 서버 설치 및 설정

23.5.7.1 이 섹션에서 무엇을 설명하는가

이 섹션은 ISC(Internet Software Consortium) DHCP 스위트를 사용하여 FreeBSD 시스템을 어떻게 DHCP 서버로 동작하도록 설정하는지 설명한다.

서버 부분은 FreeBSD의 기본 시스템에서 제공하지 않기 때문에 이 서비스를 제공하려면 net/isc-dhcp3-server 포트를 설치해야 한다. 포트 컬렉션 사용에 대한 더 많은 정보는 4 장을 본다.

23.5.7.2 DHCP 서버 설치

FreeBSD 시스템을 DHCP 서버로 설정하려면 bpf(4) 장치를 커널에 컴파일 해야 된다. 커널에 컴파일 하려면 *device bpf* (FreeBSD 4.X 에서는 *pseudo-device bpf*)를 커널 설정 파일에 추가하고 커널을 다시 빌드 한다. 커널 빌드에 대한 더 많은 정보는 8 장을 본다.

bpf 장치는 FreeBSD GENERIC 커널에 포함되어 있으므로 DHCP 동작을 위해 사용자 커널을 생성할 필요는 없다.

> Note: 특히 보안에 민감하다면 bpf 도 패킷 스니퍼가 정확하게 동작하도록 하는 장치라는 것을 인식한다(root 로 실행해야 되더라도). 보안에 민감하다면 DHCP 를 정말 사용할 때까지 bpf 를 커널에 추가하면 안 된다.

다음에 해야 될 일은 net/isc-dhcp3-server 포트로 설치된 샘플 dhcpd.conf를 수정해야 된다. 기본적으로 이 파일은 /usr/local/etc/dhcpd.conf.sample 파일이고 변경하기 전에 이 파일을 /usr/local/etc/dhcpd.conf로 복사한다.

23.5.7.3 DHCP 서버 설정하기

dhcpd.conf 는 서브넷과 호스트에 관한 선언을 포함하고 있으며, 예제를 이용하여 아주 쉽게 설명되어있다:

```
option domain-name "example.com"; ❶
option domain-name-servers 192.168.4.100; ❷
option subnet-mask 255.255.255.0; ❸

default-lease-time 3600; ❹
max-lease-time 86400; ❺
ddns-update-style none; ❻

subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.129 192.168.4.254; ❼
  option routers 192.168.4.1; ❽
}

host mailhost {
  hardware ethernet 02:03:04:05:06:07; ❾
  fixed-address mailhost.example.com; ❿
}
```

❶ 이 옵션은 기본 검색 도메인으로 클라이언트에 제공되는 도메인을 지정한다. 이 의미에 대한 더 많은 정보는 resolv.conf(5)를 본다.

❷ 이 옵션은 클라이언트가 사용해야 되는 DNS 서버 리스트를 컴마(,)로 나누어 지정한다.

❸ 클라이언트에 제공할 넷 마스크

❹ 클라이언트는 유효한 임대기간으로 특정 기간을 요청할 것이다. 그렇지 않으면 서버는 이 만료 값을(초 단위) 임대 기간으로 할당한다.

❺ 이것은 서버가 제공하는 최대 임대기간이다. **오직 *max-lease-time* 초가 유효하지만 클라이언트가 더 오랜 임대기간을 요청해야 되기 때문에 문제가 된다.**

❻ 이 옵션은 임대가 허용되거나 다시 임대될 때 DHCP 서버가 DNS 를 업데이트 할지 지정한다. ISC 에서 이 옵션이 *필요하다*.

❼	이것은 클라이언트에 할당하기 위해 남겨 둔 풀(pool)에서 어떤 IP 주소를 사용할지를 의미한다. 한 쪽은 시작되는 IP 주소고 다른 쪽은 끝나는 주소다.
❽	클라이언트에 공급할 기본 게이트웨이를 선언한다.
❾	호스트의 하드웨어 MAC 주소(그래서 DHCP 서버는 요청이 들어왔을 때 호스트를 인지할 수 있다).
❿	특정 호스트에게 항상 같은 IP 주소를 할당하도록 지정한다. 제공한 정보를 회수하기 전에 DHCP 서버는 호스트 이름을 분석하기 때문에 이곳에 호스트 이름도 사용할 수 있다.

dhcpd.conf 작성이 끝나면 다음 명령으로 서버를 시작할 수 있다:

```
# /usr/local/etc/rc.d/isc-dhcpd.sh start
```

나중에 서버 설정을 변경할 때 대부분의 데몬처럼 설정을 다시 로드하는 동안 **dhcpd** 는 *SIGHUP* 신호를 보내도 응답하지 않는다. *SIGTERM* 신호를 보내어 프로세스를 정지시키고 위의 명령을 사용하여 재 시작해야 된다.

23.5.7.4 파일

- /usr/local/sbin/dhcpd

  dhcpd 는 /usr/local/sbin 내에 정적으로 링크되어 있다. 포트로 설치된 dhcpd(8) 매뉴얼 페이지에서 dhcpd 에 대한 더 많은 정보를 제공한다.

- /usr/local/etc/dhcpd.conf

  dhcpd 는 클라이언트에게 서비스를 제공하기 전에 /usr/local/etc/dhcpd.conf 파일의 설정이 필요하다. 이 파일은 서버 운용에 대한 정보와 함께 서비스를 받는 클라이언트에게 제공할 모든 정보를 포함해야 된다. 이 설정 파일은 포트로 설치된 dhcpd.conf(5) 매뉴얼 페이지에 설명되어 있다.

- /var/db/dhcpd.leases

  DHCP 서버는 로그처럼 작성된 이 파일에 필요한 데이터베이스를 가지고 있다. 포트로 설치한 dhclient.leases(5) 매뉴얼 페이지에 좀더 자세한 설명이 있다.

- /usr/local/sbin/dhcrelay

  dhcrelay는 하나의 DHCP 서버가 클라이언트의 요청을 다른 네트워크의 DHCP 서버에게 포워드하는 고급 환경에 사용된다. 이 기능이 필요하다면 net/isc-dhcp3-server 포트를 설치한다. 포트로 제공되는 dhcrelay(8) 매뉴얼 페이지에 더 자세한 정보가 있다.

23.6 DNS

23.6.1 요약

FreeBSD 는 아주 일반적인 DNS 프로토콜을 실행하는 BIND(Berkeley Internet Name Domain) 버전을 기본적으로 사용한다. DNS 는 이름을 IP 주소로 매핑시키는 프로토콜이다. 예를 들어 www.FreeBSD.org 로 쿼리를 보내면 FreeBSD 의 웹 서버의 IP 주소를 응답으로 받지만 ftp.FreeBSD.org 로 쿼리 하면 그에 맞는 FTP 머신의 IP 주소를 돌려준다. 이와 반대의 경우가 발생할 수 있다. IP 주소로 쿼리를 보내면 호스트 이름으로 해석된다. 시스템에서 DNS lookup 을 수행하려고 네임서버를 운용할 필요는 없다.

DNS 는 좀더 복잡한 시스템의 authoritative root 네임서버와 호스트 및 각각의 도메인 정보를 캐시하는 작은 규모의 네임서버와 인터넷을 통해 공조한다.

이 문서는 FreeBSD에서 사용되는 안정 버전인 BIND 8.x를 설명한다. FreeBSD에서 BIND 9.x는 net/bind9 포트로 설치할 수 있다.

RFC 1034 와 RFC 1035 에 DNS 프로토콜이 명시되어있다.

현재 BIND 는 인터넷 소프트웨어 컨소시엄에서(www.isc.org) 관리한다.

23.6.2 용어

이 문서를 이해하려면 DNS 와 관련된 몇 가지 용어를 이해해야 된다.

용어	정의
순방향 DNS	호스트 이름을 IP 주소로 매핑
Origin	특정 존 파일로 커버하는 도메인을 의미한다.
named, BIND, 네임서버	FreeBSD 에서 BIND 네임서버 패키지를 부르는 일반적인 이름.
리졸버	머신이 존 정보를 네임서버에 쿼리 하는 시스템 프로세스
역방향 DNS	순방향 DNS 의 반대; IP 주소를 호스트 이름으로 매핑
root 존	인터넷 존 계층의 시작점. 파일 시스템에서 root 디렉터리 아래로 모든 파일이 있는 것과 비슷하게 모든 존은 root 존 아래에 있다.
존	개개의 도메인, 서브 도메인 또는 같은 권한을 가진 DNS 관할 영역

존 예제:

- .은 root 존이다.
- org.은 root 존 아래의 존이다.
- example.org는 org. 존의 아래 존이다.
- foo.example.org.은 example.org. 존 아래의 서브 도메인이다.
- 1.2.3.in-addr.arpa는 3.2.1.* IP 공간 아래에 있는 모든 IP 주소를 의미하는 존이다.

위에서 보았듯이 더 명확한 호스트 이름 부분이 왼쪽에 나타난다. 예를 들어 example.org.은 org.보다 더 명확하고 org.은 root 존보다 더 명확하다. 각 호스트 이름의 레이아웃은 파일 시스템과 더 비슷하다: /dev 디렉터리는 root 아래에 있다.

23.6.3. 네임서버를 운용하는 이유

네임서버는 보통 두 종류가 있다: 권한을 가진 네임서버와 케싱 네임서버.

권한을 가진 네임서버는 다음과 같은 경우에 필요하다:

- 쿼리에 대해 신뢰할 수 있는 DNS 정보를 전 세계에 제공하고자 할 때.
- example.org와 같은 도메인이 등록되어 있고 이 도메인 아래의 호스트 이름에 IP 주소를 할당해야 될 때.
- IP 주소 블록이 역방향 DNS 엔트리를 필요로 할 때(IP를 호스트 이름으로).
- 슬레이브라고 하는 백업 네임서버는 주 네임서버가 다운되거나 접근할 수 없을 때 쿼리에 응답해야 된다.

케싱 네임서버는 다음과 같은 때 필요하다:

- 로컬 DNS 서버는 캐시 하여 외부의 네임서버에 쿼리 하는 것보다 더 빠르게 응답한다.
- 네트워크 전체의 트래픽을 감소해야 될 때(DNS 트래픽은 전체 인터넷 트래픽의 5% 정도를 감소시킨다.)

www.FreeBSD.org 를 쿼리 할 때 리졸버는 보통 위 단의 ISP 네임서버에 쿼리해서 응답을 전달한다. 케싱 DNS 서버인 로컬 네임서버는 DNS 서버를 케싱해서 외부에 한번만 쿼리 하면 된다. 정보가 로컬에 케시되어 있기 때문에 추가적인 쿼리는 로컬 네트워크에서 해결된다.

23.6.4 어떻게 동작 하는가

FreeBSD 에서 BIND 데몬은 특별한 이유로 named 라고 한다.

파일	설명
named	BIND 데몬
ndc	네임 데몬 제어 프로그램

/etc/namedb	BIND 존 정보가 위치한 디렉터리
/etc/namedb/named.conf	데몬 설정파일

존 파일은 보통 /etc/namedb 디렉터리에 있고 네임서버가 제공하는 DNS 존 정보를 가지고 있다.

23.6.5 BIND 시작

BIND 는 기본적으로 설치되기 때문에 모든 설정은 비교적 간단하다.

네임 데몬을 부팅할 때 시작하려면 /etc/rc.conf 를 다음과 같이 수정한다:

```
named_enable="YES"
```

데몬을 수동으로 시작하려면(설정을 한 후) 다음 명령을 사용한다.

```
# ndc start
```

23.6.6 설정 파일

23.6.6.1 make-localhost 사용

로컬 역방한 DNS 존 파일을 /etc/namedb/localhost.rev 에 정확하게 생성한다.

```
# cd /etc/namedb
# sh make-localhost
```

23.6.6.2 /etc/namedb/named.conf

```
// $FreeBSD$
//
// Refer to the named(8) manual page for details.  If you are ever going
// to setup a primary server, make sure you've understood the hairy
// details of how DNS is working.  Even with simple mistakes, you can
// break connectivity for affected parties, or cause huge amount of
// useless Internet traffic.

options {
        directory "/etc/namedb";

// In addition to the "forwarders" clause, you can force your name
// server to never initiate queries of its own, but always ask its
// forwarders only, by enabling the following line:
//
//      forward only;

// If you've got a DNS server around at your upstream provider, enter
// its IP address here, and enable the line below.  This will make you
// benefit from its cache, thus reduce overall DNS traffic in the
Internet.
/*
        forwarders {
                127.0.0.1;
        };
*/
```

주석문에서 설명하듯이 업 링크 케시의 장점을 활용하도록 여기서 *forwarders* 를 활성화할 수 있다. 일반적인 상황에서 네임서버는 찾고 있는 응답을 얻을 때까지 특정 네임서버에 반복적으로 쿼리를 보낸다. 이것이 활성화되면 업 링크 네임서버에(또는 제공된 네임서버) 처음으로 쿼리 해서 케시의 장점을 얻을 수 있다. 업 링크 네임서버에 과부하가 걸렸다면 빠른 네임서버를 활성화하면 좋을 것이다.

> **주의:** 127.0.0.1 은 여기서 동작하지 않는다. 이 IP 를 여러분의 업 링크 네임서버의 주소로 변경한다.

```
/*
         * If there is a firewall between you and name servers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
        // query-source address * port 53;

        /*
         * If running in a sandbox, you may have to specify a different
         * location for the dumpfile.
         */
        // dump-file "s/named_dump.db";
};

// Note: the following will be supported in a future release.
/*
host { any; } {
        topology {
                127.0.0.0/8;
        };
};
*/

// Setting up secondaries is way easier and the rough picture for this
// is explained below.
//
// If you enable a local name server, don't forget to enter 127.0.0.1
// into your /etc/resolv.conf so this server will be queried first.
// Also, make sure to enable it in /etc/rc.conf.

zone "." {
        type hint;
        file "named.root";
};
```

```
zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};

zone
"0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.INT" {
        type master;
        file "localhost.rev";
};

// NB: Do not use the IP addresses below, they are faked, and only
// serve demonstration/documentation purposes!
//
// Example secondary config entries.  It can be convenient to become
// a secondary at least for the zone where your own domain is in.  Ask
// your network administrator for the IP address of the responsible
// primary.
//
// Never forget to include the reverse lookup (IN-ADDR.ARPA) zone!
// (This is the first bytes of the respective IP address, in reverse
// order, with ".IN-ADDR.ARPA" appended.)
//
// Before starting to setup a primary zone, better make sure you fully
// understand how DNS and BIND works, however.  There are sometimes
// unobvious pitfalls.  Setting up a secondary is comparably simpler.
//
// NB: Don't blindly enable the examples below. :-)  Use actual names
// and addresses instead.
//
// NOTE!!! FreeBSD runs BIND in a sandbox (see named_flags in rc.conf).
// The directory containing the secondary zones must be write accessible
// to BIND.  The following sequence is suggested:
//
//      mkdir /etc/namedb/s
```

```
//      chown bind:bind /etc/namedb/s
//      chmod 750 /etc/namedb/s
```

샌드 박스에서 BIND 를 운용하는 방법에 대한 더 많은 정보는 샌드 박스에서 네임서버 운용하기를 본다(http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-dns.html#NETWORK-NAMED-SANDBOX)

```
/*
zone "example.com" {
        type slave;
        file "s/example.com.bak";
        masters {
                192.168.1.1;
        };
};

zone "0.168.192.in-addr.arpa" {
        type slave;
        file "s/0.168.192.in-addr.arpa.bak";
        masters {
                192.168.1.1;
        };
};
*/
```

named.conf 에서 이 설정은 순방향과 역방향 존의 슬레이브 엔트리 예제다.

새로 추가된 각 존의 새로운 존 엔트리를 named.conf 에 추가해야 한다.

예를 들어 example.org 의 가장 간단한 존 엔트리는 다음과 비슷할 수 있다:

```
zone "example.org" {
    type master;
    file "example.org";
};
```

type 구문에서 보여주듯이 이 존은 마스터다. 존 정보는 *file* 구문에서 보여주듯이 /etc/namedb/example.org 에 있다.

```
zone "example.org" {
    type slave;
    file "example.org";
};
```

슬레이브의 경우 존 정보는 마스터 서버의 특정 존을 전송해서 지정된 파일에 저장한다. 마스터 서버가 죽었거나 접속할 수 없을 슬레이브 네임서버는 전송된 존 정보를 제공할 수 있다.

23.6.6.3 존 파일

example.org(/etc/namedb/example.org 에 있는)의 마스터 존 예제는 다음과 같다:

```
$TTL 3600

example.org. IN SOA ns1.example.org. admin.example.org. (
                        5               ; Serial
                        10800           ; Refresh
                        3600            ; Retry
                        604800          ; Expire
                        86400 )         ; Minimum TTL

; DNS Servers
@       IN NS           ns1.example.org.
@       IN NS           ns2.example.org.

; Machine Names
localhost       IN A    127.0.0.1
ns1             IN A    3.2.1.2
ns2             IN A    3.2.1.3
mail            IN A    3.2.1.10
```

```
@                   IN A      3.2.1.30

; Aliases
www                 IN CNAME          @

; MX Record
@                   IN MX     10        mail.example.org.
```

"."으로 끝나는 모든 호스트 이름은 정확한 호스트 이름이고, 반대로 끝에 "."이 없는 것은 Origin을 참조한다. 예를 들어 *www*는 *www + Origin*이다. 가상 존 파일에서 origin은 example.org기 때문에 www는 www.example.org.로 변경된다.

존 파일 포맷은 다음과 같다:

```
recordname      IN recordtype     value
```

아주 일반적으로 사용되는 DNS 레코드:

SOA	존의 권한 시작
NS	권한을 가진 네임서버
A	호스트 주소
CNAME	엘리어스의 캐노니컬(canonical) 이름
MX	메일 익스체인져
PTR	도메인 네임 포인터 (역방향 DNS 에 사용되는)

```
example.org. IN SOA ns1.example.org. admin.example.org. (
                              5               ; Serial
                              10800           ; Refresh after 3 hours
                              3600            ; Retry after 1 hour
                              604800          ; Expire after 1 week
                              86400 )         ; Minimum TTL of 1 day
```

example.org.	도메인 이름이면서 이 존 파일의 Origin
ns1.example.org.	이 존에 대한 주/권한을 가진 네임서버

admin.example.org.	이 존에 대한 책임이 있는 사람, "@의 메일주소가 .으로 대체된다. 따라서 <admin@example.org>는 admin.example.org 로 바뀐다.
5	파일의 시리얼 번호. 이것은 존 파일이 수정될 때마다 증가해야 된다. 요즘 많은 관리자는 시리얼 번호로 *yyyymmddrr* 포맷을 선호한다. *2001041002*는 2001 년 04 월 10 일에 마지막으로 수정되었음을 의미하고, 마지막의 02 는 이날 존 파일이 수정된 횟수를 의미한다. 시리얼 번호는 존이 업데이트 되었을 때 슬레이브 네임서버에게 알려 주기 때문에 중요하다.

```
@           IN NS                ns1.example.org.
```

이것은 *NS* 엔트리다. 존에 대한 응답 권한을 가진 모든 네임서버는 이런 엔트리를 하나씩 가지고 있다. 여기서 본 @ 표시는 example.org.가 될 수 있다. @ 표시는 Origin 으로 변경된다.

```
localhost       IN A     127.0.0.1
ns1             IN A     3.2.1.2
ns2             IN A     3.2.1.3
mail            IN A     3.2.1.10
@               IN A     3.2.1.30
```

레코드는 머신 이름을 보여 준다. 위에서 보았듯이 ns1.example.org 는 3.2.1.2 로 해석된다. 역시 Origin 표시 @가 여기서도 사용되었기 때문에 example.org 의미는 3.2.1.30 으로 해석된다.

```
www             IN CNAME         @
```

케노니컬 이름 레코드는 보통 머신에 엘리어스를 할당하는데 사용된다. 예제에서 www 는 origin 으로 표시된 머신이나 example.org(3.2.1.30)의 엘리어스다. *CNAMES*는 호스트 이름의 엘리어스를 제공하거나 라운드 로빈의 여러 머신 중 호스트 이름 하나다.

```
@               IN MX    10      mail.example.org.
```

MX 레코드는 존으로 들어오는 메일을 어떤 메일 서버가 제어하는지 지정한다. mail.example.org 는 메일 서버의 호스트 이름이고 10 은 메일 서버의 우선순위다.

3, 2, 1 의 우선순위로 여러 대의 메일 서버를 가지고 있을 수 있다. 메일 서버는 처음으로 example.org 의 가장 높은 우선 순위의 MX 에 전달을 시도하고 두 번째 우선 순위 등으로 메일이 정확하게 전달될 때까지 시도한다.

80.90.210.in-addr.arpa 존 파일에는(역방향 DNS) *A* 또는 *CNAME* 대신 PTR 엔트리를 사용하는 것을 제외하고 같은 포맷이 사용된다.

```
$TTL 3600

1.2.3.in-addr.arpa. IN SOA ns1.example.org. admin.example.org. (
                              5               ; Serial
                              10800           ; Refresh
                              3600            ; Retry
                              604800          ; Expire
                              3600 )          ; Minimum

@         IN NS     ns1.example.org.
@         IN NS     ns2.example.org.

2         IN PTR    ns1.example.org.
3         IN PTR    ns2.example.org.
10        IN PTR    mail.example.org.
30        IN PTR    example.org.
```

이 파일은 적당한 IP 주소를 가상의 도메인 호스트 이름으로 매핑해 준다.

23.6.7 케싱 네임서버

케싱 네임서버는 존에 대한 권한이 없는 네임서버다. 단순히 쿼리하고 나중에 사용하기 위해 쿼리한 내용을 기억하고 있다. 설정은 존을 포함하지 않은 일반적인 네임서버로 설정한다.

23.6.8 Sandbox 에서 named 운용

named(8)의 보안을 강화하기 위해 특권이 없는 유저로 실행하고 sandbox 디렉터리로 chroot(8)하도록 설정한다. 이 설정은 sandbox 외부에서 named 데몬으로 모든 접근을 차단한다. 이렇게 해서 named 에서 발생할 피해를 줄이는데 도움이 된다. 기본적으로 named 를 실행하는 유저로 FreeBSD 는 bind 라는 유저와 그룹을 가지고 있다.

> Note: 많은 사람들이 named 를 chroot 로 설정하는 대신 jail(8) 내부에서 named 를 운용하도록 권장한다. 이 섹션에서는 이 상황에 대해 다루지 않는다.

named 는 sandbox 외부로(공유 라이브러리와 로그 소켓 등) 모든 접근이 차단되기 때문에 named 기능이 정확하게 동작하도록 몇 개의 단계가 필요하다. 다음 체크 리스트에서 sandbox 의 경로는 /etc/named 이고 이 디렉터리의 내용을 수정하지 못하도록 설정하였다고 가정한다. root 로 다음 단계를 수행한다.

- named에 필요한 모든 디렉터리를 생성한다:

```
# cd /etc/namedb
# mkdir -p bin dev etc var/tmp var/run master slave
# chown bind:bind slave var/* ❶
```

❶ named 만 이들 디렉터리에 쓰기 권한이 필요하다.

- 기본 존과 설정파일을 다시 준비하고 생성한다:

```
# cp /etc/localtime etc ❷
# mv named.conf etc && ln -sf etc/named.conf
# mv named.root master
# sh make-localhost && mv localhost.rev localhost-v6.rev master
# cat > master/named.localhost
$ORIGIN localhost.
$TTL 6h
@       IN   SOA localhost. postmaster.localhost. (
                    1     ; serial
                    3600       ; refresh
```

```
                1800      ; retry
                604800    ; expiration
                3600 )    ; minimum
        IN  NS  localhost.
        IN  A       127.0.0.1
```

❷ 이 설정은 named 가 syslogd(8)에 정확한 시간으로 로그를 남기도록 한다.

- FreeBSD 4.9-RELEASE 이전 버전을 사용 중이라면 named-xfer를 정적으로 링크한 복사본을 빌드해서 sandbox에 복사한다:

```
# cd /usr/src/lib/libisc
# make cleandir && make cleandir && make depend && make all
# cd /usr/src/lib/libbind
# make cleandir && make cleandir && make depend && make all
# cd /usr/src/libexec/named-xfer
# make cleandir && make cleandir && make depend && make
NOSHARED=yes all
# cp named-xfer /etc/namedb/bin && chmod 555
/etc/namedb/bin/named-xfer ❸
```

정적으로 링크한 named-xfer 가 설치된 후 소스 트리에 라이브러리나 프로그램의 필요 없는 복사본이 남지 않도록 설치된 것을 정리해야 된다:

```
# cd /usr/src/lib/libisc
# make cleandir
# cd /usr/src/lib/libbind
# make cleandir
# cd /usr/src/libexec/named-xfer
# make cleandir
```

❸ 이 단계에서 문제가 발생한다고 종종 보고되었다. 문제가 발생하면 다음 명령을 사용한다:

```
# cd /usr/src && make cleandir && make cleandir
```

그리고 /usr/obj 트리를 삭제한다:

```
# rm -fr /usr/obj && mkdir /usr/obj
```

이렇게 소스 트리에서 "cruft"를 정리한 후 위의 단계를 다시 시도하면 정확히 동작한다.

FreeBSD 버전 4.9-RELEASE 나 이 후 버전을 사용한다면 named-xfer 을 /usr/libexec 에 복사하면 정적으로 링크가 되고, 이것을 sandbox 에 복사할 때 간단히 cp(1) 명령을 사용할 수 있다.

- named가 쓰기를 할 수 있도록 dev/null을 만든다:

```
# cd /etc/namedb/dev && mknod null c 2 2
# chmod 666 null
```

- /var/run/ndc를 /etc/namedb/var/run/ndc로 심볼릭 링크한다:

```
# ln -sf /etc/namedb/var/run/ndc /var/run/ndc
```

Note: 위 설정은 단순히 ndc(8)을 실행할 때마다 *-c* 옵션을 지정할 필요가 없게 한다. 부팅할 때 /var/run 의 내용이 삭제되기 때문에 유용하다고 생각되면 *@reboot* 옵션을 사용하도록 이 명령을 root crontab 에 추가할 수 있다. 이에 관한 더 많은 정보는 crontab(5)에서 볼 수 있다.

- named가 작성할 수 있는 추가적인 log 소켓을 생성하도록 syslogd(8)을 설정한다. 설정은 /etc/rc.conf의 *syslogd_flags* 변수에 *-l /etc/namedb/dev/log*을 추가한다.

- named가 시작해서 sandbox로 chrroot하도록 다음 라인을 /etc/r.conf에 추가하여 준비한다:

```
named_enable="YES"
named_flags="-u bind -g bind -t /etc/namedb /etc/named.conf"
```

Note: 설정파일 */etc/named.conf* 에 *sandbox* 와 관련된 절대경로를 표시한다. 예를 들면 위의 라인에서 실제파일은 /etc/namedb/etc/named.conf 임을

의미한다.

다음 단계는 /etc/namedb/etc/named.conf 에 named 가 어떤 존을 로드하고 디스크의 어느 곳에서 찾아야 되는지 지정한다. 다음은 예제와 설명이다(여기에서는 sandbox 에서 운용되는 DNS 서버를 위한 특별한 다른 설정은 없다):

```
options {
        directory "/"; ❶
        named-xfer "/bin/named-xfer"; ❷
        version "";       // Don't reveal BIND version
        query-source address * port 53;
};
// ndc control socket
controls {
        unix "/var/run/ndc" perm 0600 owner 0 group 0;
};
// Zones follow:
zone "localhost" IN {
        type master;
        file "master/named.localhost"; ❸
        allow-transfer { localhost; };
        notify no;
};
zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "master/localhost.rev";
        allow-transfer { localhost; };
        notify no;
};
zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.int" {
    type master;
    file "master/localhost-v6.rev";
    allow-transfer { localhost; };
    notify no;
};
zone "." IN {
```

```
        type hint;
        file "master/named.root";
};
zone "private.example.net" in {
        type master;
        file "master/private.example.net.db";
    allow-transfer { 192.168.10.0/24; };
};
zone "10.168.192.in-addr.arpa" in {
        type slave;
        masters { 192.168.10.2; };
        file "slave/192.168.10.db"; ❹
};
```

❶ *directory* 구문은 named 에게 필요한 모든 파일이 이 디렉터리에 있다는 것을 지시하기 위해 /로 지정되어 있다(일반 유저의 /etc/namedb 와 동일함을 기억한다).

❷ named-xfer 바이너리의 절대경로 지정(named 의 프래임 레퍼런스에서). 이 설정은 기본적으로 named 가 /usr/libexec 에서 named-xfer 를 찾도록 컴파일 되어 있기 때문에 필요하다.

❸ named 가 이 존의 존 파일을 찾을 수 있는 파일이름을 지정한다(위의 *directory* 구문과 관련 있다).

❹ 존 파일이 마스터 서버에서 완벽하게 전송된 후 named 가 존 파일의 복사본을 작성하는 파일이름을 지정한다(위의 directory 구문과 관련 있다). 이것은 slave 디렉터리의 소유자를 위의 설정 단계에서 왜 bind 로 왜 바꿔야 됐는지를 보여준다.

위 단계를 마치고 서버를 재 부팅하거나 syslogd(8)을 재 시작한 후 named(8)을 시작하고 *syslogd_flags* 와 *named_flags* 에 지정한 새로운 옵션이 사용되는지 확인한다. 이제 named 를 복사한 sandbox 를 운용할 수 있다.

23.6.9 보안

BIND 가 DNS 로 가장 보편적으로 사용되고 있지만 항상 보안 문제가 발생했다. 따라서

앞으로도 취약점이 발견될 가능성은 충분하다.

CERT 의(http://www.cert.org/) 보안 권고를 읽고 현재 인터넷과 FreeBSD 보안 문제에 대한 최신 이슈를 접할 수 있도록 FreeBSD 보안 공고 메일링 리스트(http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html#ERESOURCES-MAIL)에 가입하는 것도 좋은 생각이다.

> **Tip:** 보안 문제가 발생하면 최신 소스를 받아서 **named** 를 새로 빌드하여 피해를 최소화할 수 있다.

23.6.10 더 많은 자료

BIND/**named** 매뉴얼 페이지: ndc(8), named(8), named.conf(5)

- 공식 ISC BIND 페이지(http://www.isc.org/products/BIND/).
- BIND FAQ(http://www.nominum.com/getOpenSourceResource.php?id=6).
- 오라일리 DNS와 BIND 4th 에디션 (http://www.oreilly.com/catalog/dns4/).
- RFC1034 - 도메인 네임 - 컨셉과 유용성(ftp://ftp.isi.edu/in-notes/rfc1034.txt).
- RFC1035 - 도메인 네임 - 툴과 설명서(ftp://ftp.isi.edu/in-notes/rfc1035.txt).

23.6.11 BIND9 과 FreeBSD

FreeBSD 5.3 릴리즈 버전에는 BIND9 DNS 서버 소프트웨어가 배포판에 포함되어있다. 이 소프트웨어에는 새로운 보안 기능과 파일 시스템 레이아웃 그리고 자동으로 생성된 chroot(8) 설정이 포함되어있다. 이 섹션은 두 개의 파트로 구성되어있으며, 첫 번째는 새로운 기능과 설정을 그리고 두 번째는 5.3 으로의 업그레이드를 돕는 내용을 작성하였다. 여기서 BIND9 서버는 단순히 named(8)로 언급한다. 이 섹션은 앞장에서 설명한 용어와 이론적인 설명은 제외한다. 따라서 더 읽기 전에 앞장을 참고하기 바란다.

named 설정 파일은 현재 /var/named/etc/namedb/에 있으며 수정이 필요하다.

24.7.1 마스터 존 설정

마스터 존을 설정하기 위해 /var/named/etc/namedb/에서 다음 명령을 실행한다:

```
# sh make-localhost
```

위 명령이 정상적으로 실행되면 새로운 파일이 master 디렉터리에 생성된다; 로컬 도메인 네임을 위한 파일 이름은 localhost.rev이고 IPv6 설정을 위한 이름은 localhost-v6.rev다. named.conf 파일에 이 파일들에 대한 설정이 있으므로 따로 수정할 필요는 없다.

24.7.2 슬레이브 존 설정

추가적인 도메인이나 서브 도메인 설정은 슬레이브 존으로 설정하면 될 것이다. 일반적으로 master/localhost.rev 파일을 slave 디렉터리에 복사한 후 수정한다. 파일을 수정한 후 named.conf 파일에도 아래 example.com 예제와 같이 추가해야 된다.

```
zone "example.com" {
        type slave;
        file "slave/example.com";
        masters {
                10.0.0.1;
        };
};

zone "0.168.192.in-addr.arpa" {
        type slave;
        file "slave/0.168.192.in-addr.arpa";
        masters {
                10.0.0.1;
        };
};
```

이 예제에서 마스터 IP(10.0.0.1)은 이 도메인에 존의 설정을 전해주는 주 도메인 서버의 IP 를 지정한다.

24.7.3 시스템 초기화 설정

시스템이 부팅할 때 named 데몬도 시작하도록 설정하려면 rc.conf 파일(/etc/rc.ocnf)에 다음 옵션을 입력한다.

```
named_enable="YES"
```

다른 옵션도 지정할 수 있지만 기본적으로 이 설정이면 된다. 이곳에 입력할 수 있는 다른 옵션은 rc.conf(5) 매뉴얼 페이지를 참고한다. rc.conf 파일에 아무것도 입력하지 않았다면 다음 명령을 실행하여 직접 named 데몬을 실행할 수 있다:

```
# /etc/rc.d/named start
```

24.7.4 BIND9 보안

FreeBSD 가 자동으로 named 를 chroot(8)에서 실행하지만 가능한 DNS 서비스 공격을 방지할 수 있는 다른 보안 메커니즘이 있다.

24.7.4.1 쿼리 접근 제어 리스트

쿼리 접근 제어 리스트는 존에 대한 쿼리를 제한하는데 사용할 수 있다. 설정은 'Acl 토큰'에 지정한 내부 네트워크와 존 설정에 원하는 IP 주소를 지정하여 설정한다. 예제 호스트에 쿼리를 허용하는 도메인을 지정하려면 다음과 같이 설정한다:

```
acl "example.com" {
        192.168.0.0/24;
};

zone "example.com" {
        type slave;
        file "slave/example.com";
        masters {
                10.0.0.1;
```

```
        };
    allow-query { example.com; };
};

zone "0.168.192.in-addr.arpa" {
        type slave;
        file "slave/0.168.192.in-addr.arpa";
        masters {
                10.0.0.1;
        };
    allow-query { example.com; };
};
```

24.7.4.2 버전 제한

DNS 서버에서 버전을 확인할 수 있도록 허용하는 것은 공격자에게 좋은 정보가 될 수 있다. 악의적인 유저는 서버를 해킹하기 위해 알려진 익스플로잇(exploits)이나 버그를 찾는데 이 정보를 사용할 것이다.

> **주의**: 버전 정보를 거짓으로 설정해도 익스플로잇 공격으로부터 서버를 보호하지 못한다. 취약성이 없는 버전으로 업그레이드하는 것만이 서버를 보호할 수 있다.

거짓 버전은 named.conf 의 'options' 섹션에 설정할 수 있다:

```
options {
        directory       "/etc/namedb";
        pid-file        "/var/run/named/pid";
        dump-file       "/var/dump/named_dump.db";
        statistics-file "/var/stats/named.stats";
        version     "None of your business";
};
```

23.7 아파치 웹 서버

23.7.1 개요

FreeBSD는 세계의 가장 유명한 웹 사이트 중 몇 곳을 운영하는데 사용된다. 인터넷에서 가장 많이 사용되는 웹 서버는 **아파치 웹 서버**다. **아파치** 소프트웨어 패키지는 FreeBSD 설치 미디어에 포함되어 있다 FreeBSD를 처음 설치할 때 **아파치**를 설치하지 않았다면 www/apache13 또는 www/apache2 포트에서 설치할 수 있다.

아파치가 성공적으로 설치되면 설정을 해야 된다.

> Note: 이 섹션은 FreeBSD에서 가장 많이 사용되는 **아파치 웹 서버** 1.3.X 버전에 대해 설명한다. 아파치 2.X가 새로운 기술을 많은 소개하겠지만 여기서는 설명하지 않는다. 아파치 2.X에 대한 더 많은 정보는 http://httpd.apache.org/를 본다.

23.7.2 설정

메인 아파치 웹 서버 설정파일은 FreeBSD 의 /usr/local/etc/apache/httpd.conf 에 설치된다. 이 파일은 # 문자로 시작되는 주석 라인이 있는 전형적인 유닉스 텍스트 설정파일이다. 가능한 모든 설정 옵션의 포괄적인 설명은 이 책의 범위를 벗어나기 때문에 여기서는 가장 일반적인 설정 방법을 설명한다.

ServerRoot "/usr/local"

> 이곳은 아파치 설치의 기본 디렉터리 구조를 지정한다. 바이너리는 서버 root(/usr/local 을 의미한다)의 bin 과 sbin 서브 디렉터리에 저장되고 설정파일은 etc/apache 에 저장된다.

ServerAdmin you@your.address

> 이 주소는 서버에 문제가 있을 때 메일을 보내는 곳이다. 이 주소는 에러 페이지처럼 서버가 생성하는 몇몇 페이지에 나타난다.

ServerName www.example.com

ServerName 는 설정한 호스트 이름이 다르다면(다시 말해 호스트의 실제 이름 대신 "www"를 사용) 서버의 클라이언트에게 보여주는 호스트 이름을 설정한다.

DocumentRoot "/usr/local/www/data"

DocumentRoot 는 여러분의 문서를 제공하는 디렉터리다. 기본적으로 모든 요청은 이 디렉터리에서 가져오지만 다른 위치를 지정하기 위해 심볼릭 링크와 엘리어스를 사용할 수 있다.

변경하기 전에 아파치 설정파일을 백업으로 복사해 두는 것이 좋다. 설정이 만족스럽다면 **아파치**를 시작할 준비가 되었다.

23.7.3 아파치 시작

아파치는 다른 네트워크 서버처럼 **inetd** 슈퍼 서버에서 실행되지 않는다. 클라이언트 웹 브라우저의 HTTP 요청을 더 빨리 처리하기 위해 standalone 로 실행하도록 설정한다. 쉘 스크립트로 가능한 간단히 시작, 정지, 재 시작할 수 있다. **아파치**를 처음으로 시작하려면 다음 명령을 실행한다:

```
# /usr/local/sbin/apachectl start
```

다음 명령을 이용하여 언제든지 서버를 정지 시킬 수 있다:

```
# /usr/local/sbin/apachectl stop
```

어떤 이유로 설정파일을 변경하면 서버를 재 시작해야 된다:

```
# /usr/local/sbin/apachectl restart
```

시스템이 시작될 때 **아파치**를 실행시키려면 다음 라인을 /etc/rc.conf 에 추가 한다:

```
apache_enable="YES"
```

시스템이 부팅할 때 추가적인 명령어 라인 옵션을 아파치 httpd 프로그램에 지정하려면 rc.conf 에 추가적인 라인으로 옵션을 지정한다:

```
apache_flags=""
```

이제 웹 서버가 시작되고 웹 브라우저에 *http:///localhost/*를 지정하여 웹 사이트를 볼 수 있다. 표시되는 기본 웹 페이지는 /usr/local/www/data/index.html 이다.

23.7.4 아파치 모듈

기본 서버에 기능을 추가하기 위해 수많은 아파치 모듈을 사용할 수 있다. FreeBSD 포트 컬렉션은 유명한 add-on 모듈을 **아파치**와 같이 설치하는 쉬운 방법을 제공한다.

23.7.4.1 mod_ssl

mod_ssl 모듈은 Secure Sockets Layer(SSL v2/v3)와 Transport Layer Security(TLS v1) 프로토콜을 통해 강력한 암호화를 제공하기 위해 OpenSSL 라이브러리를 사용한다. 이 모듈은 신뢰되는 증명서를 제공하는 곳에 증명서를 요청하기 위해 필요한 모든 것을 제공하기 때문에 FreeBSD 에서 보안 웹 서버를 운용할 수 있다.

아직 아파치를 설치하지 않았다면 **mod_ssl**이 포함된 **아파치** 버전을 www/apache13-modssl 포트로 설치할 수 있을 것이다.

23.7.4.2 mod_perl

아파치/펄 통합 프로젝트는 강력한 펄 프로그래밍 언어와 아파치 웹 서버를 통합하고 있다. **mod_perl** 모듈로 전적으로 펄을 이용한 아파치 모듈을 작성할 수 있다. 게다가 서버에 내장된 인터프리터로 외부 인터프리터의 오버헤드와 펄을 시작할 때의 페널티(penalty)를 최소화할 수 있다.

아직 **아파치**를 설치하지 않았다면 mod_perl이 포함된 아파치 버전을 www/apache13-modperl 포트로 설치할 수 있을 것이다.

23.7.4.3 PHP

"PHP: Hypertext Preprocessor" PHP 는 오픈 소스의 일반적인 스크립트 언어에 광범위하게 사용되고 특히 웹 개발과 HTML 에 내장되는 곳에 적합하다. 구문이 C, Java 그리고 펄과 비슷하기 때문에 배우기 쉽다. 언어의 가장 주된 목적은 웹 개발자들이 동적으로 생성되는 웹 페이지를 빠르게 만들 수 있게 하지만 PHP 로 더 많은 것을 할 수 있다.

PHP는 lang/php5 포트로 설치할 수 있을 것이다.

23.8 파일 전송 프로토콜 (FTP)

23.8.1 개요

파일 전송 프로토콜(FTP)는 FTP 서버로부터 유저에게 간편하게 파일을 전송하는 방법을 제공한다. FreeBSD 는 기본 시스템에 FTP 서버 소프트웨어 **ftpd** 가 포함되어 있다. 여기서는 FreeBSD 에서 FTP 서버를 설치하고 관리하도록 아주 직선적으로 설명한다.

23.8.2 설정

가장 중요한 설정 단계는 FTP 서버를 어떤 계정이 사용할 수 있도록 할 것인지 선택하는 것이다. 일반적인 FreeBSD 시스템은 다양한 데몬에 사용하는 수많은 시스템 계정을 가지고 있지만 unknown 유저가 이들 계정에 로그인할 수 없어야 된다. /etc/ftpusers 파일은 FTP 를 사용할 수 없는 유저를 나열하고 있다. 기본적으로 앞에서 말한 시스템 계정이 포함되어 있지만 FTP 를 사용하지 말아야 되는 유저를 이곳에 추가할 수도 있다.

어떤 유저의 FTP 사용을 완전히 막지 않고 제한하고 싶을 것이다. 이것은 /etc/ftpchroot 파일로 설정할 수 있다. 이 파일에 FTP 사용을 제한하려는 유저와 그룹을 나열한다. ftpchroot(5) 매뉴얼 페이지에 자세한 설명이 있기 때문에 여기서는 더 이상 설명하지 않는다.

여러분의 서버에 익명 FTP 사용을 허용하려면 FreeBSD 시스템에 ftp 라는 이름의 유저를 생성해야 된다. 그러면 유저는 유저 이름 ftp 나 anonymous 와 패스워드(전통적으로 패스워드에 유저의 전자 메일 주소를 사용한다)로 여러분의 FTP 서버에 로그인할 수 있다. 익명 유저가 로그인하면 ftp 유저의 홈 디렉터리만 사용하도록 제한하기 위해 FTP 서버는 chroot(2)를 호출한다.

FTP 클라이언트에 보여주기 위해 환영 메시지를 입력할 수 있는 두 개의 텍스트 파일이 있다. /etc/ftpwelcom 파일의 내용이 로그인 프롬프트가 나타나기 전에 유저들에게 보여진다. 로그인에 성공하면 /etc/ftpmotd 파일의 내용을 보여준다. 이 파일 경로는 로그인 환경과 상대적이기 때문에 익명 유저에게 ~ftp/etc/ftpmotd 가 보여진다.

FTP 서버를 정확히 설정한 후 /etc/inetd.conf 를 활성화해야 된다. 활성화하는 방법은 단순히 **ftpd** 라인 앞부분의 # 표시를 삭제한다:

```
ftp     stream  tcp nowait  root    /usr/libexec/ftpd   ftpd -l
```

예제 23-1 에서 설명한 것처럼 이 설정파일을 수정한 후 inetd 에게 설정파일을 다시 읽도록 신호를 보내야 된다.

다음과 같이 입력하여 FTP 서버에 로그인할 수 있다:

```
% ftp localhost
```

23.8.3 관리

ftpd 데몬은 로그 메시지에 syslog(3)를 사용한다. 기본적으로 시스템 로그 데몬은 /var/log/xferlog 파일의 FTP 관련된 위치에 메시지를 저장한다. FTP 로그는 /etc/syslog.ocnf 파일의 다음 라인을 변경하여 수정할 수 있다:

```
ftp.info        /var/log/xferlog
```

익명 FTP 서버를 운영하면 중요한 문제가 생길 수 있다는 것을 알고 있어야 된다. 특히 익명 유저가 파일을 업로드 할 수 있도록 하는 것에 대해 다시 생각해 본다. 여러분의 FTP 사이트가 불법 소프트웨어의 거래 장소가 되는 것을 알게 될 것이다. FTP 업로드를 허용할

필요가 있다면 퍼미션을 설정하여 다른 익명 유저들이 이들 파일을 볼 수 없도록 한다.

23.9 Microsoft® Windows® 클라이언트를 위한 파일과 프린트 서비스 (Samba)

23.9.1 개요

삼바는 마이크로소프트 윈도우 클라이언트에 파일과 프린트 서비스를 제공하는 유명한 오픈 소스 소프트웨어 패키지다. 이러한 클라이언트는 FreeBSD 의 로컬 디스크 드라이브나 로컬 프린터에 연결하여 사용할 수 있다.

삼바 소프트웨어 패키지는 FreeBSD 설치 미디어에 포함되어 있다. FreeBSD를 처음 설치할 때 삼바를 설치하지 않았다면 net/samba3 포트 패키지로 설치할 수 있다.

23.9.2 설정

기본 삼바 설정 파일은 /usr/local/etc/smb.conf.default 에 설치된다. 이 파일을 /usr/local/etc/smb.conf 에 복사하여 삼바를 사용하기 전에 수정한다.

smb.conf 파일은 프린터와 윈도우 클라이언트에 공유하려는 "파일시스템" 정의와 같은 **삼바**의 런타임 설정정보를 포함하고 있다. **삼바** 패키지는 smb.conf 파일을 쉽게 설정할 수 있는 **swat** 라는 웹 기반 툴을 포함하고 있다.

23.9.2.1 삼바 웹 기반 툴 사용 (SWAT)

삼바 웹 기반 툴(SWAT)은 inetd 데몬으로 실행된다. 따라서 삼바 설정에 swat 을 사용하기 전에 /etc/inetd.conf 에서 다음 라인의 주석을 풀어야 된다:

```
swat    stream   tcp      nowait/400      root     /usr/local/sbin/swat
```

예제 23-1 에서 설명하듯이 이 설정파일을 수정한 후 **inetd** 가 설정파일을 다시 읽도록

해야 된다.

inetd.conf에서 **swat**이 활성화되면 브라우저에 http://localhost:901을 입력하여 연결할 수 있다. 처음엔 시스템 root 계정으로 로그인해야 된다.

메인 삼바 설정 페이지에 성공적으로 로그인한 후 시스템 문서를 탐색하거나 'Globals' 탭을 클릭하여 설정을 시작할 수 있다. 이 Globals 섹션은 /usr/local/etc/smb.conf 의 *[global]*에 설정된 변수와 일치한다.

23.9.2.2 Global 설정

swat 를 사용하거나 /usr/local/etc/smb.conf 를 직접 수정하더라도 **삼바**를 설정할 때 마주치게 될 첫 번째 설정은 다음과 같다:

workgroup

이 서버를 사용할 컴퓨터의 NT 도메인 이름이나 Workgroup 이름 설정.

netbios name

삼바 서버가 알려질 NetBIOS 이름을 설정한다. 기본적으로 호스트 DNS 이름의 첫 번째 컴포넌트와 같다.

server string

net view 명령과 서버에 대한 설명을 찾는 다른 네트워크 툴에 표시되는 문자열을 설정한다.

23.9.2.3 보안 설정

/usr/local/etc/smb.conf 에서 가장 중요한 두 가지 설정은 보안 모델 선택과 클라이언트 유저의 패스워드 포맷이다. 다음 설정이 이들 옵션을 제어한다:

security

여기서 가장 일반적인 두 가지 옵션은 *security = share* 와 *security = user* 다. FreeBSD 머신의 유저이름과 같은 유저 이름을 클라이언트가 사용한다면 user 레벨 보안을 사용하려고 할 것이다. 이것이 기본 보안 정책이고 공유된 자원에 접근하기 전에 클라이언트는 로그인이 필요하다.

share 레벨 보안에서 공유된 자원에 연결하기 전에 클라이언트는 유효한 유저 이름과 패스워드로 서버에 로그인할 필요가 없다. 이 설정이 예전 **삼바** 버전의 기본 보안 모델이다.

passdb backed

삼바는 여러 개의 backend 인증 모델을 가지고 있다. LADP, NIS+, SQL 데이터베이스나 수정된 패스워드 파일로 클라이언트를 인증할 수 있다. 기본 인증 방법은 *smbpasswd* 이고 여기서 설명한다.

smbpasswd 를 사용한다면 **삼바**가 클라이언트를 인증할 수 있도록 /usr/local/private/smbpasswd 파일을 생성해야 된다. 모든 유닉스 유저 계정들에게 접근 권한을 주려면 다음 명령을 사용한다:

```
# cat /etc/passwd | grep -v "^#" | make_smbpasswd > /usr/local/private/smbpasswd
# chmod 600 /usr/local/private/smbpasswd
```

설정 옵션에 대한 추가적인 정보는 **삼바** 문서를 확인해 본다. 이곳의 기본적인 개요에서 **삼바**를 실행하기 위해 필요한 모든 것을 얻을 수 있다.

23.9.3 삼바 시작

시스템이 부팅할 때 삼바를 시작하려면 /etc/rc.conf 에 다음 라인을 추가한다:

```
samba_enable="YES"
```

다음 명령을 실행하여 언제라도 **삼바**를 시작할 수 있다:

```
# /usr/local/etc/rc.d/samba.sh start
Starting SAMBA: removing stale tdbs :
Starting nmbd.
Starting smbd.
```

삼바는 사실 3 개의 데몬으로 구성되어 있다. **nmbd** 와 **smbd** 데몬은 samba.sh 스크립트가 시작한다. smb.conf 에서 winbind name resolution 서비스를 활성화했다면 winbind 데몬도 시작되는 것을 볼 수 있다.

다음 명령으로 언제라도 **삼바**를 정지 시킬 수 있다:

```
# /usr/local/etc/rc.d/samba.sh stop
```

삼바는 마이크로소프트 윈도우 네트워크와 기능적으로 통합되는 복잡한 소프트웨어다. 여기서 설명하는 기본 설치 이상의 기능에 대한 더 많은 정보는 http://www.samba.org를 확인한다.

23.10 NTP

23.10.1 개요

시간이 지나면 컴퓨터의 시간은 정확성이 떨어진다. NTP(네트워크 시간 프로토콜)는 시간을 정확하게 하는 한가지 방법이다.

많은 인터넷 서비스는 컴퓨터 시간과 연관성이 있고 시간이 정확해야 이점을 얻을 수 있다. 예를 들어 웹 서버는 파일이 특정 시간 이후에 수정되었다면 이 파일을 보내달라는 요청을 받을 것이다. cron(8)과 같은 서비스는 주어진 시간에 명령을 수행한다. 시간이 맞지 않는다면 이들 명령은 예정된 시간에 실행되지 않는다.

FreeBSD 는 머신의 시간을 맞추기 위해 다른 NTP 서버에 쿼리를 보내거나 다른 머신에 시간 서비스를 제공하는데 사용되는 NTP 서버 ntpd(8)을 가지고 있다.

23.10.2 적당한 NTP 서버 선택

시간을 동기화하려면 한대 또는 한대 이상의 NTP 서버를 찾아야 한다. 여러분의 네트워크 관리자나 ISP 가 이런 목적으로 NTP 서버를 설정하였을 것이다 -- 이러한 경우 그들이 제공하는 문서를 확인한다. 여러분에게 가까운 NTP 서버를 찾아서 공적으로 사용할 수 있는 NTP 서버 리스트가 있다(http://www.eecis.udel.edu/~mills/ntp/servers.html). 선택한 서버의 정책을 인지하고 필요한 경우 허락을 얻는다.

여러분이 사용하던 서버를 사용할 수 없거나 시간이 부정확할 수 있기 때문에 연결이 되지 않은 여러 개의 NTP 서버를 선택하는 것도 좋은 생각이다. ntpd(8)은 다른 서버의 응답을 지능적으로 받는다 -- 이것은 안정된 서버보다 불안정한 서버에 더 유용하다.

23.10.3 머신 설정

23.10.3.1 기본 설정

머신이 부팅할 때 시간을 동기화하려면 ntpdate(8)을 사용할 수 있다. 이것은 자주 재부팅하고 가끔 동기화가 필요한 몇몇 데스크톱 머신에는 적당하지만 대부분의 머신은 ntpd(8)을 사용하는 것이 좋다.

부팅할 때 ntpdate(8)을 사용하는 것은 ntpd(8)을 운용하는 머신에도 좋은 생각이다. ntpd(8)은 시간을 점차적으로 변경하지만 ntpdate(8)은 시간을 설정한다. 따라서 현재 머신에 설정된 시간이 정확한 시간과 얼마나 큰 차이가 있는지 문제되지 않는다.

부팅할 때 ntpdate(8)을 활성화하려면 /etc/rc.conf 에 *ntpdate_enable="YES"*를 추가한다. 그리고 동기화하려는 모든 서버를 ntpdate(8)에 적용할 플래그와 함께 *ntpdate_flags* 에 지정해야 된다.

23.10.3.2 일반적인 설정

NTP 는 ntp.conf(5)에 설명된 포맷으로 /etc/ntp.conf 에 설정한다. 여기 샘플 예제가 있다:

```
server ntplocal.example.com prefer
server timeserver.example.org
server ntp2a.example.net

driftfile /var/db/ntp.drift
```

server 옵션으로 어떤 서버를 사용할지 각 라인에 서버 한대씩 지정한다. 서버가 ntplocal.example.com 과 같이 prefer 인자로 지정되어 있다면 이 서버는 다른 서버보다 우선권이 있다. 우선권이 있는 서버의 응답이 다른 서버의 응답과 많은 차이가 있다면 폐기되고 그렇지 않으면 다른 응답을 고려하지 않고 사용된다. *prefer* 인자는 보통 특별한 시간 모니터링 하드웨어처럼 아주 정확하다고 알려진 NTP 서버에 사용된다.

driftfile 옵션은 시스템 시간의 빈번한 옵셋을 저장할 파일을 지정한다. ntpd(8)은 시간이 맞지 않는 것을 자동으로 보상하는데 이 옵션을 사용하고, 외부의 모든 시간이 일정 시간 동안 차단되더라도 적절히 정확한 설정을 유지한다.

driftfile 옵션은 사용중인 NTP 서버의 기존 응답에 대한 정보를 저장할 파일을 지정한다. 이 파일은 NTP 의 내적인 정보를 포함한다. 다른 프로세스가 이 파일을 수정하면 안 된다.

23.10.3.3 서버 접근 제어

기본적으로 NTP 서버는 인터넷의 모든 호스트가 접근할 수 있다. /etc/ntp.conf 의 *restrict* 옵션으로 서버에 접근할 머신을 제어할 수 있다.

NTP 서버에 모든 머신의 접근을 거부하려면 다음 라인을 /etc/ntp.conf 에 추가한다:

```
restrict default ignore
```

여러분의 네트워크에 있는 머신만 서버로 시간을 동기화하고, 서버를 설정하거나 동기화를 거부하지 않을 것이라면 다음 라인을 추가한다:

```
restrict 192.168.1.0 mask 255.255.255.0 notrust nomodify notrap
```

192.168.1.0 은 여러분의 네트워크 주소고 255.255.255.0 는 넷 마스크다.

/etc/ntp.conf 에 여러 개의 *restrict* 옵션을 지정할 수 있다. 더 자세한 사항은 ntp.conf(5)의 접근제어 지원에 대한 부분을 본다.

23.10.4 NTP 서버 운용

부팅할 때 NTP 서버를 시작하려면 *xntpd_enable="YES"* 라인을 /etc/rc.conf 에 추가한다. 추가적인 플래그를 ntpd(8)에 적용하려면 /etc/rc.conf 의 *xntpd_flags* 매개변수를 수정한다.

머신을 재 부팅하지 않고 서버를 시작하려면 /etc/rc.conf 의 *xntpd_flags* 에 부가적인 매개변수를 지정하고 ntpd 를 실행한다. 예를 들면 다음과 같이 실행한다:

```
# ntpd -p /var/run/ntpd.pid
```

> **Note:** FreeBSD 5.X 에서 /etc/rc.conf 의 여러 가지 옵션 이름이 바뀌었다. 따라서 *xntpd* 의 모든 인스턴스를 위 옵션의 *ntpd* 로 대체한다.

23.10.5 일시적인 인터넷 연결에 ntpd(8) 사용

ntpd 의 적절한 기능 때문에 인터넷에 계속적으로 연결할 필요가 없다. 그러나 요청에 의해 전화접속으로 일시적으로 연결한다면 전화 걸기를 시작하거나 연결을 유지하는 동안 NTP 트래픽을 방지하는 것도 좋은 생각이다. 유저 PPP 를 사용한다면 /etc/ppp/ppp.conf 에 *filter* 기능을 사용할 수 있다. 예를 들면 다음과 같이 설정할 수 있다:

```
set filter dial 0 deny udp src eq 123
# Prevent NTP traffic from initiating dial out
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Prevent incoming NTP traffic from keeping the connection open
set filter alive 1 deny udp dst eq 123
# Prevent outgoing NTP traffic from keeping the connection open
set filter alive 2 permit 0/0 0/0
```

더 자세한 사항은 ppp(8)의 패킷 필터링 섹션과 /usr/share/examples/ppp/의 예제를

확인한다.

> Note: 어떤 ISP 는 미리 정의되어 있는 포트번호를 막아서 응답이 머신에 도달하지 못하므로 NTP 기능을 사용하지 못한다.

23.10.6 더 많은 정보

NTP 서버 문서는 /usr/share/doc/ntp/에서 HTML 문서를 찾을 수 있다.