DNS HOWTO

Nicolai Langfeldt
Jamie Norrish
and others.

janl (at) linpro.no

도운이

김지환

e4261 (at) hanmail.net
http://www.freechal.com/linuxdocs

Version 3.1, $Date: 2001/09/02 09:41:09 $

HOWTO become a totally small time DNS admin.

1. 서문

이문서는 LDP의 일부입니다.

1.1. 법적효력(원문그대로 실음)

1.2. 도움에 대한 신용과 요청

이 문서 초안을 수도 없이 읽어 주고 많은 제안을 해준 Arnt Gulbrandsen 씨에게 감사의 말을 전하는 것이 순서에 맞을 것 같다. 또한 e-mail로 의견과 유용한 내용을 보낸 준 사람들에게도 감사의 말을 전한다.

이 문서는 완결된 문서가 아니다. DNS를 설정하여 사용할 때 발생하는 문제점이나 그에 따른 해결책들이 있을 것이다. 그러한 내용들은 e-mail로 보내 준다면, 다음 번에는 더 좋은 DNS-HOWTO가 나올 수 있을 것이다. money나 의견 또는 의문점은 janl (at) math.uio.no 앞으로 보내 주길 바란다. 아니면 내 DNS 책을 사기 바란다. 아니면 여러 정보를 찾아보길 또한 바란다. e-mail을 보내기 전에 반드시 자신의 e-mail 주소가 올바른지 확인하도록 한다. 그래야 답신을 받을 수 있다는 것은 당연한 이야기 일 것이다. 또한 메일을 보내기 전에 질문과 답 절을 읽어 보기 바란다. 하나더. 나는 노르웨이어와 영어만 할 줄 안다는 것도 알아두길.

이 HOWTO문서는 LDP의 일부로 1995년부터 있었고 2000년까지 같은 주제로 글을 썼다. 이것이 HOWTO문서이고 책에 유사한 내용이 많아도 이것이 이것저것 희석하거나 시중의 책을 배낀것이 아님을 알아주길 바란다. 그리고 이책이 참고한 문헌은 끝에서 확인할 수 있다. 이 HOWTO를 읽는 분들이 DNS의 어려운점을 여기에서 해결하기를 바란다. 또한 이 문서는 책을 보조하는 것이지만, 반대로 이 문서를 보조하기 위해 책을 보는 경우도 있을 것이다. HOWTO가 책을 생기게 했고 책이 이 HOWTO 버전을 3으로 만들었다. 나에게 기회를 준 Que 출판사에 감사를 하는 바이다.

1.3. 헌정

이 HOWTO를 Anne Line Norheim Langfeldt에 바친다. 그녀는 이 문서를 읽어 보지도 않겠지만, 그녀는 정말로 특별한 여자이다.

2. 소개

DNS란 무엇인가?

DNS는 Domain Name Server이다. DNS는 모든 넷상에 속해있는 컴퓨터가 가지고 있는 이름을 IP주소로 바꾸어 주는 역할을 한다. 이것은 이름을 IP로 바꾸어 주기도 하고 IP를 이름이나 다른 것으로 바꾸어 주기도 한다(혹은 mapping한다고도 한다). 이 HOWTO 문서는 리눅스에서 특수화된 몇가지 부분을 포함한 유닉스시스템의 이와같은 mapping 용례에 대해 정의할 것이다.

이러한 mapping은 단순하게는 `ftp.linux.org.`같은 컴퓨터의 이름과 `199.249.150.4` 같은 IP 넘버(혹은 어드레스)사이의 조합이라고 볼 수 있다. 또한 DNS는 다른 방식의 mapping도 할 수 있는데 이것은 'reverse mapping'이라고 하며 IP를 컴퓨터 이름으로 바꾸어 주는 것이다.

DNS는 네트워크 관리 부분에서 상당히 모호한 부분이지만 실제로 그렇게 어렵지는 않다. 이 문서는 몇가지 부분을 명확하게 하려고 노력할 것이다. 도메인에 대한 1차 DNS서버에 대한 설정과 케쉬서버를 포함해서 간단한 DNS name server에 대한 설정작업에 대해 기술할 것이다. 더 복잡한 설정은 이 문서의 Q&A부분을 참조하길.... 이 문서에 없는 부분은 상세한 문서를 참조하길 바란다. 마지막 장에 상세한 문서에 대한 부분을 기술할 것이다.

시작하기 전에 telnet을 사용가능하게 하며, 다른 넷상의 모든 연결이 가능하게 설정이 되야한다. 그리고 `telnet 127.0.0.1` 이 실행시 자신의 컴퓨터에 대한 정보를 가져와야 한다.(실행해 보시길) 또한 시작부분인 `/etc/nsswitch.conf`, `/etc/resolv.conf`, `/etc/hosts` 파일이 올바르게 설정되어야 한다. 여기에서는 이들의 기능에 대해서는 설명하지 않는다. 이러한 전반적인 기능이 수행이 되지 않는다면 Howto의 Networking 부분을 참조하고 읽어보길 바란다. (모르면 알아서 공부하라는 뜻같군....--.)

이 문서에서 '컴퓨터'는 일반적으로 이러한 DNS 설정이 되어 있는 것을 의미하며 그외의 일반적인 컴퓨터를 통칭하는 부분은 네트워크와 무관한 경우이다.

이름질의(name query)를 막는 방화벽은 제거되야 하며 특별한 설정이 요구되면 역시 Q&A를 참고하라.

유닉스 상의 네임서버기능은`named`라는 프로그램에 의해 이루어 진다. 이것은 BIND 패키지에 있는 것이며 ISC(Internet Sotfware Consortium)이 제공한다. 이 `named`는 거의 모든 리눅스 프로그램에 포함되어 있는 것이며 이것은 일반적으로 `BIND` 패키지로 설치된 경우 `/usr/sbin/named`부분에 위치하게 된다.

만약 컴퓨터에 named가 있으면 바로 사용이 가능하다. 만약에 설치가 되있지 않다면 ftp://ftp.isc.org/isc/bind/src/에서 최신 버젼과 소스를 구할 수 있다. 이것은 BIND 버젼 8에 대한 문서이다. 구버젼인 버젼4의 경우는 http://www.math.uio.no/~janl/DNS/에서 자료를 구할 수 있을 것이다. 만약 버젼이 8이라면 named의 manpage에서 `named.conf`를 언급할 것이고 버젼이 4라면 `named.boot`에 대한 언급을 할 것이다. 만약 버젼 4를 가지고 있다면 보안의 측면을 고려해서 버젼 8로 버젼업을 해야 할 것이다

DNS는 광범위한 넷상의 데이터베이스이다. 무엇이 삽입되어야 하는지 신중하게 고려해야 한다. 쓸모없는 정보가 들어간다면 보는 이들 역시 쓸모 없는 정보를 얻게 된다. DNS를 간소하고 필요한 것으로 체워야 하며, 그러면 좋은 서비스를 받게 될 것이다. 사용법, 유지법, 디버깅법을 배우면 잘못된 관리로 인한 넷상에서의 실패를 방지하는 훌륭한 유지를 하게 될 것이다.

> 작은 정보**:** 이러한 파일에 대한 모든 백업 파일을 만드시오. 그래서 작업이후 작동이 되지 않으면 다시 복구할 수 있도록 대비하시길....

3. caching name server의 구축

우선 **DNS** 설정은 다이얼업**,** 케이블모뎀**, ADSL** 사용자들에게 매우 유용하다**.**

레드헷이나 레드헷 기반 배포본은 이미 `bind`나 `bind-utils`와 caching nameserver가 설치되었다. 데비안 사용자라면 `bind`와`bind-doc`가 설치되었을 것이다. 물론 설치시 이런 문서처럼 자세하게 설치에 대해 나오지 않을 것이고, 설치된 파일을 설정하는 법을 같이 읽어야 할 것이다.

caching nameserver는 단지 이름질의에 대한 대답만을 찾고 그 해답을 기억해서 다음에 필요할때 사용가능하게 한다. 이것은 기다리는 시간을 단축시킬 수 있다.

우선 `/etc/named.conf` (Debian: `/etc/bind/named.conf`)가 필요하며 이것은 named가 구동될 때 필요한 것이다.포함된 내용을 살펴보자

```
// Config file for caching only name server

options {
directory "/var/named";

// Uncommenting this might help if you have to go through a
// firewall and things are not working out.  But you probably
// need to talk to your firewall admin.

// query-source port 53;
```

```
};

zone "." {
        type hint;
        file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "pz/127.0.0";
};
```

리눅스 배포본이 여기에서 언급되는 모든 종류의 파일이름과 다른 이름을 쓸지도 모를 일이지만 안의 내용을 위와 같이 모두 같다.

'directory' : named프로그램이 설정 파일을 찾는 위치를 지정해 준다. named에 관련된 파일들은 위와 같다. 예를 들어 이후에 pz란 디렉토리 가 나오면 그것은 /var/named/pz라는 것을 의미하는 것일 것이다. 위의 예시와 같은 설정이 일반적인 설정이다.

/var/named/root.hints라는 파일이 안에 지정되 있다. /var/named/root.hints는 아래의 내용을 포함해야 한다. (만약 이 문서에서 이 내용들을 잘라서 첨부하게 된다면, 파일 앞에 있는 공백들을 제거하기를 바란다. 모든 라인의 시작은 공백이 없게 구성이 되어 있다. 몇몇 문서 프로그램들은 인위적으로 이러한 공백들을 만들기도 하는데 모든 초기 공백들은 제거되야 한다.

```
;
; There might be opening comments here if you already have this file.
; If not don't worry.
;
.                       6D IN NS        M.ROOT-SERVERS.NET.
.                       6D IN NS        I.ROOT-SERVERS.NET.
.                       6D IN NS        E.ROOT-SERVERS.NET.
.                       6D IN NS        D.ROOT-SERVERS.NET.
.                       6D IN NS        A.ROOT-SERVERS.NET.
.                       6D IN NS        H.ROOT-SERVERS.NET.
.                       6D IN NS        C.ROOT-SERVERS.NET.
.                       6D IN NS        G.ROOT-SERVERS.NET.
.                       6D IN NS        F.ROOT-SERVERS.NET.
.                       6D IN NS        B.ROOT-SERVERS.NET.
.                       6D IN NS        J.ROOT-SERVERS.NET.
.                       6D IN NS        K.ROOT-SERVERS.NET.
.                       6D IN NS        L.ROOT-SERVERS.NET.
;
M.ROOT-SERVERS.NET.     6D IN A         202.12.27.33
I.ROOT-SERVERS.NET.     6D IN A         192.36.148.17
E.ROOT-SERVERS.NET.     6D IN A         192.203.230.10
D.ROOT-SERVERS.NET.     6D IN A         128.8.10.90
A.ROOT-SERVERS.NET.     6D IN A         198.41.0.4
H.ROOT-SERVERS.NET.     6D IN A         128.63.2.53
```

```
C.ROOT-SERVERS.NET.     6D IN A         192.33.4.12
G.ROOT-SERVERS.NET.     6D IN A         192.112.36.4
F.ROOT-SERVERS.NET.     6D IN A         192.5.5.241
B.ROOT-SERVERS.NET.     6D IN A         128.9.0.107
J.ROOT-SERVERS.NET.     6D IN A         198.41.0.10
K.ROOT-SERVERS.NET.     6D IN A         193.0.14.129
L.ROOT-SERVERS.NET.     6D IN A         198.32.64.12
```

이 파일은 세상에 존재하는 root name server에 대하여 기술해 놓는다. 이 서버들은 시간마다 바뀌고 항상 현재 상태로 유지되야 한다. 유지섹션(maintain section)에서 어떤 방식으로 업데이트가 이루어 지는지 확인하길....

named.conf의 마지막 부분인 zone 부분에 대해서는 마지막 장에 기술해 놓기로 한다. 지금은 단지 pz 디렉토리 안에 있는 127.0.0이라는 파일 구성에 대해 보기로 한다. (누차 이야기하지만 잘라 넣을 시 초기 공백은 제외하라....)

```
$TTL 3D
@               IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
1       ; Serial
8H ; Refresh
2H      ; Retry
4W ; Expire
1D) ; Minimum TTL
NS      ns.linux.bogus.
1 PTR localhost.
```

다음으로/etc/resolv.conf부분을 확인 하도록 하자.

```
search subdomain.your-domain.edu your-domain.edu
nameserver 127.0.0.1
```

'search'부분은 연결하고자 하는 호스트 이름은 어느 도메인에서 찾아져야 하는가를 결정해 준다.'nameserver'부분은 nameserver의 주소부분을 나타내는데 위와 같은 경우는 named가 구동되는 서버 자신을 의미한다.(127.0.0.1 이 정상이다. 다른 컴퓨터들이 마찬가지로 이 주소를 가지고 있다고 해도 문제될 것 없다.) 원한다면 'nameserver'부분에 여러 nameserver들을 쓸 수도 있다. (Note1:named 데몬이 이 파일을 읽는 것은 아니다. named가 작동하는 것을 이용하는 resolver들이 읽는다, Note2: 이 파일(resolv.conf)에서 domain이란 용어가 사용된다. 그러나 "search"와 "domain"이 같이 사용되진 않는다. 둘중 하나만이 작동할 것이다.)

이러한 파일의 작동을 확인하기 위해서: 만약 client가 foo라는 것을 찾는다면, foo.subdomain.your-domain.edu을 우선 확인하고, 그리고 나서 foo.your-domain.edu, 마지막으로 foo. 를 찾게 된다. search부분에서 너무많은 도메인을 넣기 않는게 좋다. 결국 기재된 모든 부분을 찾는데 시간이 소요되기 때문이다.

이 예에서 subdomain.your-domain.edu에 속해있다고 볼때, 여러분들의 컴퓨터는 your-machine.subdomain.your-domain.edu라고 불리게 될 것이다. 이러한 search 라인에

여러분들의 TLD(Top Level Domain, 예를들어edu)을 포함하면 안된다. 만약 다른 도메인을 연결하려면 예시와 같이 추가연결하면 된다.

```
search subdomain.your-domain.edu your-domain.edu other-domain.com
```

정확히 이야기 한다면 이런 예를 바탕으로 실제 도메인명을 써야한다. 도메인 명 마지막에 점이 없다는 것을 유의하자. 중요하다. 다시 상기하길....

3.1. named의 시작

이제 named를 시작할 시간이다. 만약에 전화접속식이면 접속부터 해라. 'ndc start'라고 코딩하고 실행해 보길... 옵션은 없이 실행한다. 데몬이 생성되지 않으면 '/usr/sbin/ndc start'라고 코딩해서 실행해 보시기를. 그래도 안되면 Q&A를 참조하라. named가 구동되는 동안에 syslog 파일을 보면 이러한 식의 결과를 보게 될 것이다.(보통 이 log파일은 /var/adm/messages안에 있거나/var/log 안에 위치하게 된다. 이 결과를 보려면 'tail -f /var/log/messages' 를 입력하면 된다.

(\은 연결되는 다음줄을 이야기한다.)

```
Dec 15 23:53:29 localhost named[3768]: starting.  named 8.2.2-P7 \
Fri Nov 10 04:50:23 EST 2000 ^Iprospector@porky.\
devel.redhat.com:/usr/src/bs/BUILD/bind-8.2.2_P7/\
src/bin/named
Dec 15 23:53:29 localhost named[3768]: hint zone "" (IN) loaded\
(serial 0)
Dec 15 23:53:29 localhost named[3768]: Zone "0.0.127.in-addr.arpa"\
(file pz/127.0.0): No default TTL set using SOA\
minimum instead
Dec 15 23:53:29 localhost named[3768]: master zone\
"0.0.127.in-addr.arpa" (IN) loaded (serial 1)
Dec 15 23:53:29 localhost named[3768]: listening on [127.0.0.1].53 (lo)
Dec 15 23:53:29 localhost named[3768]: listening on [10.0.0.129].53\
(wvlan0)
Dec 15 23:53:29 localhost named[3768]: Forwarding source address is\
[0.0.0.0].1034
Dec 15 23:53:29 localhost named[3769]: Ready to answer queries.
```

만약 여기에 에러 메시지가 나온다면 뭔가 실수가 있는 것이다. named 가 위 예에서처럼 나오는 파일들을 지정해 주기 때문에 에러시 다시 원점으로 돌아가서 여기에 나오는 파일을 체크하도록 하자.

다음은 셋업된 부분을 체크해 보도록 한다. 전통적으로 nslookup 파일을 사용했지만 요즘은 dig dig를 사용한다.

```
$ dig -x 127.0.0.1
```

```
; <<>> DiG 8.2 <<>> -x
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; QUERY SECTION:
;;      1.0.0.127.in-addr.arpa, type = ANY, class = IN

;; ANSWER SECTION:
1.0.0.127.in-addr.arpa.  1D IN PTR  localhost.

;; AUTHORITY SECTION:
0.0.127.in-addr.arpa.   1D IN NS        ns.penguin.bv.

;; Total query time: 30 msec
;; FROM: lookfar to SERVER: default -- 127.0.0.1
;; WHEN: Sat Dec 16 00:16:12 2000
;; MSG SIZE  sent: 40  rcvd: 110
```

이러한 식으로 나온다면 제대로 동작하는 것이다. 그렇지 않으면 돌아가서 다시 처음부터 체크해 보도록. `named.conf`를 수정할때 마다 `ndc restart`를 수행해야 한다.

이제 질의를 던져보자. 당신과 가까이 있는 컴퓨터를 조사해 봐라. 나는 오슬로대학의 `pat.uio.no`가 제일 가깝다.

```
$<65533>ig pat.uio.no

; <<>> DiG 8.2 <<>> pat.uio.no
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
;; QUERY SECTION:
;;      pat.uio.no, type = A, class = IN

;; ANSWER SECTION:
pat.uio.no.             1D IN A         129.240.130.16

;; AUTHORITY SECTION:
uio.no.                 1D IN NS        nissen.uio.no.
uio.no.                 1D IN NS        ifi.uio.no.
uio.no.                 1D IN NS        nn.uninett.no.

;; ADDITIONAL SECTION:
nissen.uio.no.          1D IN A         129.240.2.3
ifi.uio.no.             1H IN A         129.240.64.2
nn.uninett.no.          1D IN A         158.38.0.181

;; Total query time: 112 msec
```

```
;; FROM: lookfar to SERVER: default -- 127.0.0.1
;; WHEN: Sat Dec 16 00:23:07 2000
;; MSG SIZE  sent: 28  rcvd: 162
```

여기에서 dig는 named에게 `pat.uio.no`라고 명칭된 기계를 찾으라고 요청하게 되고 그러면 `root.hints`파일에 명시된 name server들 중에 하나에 접속하게 되며, 그곳으로 부터 도달하는 방법을 질문한다. 결과를 얻기 전에 약간의 시간이 걸릴 수도 있는데 이것은 `/etc/resolv.conf` 에 지정된 모든 도메인을 찾을 경우가 있기 때문이다. flag: 부분의 aa부분을 주목하길. 이것은 질의에 대한 결과가 인증되었다는 것을 의미한다. 다시 말하면 인증된 서버로 부터 갱신이 되었다는 것을 의미한다. 다음에 이 인증부분에 대해 다시 설명할 것이다.

만약 다시 똑같은 질의를 보낸다면 다름과 같은 결과를 얻게 될 것이다.

```
$ dig pat.uio.no

; <<>> DiG 8.2 <<>> pat.uio.no
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
;; QUERY SECTION:
;;      pat.uio.no, type = A, class = IN

;; ANSWER SECTION:
pat.uio.no.             23h59m58s IN A  129.240.130.16

;; AUTHORITY SECTION:
UIO.NO.                 23h59m58s IN NS  nissen.UIO.NO.
UIO.NO.                 23h59m58s IN NS  ifi.UIO.NO.
UIO.NO.                 23h59m58s IN NS  nn.uninett.NO.

;; ADDITIONAL SECTION:
nissen.UIO.NO.          23h59m58s IN A  129.240.2.3
ifi.UIO.NO.             1d23h59m58s IN A  129.240.64.2
nn.uninett.NO.          1d23h59m58s IN A  158.38.0.181

;; Total query time: 4 msec
;; FROM: lookfar to SERVER: default -- 127.0.0.1
;; WHEN: Sat Dec 16 00:23:09 2000
;; MSG SIZE  sent: 28  rcvd: 162
```

질의에 대한 결과 부분에서 "flag"부분의 "aa"가 없다는 사실을 주목하길. 이것은 cache가 정보를 가지고 있을때 더이상 named가 질의를 하기 위해 네트워크 밖으로 나가지 않는다는 것을 의미한다. 저장된 정보가 오래 되었을 수도 있다. 매우 희박하겠지만 가끔 이러한 정보를 통해서 "aa"가 없는지를 확인해야 할 것이다. 이제 자신의 cache서버가 작동한다는 것을 알게 되었다.

3.2. Resolvers

표준 C API가 구현되는 모든 종류의 OS들은 gethostbyname 과 gethostbyaddr를 호출한다. 이러한 함수(아마도 API구현 함수종류인듯 합니다.) 는 다른 여러 소스로 부터 정보들을 가져오게 된다. 가져온 이런 정보들은 리눅스나 다른 유닉스에서는 `/etc/nsswitch.conf` 부분에 설정되어 있는 데로 받게된다. 이것은 여러 다른 파일이나 DB에서 서로 다른 종류의 데이터를 얻는 것이다. 맨위에 자세한 주석이 있으며 반드시 읽기를 바란다. '`hosts:`'로 시작하는 라인을 찾아 보자. 거기에 이렇게 쓰여 있을 것이다.

```
hosts:      files dns
```

(복사할 경우 생기는 공백을 잊지 말기를... 이제 더 이야기 하지 않을 것이다.)

만약 '`hosts:`'라는 라인이 보이지 않는다면 위와같이 기재 해 놓기를... 이것은 프로그램들이 우선 `/etc/hosts`를 찾아야 한다는 것을 이야기 하며 그리고 `resolv.conf`에 지정된 DNS를 체크한다는 이야기 이다.

3.3. 축하

caching name server 설정법을 알게 되었다. 자축하는 의미에서 맥주나 우유, 아니면 다른 걸로 축하하시길...

4. Forwarding

대규모로 잘 조직된 학내망이나 인터넷 서비스업체(ISP)의 네트워크를 보면 가끔 DNS를 포워딩 세팅해서 내부 네트워크에서의 부하와 외부 네트워크로 나가는 부하를 줄이는 경우를 볼 수 있다. 내부에 이러한 네트워크가 존재하는지에 대한 여부를 아는 것은 쉽지 않다. 그게 중요한 것은 아니고 네트워크에서 제공되는 DNS를 "forwarder"로 사용함으로서 질의에 대한 응답을 더 빨리 얻을 수도 있고 네트워크의 부하역시 줄일 수 있다. 모뎀을 사용한다면 아주 좋을 것이다. 네트워크 공급자가 사용하고 싶은 네임 서버 2개를 확보하고 있으며 그 ip가 `10.0.0.1`과 `10.1.0.1` 이라고 가정해 보자. 그러면 `named.conf`부분에 "option"이라고 불리는 부분의 시작에 다음과 같은 식으로 코딩을 추가하면 된다.

```
forward first;
forwarders {
     10.0.0.1;
     10.1.0.1;
 };
```

모뎀사용하는 DNS에게 forwarding은 아주 좋은 방법이다.Q&A에 기술해 놓았다.

nameserver를 재가동하고 dig로 확인해 보기를. 아마 잘 작동할 것이다.

5. 간단한 도메인

도메인 구축 방법.

5.1. 5.1 우선은 상투적인 이론부터

우선적으로 이전에 다른 여러 관련서적을 읽고 숙지했는가? 그래야만한다.

이 장을 시작하기 전에 DNS가 구동되는 약간의 이론에 대해 설명하기로 한다. 그리고 이 장을읽는것이 좋다. 읽기 싫으면 적어도 `named.conf`가 나오기 전까지 속독이라도 하라.

DNS는 tree구조로 되있는 계층구조이다. 일반적인 tree데이터-구조에서 상위는 '.' 으로 되어 있고 이것은 'root'로 다시 불리워 진다. '.' 아래에는 수많은 최상위 도메인들이 있다(TLDs:Top Level Domains);가장 잘 알려진 것은 ORG, COM, EDU, NET 등이 있다. 나무처럼 뿌리(root)도 있고 가지(branch)도 있다고 생각하자. 컴퓨터과학에 대한 지식이 있다면 DNS를 tree부분, 마디, 잎마디, 봉우리로 인식하게 될 것이다. 점(.)이 분기점이고 봉우리가 이름들이다.

질의는 root로 부터 시작되는 구조에 대해서 순환적으로 진행된다. `prep.ai.mit.edu.` 이라는 주소를 찾으려 한다면 네임서버는 어딘가에 질의를 할것이다. 우선은 캐쉬에서부터 질의를 한다. 캐쉬가 답을 가지고 있고 답변이 오게 된다면 바로 전 장에서와 같은 결과를 보게 될 것이다.여기에서 답이 나오지 않으면 이름이 시작되는 왼쪽부터 지워 나가게 된다. `ai.mit.edu.` 그리고는 `mit.edu.` 또 그리고 `edu.`를 네임서버가 알고 있는지 확인하고, 모른다면`.` 에게 질의를 하게 된다. `.` 은 hints파일을 가르키기 때문이다. 그러면 `.` 서버들은 `prep.ai.mit.edu` 에 대해 질의를 하게 되며, `prep.ai.mit.edu` 서버가 답을 모르더라도 다른 곳 어디를 찾아야 할지를 당신의 서버에 알려주고, 찾아간 곳에서 질의에 대한 답을 얻게 된다. 이러한 것을 이제 보여줄 것이다. dig에서 `+norec`는 비순환적인 질의를 보내서 우리에게 돌아온 값만을 얻게 된다.이 옵션은 dig 과정을 줄이고 페이지 수를 줄일것이다. (질의 보내고 다시 질의를 찾는 식의 순환이 아니라 질의 한번에 결과 한번을 보여준다는 의미로 이해됩니다.^^;)

```
$ dig +norec +noH +noques +nostats +nocmd prep.ai.mit.edu.
;; res options: init defnam dnsrch
;; got answer:
; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 13
;; AUTHORITY SECTION:
.                       5d23h48m47s IN NS  I.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  E.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  D.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  A.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  H.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  C.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  G.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  F.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  B.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  J.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  K.ROOT-SERVERS.NET.
```

```
.                       5d23h48m47s IN NS  L.ROOT-SERVERS.NET.
.                       5d23h48m47s IN NS  M.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
I.ROOT-SERVERS.NET.     6d23h48m47s IN A  192.36.148.17
E.ROOT-SERVERS.NET.     6d23h48m47s IN A  192.203.230.10
D.ROOT-SERVERS.NET.     6d23h48m47s IN A  128.8.10.90
A.ROOT-SERVERS.NET.     6d23h48m47s IN A  198.41.0.4
H.ROOT-SERVERS.NET.     6d23h48m47s IN A  128.63.2.53
C.ROOT-SERVERS.NET.     6d23h48m47s IN A  192.33.4.12
G.ROOT-SERVERS.NET.     6d23h48m47s IN A  192.112.36.4
F.ROOT-SERVERS.NET.     6d23h48m47s IN A  192.5.5.241
B.ROOT-SERVERS.NET.     6d23h48m47s IN A  128.9.0.107
J.ROOT-SERVERS.NET.     6d23h48m47s IN A  198.41.0.10
K.ROOT-SERVERS.NET.     6d23h48m47s IN A  193.0.14.129
L.ROOT-SERVERS.NET.     6d23h48m47s IN A  198.32.64.12
M.ROOT-SERVERS.NET.     6d23h48m47s IN A  202.12.27.33
```

가이드라인만(referral) 들어왔다. 우리에게 "AUTHORITY SECTION(인증 섹션)"만을 주고 "Answer Section(응답섹션)"은 주지 않았다. 우리 네임서버가 하나의 네임서버를 선택할 것이다. 아무거나 집어보자.(추천된 서버중 하나를 집어내서 다시 질의를 보내자는 이야기겠져...)

```
$ dig +norec +noH +noques +nostats +nocmd prep.ai.mit.edu. @H.ROOT-
SERVERS.NET.
; (1 server found)
;; res options: init defnam dnsrch
;; got answer:
; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 3
;; AUTHORITY SECTION:
MIT.EDU.                2D IN NS        BITSY.MIT.EDU.
MIT.EDU.                2D IN NS        STRAWB.MIT.EDU.
MIT.EDU.                2D IN NS        W20NS.MIT.EDU.

;; ADDITIONAL SECTION:
BITSY.MIT.EDU.          2D IN A         18.72.0.3
STRAWB.MIT.EDU.         2D IN A         18.71.0.151
W20NS.MIT.EDU.          2D IN A         18.70.0.160
```

이제 우리에게 MIT.EDU를 선택할 수 있다. 다시 하나 집어내 보자.

```
$ dig +norec +noH +noques +nostats +nocmd prep.ai.mit.edu. @bitsy.mit.edu
; (1 server found)
;; res options: init defnam dnsrch
;; got answer:
; flags: qr ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4
;; ANSWER SECTION:
prep.ai.mit.edu.        3h50m7s IN A    198.186.203.18
```

```
;; AUTHORITY SECTION:
AI.MIT.EDU.               6H IN NS        FEDEX.AI.MIT.EDU.
AI.MIT.EDU.               6H IN NS        LIFE.AI.MIT.EDU.
AI.MIT.EDU.               6H IN NS        ALPHA-BITS.AI.MIT.EDU.
AI.MIT.EDU.               6H IN NS        BEET-CHEX.AI.MIT.EDU.

;; ADDITIONAL SECTION:
FEDEX.AI.MIT.EDU.         6H IN A         192.148.252.43
LIFE.AI.MIT.EDU.          6H IN A         128.52.32.80
ALPHA-BITS.AI.MIT.EDU.    6H IN A         128.52.32.5
BEET-CHEX.AI.MIT.EDU.     6H IN A         128.52.32.22
```

이제 "Answer Section"을 얻었고, 우리는 질의에 대한 답을 얻게 되었다. "AUTHORITY SECTION"은 어느 서버에 ai.mit.edu를 다음에 물어야 하는 지에 대한 정보를 가지고 있다. 그래서 바로 다음에 ai.mit.edu를 물어볼 수 있게 된다.

. 에서부터 출발해서 우리는 각 도메인 네임의 레벨에서 네임서버를 성공적으로 발견해 내었다. 다른 서버를 사용하지 않고 여러분의 DNS서버를 사용했다면 당연히 그 정보를 캐쉬에 저장해 둘 것이고 당분간은 다시 묻지 않을 것이다.

이 트리구조에서 이름안에 있는 "." 들은 분기점이다. 그리고 "."점들 사이의 부분은 이 트리구조에서 개별적으로 분기되는 부분이다. 우리가 원하는 이름을 (prep.ai.mit.edu) 찾기 위해서 트리구조를 보면 우선 캐쉬에 있는 정보나 아니면 뿌리(.)에게 어느 서버가 prep.ai.mit.edu로 가는 시작점(원본은 father라는 동사를 씁니다.원류... 근원...)인지를 물어본다. 캐쉬에 정보가 없으면 밖으로 나가서 그 이름에 대한 좀더 가까운 힌트(referral,지정된 서버라고 번역되는데 힌트가 더 적당할 수 있겠습니다.)를 얻기 위해 재귀적으로 계속해서 서버들에게 질문을 하게 된다.

자주 이야기 되지는 않지만 중요한 도메인이 in-addr.arpa 이다. '일반'도메인과 유사하다. in-addr.arpa는 우리가 주소를 알고 있을 경우에 그 이름을 알려준다. 숙지해야 할 중요한 부분은 ip 주소가 in-addr.arpa 도메인에서는 거꾸로 사용된다는 것이다. 만약에 주소 192.148.52.43을 알고 있다고 하자. 그러면 prep.ai.mit.edu 예에서 했던 것과 같은 식으로 진행이 된다. 먼저 arpa. 서버를 찾고 in-addr.arpa. 서버를 찾게된다. 그리고 나서 192.in-addr.arpa. 로, 148.192.in-addr.arpa. 로, 52.148.192.in-addr.arpa. 서버로 해서 43.52.148.192.in-addr.arpa.서버를 찾게된다. 영리하지 않은가? (그렇다고 대답하라고 원문은 강요도 한다.-.-) 숫자를 뒤집어 놓은 것이 다소간은 혼란스러울 것이다.)

5.2. 자신의 도메인

이제 우리 도메인을 정의해 보자. 우리는 linux.bogus라는 도메인을 가지고 우리 서버에 만들어 놓을 것이다. 나는 이러한 것이 외부로 실제 나가지 않게 가짜 도메인 이름을 쓸 것이다.

시작전에 하나 알아둬야 한다. 모든 문자가 다 도메인이 되는 것은 아니다. 제한적으로 영어 알파벳이 쓰인다.: a-z, 0-9, 그리고 '-'(대쉬)가 사용된다. 대소문자는 구별이 없다는 점도 알아두기를. 그래서 pat.uio.no 는 Pat.UiO.No 와 같이 쓰인다.

우리는 이미 named.conf에서 이 부분을 했었다.:

```
zone "0.0.127.in-addr.arpa" {
type master;
file "pz/127.0.0";
};
```

이 파일에서는 도메인 네임의 마지막에 '.'이 없다는 사실을 주의해라. 이 부분은 우리가 0.0.127.in-addr.arpa 존에 대한 정의이고 마스터 서버에 대한 정의가 있으며 pz/127.0.0이라는 파일 안에 저장이 된다. 우리는 이미 이 파일을 세팅했다. 보면...

```
$TTL 3D
@               IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
1       ; Serial
8H ; Refresh
2H      ; Retry
4W ; Expire
1D) ; Minimum TTL
NS      ns.linux.bogus.
1 PTR localhost.
```

위의 named.conf와는 다르게 이 파일안에 있는 모든 도메인명의 마지막에는 '.'이 있다는 사실을 알아두길 바란다. 어떤 이들은 모든 존 파일을 $ORIGIN으로 시작하기도 하는데 이것은 불필요한 것이다. 도메인 계층안에 포함되어 있는 이 존 파일들의 근원지는 이미 named.conf에서 다 명시가 되어 있기 때문이다. 이 경우에는 0.0.127.in-addr.arpa 이다.

이 'zone' 파일은 3개의 '자원레코드(RR:Resource Records)'를 가지고 있다. SOA, NS, PTR이다. SOA는 Start Of Authority의 약어이다. @는 원본을 의미하는 것으로 이것은 'domain' 컬럼에서 0.0.127.in-addr.arpa를의미하는 것이다.

```
0.0.127.in-addr.arpa. IN SOA ...
```

NS는 NameServer의 약자이다. 시작부분에 '@'가 없다. 이미 앞 라인이 @로 시작되면서 @가 내제되어있다. 타이핑 수를 줄여라. 그래서 NS라인은 이렇게 쓰여진다.

```
0.0.127.in-addr.arpa. IN NS ns.linux.bogus
```

이것은 DNS에게 어떤 컴퓨터가 도메인 0.0.127.in-addr.arpa의 nameserver인지 알려주게 된다. 여기서는 ns.linux.bogus이다. 'ns'는 보편적으로 nameserver에 붙는 말이다. 그러나 웹서버의 이름이 일반적으로 www.**something** 이듯이 다른 이름을 주어도 무방하다.

그리고 마지막으로 PTR(Domain Name Pointer)레코드는 서브넷 0.0.127.in-addr.arpa의 호스트 어드레스가 1 임을 이야기 한다. 즉 127.0.0.1의 이름이 localhost이다.

SOA 레코드는 모든 zone 파일의 서두에 해당되며 각 존 파일마다 반드시 하나씩 있어야 한다. 이것은 그 zone 파일이 무엇이라 불리는지(여기서는 ns.linux.bogus라 불린다.), 누가 이

contents를 관리하는지(hostmaster@linux.bogus; email주소를 필히 넣어야 한다), 이 존 파일의 버젼이 무엇인지(serial:1), 캐시서버와 2차 DNS서버는 무엇인지 등을 기재하게 된다. 나머지 필드(refresh, retry,expire,minimum)은 HOWTO 에 기술된 것으로 해라. 이것이 안정적일 것이다. SOA 라인이 쓰여지기 전에 $TTL 3D란 것을 위에 기재하길 바란다. 이것은 모든 zone file에 넣어야 한다.

이제 named 를 재구동시켜보자.(ndc restart) 그리고 dig명령어로 질의를 해 보자(옵션 -x)

```
$ dig -x 127.0.0.1

; <<>> DiG 8.2 <<>> -x
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; QUERY SECTION:
;;      1.0.0.127.in-addr.arpa, type = ANY, class = IN

;; ANSWER SECTION:
1.0.0.127.in-addr.arpa.  1D IN PTR  localhost.

;; AUTHORITY SECTION:
0.0.127.in-addr.arpa.   1D IN NS        ns.penguin.bv.

;; Total query time: 5 msec
;; FROM: lookfar to SERVER: default -- 127.0.0.1
;; WHEN: Sat Dec 16 01:13:48 2000
;; MSG SIZE  sent: 40  rcvd: 110
```

위에서 127.0.0.1에서 localhost를 찾는데 성공했다. 이제 주요 작업인 linux.bogus 도메인을 만들어 보자. named.conf의 'zone'섹션에 추가하라

```
zone "linux.bogus" {
notify no;
type master;
file "pz/linux.bogus";
};
```

named.conf 파일의 도메인 이름에는 '.'이 없다는 것을 다시한번 상기하자.

linux.bogus의 존파일에는 전부 가짜 데이터를 넣을 것이다.

```
;
; Zone file for linux.bogus
;
; The full zone file
;
$TTL 3D
```

```
@ IN SOA ns.linux.bogus. hostmaster.linux.bogus. (
199802151 ; serial, todays date + todays serial #
8H ; refresh, seconds
2H ; retry, seconds
4W ; expire, seconds
1D ) ; minimum, seconds
;
NS ns ; Inet Address of name server
MX 10 mail.linux.bogus ; Primary Mail Exchanger
MX 20 mail.friend.bogus. ; Secondary Mail Exchanger
;
localhost A 127.0.0.1
ns A 192.168.196.2
mail A 192.168.196.4
```

SOA 레코드 부분에서 2가지 부분을 주의해서 보자. `ns.linux.bogus`는 A 레코드를 가진 실제 컴퓨터여야 한다. SOA레코드가 이야기하는 컴퓨터의 이름을 CNAME으로 하는 것은 규정에 어긋난다. 이 이름에 'ns'가 들어갈 필요는 없으며 아무 이름이라도 규정에만 어긋나지 않으면 된다. 다음으로 hostmaster.linux.bogus 는 hostmaster@linux.bogus로 봐야 한다. 이것은 mail alias나 mailbox로서, DNS관리자가 이 주소로 메일을 받게 될 것이다. 이름이 꼭 hostmaster일 필요는 없다. 일반 메일주소를 넣어도 되지만 'hostmaster' 인 것이 효율적일 것이다.

MX라는 새로운 레코드가 보인다. 이것은 Mail eXchanger로 메일 시스템에게 어디로 메일을 보내는지 알려준다. `mail.linux.bogus` 혹은 `mail.friend.bogus`로 인해 주소는 `someone@linux.bogus`가 된다. 각 computer이름앞의 숫자는 MX레코드의 우선 순위를 나타낸다. 가장 낮은 숫자의 레코드(10)에 가능한 우선 순위가 주어진다. 이것이 실패할 경우 더 큰 숫자를 찾아서 가는에 여기에서는 20번의 `mail.friend.bogus`가 된다.

다시 `ndc restart`한 후 dig 결과를 살펴보자.

```
$ dig any linux.bogus +pfmin
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23499
;; QUERY: 1, ANSWER: 4, AUTHORITY: 1, ADDITIONAL: 1
;; QUERY SECTION:
;;      linux.bogus, type = ANY, class = IN

;; ANSWER SECTION:
linux.bogus.            3D IN MX        10 mail.linux.bogus.linux.bogus.
linux.bogus.            3D IN MX        20 mail.friend.bogus.
linux.bogus.            3D IN NS        ns.linux.bogus.
linux.bogus.            3D IN SOA       ns.linux.bogus. hostmas-
ter.linux.bogus. (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
```

```
                                             4W              ; expiry
                                             1D )            ; minimum
```

잘 보면 버그가 있다. 라인의

```
linux.bogus.            3D IN MX        10 mail.linux.bogus.linux.bogus.
```

가 잘못되었다.이것은

```
linux.bogus.            3D IN MX        10 mail.linux.bogus.
```

로 나와야 한다. 일부러 버그를 만들어서 보여줬다. 존 파일 부분에서 우리가 찾아낸 부분을 살펴보자

```
MX 10 mail.linux.bogus ; Primary Mail Exchanger
```

마침표가 없거나 'linux.bogus'가 중복이 되어 있다. 끝부분에 마침표를 찍지 않으면 존파일의 origin이 더해져서 `linux.bogus.linux.bogus`형태가 된다. 그래서

```
MX 10 mail.linux.bogus. ; Primary Mail Exchanger
```

이거나

```
MX 10 mail ; Primary Mail Exchanger
```

의 형태로 되어야 한다. 나는 후자를 선호한다. 타이핑수를 줄이게 되니까. BIND 전문가들은 이러한 부분을 동의하기도 하고 반대하기도 한다. 존 파일에서 도메인은 끝이 '.'로 끝나던가 아니면 전부 포함되지 말아야 하며 전부 포함되지 않을 경우는 디폴트로 origin이 들어간다.

분명히 named.conf 파일에는 '.' 이 들어가지 않는다고 했다. '.'이 도메인 이름 다음에 얼마나 많게 혹은 적게 쓰이거나 하는 차이로 세팅이 안되고 사람들을 혼란스럽게 하는 일이 다반사이다.

그래서 새로운 존 파일의 생성시에는 그에 맞게 추가 정보가 주어져야 한다.

```
;
; Zone file for linux.bogus
;
; The full zone file
;
$TTL 3D
@ IN SOA ns.linux.bogus. hostmaster.linux.bogus. (
199802151 ; serial, todays date + todays serial #
8H ; refresh, seconds
2H ; retry, seconds
```

```
4W ; expire, seconds
1D ) ; minimum, seconds
;
TXT "Linux.Bogus, your DNS consultants"
NS ns ; Inet Address of name server
NS ns.friend.bogus.
MX 10 mail ; Primary Mail Exchanger
MX 20 mail.friend.bogus. ; Secondary Mail Exchanger

localhost A 127.0.0.1

gw A 192.168.196.1
HINFO "Cisco" "IOS"
TXT "The router"

ns A 192.168.196.2
MX 10 mail
MX 20 mail.friend.bogus.
HINFO "Pentium" "Linux 2.0"
www CNAME ns

donald A 192.168.196.3
MX 10 mail
MX 20 mail.friend.bogus.
HINFO "i486" "Linux 2.0"
TXT "DEK"

mail A 192.168.196.4
MX 10 mail
MX 20 mail.friend.bogus.
HINFO "386sx" "Linux 1.2"

ftp A 192.168.196.5
MX 10 mail
MX 20 mail.friend.bogus.
HINFO "P6" "Linux 2.1.86"
```

새로운 레코드가 보인다. HINFO(Host INFOmation)는 2개의 파트로 구성되어 있다. 첫째 부분은 하드웨어나 cpu부분, 둘째 부분은 소프트웨어나 OS를 표시한다. 'ns'라 불리는 컴퓨터는 펜티엄 cpu를 가지고 있고 Linux 2.0으로 구현된다. CNAME(Canonical Name)은 각 computer에게 여러 이름을 주는 방법이다. 그래서 www는 ns의 alias가 된다.

CNAME 레코드는 약간의 문제점이 있다. 그러나 규정대로만 하면 문제될 부분은 없다. MX,CNAME,SOA 레코드가 CNAME 레코드에 연계되어서는 안된다. 이것들은 A 레코드에만 연계되어야 하기 때문이다. 그래서 아래와 같은 경우는 안된다.(www가 이미 CNAME이다.)

```
foobar CNAME www ; NO!
```

그러나 이런 경우는 된다.

```
foobar CNAME ns ; Yes!
```

CNAME이 email 주소를 위한 hostname으로 쓰기에는 적합하지 않다고 여기는 것이 현명할 것이다. `webmaster@www.linux.bogus`는 위의 setup과 다른 잘못된 경우이다. 많은 mail 관리자들이 이것이 설령 되더라도 이 방식을 거의 사용하지 않을 것이다. 이러한 오류를 피하는 방식은 A 레코드를 사용하는 것이다.(다른 MX 레코드 처럼)

```
www A 192.168.196.2
```

많은 DNS전문가들이 CNAME을 사용하지 않을 것을 권고한다. 그러나 사용가능 여부는 이 HOWTO에서 논할 부분은 아니다.

하지만 많은 HOWTO와 많은 싸이트들이 이 규칙을 따르지는 않는다.

새 database를 로딩하고 `ndc reload`를 실행해보자.

```
$ dig linux.bogus axfr

; <<>> DiG 8.2 <<>> linux.bogus axfr
$ORIGIN linux.bogus.
@                       3D IN SOA       ns hostmaster (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        4W              ; expiry
                                        1D )            ; minimum

                        3D IN NS        ns
                        3D IN NS        ns.friend.bogus.
                        3D IN MX        10 mail
                        3D IN MX        20 mail.friend.bogus.
                        3D IN TXT       "Linux.Bogus, your DNS consultants"
gw                      3D IN TXT       "The router"
                        3D IN HINFO     "Cisco" "IOS"
                        3D IN A         192.168.196.1
localhost               3D IN A         127.0.0.1
mail                    3D IN HINFO     "386sx" "Linux 1.2"
                        3D IN MX        10 mail
                        3D IN MX        20 mail.friend.bogus.
                        3D IN A         192.168.196.4
www                     3D IN CNAME     ns
donald                  3D IN TXT       "DEK"
                        3D IN HINFO     "i486" "Linux 2.0"
                        3D IN MX        10 mail
                        3D IN MX        20 mail.friend.bogus.
                        3D IN A         192.168.196.3
```

```
ns                      3D IN HINFO     "Pentium" "Linux 2.0"
                        3D IN MX        10 mail
                        3D IN MX        20 mail.friend.bogus.
                        3D IN A         192.168.196.2
ftp                     3D IN HINFO     "P6" "Linux 2.1.86"
                        3D IN MX        10 mail
                        3D IN MX        20 mail.friend.bogus.
                        3D IN A         192.168.196.5
@                       3D IN SOA       ns hostmaster (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        4W              ; expiry
                                        1D )            ; minimum

;; Received 29 answers (29 records).
;; FROM: lookfar to SERVER: 127.0.0.1
;; WHEN: Sat Dec 16 01:35:05 2000
```

good! 결과가 존 파일과 비슷하게 보인다. www부분에 대한 정보를 확인해보자

```
$<65533>ig www.linux.bogus +pfmin
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27345
;; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 1
;; QUERY SECTION:
;;      www.linux.bogus, type = A, class = IN

;; ANSWER SECTION:
www.linux.bogus.        3D IN CNAME     ns.linux.bogus.
ns.linux.bogus.         3D IN A         192.168.196.2
```

다시 보면 `www.linux.bogus`의 실제 이름은 `ns.linux.bogus`이다. 그리고 ns가 가진 정보도 보여주며 충분히 연결될 수 있다.

이제 절반 했다.

5.3. The reverse zone(역변환존)

프로그램들은 linux.bogus의 이름을 접속가능한 주소로 바꿔야 접속이 가능하다. 그러나 역변환존(reverse zone)이 있어야 DNS가 주소를 이름으로 바꾸는 것이 가능하다. 이러한 이름은 당신이 이 서비스를 사용할지 말지, 그리고 사용한다면 얼마만큼의 우선순위가 주어지는지 등을 결정할 다양한 종류의 서버 이름들(FTP,IRC,WWW 그외)에서 사용된다. 인터넷에서 모든 서비스에 대한 완전한 접속을 위해서는 reverse zone이 필요하다.

`named.conf`에 추가해 보자

```
zone "196.168.192.in-addr.arpa" {
notify no;
        type master;
        file "pz/192.168.196";
};
```

`0.0.127.in-addr.arpa`와 같다. 내용도 유사하다.

```
$TTL 3D
@ IN SOA ns.linux.bogus. hostmaster.linux.bogus. (
199802151 ; Serial, todays date + todays serial
8H ; Refresh
2H      ; Retry
4W ; Expire
1D) ; Minimum TTL
NS      ns.linux.bogus.

1 PTR gw.linux.bogus.
2 PTR ns.linux.bogus.
3 PTR donald.linux.bogus.
4 PTR mail.linux.bogus.
5 PTR ftp.linux.bogus.
```

이제 named를 재구동시키고(`ndc restart`) dig로 다시 검사해 보자

```
$ dig -x 192.168.196.4 +pfmin
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8764
;; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;; QUERY SECTION:
;;      4.196.168.192.in-addr.arpa, type = ANY, class = IN

;; ANSWER SECTION:
4.196.168.192.in-addr.arpa.  3D IN PTR  mail.linux.bogus.
```

괜찮아 보인다. 모든 부분을 검사해 보자

```
dig -x 192.168.196 AXFR

; <<>> DiG 8.2 <<>> -x AXFR
$ORIGIN 196.168.192.in-addr.arpa.
@                        3D IN SOA       ns.linux.bogus. hostmas-
ter.linux.bogus. (
                                         199802151       ; serial
                                         8H              ; refresh
                                         2H              ; retry
```

```
                                                4W              ; expiry
                                                1D )            ; minimum

                        3D IN NS        ns.linux.bogus.
4                       3D IN PTR       mail.linux.bogus.
2                       3D IN PTR       ns.linux.bogus.
5                       3D IN PTR       ftp.linux.bogus.
3                       3D IN PTR       donald.linux.bogus.
1                       3D IN PTR       gw.linux.bogus.
@                       3D IN SOA       ns.linux.bogus. hostmas-
ter.linux.bogus. (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        4W              ; expiry
                                        1D )            ; minimum

;; Received 8 answers (8 records).
;; FROM: lookfar to SERVER: 127.0.0.1
;; WHEN: Sat Dec 16 01:44:03 2000
```

잘 된다! 결과가 위와 같지 않으면 syslog의 에러 메시지를 봐라. 이미 첫부분의 Starting Named에서 설명해 놨다.

5.4. 주의사항

추가할 부분이 있다. 위에 명시된 IP 주소들은 전부 '사설망'에 국한된 예시이다. 이것은 일반 인터넷에는 적용되지 않는다. 그래서 안전하게 예시를 다룰 수 있다. 두번째 부분은 'notify no;'라는 부분이다. named가 존파일이 업데이트 될 때 슬레이브서버에 공지가 되어서는 안된다는 이야기이다. BIND-8 버젼의 named는 zone파일이 업데이트 될 때 zone 파일에 있는 NS레코드에 등록되 있는 다른 서버들에게 공지를 하기 때문이다. 이것은 일반적인 이용에는 편할 것이지만 개인적인 연습에서는 이러한 부분을 사용하지 말아야 할 것이다. 우리가 이러한 것으로 인터넷을 어지럽힐 수는 없지 않는가?

그리고 이것은 물론 완전히 허구이다. 실례를 보려면 다음 장을 참조하라

5.5. 왜 **reverse lookups** 이 작동하지 않는가**?**

역변환 존이 설정되었을 때 가끔 name lookup(이름검사)이 되지 않아서 '아차'하는 순간들이 있다. 수행전에 당신의 nameserver에 대해 역변환 검사 (reverse lookups)를 할 필요가 있다. 작동하지 않으면 이 세팅을 진행하기 전에 뒤로 가서 고쳐야 한다.

일반적으로 역변환 검사가(reverse looksup) 되지 않는 2가지 이유를 들어 보겠다.

5.5.1. 역변환존이 제대로 위임되지 않는 경우.

네트워크 주소와 도메인을 공급하는 업체에 일반적으로 도메인 이름을 위탁하게 된다. 그러면 업체는 NS 레코드에 추가하고 앞에서 설명한 이론대로 네임서버를 찾는다. 읽어봤겠지만 제대로 작동되지 않으면 다시 돌아가서 읽어봐라.

역변환 존 역시 맡겨질 것인데 만약 `192.168.196`대의 망을 업체로부터 `linux.bogus`라고 얻었다면 `NS`레코드에 포워드 존처럼 역변환존을 추가할 것이다. `in-addr.arpa`를 따르고 이것으로 망을 구성하면 이 안의 오류를 찾아야 할 것이며 대부분은 업체의 문제이다. 업체와 연락해서 에러를 수정하라.

5.5.2. 클래스가 없는 서브넷을 얻는 경우

좀더 발전된 형식이지만 요사이는 당신이 소규모 업체이면 가지고 있을 법한 일반적인 기술이 되었다.

classless 서브넷은 요새 인터넷의 추세이다. 얼마전만해도 IP 부족으로 고생이 많았다. IETF(Internet Engineering Task Force)의 똑똑한 이들이 머리를 맞대고 문제를 해결했다. 그 결과로 C 클래스보다 작은 서브넷을 얻게 되었는데 여기에 약간의 문제가 있다. Mr. DNS (http://www.acmebw.com/askmrdns/00007.htm)의 사이트에 가서 사용법과 해결법도 참고하라.

읽어보지 않았다면 확인하길 바란다.

첫번째 부분은 인터넷 공급업체가 Mr.DNS의 기술을 인식하지 못한다는 문제이다. 모든 인터넷 업체가 다 그러는 것은 아니지만 만약 이런 경우가 생기면 당신이 원인을 설명하고 보충해 줘야 한다. 물론 당신이 이 부분을 먼저 이해하고 있어야 한다. 그러면 그들이 역변환 존을 설정하고 dig로 확인할 것이다.

두번째 부분은 기술을 명확하게 이해하지 못한다는 것이다. 앞부분이 확실하지 않다면 다시 돌아가서 확인해보길. 그들은 Dr.DNS에서 나온 데로 클래스 없는 역변환 존을 생성할 것이다.

또다른 복병이 있다. 오래된 resolver는 `CNAME` 트릭을 따르지 않고 역변환 존의 가동을 막을 것이다. 잘못된 클래스를 할당하고 접속을 거부하기도한다. 이러한 경우의 해결책은 인터넷 공급업체에 문의해서 classless 존 파일에 CNAME 대신 PTR레코드를 직접 넣는 것이다.

몇몇 ISP업체들은 automasical system이나 역도메인을 메핑하는 식의 웹기반 폼으로 이것을 조절하기도 한다.

5.6. Slave servers(2차 네임서버)

master 서버에 존파일 설정을 제대로 했다면 이제는 하나 이상의 slave서버도 설정해야 할 것이다. slave 서버는 꼭 필요한 것이다. 1차 도메인이 죽어 버려도 사람들은 2차 네임서버로 찾아가 정보를 얻게 될 것이다. slave 서버는 1차 서버와 멀리 떨어져 있는것이 좋다. 1차 DNS서버와 2차 DNS서버는 될 수 있으면 Power Supply,LAN,ISP,도시, 국가등.. 공유하는 것이 적을수록 좋다. master와 slave 가 위의 사항들이 모두 다른 경우라면 당신의 slave는 매우 잘된 예이다.

slave는 단순히 master로부터 존파일들을 복사해온다. 아래와 같이 설정하면 된다.

```
zone "linux.bogus" {
type slave;
file "sz/linux.bogus";
masters { 192.168.196.2; };
};
```

단순히 데이터가 복사되는 구조이다. 이러한 파일복사는 SOA에 의해 조정된다.

```
@ IN SOA ns.linux.bogus. hostmaster.linux.bogus. (
199802151 ; serial, todays date + todays serial #
8H ; refresh, seconds
2H ; retry, seconds
4W ; expire, seconds
1D ) ; minimum, seconds
```

master의 시리얼넘버가 단지 slave보다 크기만 하면 존파일이 전송된다. slave의 refresh는 master가 업데이트 될 때마다 체크할 것이다. 만약에 체크가 되지 않으면(master가 죽어서) 매번 재시도 하면서 체크할 것이다. expire기간까지 실패를 한다면 slave는 존파일을 제거하고 더이상 네임서버의 기능을 하지 않을 것이다.

6. 기본보안옵션

By Jamie Norrish

설정옵션을 이용하여 문제의 소지를 줄이기.

서버의 부하를 줄이고 보안을 높히기 위한 몇가지 간단한 단계들이 있다. 여기 나오는 부분이 시작할때 나온 부분보다 더 나가는 것은 없다. 만약 당신이 보안에 관심이 없다면 (아마 그럴것이지만) 넷상의 다른 자원들을 살펴보길 바란다(마지막 장 참조)

설정은 `named.conf`에서 이루어진다. file `options`부분이 지정되면 파일에 지정된 모든 `zone` 파일들이 적용이 된다. `zone` 파일부분이 지정이 되면 그 zone에서만 적용된다. `zone` 부분은 `options` 부분을 override한다.

6.1. 전송에서의 제한

slave 서버가 도메인에 대해 응답할 수 있게 하려면 주서버로부터 zone 정보를 전달받아야 한다. 이러한 요청을 하는 부분은 일부분이면 된다. `allow-transfer` 옵션으로 전송을 제한 할 수 있다.192.168.1.4 주소는 ns.friend.bogus의 주소이고 디버깅 목적으로 추가되었다.

```
zone "linux.bogus" {
      allow-transfer { 192.168.1.4; localhost; };
```

```
};
```

zone 전송을 제한함으로서 정보가 필요한 사람만이 질의를 할 수 있고 그외에는 DNS 셋업에 대한 정보를 얻을 수 없다..

6.2. spoofing으로부터의 보호

우선 자신의 컴퓨터 이외에는 어떤 질의도 불가능하게 하라(내부/local 은 제외). 이것은 악의적인 DNS사용을 막아줄 것이고 서버의 불필요한 이용을 줄여 줄 것이다.

```
options {
      allow-query { 192.168.196.0/24; localhost; };
};

zone "linux.bogus" {
      allow-query { any; };
};

zone "196.168.192.in-addr.arpa" {
      allow-query { any; };
};
```

그리고 내부/local을 제외한 나머지에서 재귀적인 질의를 불가능하게 하라. 이것은 cache 공격의 위험(잘못된 데이터를 전송하는 경우)을 줄여줄 것이다.

```
options {
allow-recursion { 192.168.196.0/24; localhost; };
};
```

6.3. root외의 계정으로 named 구동

named를 root가 아닌 user로 구동시키는 것은 아주 좋은 생각이다. 이렇게 되면 크래커에게 권한을 뺏겨도 제한적일 수 밖에 없다. named를 구동할 user와 group을 생성하고 named가 구동되게 init script를 수정하라. named를 새 user의 group이 조정하도록 -u 와 -g flag를 줘라.

예를 들어 Debian GNU/Linux2.2 경우 `/etc/init.d/bind` script는 아래와 같이 수정할 수 있다. (user와 group이 생긴 상황에서)

```
start-stop-daemon --start --quiet --exec /usr/sbin/named -- -u named -
g named
```

RedHat이나 다른 배포판도 마찬가지이다. Dave Lugo가 secure dual chroot setup에 대해서 http://www.etherboy.com/dns/chrootdns.html에 기술해 놓았다. 관심있으면 참고하길

7. 실도메인의 예

실제 도메인의 예

이제 실제 예시로 사용이 되고 있는 예를 들어보기로 한다

나는 예제로 Dave Bullock의 LAND-5를 예로 들어 본다. 이 파일들은 1996년 9월 24일에 만들어 졌고 BIND-8규정에 맞게 약간 확장편집을 해 놓았다. 지금의 LAND-5 named 서버를 질의한다면 약간의 차이가 있을 것이다.

7.1. /etc/named.conf (or /var/named/named.conf)

이제 127.0.0 net과 LAND-5의 `206.6.177` 서브넷에 필요한 2개의 reverse zone이 있는 master zone부분에 대해서 보기로 하자. 그리고 land-5의 포워드 존인 `land-5.com`도 보기로 하자. HOWTO에서 쓰던 pz 대신에 `zone`이라는 디렉토리를 쓰는 것도 주의하길 바란다.

```
// Boot file for LAND-5 name server

options {
directory "/var/named";
};

zone "." {
type hint;
file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
type master;
file "zone/127.0.0";
};

zone "land-5.com" {
type master;
file "zone/land-5.com";
};

zone "177.6.206.in-addr.arpa" {
type master;
file "zone/206.6.177";
};
```

만약 위와 같이 named.conf파일을 설정했다면 제발! 2개의 `land-5` zone에 `"notify no;"`라고 적어서 충돌을 방지해주길 바란다.

7.2. /var/named/root.hints

이 파일들이 유동적이라는 사실을 기억해라. 그리고 이것이 옛날 것이라는 것도 기억하라. 이전에 설명한 dig 로 지금의 것을 측정하는 것이 훨씬 나을 것이다.

```
; <<>> DiG 8.1 <<>> @A.ROOT-SERVERS.NET.
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;; ., type = NS, class = IN

;; ANSWER SECTION:
. 6D IN NS G.ROOT-SERVERS.NET.
. 6D IN NS J.ROOT-SERVERS.NET.
. 6D IN NS K.ROOT-SERVERS.NET.
. 6D IN NS L.ROOT-SERVERS.NET.
. 6D IN NS M.ROOT-SERVERS.NET.
. 6D IN NS A.ROOT-SERVERS.NET.
. 6D IN NS H.ROOT-SERVERS.NET.
. 6D IN NS B.ROOT-SERVERS.NET.
. 6D IN NS C.ROOT-SERVERS.NET.
. 6D IN NS D.ROOT-SERVERS.NET.
. 6D IN NS E.ROOT-SERVERS.NET.
. 6D IN NS I.ROOT-SERVERS.NET.
. 6D IN NS F.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
G.ROOT-SERVERS.NET. 5w6d16h IN A 192.112.36.4
J.ROOT-SERVERS.NET. 5w6d16h IN A 198.41.0.10
K.ROOT-SERVERS.NET. 5w6d16h IN A 193.0.14.129
L.ROOT-SERVERS.NET. 5w6d16h IN A 198.32.64.12
M.ROOT-SERVERS.NET. 5w6d16h IN A 202.12.27.33
A.ROOT-SERVERS.NET. 5w6d16h IN A 198.41.0.4
H.ROOT-SERVERS.NET. 5w6d16h IN A 128.63.2.53
B.ROOT-SERVERS.NET. 5w6d16h IN A 128.9.0.107
C.ROOT-SERVERS.NET. 5w6d16h IN A 192.33.4.12
D.ROOT-SERVERS.NET. 5w6d16h IN A 128.8.10.90
E.ROOT-SERVERS.NET. 5w6d16h IN A 192.203.230.10
I.ROOT-SERVERS.NET. 5w6d16h IN A 192.36.148.17
F.ROOT-SERVERS.NET. 5w6d16h IN A 192.5.5.241

;; Total query time: 215 msec
;; FROM: roke.uio.no to SERVER: A.ROOT-SERVERS.NET.  198.41.0.4
;; WHEN: Sun Feb 15 01:22:51 1998
;; MSG SIZE  sent: 17  rcvd: 436
```

7.3. /var/named/zone/127.0.0

기본적으로 SOA레코는 필수이고, 이 레코드는 127.0.0.1를 `localhost`로 매핑한다. 둘 다 필요하다. 파일 안에 이 이상 필요하지도 않으며 네임서버나 hostmaster 주소가 바뀌지 않는 이상은 업데이트 하지 않는게 좋다.

```
@               IN      SOA     land-5.com. root.land-5.com. (
                                199609203       ; Serial
                                28800   ; Refresh
                                7200    ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      land-5.com.

1                       PTR     localhost.
```

BIND를 그냥 인스톨했다면(rpm이나 옵션이 없는 경우를 이야기하는 것 같다.) 위에 `$TTL`라인이 위와 같이 빠져 있을 것이다. 전에는 쓰지 않다가 BIND 8.2 버젼부터 이것을 추가하기 시작했다. 이 라인이 빠진것을 보는데로 넣는 것이 좋을 것이다.

7.4. /var/named/zone/land-5.com

우리는 SOA 레코드에서 NS레코드가 쓰이는 것을 봤다. 우리는 2차 네임서버로 `ns2.psi.net`이라고 되있는 것을 보게 될 것이다. 이것은 2차 서버로 항상 off라인상에 있어야 하며 백업용으로 사용된다는 것이다. 또한 master 호스트가 `land-5` 이고 이것은 여러 다른 서비스를 하게 된다.여기서는 CNAME을 사용한다. (대체할 수 있는 것으로는 A레코드가 있다.)

i SOA 항목에서 알 수 있듯이, 존 파일은 origin이 `land-5.com` 이며 관리자는 `root@land-5.com`이다. `hostmaster`는 관리자의 주소로 자주 사용되는 것이다. 시리얼 넘버는 의례적으로 yyyymmdd 형식에 그날의 시리얼 넘버를 덧붙인다. 아래서 보면 아마 이 지역 파일은 1996년 9월 20일 버전 6일 것이다. 시리얼 넘버는 한방향으로만 증가해야 함을 명심하자. 여기서는 그날의 시리얼 넘버가 한자리다. 그러므로 9번을 편집하고 나서 또 편집하려면 내일을 기다려야 할 것이다. 두 자리수 사용을 고려하자.

```
@       IN      SOA     land-5.com. root.land-5.com. (
                        199609206       ; serial, todays date + todays se-
rial #
                        8H ; refresh, seconds
                        2H ; retry, seconds
                        4W ; expire, seconds
                        1D ) ; minimum, seconds
                NS      land-5.com.
                NS      ns2.psi.net.
                MX      10 land-5.com.  ; Primary Mail Exchanger
                TXT     "LAND-5 Corporation"
```

```
localhost A 127.0.0.1

router          A       206.6.177.1

land-5.com.     A       206.6.177.2
ns A 206.6.177.3
www             A 207.159.141.192

ftp             CNAME   land-5.com.
mail            CNAME   land-5.com.
news            CNAME   land-5.com.

funn            A       206.6.177.2

;
;       Workstations
;
ws-177200 A       206.6.177.200
                MX      10 land-5.com.   ; Primary Mail Host
ws-177201       A       206.6.177.201
                MX      10 land-5.com.   ; Primary Mail Host
ws-177202       A       206.6.177.202
                MX      10 land-5.com.   ; Primary Mail Host
ws-177203       A       206.6.177.203
                MX      10 land-5.com.   ; Primary Mail Host
ws-177204       A       206.6.177.204
                MX      10 land-5.com.   ; Primary Mail Host
ws-177205       A       206.6.177.205
                MX      10 land-5.com.   ; Primary Mail Host
; {Many repetitive definitions deleted - SNIP}
ws-177250       A       206.6.177.250
                MX      10 land-5.com.   ; Primary Mail Host
ws-177251       A       206.6.177.251
                MX      10 land-5.com.   ; Primary Mail Host
ws-177252       A       206.6.177.252
                MX      10 land-5.com.   ; Primary Mail Host
ws-177253       A       206.6.177.253
                MX      10 land-5.com.   ; Primary Mail Host
ws-177254       A       206.6.177.254
                MX      10 land-5.com.   ; Primary Mail Host
```

확인해 보면 알겠지만 host 명이 ws_**number** 형식으로 되어있다. BIND4 버젼에서는 named를 시작할 때 쓰이는 호스트명이 강제적으로 제한되었다. 그러나 BIND 8에서는 이러한 것이 적용되지 않으므로 '_'(underline) 대신에 '-'(dash)로 바꾸었다.

또 하나 주목해야 할 부분은 workstation 의 이름이 개인의 이름이 아니고 IP 주소의 끝 2부분이라는 것이다. 이러한 것들은 관리를 편하게 할 수 있겠지만 조금은 비인간적이고 고객과의 불신의 요인이 될 수도 있다.

또한 `funn.land-5.com`이 `land-5.com` 를 CNAME대신에 A 레코드를 썼다는 사실을 주의해야 한다. 이것은 전에 이야기 했듯 좋은 방침이다.

7.5. /var/named/zone/206.6.177

아래에 이 파일의 설명이 있다

```
@               IN      SOA     land-5.com. root.land-5.com. (
                                199609206       ; Serial
                                28800   ; Refresh
                                7200    ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      land-5.com.
                        NS      ns2.psi.net.
;
;       Servers
;
1       PTR     router.land-5.com.
2       PTR     land-5.com.
2       PTR     funn.land-5.com.
;
;       Workstations
;
200     PTR     ws-177200.land-5.com.
201     PTR     ws-177201.land-5.com.
202     PTR     ws-177202.land-5.com.
203     PTR     ws-177203.land-5.com.
204     PTR     ws-177204.land-5.com.
205     PTR     ws-177205.land-5.com.
; {Many repetitive definitions deleted - SNIP}
250     PTR     ws-177250.land-5.com.
251     PTR     ws-177251.land-5.com.
252     PTR     ws-177252.land-5.com.
253     PTR     ws-177253.land-5.com.
254     PTR     ws-177254.land-5.com.
```

역변환 존은 대부분의 비극을 야기시키는 설정이다. 이것은 IP주소를 가지고 있으면 hostname을 찾는데 사용된다. 예를 보자: IRC 서버가 있고 IRC 클라이언트로부터 접속을 허용할 경우를 보자. 노르웨이 IRC 서버를 사용하며 노르웨이와 스칸디나비아반도 클라이언트와의 접속만 허용하려 한다. 클라이언트 접속을 할 때 C 라이브러리가 클라이언트의 IP 번호를 알려준다. 클라이언트의 IP번호가 넷상으로 오는 패킷에 포함되어 있기 때문이다. 이제 gethostbyaddr이라는 함수를 호출해서 주어진 IP로 그 컴퓨터를 찾는다. gethostbyaddr은 DNS 서버에 질의를 할 것이며 DNS는 찾은 컴퓨터 이름으로 변환할 것이다. 클라이언트가 ws-177200.land-5.com에 접속했다고 가정해 보자. IRC서버를 위해 C라이브러리는 IP가 206.6.177.200임을 알아내고 컴퓨터의 이름을 알아내기 위해

200.177.6.206.in-addr.arpa을 찾게 된다. DNS는 우선 arpa.을 찾게되며 그리고는 in-addr.arpa., 역순으로 206, 6 을 추적하고 마지막으로 177.6.206.in-addr.arpa를 LAND-5 존으로부터 찾아내게 된다. 거기에서 우리는 최종적으로 200.177.6.206.in-addr.arpa를 "PTR ws-177200.land-5.com" 레코드로부터 얻게 된다. 이것은 206.6.177.200은 ws-177200.land-5.com을 의미하는 것이다. 위의 설명은 prep.ai.mit.edu에서도 했고 여기에서는 허구가 곁들여져 있다.

IRC 서버 이야기로 돌아와서, IRC서버는 이제 스칸디나비아 지역의 접속만을 허가한다. *.no,*.se, *.dk 같은 것을 허가할 것이고 ws-177200.land-5.com 은 적합하지 않고 접속을 거부할 것이다.in-addr.arpa 존으로부터 206.2.177.200 에 대한 역변환 mapping이 없으면 서버는 이름들을 전혀 찾지 못하며 206.2.177.200은 매치가 안되는 *.no, *.se , *.dk와 비교할 것이다.

혹자는 역변환 mapping을 찾는 것이 단지 서버에게만 필요한 일이거나 중요하지 않다고 하는데, 결코 그렇지 않다. 많은 ftp, news,IRC와 심지어는 http(www)서버도 이름을 찾지 못하면 접속하지 않을 것이다. 역변환 mapping은 사실 필수적인 것이다.

8. 유지

작동이 되게 유지하라.

named의 기능이 유지되게 해야 하며 그것이 지속적으로 구동되게 해야 한다. 그럴려면 root.hints를 업데이트 해야 한다. 가장 손쉬운 방법은 dig를 이용하는 것이다. 먼저 어떤 인수도 없이 dig 를 구동시키면 자신의 서버에 해당하는 root.hints를 얻게 될 것이다. 그리고 나서는 dig @rootserver로 나열된 루트서버중 하나에 질의를 한다. root.hints 파일을 나타내는 출력결과를 주시해야 한다. 파일을 저장하고(dig @e.root-servers.net . ns>root.hints.new) root.hints 파일을 교체하면 된다.

항상 캐쉬파일을 교체하고 나면 named를 재구동 해야 한다는 사실을 염두해 두길.

Al Longyear씨가 자동적으로 root.hints를 갱신할 수 있는 스크립트를 제공해 주었다. crontab에 저장하고 한달에 한번씩 돌아가게 설정하면 그뒤에는 신경쓰지 않아도 된다. 이 스크립트는 여러분의 메일이 작동하고 mail-alias 'hostmaster'가 정의되어 있다고 가정한다. 수정해서 자신에게 맞게 설정해야 한다.

```
#!/bin/sh
#
# Update the nameserver cache information file once per month.
# This is run automatically by a cron entry.
#
# Original by Al Longyear
# Updated for BIND 8 by Nicolai Langfeldt
# Miscelanious error-conditions reported by David A. Ranch
# Ping test suggested by Martin Foster
# named up-test suggested by Erik Bryer.
#
```

```
(
 echo "To: hostmaster <hostmaster>"
 echo "From: system <root>"

 # Is named up? Check the status of named.
 case `ndc status 2>&1` in
    *'cannot connect to command channel'*)
        echo "named is DOWN. root.hints was NOT updated"
        echo
        exit 0
        ;;
 esac

 PATH=/sbin:/usr/sbin:/bin:/usr/bin:
 export PATH
 # NOTE: /var/named must be writable only by trusted users or this script
 # will cause root compromise/denial of service opportunities.
 cd /var/named 2>/dev/null || {
    echo "Subject: Cannot cd to /var/named, error $?"
    echo
    echo "The subject says it all"
    exit 1
 }

 # Are we online?  Ping a server at your ISP
 case `ping -qnc 1 some.machine.net 2>&1` in
   *'100% packet loss'*)
        echo "Subject: root.hints NOT updated.  The network is DOWN."
echo
echo "The subject says it all"
exit 1
;;
 esac

 dig @e.root-servers.net . ns >root.hints.new 2> errors

 case `cat root.hints.new` in
   *NOERROR*)
# It worked
:;;
   *)
echo "Subject: The root.hints file update has FAILED."
        echo
    echo "The root.hints update has failed"
echo "This is the dig output reported:"
    echo
    cat root.hints.new errors
        exit 1
;;
```

```
 esac

 echo "Subject: The root.hints file has been updated"
 echo
 echo "The root.hints file has been updated to contain the following
information:"
 echo
 cat root.hints.new

 chown root.root root.hints.new
 chmod 444 root.hints.new
 rm -f root.hints.old errors
 mv root.hints root.hints.old
 mv root.hints.new root.hints
 ndc restart
 echo
 echo "The nameserver has been restarted to ensure that the up-
date is complete."
 echo "The previous root.hints file is now called
/var/named/root.hints.old."
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

혹자들은 `root.hints` 파일을 인터닉의 ftp로부터 가져오는 것이 유용하다고 한다. ftp를 사용하지 말고 위의 방법으로 `root.hints` 를 업데이트 하기를. 위의 방법이 더 net상에서, 그리고 internic 에도 좋은 것이다.

9. 버전 4에서 8로의 변환

이 부분은 David E. Smith (dave@bureau42.ml.org)씨가 쓴 'using bind 8'에 있던 절이다. 새로운 절의 이름에 맞도록 약간 편집을 가했다.

별로 해야할 것은 없다. named.boot대신 named.conf를 사용하는 점 말고는 모든 것이 동일하다. bind8은 펄 스크립트로 옛 형식의 파일들을 새로운 형식에 맞게 변환한다. 다음은 옛형식으로 된 캐시 전용 네임 서버의 예이다.

```
directory /var/named
cache .                                 root.hints
primary 0.0.127.IN-ADDR.ARPA                    127.0.0.zone
primary localhost localhost.zone
```

아래의 명령을 bind8/src/bin/named 디렉토리 안에서 실행하라 (이것은 소스로 컴파일하는 배포본의 경우이며 바이너리 패키지를 사용했다면 스크립트가 어딘가에 있을 것이다. 어디에 있는지는 확신할 수 없다)

```
./named-bootconf.pl < named.boot > named.conf
```

이제 named.conf가 만들어 졌다.

```
// generated by named-bootconf.pl

options {
directory "/var/named";
};

zone "." {
type hint;
file "root.hints";
};

zone "0.0.127.IN-ADDR.ARPA" {
type master;
file "127.0.0.zone";
};

zone "localhost" {
type master;
file "localhost.zone";
};
```

`named.boot`안에 있던 내용들은 그대로 작동하지만 BIND-8의 새로운 기능이나 설정은 작동하지 않는다. 여기에 비슷하지만 더 효과적인 보다 정교한 `named.conf`가 있다

```
// This is a configuration file for named (from BIND 8.1 or later).
// It would normally be installed as /etc/named.conf.
// The only change made from the 'stock' named.conf (aside from this
// comment :) is that the directory line was uncommented, since I
// already had the zone files in /var/named.

options {
directory "/var/named";
datasize 20M;
};

zone "localhost" IN {
type master;
file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" IN {
type master;
file "127.0.0.zone";
};
```

```
zone "." IN {
type hint;
file "root.hints";
};
```

BIND 8 배포본의 bind8/src/bin/named/test에서 테스트 할 수 있으며 존 파일을 복사할 수 있다. 사람들은 이러한 식으로 입력해서 즉시 사용하곤 한다.

존 파일과 `root.hints` 파일은 서로 형식이 동일하며 그들을 업데이트하는 명령 또한 동일하다.

10. 질문과 답변

솔찍히 이해가 안가는 부분이 많은 부분입니다. 원문도 같이 살펴봐 주세요. 우선 나에게 메일 보내기 전에 이부분을 읽기를 바란다.

1. 내 네임서버가 named.boot를 찾고 있어요.

   이 HOWTO문서를 제대로 읽지 않은 경우이다. 이전 버젼인 BIND 4의 HOWTO를 보려면 아래를 참고하라. http://www.math.uio.no/~janl/DNS/

2. 방화벽 안에 있는 DNS를 사용하는 방법은?

   A hint: `forward only;`forward 밖에 없다.;. 또한

   ```
   query-source port 53;
   ```

   `named.conf`의 "option" 부분안에 넣으면 된다. caching 부분에 기술했었다.

3. 예를들어 `www.busy.site`같이 어떻게 해야 부하를 효과적이거나 비슷하게 유지하면서 DNS를 돌릴수 있는지?

   `www.busy.site`에 맞는 **A** 레코드를 사용하고 BIND 4.9.3 이나 그 이후 버젼을 사용하라. 그러면 BIND 가 알아서 돌면서 질의를 답해줄 것이며 이전 버젼에서는 작동하지 않을 것이다.

4. 인트라넷에(외부와 단절된) DNS 를 설정하고자 합니다. 뭘 해야 하는지?

   `root.hints` 파일을 제외하고 zone파일을 만들어라. 이것은 항상 새 hints 파일을 갱신할 필요가 없다는 이야기이다.

5. 2차(slave) 네임서버를 어떻게 설정합니까?

   만약에 1차 네임서버(primary/master)가 127.0.0.1 이면 당신의 2차 네임서버의 named.conf에 다음을 추가하면 된다.

```
  zone "linux.bogus" {
type slave;
file "sz/linux.bogus";
masters { 127.0.0.1; };
  };
```

`masters` 부분에 ';' (semicolon)으로 구분해서 여러 대체 master를 넣으면 그것도 같이 복사가 된다.

6. 넷상의 접속을 끊고 나서도 BIND가 구동되게 하고 싶다면.

   4가지 방법이 있다.

- BIND 8을 특별화하기, Adam L Rice가 내게 다이얼업 컴퓨터에서 DNS를 문제없이 사용하는 방법을 email로 알려왔다.

  ```
  나는 새 BIND 버젼에서 [<em/shuffeling files, -ed/] 이 더이상 필요하지 않다
  는
  것을 알았다. "forwarders"를 포함한 "forward" 로 어떻게 이것을 통제하는지를 알
  았다.
  기본설정은 "forward first"로 이것은 각각의 forwarders들에게 질의를 하고, 이
  것이
  실패하면 일반적으로 자신을 철저하게 조사한다. 이것은 gethostbyname()와 유사한
  결과를 주며 일반적으로 링크가 안될 때 오랜 시간을 허비한다.그러
  나 "forward only" 가
  설정되면, forwarders로 부터 응답이 없을 경우 바로 포기를 하
  고 gethostbyname() 로
  넘어간다. 고로 /etc파일을 솜씨있게 설정할 필요도 없고 서버를 재시작할 필요도 없
  다.

  내 경우에는 단지 몇줄만을 추가했다.

  forward only;
  forwarders { 193.133.58.5; };

  내 named.conf file에 설정했고 잘 작동된다. 이것의 단점은 단지 dump cache 상
  황을
  보는 DNS 프로그램의 정확도를 낮춘다는 것이다.좀더 확장성을 갖기 위해, 이러
  한 dump
  cache 프로그램을 돌리고 싶지만 리눅스에서 그다지 필요한 프로그램은 아닌 것 같다.
  ```

- Ian Clark에게서도 편지를 받았다.`<ic@deakin.edu.au>` 그는 그가 한 방식을 설명한다.

  ```
  나는 named를 'Masquerading'상태에서 구동한다.나는 2개
  의 root.hints file이 있고
  하나는 root.hints.real로 명명했다. 이것은 실제  root server 이름을 가지고 있
  고
  그리고 root.hints.fake 라 불리는 다른 파일은...
  ```

```
----
; root.hints.fake
; this file contains no information
----
```

내가 off라인일때 나는 root.hints.fake file 를 root.hints 로 복사하
고 named를
구동 시킨다..

내가 online일때 root.hints.real 을 root.hints 로 복사하고 named를 재시작한
다.

이것은 ip-down & ip-up 일 경우 행해진다.

내가 처음으로 off line 에서 질의를 했을 때 정밀하게 작동하지 않았으며 아래와 같
은
메시지만 출력榮.

```
Jan 28 20:10:11 hazchem named[10147]: No root nameserver for class IN
```

which I can live with.

잘 작동하는 것 같다. 외부 도메인에 대한 지연시간 없이 로컬 컴퓨터에서 네임서버를 운영할 수 있었으며 외부 도메인이 작동하는지 질의를 보낼 수 있었다.

Peter Denison 은 Ian 이 충분치 못하다고 생각하고 글을 보냈다.

```
연결되었을때  ) 캐쉬되어진(지역네트워크 포함) 목록은 즉시 서비스
                캐쉬되어지지 않은 부분은 ISP의 네임서버로 포워딩된다.
연결되지않을때) 지역네트워크 질의는 즉시 이루어짐
                다른 질의는 즉시 실패함
```

루트 캐쉬파일과 포워딩 질의의 변경조합은 잘 작동하지 않는다.

그래서 나는 내부LUG들과 토론을 해서 2개의 nameds를 다음과 같이 만들었다:

```
named-online:   forwards to ISPs nameserver
                master for localnet zone
                master for localnet reverse zone (1.168.192.in-
addr.arpa)
                master for 0.0.127.in-addr.arpa
                listens on port 60053

named-offline:  no forwarding
                "fake" root cache file
                slave for 3 local zones (master is 127.0.0.1:60053)
```

```
                listens on port 61053
그리고 이것을 포트 포워딩과 결합해서 off라인일때 port 53 부터 61053 까지 보내
며
온라인일 경우는 port 60053 을 사용했다. (나는 2.3.18대의 새 netfilter를 사용
했지만
옛날 (ipchains) 방식이 작동되었다.)

구버젼에서는 작동하지 않고 BIND 8.2에는 약간의 버그가 있다는 것을 명심해라.
슬레이브와 마스터가 비록 포트가 다른 경우라 해도 같은 IP를 쓰지 말아야 한다. 이
것은
trial 버젼이며 곳 나아질 것이다.
```

- Karl-Max Wanger로부터 BIND가 오프라인에서 NFS와 port mapper와 상호작용하는 법에 대한 정보를 받았다:

```
나는 모뎀으로 내 nameserver를 돌리고 네임서버는 cache 네임서버의 기능만을 한
다.
인증이나 모든 root.cache file에 대한 재질의는 없다. Slackware처
럼 nfsd 나 mountd
이전에 구동된다.

내 컴퓨터중 하나로 (a Libretto 30 notebook) 내 local LAN을 통해 다른 시스템
에
마운트를 할수 있는데 대부분은 그렇게 되지 않는 문제를 가지고 있다. 직렬포트에서
작동하는 PLIP, a PCMCIA ethernet card 나 PPP 등을 무시해도 같은 결과가 나온
다.

계속 추측하고 실험하면서 나는 무의미한 nfsd와 mountd 등록과정이 portmapper가
시작파일에서 실행될 때 계속 진행된다는 것을 알았다.(이러한 daemons 들을 의례히
부트부분에 넣었다.) named 를 nfsd 와 mountd 이후에 수행하면 이러한 문제는 사
라진다.

이러한 boot sequence를 수정하는데 특별한 불이익은 없다.이러한 방식을 권유해서
문제점을 방지했으면 한다.
```

- 마지막으로, 이 부분에 관한 HOWTO 정보이다. Ask Mr. DNS at (http://www.acmebw.com/askmrdns/#linux-dialup)에서 찾아보길. BIND 4 에 대한 정보이지만 BIND 8에도 많이 적용된다고 이야기 한다.

7. 캐쉬 전용 네임 서버는 그 캐쉬 정보를 어디에 저장하나? 캐쉬 크기를 제어할수 있는 방법은 없는가?

   캐시된 정보는 모두 메모리에 저장된다. 디스크에는 기록되지 않는다. named를 죽일 때마다 캐시는 사라진다. 캐시 정보는 어떤 방법으로든 제어할 수 없다. named는 어떤 간단한 규칙에 따라 캐시를 다루는데 다음과 같다. 어떤 목적으로든 캐시 정보나 캐시 크기를 제어할 방법은 없다. 그러고 싶다면 named를 해킹해서 수정하면 된다. 그러나 권하지는 않는다.

8. named가 재시작되는 동안은 캐시를 저장하는가? 저장하도록 할 수 있는가?

   없다. named는 멈출 때 캐시를 저장하지 않는다. 즉, named가 멈추었다가 다시 시작할 때마다 캐시는 새로 만들어 진다. named로 하여금 캐시를 파일로 저장하게 할 수는 없다. 그러고 싶다면 named를 해킹해서 수정하면 된다. 그러나 권하지는 않는다.

9. 어떻게 도메인을 얻어야 하고, 예를들어 `linux-rules.net`같은 도메인을 어떻게 구축해야 하며, 어떻게 하면 나에게 적합한 도메인을 할당할 수 있는가?

   서비스 공급업체에 연락을 해라. 이러한 부분들을 도와줄 것이다.대부분은 도메인을 얻기 위해 돈을 지불한다는 사실을 명심해라.

10. DNS server의 보안을 높히고 DNS를 분리하는 방법은?

    둘다 진보한 주제이다. 둘다 http://www.etherboy.com/dns/chrootdns.html 에 기술 되어 있다. 여기서는 더이상 설명하지 않는다.

11. DNS 관리자로서 더 많은 시간을 소비하려면

문서와 툴들

온라인 상과 출판등 문서들이 여럿있다. 이러한 문서들을 읽는 것은 더 많은 시간이 든다. Que (ISDN 0-7897-2273-9)에서 출판된 **The Concise Guide to DNS and BIND** (by Nicolai Langfeldt)가 있다. 이 책은 HOWTO와 비슷하지만 더 많고 자세한 내용을 기술해 놓았다. 그러나 일반적으로는 C. Liu and P. Albitz가 쓴 O'Reilly & Associates (ISBN 0-937175-82-X)의 **DNS and BIND** 가 있다. 이것 역시 매우 좋다. 3판까지 나왔다. BIND 4만큼이나 BIND 8을 많이 설명해 놓았다. Craig Hunt가 쓴 O'Reilly (ISBN 0-937175-82-X)사의 **TCP/IP Network Administration**의 DNS 부분도 좋다. Robert M. Pirsig의 **Zen and the Art of Motorcycle Maintenance** 역시 훌륭하다(ISBN 0688052304).

인터넷에서는 http://www.dns.net/dnsrd/ , http://www.isc.org/bind.html에서 관련 내용을 찾을 수 있다. FAQ, 레퍼런스 매뉴얼(BOG; Bind Operations Guide), 기사, 프로토콜 정의, DNS 해킹 (전부는 아니지만, 이 문서와 rfcs 대부분이 bind 배포본에 포함되어 있다.) 필자는 이들 대부분을 읽지 보지 않았다. 어쨌든 필자는 전문적으로 DNS를 관리하는 관리자는 아니다. 반면 Arnt Gulbrandsen은 BOG를 읽었고 그 사실에 황홀해 한다. :-) news:comp.protocols.tcp-ip.domains가 DNS 관련 뉴스그룹이다. 또한, DNS에 관한 RFC가 많이 있다. 아마도 가장 중요한 것은 이것들일 것이다.

RFC 2671

P. Vixie, **Extension Mechanisms for DNS (EDNS0)** August 1999.

RFC 2317

, BCP 20, H. Eidnes et. al. **Classless IN-ADDR.ARPA delegation**, March 1998. This is about CIDR, or classless subnet reverse lookups.

RFC 2308

, M. Andrews, **Negative Caching of DNS Queries**, March 1998. About negative caching and the $TTL zone file directive.

RFC 2219

, BCP 17, M. Hamilton and R. Wright, **Use of DNS Aliases for Network Services**, October 1997. About CNAME usage.

RFC 2182

, BCP 16, R. Elz et. al., **Selection and Operation of Secondary DNS Servers**, July 1997.

RFC 2052

A. Gulbrandsen, P. Vixie, **A DNS RR for specifying the location of services (DNS SRV)**, October 1996

RFC 1918

Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, **Address Allocation for Private Internets**, 02/29/1996.

RFC 1912

D. Barr, **Common DNS Operational and Configuration Errors**, 02/28/1996.

RFC 1912 Errors

B. Barr **Errors in RFC 1912**, this is available at http://www.cis.ohio-state.edu/~barr/rfc1912-errors.html

RFC 1713

A. Romao, **Tools for DNS debugging**, 11/03/1994.

RFC 1712

C. Farrell, M. Schulze, S. Pleitner, D. Baldoni, **DNS Encoding of Geographical Location**, 11/01/1994.

RFC 1183

R. Ullmann, P. Mockapetris, L. Mamakos, C. Everhart, **New DNS RR Definitions**, 10/08/1990.

RFC 1035

P. Mockapetris, **Domain names - implementation and specification**, 11/01/1987.

RFC 1034

P. Mockapetris, **Domain names - concepts and facilities**, 11/01/1987.

RFC 1033

M. Lottor, **Domain administrators operations guide**, 11/01/1987.

RFC 1032

M. Stahl, **Domain administrators guide**, 11/01/1987.

RFC 974

C. Partridge, **Mail routing and the domain system**, 01/01/1986.