



Building a Linux IPv6 DNS Server

By David Gordon and Ibrahim Haddad Open Systems Lab – Ericsson Research Corporate Unit

This article presents a tutorial on building an IPv6 DNS Linux server that provides IPv6 name resolution as part of an IPv6 network.

Introduction

This article falls within a series of LJ articles on the subject of IPv6 and Linux. The purpose of this series is to provide beginner and intermediate users with IPv6 tutorials that help them experiment with IPv6 and Linux, and to gain knowledge needed to ease the migration process from IPv4 to IPv6.

IPv6 is the next generation protocol designed by the Internet Engineering Task Force (IETF) to replace IPv4. IPv4 has been remarkably resilient. However, its initial design did not take into consideration several issues that are of importance today such as a large address space, mobility, security, autoconfiguration, and quality of service. To address these concern, IETF has developed a suite of protocols and standards known as IPv6, which incorporates many of the concepts and proposed methods for updating IPv4. As a result, IPv6 fixes a number of problems in IPv4 and adds many improvements and features to cater for the future (Mobile) Internet.

IPv6 is expected to gradually replace IPv4, with the two coexisting for a number of years during a transition period, where servers will be dual stack supporting both IPv4 and IPv6.

In this article, we look closely at IPv6 name resolution. The article aims at providing a technical tutorial to help readers setup their own IPv6 Linux DNS server to allow IPv6 name resolution using the latest version of BIND 9.x.

General network overview

In this section, we present a sample network scheme (Figure 1) with different IPv6 servers.

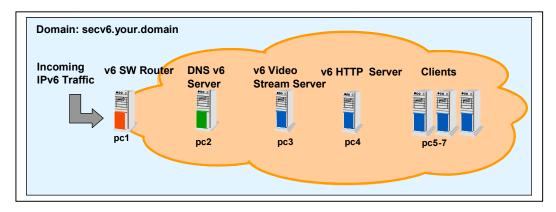


Figure 1: Sample network architecture

The following nodes are represented in this architecture:

- Routing server [pc1] acts as an IPv6 software router server and provides router advertisement for all IPv6 nodes.
- DNS IPv6 server [pc2] provides IPv6 name resolution (the main subject of this article)
- Application servers: we have two types of application servers: one providing video streaming [pc3] and the other is an Apache based web server [pc4]. (A follow up article "Apache talking IPv6" will be published in LJ in a later issue).
- Client machines [pc5-7] are used for testing purposes.

IPv6 name resolution

Domain names are a meaningful and easy-to-remember "handle" for Internet addresses. The domain name system is the way that Internet domain names are located and translated into Internet protocol addresses. Because maintaining a central list of domain name/IP address correspondences is not practical, the lists of domain names and IP addresses are distributed throughout the Internet in a hierarchy of authority. Typically, there is a DNS server within close geographic proximity to your access provider; this DNS server maps the domain names in DNS requests or forwards them to other servers in the Internet. For IPv6 DNS requests, both A6 and AAAA syntax are used to express IPv6 addresses.

AAAA resource record (called quad A record) is formatted as a fixed-length data. With AAAA, we can define DNS records for IPv6 name resolution as follows, just like A records in IPv4.

```
$ORIGIN X.EXAMPLE.
N AAAA 2345:00C1:CA11:0001:1234:5678:9ABC:DEF0
N AAAA 2345:00D2:DA11:0001:1234:5678:9ABC:DEF0
N AAAA 2345:000E:EB22:0001:1234:5678:9ABC:DEF0
```

A6 resource record is formatted as a variable-length data. With A6, it is possible to define an IPv6 address by using multiple DNS records. Here is an example taken from RFC2874:

If we translate the above into AAAA records, it will be as follows:

Once IPv6 name resolution is configured, we will add Domain Name System (DNSSEC) to our DNS server. DNSSEC provides three distinct services: key distribution, data origin authentication, and transaction and request authentication. The complete definition of DNSSEC is defined in RFC2535.

Supporting IPv6 in the Kernel and in the network binaries

An essential step before installing the IPv6 compliant BIND version is to enable IPv6 support in the Linux kernel and for the networking binaries on the system supporting IPv6. We have covered this topic in a previous article, "Supporting IPv6 on a Linux Server Node", in the August issue of LJ (http://www.linuxjournal.com/article.php?sid=4763). After following the tutorial presented in that article, you will be ready to install the latest BIND version with IPv6 support.

BIND and IPv6 support

The latest version of BIND is available from the Internet Software Consortium website (http://www.isc.org/products/BIND/BIND9.html). BIND version 9 is a major rewrite of nearly all aspects of the underlying BIND architecture. Many important features and enhancements were introduced in version 9, the most relevant to this article is the support for IPv6. BIND 9.x allows the DNS server to answer DNS queries on IPv6 sockets, provides support for IPv6 resource records (A6, DNAME, etc.), and supports bitstring labels, in addition to the availability of an experimental IPv6 resolver library. Many other features are also available and you can read more about them from the BIND website.

Installing BIND 9.x

BIND 9.2.1 is the latest stable release available at the time of writing. Our installation and configuration will follow this version. To install BIND, follow these steps:

- 1) Download the latest BIND version into /usr/src
- 2) Uncompress the package:

```
% tar -xzf bind-9.2.1.tar.gz
% cd bind-9.2.1
```

3) Although IPv6 support is native to BIND, it must be explicitly specified when compiling. In addition, since we want to support DNSSEC, we need to compile BIND with crypto support. OpenSSL 0.9.5a or newer should be installed. Running the configuration script with the needed options will look as follow:

```
% ./configure -enable-ipv6 -with-openssl
```

4) Compile and install as root:

```
% make && make install
```

By default, the BIND 9 files will be distributed in the file system as follows:

- Configuration files will be under /etc/named.conf
- The binary "named" will be in /usr/local/sbin
- All other related configuration files will go in /var/named

In the next section, we will demonstrate how to configure an authoritative server for our own domain.

Configuring IPv6 DNS and DNSSEC

DNS queries can be resolved in many different ways. For instance, a DNS server can use its cache to answer a query or contact other DNS servers on behalf of the client to fully resolve the name. When the

DNS server receives a query, it first checks to see if it can answer it authoritatively based on the resource record information contained in a locally configured zone on the server. If the queried name matches a corresponding resource record in the local zone information, the server answers authoritatively, using this information to resolve the queried name.

For a complete DNS query process, there are four existing DNS zones:

- 1. **Master**: The server has the master copy of the zone data and provides authoritative answers for it.
- 2. **Slave**: A slave zone is a copy of a master zone. Each slave zone has a list of masters that it may query to receive updates to its copy of the zone. A slave may, optionally, keep a copy of the zone saved on disk to speed startups. A single master server can have any number of slaves in order to distribute load.
- 3. **Stub**: A stub zone is much like a slave zone, and behaves similarly, but it only replicates the NS records of a master zone rather than the whole zone. Stub zones keep track of which DNS servers are authoritative for the organization. They directly contact the root DNS server to determine which servers are authoritative for which domain.
- 4. Forward: A forward zone directs all queries in the zone to other servers. As such, it acts as a caching DNS server for a network, or provides Internet DNS services to a network behind a firewall that limits outside DNS queries (obviously the forwarding DNS server must have DNS access to the internet). Note that this is similar to the global forwarding facility, but allows per-zone selection of forwarders.

To map this to our network (Figure 1), we need to create a master server for our own domain, secv6.your.domain. Listing 1 provides a sample /etc/named.conf configuration:

```
options {
directory "/var/named";
// a caching only nameserver config
zone "
      "." IN {
type hint;
file "named.ca";
// this defines the loopback name lookup zone "localhost" IN \{
type master;
file "master/localhost.zone";
allow-update { none; };
// this defines the loopback reverse name lookup
zone "0.0.127.in-addr.arpa" IN {
type master;
file "master/localhost.rev";
allow-update { none; };
// This defines the secv6 domain name lookup
// Secure (signed) zone file is secv6.your.domain.signed
// Regular zone file is secv6.your.domain
zone "secv6.your.domain" IN {
type master;
file "master/secv6.your.domain.signed";
// file "master/secv6.your.domain";
// this defines the secv6 domain reverse name lookup (AAAA)
zone "secv6.int" IN {
type master;
file "master/secv6.int";
// this defines the secv6 domain reverse name lookup (A6)
```

```
zone "secv6.arpa" IN {
type master;
file "master/secv6.rev";
};
key "key" {
algorithm hmac-md5;
secret "HxbmAnSOOKtTgBWWXPRyOGayhbBaSntquVxcxBDjmAmjrmhgDUVFcFNcfmHC";
};
```

Listing 1: /etc/named.conf

The next step is to define the configuration files that will describe our domain. Notice that until now we have not touched on the specifics of IPv6. As for DNSSEC, the file <code>/var/named/master/secv6.your.domain.signed</code> will be the domain file signed by the zone key of the DNS server. This is important to DNSSEC, since clients will be able to authenticate all subsequent DNS requests. The DNS server zone key is different from the key in the configuration file and the details on how to generate a zone key will be discussed later in the article.

The next file to edit is /var/named/master/secv6.your.domain. The following example (Listing 2) uses both AAAA and A6 formats. The \$INCLUDE directive at the end includes the public portion of the zone key. Keep the private portion of the key private! The private key has 'private' appended at the end, whereas 'key' postfixes the public key. If you have any concerns regarding DNSSEC keys and their permissions, consult the BIND manual.

In Listing 2, we display a typical IPv6 DNS domain configuration for secv6.your.domain:

```
$TTL 86400
$ORIGIN secv6.your.domain.
@ IN SOA secv6.your.domain. hostmaster.secv6.your.domain. (
2002011442; Serial number (yyyymmdd-num)
3H ; Refresh
15M ; Retry
1W ; Expire
1D ) ; Minimun
IN MX 10 noah.your.domain.
IN NS ns.secv6.your.domain.
$ORIGIN secv6.your.domain.
ns 1D IN AAAA fec0::1:250:b7ff:fe14:35d0
1D IN A6 0 fec0::1:250:b7ff:fe14:35d0
secv6.your.domain. 1D IN AAAA fec0::1:250:b7ff:fe14:35d0 1D IN A6 0 fec0::1:250:b7ff:fe14:35d0
pc5 1D IN A6 0 fec0::1:250:b7ff:fe14:361b
                                  1D IN AAAA fec0::1:250:b7ff:fe14:361b
                                  1D IN AAAA fec0::1:250:b7ff:fe14:365a
pc7 1D IN A6 0 fec0::1:250:b7ff:fe14:365a
                                 1D IN AAAA fec0::1:250:b9ff:fe00:12e
pc1 1D IN A6 0 fec0::1:250:b9ff:fe00:12e
pc1 1D IN A6 0 fec0:0:0:1::1
                                  1D IN AAAA fec0:0:0:1::1
$INCLUDE "/var/named/master/Ksecv6.your.domain.+003+27034.key"
```

Listing 2: /var/named/master/secv6.your.domain

For configuration files in /var/named/master, Hostmaster is actually the email of the administrator where the first dot replaces '@' because of syntax restrictions. In addition, the first number for IN SOA structure at the beginning of Listing 2 is the serial number conventionally expressed as YYYYMMDDNN where NN is a number incremented each time the DNS zone is updated.

Now, we will examine how to generate a zone key. The working directory is important since the keys will be placed there. We suggest placing the keys in $\sqrt{\sqrt{named/master}}$. The following command will generate a 768 bit DSA key for the zone:

```
% dnssec-keygen -a DSA -b 768 -n ZONE secv6.your.domain
```

By default, all zone keys, which have an available private key, are used to generate signatures. The keys must be either in the working directory or included in the zone file. The following command signs the secv6.your.domain zone, assuming it is in a file called /var/named/master/secv6.your.domain:

```
% dnssec-signzone -o secv6.your.domain secv6.your.domain
```

One output file is produced: /var/named/master/secv6.your.domain.signed. This file should be referenced by /etc/named.conf as the input file for the zone.

The remaining configuration files are localhost.zone (Listing 3), localhost.rev (Listing 4), secv6.rev (Listing 5), and secv6.int (Listing 6) are presented each in their respective listing. The difference between reverse lookup zone files <code>secv6.rev</code> and <code>secv6.int</code> is that one can be specified using A6 strings (that do not need to be reversed in <code>secv6.rev</code>) and the other with reverse AAAA format addresses in <code>secv6.int</code>. For instance, ping6 can only refer to <code>secv6.int</code> domain because it does not support A6 format.

```
// localhost.zone Allows for local communications using the loopback interface $TTL 86400 $ORIGIN localhost.
@ 1D IN SOA @ root (
42 ; serial (d. adams)
3H ; refresh
15M ; retry
1W ; expiry
1D ) ; minimum
1D IN NS @
1D IN A 127.0.0.1
```

Listing 3: /var/named/master/localhost.zone

Listing 4: /var/named/master/localhost.rev

```
// secv6.rev Defines reverse lookup for secv6 domain in A6 format
$TTL 86400
$ORIGIN secv6.arpa.
@ IN SOA secv6.arpa. hostmaster.secv6.your.domain. (
2002011442; Serial number (yyyymmdd-num)
3H; Refresh
15M; Retry
```

```
1W ; Expire
1D ) ; Minimun
NS ns.secv6.your.domain.
MX 10 noah.your.domain.
; fec0:0:0:1::/64
$ORIGIN \[xfec0000000000001/64].secv6.arpa.
\[x0250b7fffe1435d0/64] 1D IN PTR pc2.secv6.your.domain.
\[x0250b9fffe000131/64] 1D IN PTR pc3.secv6.your.domain.
\[x0250b7fffe143617/64] 1D IN PTR pc6.secv6.your.domain.
\[x0250b7fffe143617/64] 1D IN PTR pc6.secv6.your.domain.
\[x0250b7fffe143616/64] 1D IN PTR pc4.secv6.your.domain.
\[x0250b7fffe14365a/64] 1D IN PTR pc5.secv6.your.domain.
\[x0250b7fffe14365a/64] 1D IN PTR pc7.secv6.your.domain.
\[x0250b9fffe00012e/64] 1D IN PTR pc1.secv6.your.domain.
```

Listing 5: /var/named/master/secv6.rev

```
// secv6.int
                      Defines reverse lookup for secv6 domain in AAAA format
$TTL 86400
$ORIGIN secv6.int.
@ IN SOA secv6.int. hostmaster.secv6.your.domain. (
2002011442 ; Serial number (yyyymmdd-num)
3H ; Refresh
15M ; Retry
1W ; Expire
1D ) ; Minimun
NS ns.secv6.your.domain.
MX 10 noah.your.domain.
; fec0:0:0:1::/64
$ORIGIN 1.0.0.0.0.0.0.0.0.0.0.0.c.e.f.secv6.int.
0.d.5.3.4.1.e.f.f.f.7.b.0.5.2.0 IN PTR pc2.secv6.your.domain.
e.2.1.0.0.0.e.f.f.f.9.b.0.5.2.0 IN PTR pc1.secv6.your.domain.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN PTR pcl.secv6.your.domain.
1.3.1.0.0.0.e.f.f.f.9.b.0.5.2.0 IN PTR pc3.secv6.your.domain.
7.1.6.3.4.1.e.f.f.f.7.b.0.5.2.0 IN PTR pc6.secv6.your.domain.
4.c.5.3.4.1.e.f.f.f.7.b.0.5.2.0 IN PTR pc4.secv6.your.domain.
b.1.6.3.4.1.e.f.f.f.7.b.0.5.2.0 IN PTR pc5.secv6.your.domain.
a.5.6.3.4.1.e.f.f.f.7.b.0.5.2.0 IN PTR pc7.secv6.your.domain.
```

Listing 6: /var/named/master/secv6.int

Starting DNS daemon

Once the installation and configuration steps are complete, you are ready to start the DNS daemon on pc2. 'Named' will use /etc/named.conf by default, although you can specify a different configuration file with '-c' option if you choose so. Depending on where you installed the daemon:

```
pc2% /usr/local/sbin/named
```

One additional configuration step is needed on the machines within the IPv6 network which is to update /etc/resolv.conf (Listing 7) to contain the DNS server's IP address. It is important that the IP address is included and not the hostname of the DNS server, because this is where the system will look to find the address of the DNS. In other words, if you specified the hostname of the DNS server here, how will the system know what IP address corresponds to the DNS' hostname?

```
# To enable secv6 domain, start named on pc2
# and use this file as /etc/resolv.conf
search secv6.your.domain
nameserver fec0::1:250:b7ff:fe14:35d0
```

Listing 7: /etc/resolv.conf on client machines

Testing the setup

To test the setup, we present two simple tests. The first verifies that A6 addresses are enabled in the DNS server; the second verifies that AAAA addresses are supported by the DNS server. The tests were performed on pc2. Please note that we present only the meaningful output, otherwise the listing would be too long.

For the first example, we use the DNS lookup utility 'dig'. We will perform a lookup on secv6 domain in A6 format (Example 1), then we will perform a lookup in AAAA format (Example 2). In both cases, we are not specifying a specific address to lookup, thus '0.0.0.0'.

```
pc2% dig 0.0.0.0 secv6.your.domain a6
                                                            # Check A6 addresses
; <<>> DiG 9.1.0 <<>> 0.0.0.0 secv6.your.domain A6
ſ...1
;secv6.your.domain. IN A6
;; ANSWER SECTION:
secv6.your.domain. 86400 IN A6 0 fec0::1:250:b7ff:fe14:35d0
;; AUTHORITY SECTION:
secv6.your.domain. 86400 IN NS ns.secv6.your.domain.
;; ADDITIONAL SECTION:
ns.secv6.your.domain. 86400 IN A6 0 fec0::1:250:b7ff:fe14:35d0
ns.secv6.your.domain. 86400 IN AAAA fec0::1:250:b7ff:fe14:35d0
Example 1: A6 DNS query
pc2% dig 0.0.0.0 secv6.your.domain aaaa
                                                            # Check AAAA addresses
; <<>> DiG 9.1.0 <<>> 0.0.0.0 secv6.your.domain AAAA
ſ...1
;secv6.your.domain. IN AAAA
;; ANSWER SECTION:
secv6.your.domain. 86400 IN AAAA fec0::1:250:b7ff:fe14:35d0
;; AUTHORITY SECTION:
secv6.your.domain. 86400 IN NS ns.secv6.your.domain.
;; ADDITIONAL SECTION:
ns.secv6.your.domain. 86400 IN A6 0 fec0::1:250:b7ff:fe14:35d0
```

Example 2: AAAA DNS query

We also include samples of an SSH session connection first using an IPv6 address (Example 3) and then using an IPv6 hostname (Example 4):

```
pc3% ssh -6 fec0::1:250:b7ff:fe14:35c4
user@fec0::1:250:b7ff:fe14:35c4's password:
Last login: Fri Nov 8 13:39:17 2002 from pc3.secv6.your.domain pc4%
```

ns.secv6.your.domain. 86400 IN AAAA fec0::1:250:b7ff:fe14:35d0

Example 3: SSH session using IPv6 address directly

```
pc3% ssh -6 pc4.secv6.your.domain
user@pc4.secv6.your.domain 's password:
Last login: Fri Nov 8 13:30:55 2002 from somewhere
pc4%
```

Example 4: SSH session using IPv6 hostname

Sample server applications using IPv6

In our IPv6 network, we presented two application servers: Apache as a web server and VideoLan for video streaming. To test IPv6 name resolution when streaming a video, a user on client node pc5 will access the video-streaming server on pc3.

The video server is on pc3 (fec0::1:250:b7ff:fe14:5768) and the video client is on pc5 (fec0::1:250:b7ff:fe50:7c). Sniffing the network communications on pc5 with tcpdump, we captured packets from the video stream. Here is a portion of the trace:

```
% tcpdump ip6  # only trace IPv6 traffic, must be run as root or setuid root
[snip...]
02:09:26.716040 fec0::1:250:b7ff:fe14:5768.32769 > fec0::1:250:b7ff:fe50:7c.1234: udp 1316
02:09:26.735805 fec0::1:250:b7ff:fe14:5768.32769 > fec0::1:250:b7ff:fe50:7c.1234: udp 1316
02:09:26.735971 fec0::1:250:b7ff:fe14:5768.32769 > fec0::1:250:b7ff:fe50:7c.1234: udp 1316
02:09:26.736082 fec0::1:250:b7ff:fe14:5768.32769 > fec0::1:250:b7ff:fe50:7c.1234: udp 1316
02:09:26.755810 fec0::1:250:b7ff:fe14:5768.32769 > fec0::1:250:b7ff:fe50:7c.1234: udp 1316
02:09:26.775787 fec0::1:250:b7ff:fe14:5768.32769 > fec0::1:250:b7ff:fe50:7c.1234: udp 1316
```

The video is properly displayed using X11 output on Linux X server; we illustrate a caption of it in Figure 2.



Figure 2: The output stream of IPv6 video

Conclusion

IPv6 is becoming a reality. For the next few years, we need to be able to support both IPv4 and IPv6 on our servers before the complete transition to IPv6 occurs. We need different pieces of the puzzle to achieve a full migration to IPv6 and one essential piece is an IPv6 compliant BIND implementation. We hope you find this article useful and educational and we look forward to present more articles on the subject of IPv6 and Linux.

References

Linux Kernel http://www.kernel.org

IPv6 How-to http://www.bieringer.de/linux/IPv6-HOWTO/IPv6-HOWTO.html

BIND http://www.isc.org/products/BIND/BIND9.html

BIND Manual http://www.crt.se/dnssec/bind9/Bv9ARM.html

Supporting IPv6 on a Linux Server Node http://www.linuxjournal.com/article.php?sid=4763

IPv6 Linux Implementations http://www.linuxjournal.com/article.php?sid=5468

IP Version 6 Addressing Architecture

http://www.rfc-editor.org/rfc/rfc2373.txt

DNSSEC and IPv6 A6

ftp://ftp.rfc-editor.org/in-notes/rfc3226.txt

DNSSEC Signing Authority

ftp://ftp.rfc-editor.org/in-notes/rfc3008.txt

Comparison of AAAA and A6 http://www.ietf.org/proceedings/02mar/l-D/draft-ietf-dnsext-aaaa-a6-01.txt

IPv6 support for DNS http://www.ietf.org/rfc/rfc2874.txt
DNSSEC http://www.ietf.org/rfc/rfc2535.txt

Acknowledgements

• Ericsson Research Corporate Unit for approving the publication of this article and the Open Systems Lab for supporting our work with Linux and IPv6.

• Simon Jubinville, Open Systems Lab, for his reviews.

About the authors

David Gordon (David.Gordon@Ericsson.ca) is a computer science intern at Ericsson Research - Open Systems Lab. He is completing his undergraduate studies in Computer Science at Sherbrooke University. His research interests include security, next-generation IP networks, and wireless technologies.

Ibrahim Haddad (Ibrahim.Haddad@Ericsson.com) is a Researcher at the Ericsson Corporate Unit of Research in Montreal, Canada.