Unbound

Unbound (https://unbound.net/) is a validating, recursive, and caching DNS resolver. According to **Wikipedia:Unbound (DNS Server)**:

> Unbound has supplanted the Berkeley Internet Name Domain (**BIND**) as the default, base-system name server in several open source projects, where it is perceived as smaller, more modern, and more secure for most applications.

Related articles

- DNSSEC
- **BIND**
- **DNSCrypt**
- dnsmasq
- **Pdnsd**

Contents

Installation

Install the `unbound (https://www.archlinux.org/packages/?name=unbound)` package.

Additionally, the **expat (https://www.archlinux.org/packages/?name=expat)** package is required for **#DNSSEC validation**.

Configuration

A default configuration is already included at `/etc/unbound/unbound.conf`. Additionally, there is a commented sample configuration file with other available options located at `/etc/unbound/unbound.conf.example`. The following sections highlight different settings for the configuration file. See **unbound.conf(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/unbound.conf.5)** for other settings and more details.

Unless otherwise specified, any options listed in this section are to be placed under the `server` section in the configuration like so:

/etc/unbound/unbound.conf

```
server:
  ...
  setting: value
  ...
```

Local DNS server

If you want to use *unbound* as your local DNS server, set your nameserver to `127.0.0.1` in your **resolv.conf**. You will want to have your nameserver be **preserved**.

Tip: A simple way to do this is to install the `openresolv (https://www.archlinux.org/packages/?name=openresolv)` package and uncomment the line containing `name_servers=127.0.0.1` in `/etc/resolvconf.conf`. Then run `resolvconf -u` to generate `/etc/resolv.conf`.

See **Resolv.conf#Testing** on how to test your settings.

Check specifically that the server being used is `127.0.0.1` after making permanent changes to **resolv.conf**.

You will also need to setup *unbound* such that it is **#Forwarding queries** to the DNS servers of your choice.

Root hints

For querying a host that is not cached as an address the resolver needs to start at the top of the server tree and query the root servers to know where to go for the top level domain for the address being queried. Unbound comes with default builtin hints, but it is good practice to use a root-hints file since the builtin hints may become outdated.

First point *unbound* to the `root.hints` file:

```
root-hints: "/etc/unbound/root.hints"
```

Then, put a *root hints* file into the *unbound* configuration directory. The simplest way to do this is to run the command:

```
# curl -o /etc/unbound/root.hints https://www.internic.net/domain/named.cache
```

It is a good idea to update `root.hints` every six months or so in order to make sure the list of root servers is up to date. This can be done manually or by using **Systemd/Timers**. See **#Roothints systemd timer** for an example.

DNSSEC validation

To use **DNSSEC** validation, point *unbound* to the server trust anchor file by adding the following setting under `server:` :

/etc/unbound/unbound.conf

```
trust-anchor-file: trusted-key.key
```

Also make sure that if general **#Forwarding queries** have been set to DNS servers that do not support DNSSEC, then comment them out; otherwise, DNS queries will fail. DNSSEC validation will only be done if the DNS server being queried supports it.

Note: Including DNSSEC checking significantly increases DNS lookup times for initial lookups before the address is cached.

Testing validation

To test if DNSSEC is working, after **starting** `unbound.service`, do:

```
$ unbound-host -C /etc/unbound/unbound.conf -v sigok.verteiltesysteme.net
```

The response should be the ip address with the word `(secure)` next to it.

```
$ unbound-host -C /etc/unbound/unbound.conf -v sigfail.verteiltesysteme.net
```

Here the response should include `(BOGUS (security failure))`.

Additionally you can use *drill* to test the resolver as follows:

```
$ drill sigfail.verteiltesysteme.net
$ drill sigok.verteiltesysteme.net
```

The first command should give an `rcode` of `SERVFAIL`. The second should give an `rcode` of `NOERROR`.

Forwarding queries

Tip: Unbound can be used with **DNSCrypt** by setting up forwarding. See **DNSCrypt#Unbound**.

If you only want to forward queries to an external DNS server, skip ahead to **#Forward all remaining requests**.

Allow local network to use DNS

If you have a local network which you wish to have DNS queries for and there is a local DNS server that you would like to forward queries to then you should include this line:

```
private-address: local_subnet/subnet_mask
```

For example:

```
private-address: 10.0.0.0/24
```

Note: You can use private-address to protect against DNS Rebind attacks. Therefore you may enable RFC1918 networks (10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 169.254.0.0/16 fd00::/8 fe80::/10). Unbound may enable this feature by default in future releases.

Include local DNS server

To include a local DNS server for both forward and reverse local addresses a set of lines similar to these below is necessary with a forward and reverse lookup (choose the IP address of the server providing DNS for the local network accordingly by changing 10.0.0.1 in the

lines below):

```
local-zone: "10.in-addr.arpa." transparent
```

This line above is important to get the reverse lookup to work correctly.

```
forward-zone:
name: "mynetwork.com."
forward-addr: 10.0.0.1
```

```
forward-zone:
name: "10.in-addr.arpa."
forward-addr: 10.0.0.1
```

Note: There is a difference between forward zones and stub zones - stub zones will only work when connected to an authoritative DNS server directly. This would work for lookups from a **BIND** DNS server if it is providing authoritative DNS - but if you are referring queries to an *unbound* server in which internal lookups are forwarded on to another DNS server, then defining the referral as a stub zone in the machine here will not work. In that case it is necessary to define a forward zone as above, since forward zones can have daisy chain lookups onward to other DNS servers. i.e. forward zones can refer queries to recursive DNS servers. This distinction is important as you do not get any error messages indicating what the problem is if you use a stub zone inappropriately.

You can set up the localhost forward and reverse lookups with the following lines:

```
local-zone: "localhost." static
local-data: "localhost. 10800 IN NS localhost."
local-data: "localhost. 10800 IN SOA localhost. nobody.invalid. 1 3600 1200 604800 10800"
local-data: "localhost. 10800 IN A 127.0.0.1"
local-zone: "127.in-addr.arpa." static
local-data: "127.in-addr.arpa. 10800 IN NS localhost."
local-data: "127.in-addr.arpa. 10800 IN SOA localhost. nobody.invalid. 2 3600 1200 604800 10800"
local-data: "1.0.0.127.in-addr.arpa. 10800 IN PTR localhost."
```

Forward all remaining requests

To use specific servers for default forward zones that are outside of the local machine and outside of the local network add a forward zone with the name `.` to the configuration file. In this example, all requests are forwarded to Google's DNS servers:

```
forward-zone:
  name: "."
  forward-addr: 8.8.8.8
  forward-addr: 8.8.4.4
```

Access control

You can specify the interfaces to answer queries from by IP address. The default, is to listen on *localhost*.

To listen on all interfaces, use the following:

```
interface: 0.0.0.0
```

To control which systems can access the server by IP address, use the `access-control` option:

```
access-control: subnet action
```

For example:

```
access-control: 192.168.1.0/24 allow
```

action can be one of `deny` (drop message), `refuse` (polite error reply), `allow` (recursive ok), or `allow_snoop` (recursive and nonrecursive ok). By default everything is refused except for localhost.

Usage

Starting Unbound

Start/enable the `unbound.service` systemd service.

Remotely control Unbound

unbound ships with the `unbound-control` utility which enables us to remotely administer the unbound server. It is similar to the **pdnsd-ctl** command of **pdnsd (https://www.archlinux.org/packages/?name=pdnsd)**.

Setting up unbound-control

Before you can start using it, the following steps need to be performed:

1) Firstly, you need to run the following command

```
# unbound-control-setup
```

which will generate a self-signed certificate and private key for the server, as well as the client. These files will be created in the `/etc/unbound` directory.

2) After that, edit `/etc/unbound/unbound.conf` and put the following contents in that. The `control-enable: yes` option is necessary, the rest can be adjusted as required.

```
remote-control:
    # Enable remote control with unbound-control(8) here.
    # set up the keys and certificates with unbound-control-setup.
    control-enable: yes

    # what interfaces are listened to for remote control.
    # give 0.0.0.0 and ::0 to listen to all interfaces.
    control-interface: 127.0.0.1

    # port number for remote control operations.
    control-port: 8953
```

```
    # unbound server key file.
    server-key-file: "/etc/unbound/unbound_server.key"

    # unbound server certificate file.
    server-cert-file: "/etc/unbound/unbound_server.pem"

    # unbound-control key file.
    control-key-file: "/etc/unbound/unbound_control.key"

    # unbound-control certificate file.
    control-cert-file: "/etc/unbound/unbound_control.pem"
```

Using unbound-control

Some of the commands that can be used with *unbound-control* are:

- print statistics without resetting them

```
# unbound-control stats_noreset
```

- dump cache to stdout

```
# unbound-control dump_cache
```

- flush cache and reload configuration

```
# unbound-control reload
```

Please refer to **unbound-control(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/unbound-control.8)** for a detailed look at the operations it supports.

Tips and tricks

Block advertising

You can use the following file and simply include it in your unbound configuration: **adservers (https://pgl.yoyo.org/adservers/serverlist.php?hostformat=unbound&showintro=0&startdate%5Bday%5D=&startdate%5Bmonth%5D=&startdate%5Byear%5D=&mimetype=plaintext)**

/etc/unbound/unbound.conf

```
server:
...
  include: /etc/unbound/adservers
```

Note: In order to return some OK statuses on those hosts, you can change the 127.0.0.1 redirection to a server you control and have that server respond with empty 204 replies, see **this page (http://www.shadowandy.net/2014/04/adblocking-nginx-serving-1-pixel-gif-204-content.htm)**

Tip: To convert a hosts file from another source to the unbound format do:

```
$ cat hosts | grep '^0\.0\.0\.0' | awk '{print "local-zone: \""$2"\" redirect\nlocal-data: \""$2" A 0.0.0.0\""}' > /etc/unbound/adservers
```

Adding an authoritative DNS server

For users who wish to run both a validating, recursive, caching DNS server as well as an authoritative DNS server on a single machine then it may be useful to refer to the wiki page **nsd** which gives an example of a configuration for such a system. Having one server for authoritative DNS queries and a separate DNS server for the validating, recursive, caching DNS functions gives increased security over a single DNS server providing all of these functions. Many users have used bind as a single DNS server, and some help on migration from bind to the combination of running nsd and bind is provided in the **nsd** wiki page.

WAN facing DNS

It is also possible to change the configuration files and interfaces on which the server is listening so that DNS queries from machines outside of the local network can access specific machines within the LAN. This is useful for web and mail servers which are accessible from anywhere, and the same techniques can be employed as has been achieved using bind for many years, in combination with suitable port forwarding on firewall machines to forward incoming requests to the right machine.

Roothints systemd timer

Here is an example systemd service and timer that update `root.hints` monthly using the method in **#Root hints**:

```
/etc/systemd/system/roothints.service
```

```
[Unit]
Description=Update root hints for unbound
After=network.target

[Service]
ExecStart=/usr/bin/curl -o /etc/unbound/root.hints https://www.internic.net/domain/named.cache
```

```
/etc/systemd/system/roothints.timer
```

```
[Unit]
Description=Run root.hints monthly

[Timer]
OnCalendar=monthly
Persistent=true

[Install]
WantedBy=timers.target
```

Start/enable the `roothints.timer` systemd timer.

Troubleshooting

Issues concerning num-threads

The man page for `unbound.conf` mentions:

```
outgoing-range: <number>
       Number of ports to open. This number of file  descriptors  can  be  opened  per thread.
```

and some sources suggest that the `num-threads` parameter should be set to the number of cpu cores. The sample `unbound.conf.example` file merely has:

```
# number of threads to create. 1 disables threading.
# num-threads: 1
```

However it is not possible to arbitrarily increase `num-threads` above `1` without causing *unbound* to start with warnings in the logs about exceeding the number of file descriptors. In reality for most users running on small networks or on a single machine it should be unnecessary to seek performance enhancement by increasing `num-threads` above `1`. If you do wish to do so then refer to **official documentation (http://www.unbound.net/documentation/howto_optimise.html)** and the following rule of thumb should work:

> *Set `num-threads` equal to the number of CPU cores on the system. E.g. for 4 CPUs with 2 cores each, use 8.*

Set the `outgoing-range` to as large a value as possible, see the sections in the referred web page above on how to overcome the limit of `1024` in total. This services more clients at a time. With 1 core, try `950`. With 2 cores, try `450`. With 4 cores try `200`. The `num-queries-per-thread` is best set at half the number of the `outgoing-range`.

Because of the limit on `outgoing-range` thus also limits `num-queries-per-thread`, it is better to compile with **libevent (https://www.archlinux.org/packages/?name=libevent)**, so that there is no `1024` limit on `outgoing-range`. If you need to compile this way for a heavy duty DNS server then you will need to compile the programme from source instead of using the **unbound (https://www.archlinux.org/packages/?name=unbound)** package.

See also

- **Fedora change to Unbound (https://fedoraproject.org/wiki/Changes/Default_Local_DNS_Resolver)**
- **Block hosts that contain advertisements (https://github.com/jodrell/unbound-block-hosts/)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Unbound&oldid=505851"
