PowerDNS Recursor Documentation

PowerDNS.COM BV

Jul 15, 2018

CONTENTS

CHAPTER

ONE

INTRODUCTION

The PowerDNS Recursor is a high-performance DNS recursor with built-in scripting capabilities. It is known to power the resolving needs of over 150 million internet connections.

The documentation is only for the 4.1 series, users of older versions are urged to upgrade!

This documentation is also available as a PDF document.

1.1 Notable features

- Can handle tens of thousands of concurrent questions. A quad Xeon 3GHz has been measured functioning very well at 400000 real life replayed packets per second.
- Relies heavily on Standard C++ Library infrastructure, which makes for little code.
- Powered by a highly modern DNS packet parser that should be resistant against many forms of buffer overflows.
- Best spoofing protection that we know about, involving both source port randomisation and spoofing detection.
- Uses 'connected' UDP sockets which allow the recursor to react quickly to unreachable hosts or hosts for which the server is running, but the nameserver is down. This makes the recursor faster to respond in case of misconfigured domains, which are sadly very frequent.
- Special support for FreeBSD, Linux and Solaris stateful multiplexing (kqueue, epoll, completion ports, /dev/poll).
- Very fast, and contains innovative query-throttling code to save time talking to obsolete or broken nameservers.
- Code is written linearly, sequentially, which means that there are no problems with 'query restart' or anything.
- The algorithm is simple and quite nifty.
- Does DNSSEC validation
- Is highly scriptable in Lua

1.2 Getting support

PowerDNS is an open source program so you may get help from the PowerDNS users' community or from its authors. You may also help others (please do).

Public support is available via several different channels:

- This documentation
- The mailing list
- `#powerdns` on irc.oftc.net

The PowerDNS company can provide help or support you in private as well. For first class and rapid support, please contact powerdns.support@powerdns.com, or see the .com website.

1.2.1 My information is confidential, must I send it to the mailing list or discuss on IRC?

Yes, we have a support policy called "Open Source Support: out in the open".

If you desire privacy, please consider entering a support relationship with us, in which case we invite you to contact powerdns.support.sales@powerdns.com.

1.2.2 I have a question!

This happens, we're here to help! Read below on how you can get help

1.2.3 What details should I supply?

Start out with stating what you think should be happening. Quite often, wrong expectations are the actual problem. Furthermore, your operating system, which version of PowerDNS you use and where you got it from (RPM, .DEB, tar.bz2). If you *compiled* it yourself, what were the `./configure` parameters.

If possible, supply the actual name of your domain and the IP address of your server(s).

1.2.4 I found a bug!

As much as we'd like to think we are perfect, bugs happen. If you have found a bug, please file a bug report on GitHub. Please fill in the template and we'll try our best to help you.

1.2.5 I found a security issue!

Please report this in private, see the *PowerDNS Security Policy*.

1.2.6 I have a good idea for a feature!

We like to work on new things! You can file a feature request on GitHub.

CHAPTER

TWO

GETTING STARTED

The PowerDNS Recursor can be installed on any modern unix-like system and is available in the software repositories for all major Linux distributions and BSDs.

2.1 Installation

The Recursor is available for many platforms, instructions are provided here for several platforms.

note: PowerDNS itself provides repositories for several Recursor versions for different operating systems. Checkout the repositories for more information.

2.1.1 Debian-based distributions

On Debian, Ubuntu, Linux Mint and related distributions, running `apt-get install pdns-recursor` as root will install the Recursor.

2.1.2 Enterprise Linux

On RedHat, CentOS and related distributions, ensure that EPEL is available. To install the PowerDNS Recursor, run `yum install pdns-recursor` as root.

2.1.3 FreeBSD

On FreeBSD the Recursor is available through the ports system. Run `pkg install powerdns-recursor` as root to install.

To compile yourself from ports, run `cd /usr/ports/dns/powerdns-recursor/ && make install clean`.

2.1.4 From Source

See *Compiling the PowerDNS Recursor* for instructions on how to build the PowerDNS Recursor from source.

2.2 Configuring the Recursor

The configuration file is called `recursor.conf` and is located in the `SYSCONFDIR` defined at compile-time. This is usually `/etc/powerdns`, `/etc/pdns`, `/etc/pdns-recursor`, `/usr/local/etc` or similar.

Run `pdns_recursor --no-config --config | grep config-dir` to find this location on you installation.

The PowerDNS Recursor listens on the local loopback interface by default, this can be changed with the *local-address* setting.

Now access will need to be granted to the Recursor. The *allow-from* setting lists the subnets that can communicate with the Recursor.

An example configuration is shown below. Change this to match the local infrastructure.

```
local-address=192.0.2.25, 2001:DB8::1:25
allow-from=192.0.2.0/24, 2001:DB8::1:/64
```

After a restart of the Recursor, it will answer queries on 192.0.2.25 and 2001:DB8::1:25, but only for queries with a source address in the 192.0.2.0/24 and 2001:DB8::1:/64 networks.

The recursor is now ready to be used. For more options that can be set in `recursor.conf` see the *list of settings*. Guidance on interaction with the Recursor is documented *here* If dynamic answer generation is needed or policies need to be applied to queries, the *scripting manual* will come in handy.

2.3 Using Ansible

The PowerDNS Recursor can also be installed and configured with Ansible. There is a role available from the PowerDNS authors.

CHAPTER

THREE

OPERATING THE POWERDNS RECURSOR

3.1 Logging

In a production environment, you will want to be able to monitor PowerDNS performance. Furthermore, PowerDNS can perform a configurable amount of operational logging.

On modern Linux distributions, the PowerDNS recursor logs to stdout, which is consumed by `systemd-journald`. This means that looking into the logs that are produced, journalctl can be used:

```
# journalctl -u pdns-recursor -n 100
```

Additionally, the Recursor *can* log to syslog on these systems. Logging to syslog is disabled in the unit file to prevent double logging. To enable this, create an drop in unit file at `/etc/systemd/systemd/pdns-recursor.service.d/use-syslog.conf`:

```
[Service]
ExecStart=
ExecStart=/usr/sbin/pdns_recursor --daemon=no --write-pid=no --enable-syslog
```

3.1.1 Logging to syslog

This chapter assumes familiarity with syslog, the unix logging device. PowerDNS logs messages with different levels. The more urgent the message, the lower the 'priority'.

By default, PowerDNS will only log messages with an urgency of 3 or lower, but this can be changed using the *loglevel* setting in the configuration file. Setting it to 0 will eliminate all logging, 9 will log everything.

By default, logging is performed under the 'DAEMON' facility which is shared with lots of other programs. If you regard nameserving as important, you may want to have it under a dedicated facility so PowerDNS can log to its own files, and not clutter generic files.

For this purpose, syslog knows about 'local' facilities, numbered from LOCAL0 to LOCAL7. To move PowerDNS logging to LOCAL0, add *logging-facility=0* to your configuration.

Furthermore, you may want to have separate files for the differing priorities - preventing lower priority messages from obscuring important ones. A sample `syslog.conf` might be:

```
local0.info                     -/var/log/pdns.info
local0.warn                     -/var/log/pdns.warn
local0.err                      /var/log/pdns.err
```

Where local0.err would store the really important messages. For performance and disk space reasons, it is advised to audit your `syslog.conf` for statements also logging PowerDNS activities. Many `syslog.conf`s have a `*.*` statement to `/var/log/syslog`, which you may want to remove.

For performance reasons, be especially certain that no large amounts of synchronous logging take place. Under Linux, this is indicated by file names not starting with a - - indicating a synchronous log, which hurts performance.

Be aware that syslog by default logs messages at the configured priority and higher! To log only info messages, use `local0.=info`

3.2 Cache Management

Sometimes a domain fails to resolve due to an error on the domain owner's end, or records for your own domain have updated and you want your users to immediatly see them without waiting for the TTL to expire. The *rec_control* tool can be used to selectively wipe the cache.

To wipe all records for the exact name 'www.example.com':

```
rec_control wipe-cache www.example.com
```

Whole subtrees can we wiped as well, to wipe all cache entries for 'example.com' and everything below it, suffix the name with a '$':

```
rec_control wipe-cache example.com$
```

Note: When wiping cache entries, matching entries in *all* caches (packet cache, recursor cache, negative cache) are removed.

When debugging resolving issues, it can be advantagious to have a dump of all the cache entries. *rec_control* can write the caches of all threads to a file:

```
rec_control dump-cache /tmp/cache
```

3.3 Tracing Queries

To investigate failures with resolving certain domain names, the PowerDNS Recursor features a "tracing" infrastructure. This infrastructure will log every step the Recursor takes to resolve a name and will log all DNSSEC related information as well.

To enable tracing for all queries, enable the *trace* setting.

Warning: Enabling tracing for all queries on a system with a high query rate can severely impact performance.

Tracing can also be enabled at runtime, without restarting the Recursor, for specific domains. These specific domains can be specified as a regular expression. This can be done using *rec_control trace-regex*:

```
rec_control trace-regex '.*\.example.com\.$'
```

Will enable tracing for any query *in* the example.com domain (but not example.com itself).

CHAPTER

FOUR

PERFORMANCE GUIDE

To get the best out of the PowerDNS recursor, which is important if you are doing thousands of queries per second, please consider the following.

A busy server may need hundreds of file descriptors on startup, and deals with spikes better if it has that many available later on. Linux by default restricts processes to 1024 file descriptors, which should suffice most of the time, but Solaris has a default limit of 256. This can be raised using the `ulimit` command or via the `LimitNOFILE` unit directive when `systemd` is used. FreeBSD has a default limit that is high enough for even very heavy duty use.

Limit the size of the caches to a sensible value. Cache hit rate does not improve meaningfully beyond 4 million *max-cache-entries* per thread, reducing the memory footprint reduces CPU cache misses. See below for more information about the various caches.

When deploying (large scale) IPv6, please be aware some Linux distributions leave IPv6 routing cache tables at very small default values. Please check and if necessary raise `sysctl net.ipv6.route.max_size`.

Set *threads* to your number of CPU cores (but values above 8 rarely improve performance).

4.1 Threading and distribution of queries

When running with several threads, you can either ask PowerDNS to start a special thread to dispatch the incoming queries to the workers by setting *pdns-distributes-queries* to true, or let the worker threads handle the incoming queries themselves. The dispatch thread enabled by *pdns-distributes-queries* tries to send the same queries to the same thread to maximize the cache-hit ratio, but it might become a bottleneck if the incoming queries rate is too high to be handled by a single thread.

If *pdns-distributes-queries* is set to false and either `SO_REUSEPORT` support is not available or the *reuseport* directive is set to false, all worker threads share the same listening sockets.

This prevents a single thread from having to handle every incoming queries, but can lead to thundering herd issues where all threads are awoken at once when a query arrives.

If `SO_REUSEPORT` support is available and *reuseport* is set to true, separate listening sockets are opened for each worker thread and the query distributions is handled by the kernel, avoiding any thundering herd issue as well as preventing the distributor thread from becoming the bottleneck.

New in version 4.1.0: The *cpu-map* parameter can be used to pin worker threads to specific CPUs, in order to keep caches as warm as possible and optimize memory access on NUMA systems.

4.2 Performance tips

For best PowerDNS Recursor performance, use a recent version of your operating system, since this generally offers the best event multiplexer implementation available (`kqueue`, `epoll`, `ports` or `/dev/poll`).

On AMD/Intel hardware, wherever possible, run a 64-bit binary. This delivers a nearly twofold performance increase. On UltraSPARC, there is no need to run with 64 bits.

Consider performing a 'profiled build' by building with `gprof` support enabled, running the recursor a bit then feed that info into the next build. This is good for a 20% performance boost in some cases.

When running with >3000 queries per second, and running Linux versions prior to 2.6.17 on some motherboards, your computer may spend an inordinate amount of time working around an ACPI bug for each call to gettimeofday. This is solved by rebooting with `clock=tsc` or upgrading to a 2.6.17 kernel. This is relevant if dmesg shows `Using pmtmr for high-res timesource`.

4.3 Connection tracking and firewalls

A Recursor under high load puts a severe stress on any stateful (connection tracking) firewall, so much so that the firewall may fail.

Specifically, many Linux distributions run with a connection tracking firewall configured. For high load operation (thousands of queries/second), It is advised to either turn off iptables completely, or use the `NOTRACK` feature to make sure DNS traffic bypasses the connection tracking.

Sample Linux command lines would be:

```
## IPv4
iptables -t raw -I OUTPUT -p udp --dport 53 -j CT --notrack
iptables -t raw -I OUTPUT -p udp --sport 53 -j CT --notrack
iptables -t raw -I PREROUTING -p udp --dport 53 -j CT --notrack
iptables -t raw -I PREROUTING -p udp --sport 53 -j CT --notrack
iptables -I INPUT -p udp --dport 53 -j ACCEPT
iptables -I INPUT -p udp --sport 53 -j ACCEPT
iptables -I OUTPUT -p udp --dport 53 -j ACCEPT
iptables -I OUTPUT -p udp --sport 53 -j ACCEPT

## IPv6
ip6tables -t raw -I OUTPUT -p udp --dport 53 -j CT --notrack
ip6tables -t raw -I OUTPUT -p udp --sport 53 -j CT --notrack
ip6tables -t raw -I PREROUTING -p udp --sport 53 -j CT --notrack
ip6tables -t raw -I PREROUTING -p udp --dport 53 -j CT --notrack
ip6tables -I INPUT -p udp --dport 53 -j ACCEPT
ip6tables -I INPUT -p udp --sport 53 -j ACCEPT
ip6tables -I OUTPUT -p udp --dport 53 -j ACCEPT
ip6tables -I OUTPUT -p udp --sport 53 -j ACCEPT
```

When using FirewallD (Centos 7+ / RedHat 7+ / Fedora 21+), connection tracking can be disabled via direct rules. The settings can be made permanent by using the `--permanent` flag:

```
## IPv4
firewall-cmd --direct --add-rule ipv4 raw OUTPUT 0 -p udp --dport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv4 raw OUTPUT 0 -p udp --sport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv4 raw PREROUTING 0 -p udp --dport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv4 raw PREROUTING 0 -p udp --sport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv4 filter INPUT 0 -p udp --dport 53 -j ACCEPT
firewall-cmd --direct --add-rule ipv4 filter INPUT 0 -p udp --sport 53 -j ACCEPT
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 0 -p udp --dport 53 -j ACCEPT
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 0 -p udp --sport 53 -j ACCEPT

## IPv6
firewall-cmd --direct --add-rule ipv6 raw OUTPUT 0 -p udp --dport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv6 raw OUTPUT 0 -p udp --sport 53 -j CT --
↪notrack
```

(continues on next page)

(continued from previous page)

```
firewall-cmd --direct --add-rule ipv6 raw PREROUTING 0 -p udp --dport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv6 raw PREROUTING 0 -p udp --sport 53 -j CT --
↪notrack
firewall-cmd --direct --add-rule ipv6 filter INPUT 0 -p udp --dport 53 -j ACCEPT
firewall-cmd --direct --add-rule ipv6 filter INPUT 0 -p udp --sport 53 -j ACCEPT
firewall-cmd --direct --add-rule ipv6 filter OUTPUT 0 -p udp --dport 53 -j ACCEPT
firewall-cmd --direct --add-rule ipv6 filter OUTPUT 0 -p udp --sport 53 -j ACCEPT
```

Following the instructions above, you should be able to attain very high query rates.

4.4 Recursor Caches

The PowerDNS Recursor contains a number of caches, or information stores:

4.4.1 Nameserver speeds cache

The "NSSpeeds" cache contains the average latency to all remote authoritative servers.

4.4.2 Negative cache

The "Negcache" contains all domains known not to exist, or record types not to exist for a domain.

4.4.3 Recursor Cache

The Recursor Cache contains all DNS knowledge gathered over time. This is also knows as a "record cache".

4.4.4 Packet Cache

The Packet Cache contains previous answers sent to clients. If a question comes in that matches a previous answer, this is sent back directly.

The Packet Cache is consulted first, immediately after receiving a packet. This means that a high hitrate for the Packet Cache automatically lowers the cache hitrate of subsequent caches.

4.5 Measuring performance

The PowerDNS Recursor exposes many *metrics* that can be graphed and monitored.

CHAPTER

FIVE

METRICS AND STATISTICS

The PowerDNS Recursor collects many statistics about itself.

5.1 Regular Statistics Log

Every half hour or so (configurable with *statistics-interval*, the recursor outputs a line with statistics. To force the output of statistics, send the process a SIGUSR1. A line of statistics looks like this:

```
Feb 10 14:16:03 stats: 125784 questions, 13971 cache entries, 309 negative entries,
↪ 84% cache hits, outpacket/query ratio 37%, 12% throttled
```

This means that there are 13791 different names cached, which each may have multiple records attached to them. There are 309 items in the negative cache, items of which it is known that don't exist and won't do so for the near future. 84% of incoming questions could be answered without any additional queries going out to the net.

The outpacket/query ratio means that on average, 0.37 packets were needed to answer a question. Initially this ratio may be well over 100% as additional queries may be needed to actually recurse the DNS and figure out the addresses of nameservers.

Finally, 12% of queries were not performed because identical queries had gone out previously and failed, saving load on servers worldwide.

5.2 Sending metrics to Graphite/Metronome over Carbon

For carbon/graphite/metronome, we use the following namespace. Everything starts with 'pdns.', which is then followed by the local hostname. Thirdly, we add 'recursor' to signify the daemon generating the metrics. This is then rounded off with the actual name of the metric. As an example: 'pdns.ns1.recursor.questions'.

Care has been taken to make the sending of statistics as unobtrusive as possible, the daemons will not be hindered by an unreachable carbon server, timeouts or connection refused situations.

To benefit from our carbon/graphite support, either install Graphite, or use our own lightweight statistics daemon, Metronome, currently available on GitHub.

To enable sending metrics, set *carbon-server*, possibly *carbon-interval* and possibly *carbon-ourname* in the configuration.

Warning: If your hostname includes dots, they will be replaced by underscores so as not to confuse the namespace.

If you include dots in *carbon-ourname*, they will **not** be replaced by underscores. As PowerDNS assumes you know what you are doing if you override your hostname.

5.3 Sending metrics over SNMP

New in version 4.1.0.

The recursor can export statistics over SNMP and send traps from *Lua*, provided support is compiled into the Recursor and *snmp-agent* set.

5.4 Getting Metrics from the Recursor

Should Carbon not be the preferred way of receiving metric, several other techniques can be employed to retrieve metrics.

5.4.1 Using the Webserver

The *API* exposes a statistics endpoint at `GET /api/v1/servers/:server_id/statistics`. This endpoint exports all statistics in a single JSON document.

5.4.2 Using `rec_control`

Metrics can also be gathered on the system itself by invoking *rec_control*:

```
rec_control get-all
```

Single statistics can also be retrieved with the `get` command, e.g.:

```
rec_control get all-outqueries
```

External programs can use this technique to scrape metrics.

5.5 Gathered Information

These statistics are gathered.

It should be noted that answers0-1 + answers1-10 + answers10-100 + answers100-1000 + answers-slow + packetcache-hits + over-capacity-drops + policy-drops = questions.

Also note that unauthorized-tcp and unauthorized-udp packets do not end up in the 'questions' count.

5.5.1 all-outqueries

counts the number of outgoing UDP queries since starting

5.5.2 answers-slow

counts the number of queries answered after 1 second

5.5.3 answers0-1

counts the number of queries answered within 1 millisecond

5.5.4 answers1-10

counts the number of queries answered within 10 milliseconds

5.5.5 answers10-100

counts the number of queries answered within 100 milliseconds

5.5.6 answers100-1000

counts the number of queries answered within 1 second

5.5.7 auth4-answers-slow

counts the number of queries answered by auth4s after 1 second (4.0)

5.5.8 auth4-answers0-1

counts the number of queries answered by auth4s within 1 millisecond (4.0)

5.5.9 auth4-answers1-10

counts the number of queries answered by auth4s within 10 milliseconds (4.0)

5.5.10 auth4-answers10-100

counts the number of queries answered by auth4s within 100 milliseconds (4.0)

5.5.11 auth4-answers100-1000

counts the number of queries answered by auth4s within 1 second (4.0)

5.5.12 auth6-answers-slow

counts the number of queries answered by auth6s after 1 second (4.0)

5.5.13 auth6-answers0-1

counts the number of queries answered by auth6s within 1 millisecond (4.0)

5.5.14 auth6-answers1-10

counts the number of queries answered by auth6s within 10 milliseconds (4.0)

5.5.15 auth6-answers10-100

counts the number of queries answered by auth6s within 100 milliseconds (4.0)

5.5.16 auth6-answers100-1000

counts the number of queries answered by auth6s within 1 second (4.0)

5.5.17 auth-zone-queries

counts the number of queries to locally hosted authoritative zones (*auth-zones*) since starting

5.5.18 cache-bytes

size of the cache in bytes

5.5.19 cache-entries

shows the number of entries in the cache

5.5.20 cache-hits

counts the number of cache hits since starting, this does **not** include hits that got answered from the packet-cache

5.5.21 cache-misses

counts the number of cache misses since starting

5.5.22 case-mismatches

counts the number of mismatches in character case since starting

5.5.23 chain-resends

number of queries chained to existing outstanding query

5.5.24 client-parse-errors

counts number of client packets that could not be parsed

5.5.25 concurrent-queries

shows the number of MThreads currently running

5.5.26 dlg-only-drops

number of records dropped because of *delegation-only* setting

5.5.27 dnssec-queries

number of queries received with the DO bit set

5.5.28 dnssec-result-bogus

number of DNSSEC validations that had the Bogus state

5.5.29 dnssec-result-indeterminate

number of DNSSEC validations that had the Indeterminate state

5.5.30 dnssec-result-insecure

number of DNSSEC validations that had the Insecure state

5.5.31 dnssec-result-nta

number of DNSSEC validations that had the NTA (negative trust anchor) state

5.5.32 dnssec-result-secure

number of DNSSEC validations that had the Secure state

5.5.33 dnssec-validations

number of DNSSEC validations performed

5.5.34 dont-outqueries

number of outgoing queries dropped because of *dont-query* setting (since 3.3)

5.5.35 ecs-queries

number of outgoing queries adorned with an EDNS Client Subnet option (since 4.1)

5.5.36 ecs-responses

number of responses received from authoritative servers with an EDNS Client Subnet option we used (since 4.1)

5.5.37 edns-ping-matches

number of servers that sent a valid EDNS PING response

5.5.38 edns-ping-mismatches

number of servers that sent an invalid EDNS PING response

5.5.39 failed-host-entries

number of servers that failed to resolve

5.5.40 ignored-packets

counts the number of non-query packets received on server sockets that should only get query packets

5.5.41 ipv6-outqueries

number of outgoing queries over IPv6

5.5.42 ipv6-questions

counts all end-user initiated queries with the RD bit set, received over IPv6 UDP

5.5.43 malloc-bytes

returns the number of bytes allocated by the process (broken, always returns 0)

5.5.44 max-cache-entries

currently configured maximum number of cache entries

5.5.45 max-packetcache-entries

currently configured maximum number of packet cache entries

5.5.46 max-mthread-stack

maximum amount of thread stack ever used

5.5.47 negcache-entries

shows the number of entries in the negative answer cache

5.5.48 no-packet-error

number of erroneous received packets

5.5.49 noedns-outqueries

number of queries sent out without EDNS

5.5.50 noerror-answers

counts the number of times it answered NOERROR since starting

5.5.51 noping-outqueries

number of queries sent out without ENDS PING

5.5.52 nsset-invalidations

number of times an nsset was dropped because it no longer worked

5.5.53 nsspeeds-entries

shows the number of entries in the NS speeds map

5.5.54 nxdomain-answers

counts the number of times it answered NXDOMAIN since starting

5.5.55 outgoing-timeouts

counts the number of timeouts on outgoing UDP queries since starting

5.5.56 outgoing4-timeouts

counts the number of timeouts on outgoing UDP IPv4 queries since starting (since 4.0)

5.5.57 outgoing6-timeouts

counts the number of timeouts on outgoing UDP IPv6 queries since starting (since 4.0)

5.5.58 over-capacity-drops

questions dropped because over maximum concurrent query limit (since 3.2)

5.5.59 packetcache-bytes

size of the packet cache in bytes (since 3.3.1)

5.5.60 packetcache-entries

size of packet cache (since 3.2)

5.5.61 packetcache-hits

packet cache hits (since 3.2)

5.5.62 packetcache-misses

packet cache misses (since 3.2)

5.5.63 policy-drops

packets dropped because of (Lua) policy decision

5.5.64 policy-result-noaction

packets that were not actioned upon by the RPZ/filter engine

5.5.65 policy-result-drop

packets that were dropped by the RPZ/filter engine

5.5.66 policy-result-nxdomain

packets that were replied to with NXDOMAIN by the RPZ/filter engine

5.5.67 policy-result-nodata

packets that were replied to with no data by the RPZ/filter engine

5.5.68 policy-result-truncate

packets that were forced to TCP by the RPZ/filter engine

5.5.69 policy-result-custom

packets that were sent a custom answer by the RPZ/filter engine

5.5.70 qa-latency

shows the current latency average, in microseconds, exponentially weighted over past 'latency-statistic-size' packets

5.5.71 query-pipe-full-drops

New in version 4.2.

questions dropped because the query distribution pipe was full

5.5.72 questions

counts all end-user initiated queries with the RD bit set

5.5.73 resource-limits

counts number of queries that could not be performed because of resource limits

5.5.74 security-status

security status based on *Security Polling*

5.5.75 server-parse-errors

counts number of server replied packets that could not be parsed

5.5.76 servfail-answers

counts the number of times it answered SERVFAIL since starting

5.5.77 spoof-prevents

number of times PowerDNS considered itself spoofed, and dropped the data

5.5.78 sys-msec

number of CPU milliseconds spent in 'system' mode

5.5.79 tcp-client-overflow

number of times an IP address was denied TCP access because it already had too many connections

5.5.80 tcp-clients

counts the number of currently active TCP/IP clients

5.5.81 tcp-outqueries

counts the number of outgoing TCP queries since starting

5.5.82 tcp-questions

counts all incoming TCP queries (since starting)

5.5.83 throttle-entries

shows the number of entries in the throttle map

5.5.84 throttled-out

counts the number of throttled outgoing UDP queries since starting

5.5.85 throttled-outqueries

idem to throttled-out

5.5.86 too-old-drops

questions dropped that were too old

5.5.87 truncated-drops

New in version 4.2.

questions dropped because they were larger than 512 bytes

5.5.88 unauthorized-tcp

number of TCP questions denied because of allow-from restrictions

5.5.89 unauthorized-udp

number of UDP questions denied because of allow-from restrictions

5.5.90 unexpected-packets

number of answers from remote servers that were unexpected (might point to spoofing)

5.5.91 unreachables

number of times nameservers were unreachable since starting

5.5.92 uptime

number of seconds process has been running (since 3.1.5)

5.5.93 user-msec

number of CPU milliseconds spent in 'user' mode

5.5.94 x-our-latency

New in version 4.1: Not yet proven to be reliable

PowerDNS measures per query how much time has been spent waiting on authoritative servers. In addition, the Recursor measures the total amount of time needed to answer a question. The difference between these two durations is a measure of how much time was spent within PowerDNS. This metric is the average of that difference, in microseconds.

5.5.95 x-ourtime0-1

New in version 4.1: Not yet proven to be reliable

Counts responses where between 0 and 1 milliseconds was spent within the Recursor. See *x-our-latency* for further details.

5.5.96 x-ourtime1-2

New in version 4.1: Not yet proven to be reliable

Counts responses where between 1 and 2 milliseconds was spent within the Recursor. See *x-our-latency* for further details.

5.5.97 x-ourtime2-4

New in version 4.1: Not yet proven to be reliable

Counts responses where between 2 and 4 milliseconds was spent within the Recursor. Since 4.1. See *x-our-latency* for further details.

5.5.98 x-ourtime4-8

New in version 4.1: Not yet proven to be reliable

Counts responses where between 4 and 8 milliseconds was spent within the Recursor. See *x-our-latency* for further details.

5.5.99 x-ourtime8-16

New in version 4.1: Not yet proven to be reliable

Counts responses where between 8 and 16 milliseconds was spent within the Recursor. See *x-our-latency* for further details.

5.5.100 x-ourtime16-32

New in version 4.1: Not yet proven to be reliable

Counts responses where between 16 and 32 milliseconds was spent within the Recursor. See *x-our-latency* for further details.

5.5.101 x-ourtime-slow

New in version 4.1: Not yet proven to be reliable

Counts responses where more than 32 milliseconds was spent within the Recursor. See *x-our-latency* for further details.

CHAPTER

SIX

DNSSEC IN THE POWERDNS RECURSOR

As of 4.0.0, the PowerDNS Recursor has support for DNSSEC processing and experimental support for DNSSEC validation.

Warning: The DNSSEC implementation in the PowerDNS Recursor 4.0.x is known to have deficiencies due to its original design. When doing DNSSEC validation, ensure you are running 4.1.0 or later which has a fully reworked (and correct) DNSSEC implementation.

6.1 DNSSEC settings

The PowerDNS Recursor has 5 different levels of DNSSEC processing, which can be set with the *dnssec* setting in the `recursor.conf`. In order from least to most processing, these are:

6.1.1 `off`

In this mode, **no** DNSSEC processing takes place. The PowerDNS Recursor will not set the DNSSEC OK (DO) bit in the outgoing queries and will ignore the DO and AD bits in queries. In this mode, the behaviour is equal to the PowerDNS Recursor 3.X.

6.1.2 `process-no-validate`

The default mode.

In this mode the Recursor acts as a "security aware, non-validating" nameserver, meaning it will set the DO-bit on outgoing queries and will provide DNSSEC related RRsets (NSEC, RRSIG) to clients that ask for them (by means of a DO-bit in the query), except for zones provided through the `auth-zones` setting. It will not do any validation in this mode, not even when requested by the client.

6.1.3 `process`

When *dnssec* is set to `process` the behaviour is similar to *process-no-validate*. However, the recursor will try to validate the data if at least one of the DO or AD bits is set in the query; in that case, it will set the AD-bit in the response when the data is validated successfully, or send SERVFAIL when the validation comes up bogus.

6.1.4 `log-fail`

In this mode, the recursor will attempt to validate all data it retrieves from authoritative servers, regardless of the client's DNSSEC desires, and will log the validation result. This mode can be used to determine the extra load and amount of possibly bogus answers before turning on full-blown validation. Responses to client queries are the same as with *process*.

6.1.5 `validate`

The highest mode of DNSSEC processing. In this mode, all queries will be be validated and will be answered with a SERVFAIL in case of bogus data, regardless of the client's request.

6.1.6 What, when?

The descriptions above are a bit terse, here's a table describing different scenarios with regards to the `dnssec` mode.

	`off`	`process-no-validate`	`process`	`log-fail`	`validate`
Perform validation	No	No	Only on +AD or +DO from client	Always (logs result)	Always
SERVFAIL on bogus	No	No	Only on +AD or +DO from client	Only on +AD or +DO from client	Always
AD in response on authenticate d data	Never	Never	Only on +AD or +DO from client	Only on +AD or +DO from client	Only on +AD or +DO from client
RRSIGs/NSECs in answer on +DO from client	No	Yes	Yes	Yes	Yes

Note: the `dig` tool sets the AD-bit in the query. This might lead to unexpected query results when testing. Set `+noad` on the `dig` commandline when this is the case.

Note: the CD-bit is honored correctly for `process` and `validate`. For `log-fail`, failures will be logged too.

6.2 Trust Anchor Management

In the PowerDNS Recursor, both positive and negative trust anchors can be configured during startup (from a persistent configuration file) and at runtime (which is volatile). However, all trust anchors are configurable.

6.2.1 Trust Anchors

The PowerDNS Recursor ships with the DNSSEC Root key built-in.

Note: it has no support for **RFC 5011** key rollover and does not persist a changed root trust anchor to disk.

Configuring DNSSEC key material must be done in the *lua-config-file*, using *`addDS()`*. This function takes 2 arguments: the node in the DNS-tree and the data of the corresponding DS record.

To e.g. add a trust anchor for the root and powerdns.com, use the following config in the Lua file:

```
addDS('.', "63149 13 1 a59da3f5c1b97fcd5fa2b3b2b0ac91d38a60d33a") -- This is not␣
↪an ICANN root
addDS('powerdns.com', "44030 8 2␣
↪D4C3D5552B8679FAEEBC317E5F048B614B2E5F607DC57F1553182D49 AB2179F7")
```

Now (re)start the recursor to load these trust anchors.

Runtime Configuration of Trust Anchors

To change or add trust anchors at runtime, use the *rec_control* tool. These runtime settings are not saved to disk. To make them permanent, they should be added to the *lua-config-file* as described above.

Adding a trust anchor is done with the `add-ta` command:

```
$ rec_control add-ta domain.example 63149 13 1␣
↪a59da3f5c1b97fcd5fa2b3b2b0ac91d38a60d33a
Added Trust Anchor for domain.example. with data 63149 13 1␣
↪a59da3f5c1b97fcd5fa2b3b2b0ac91d38a60d33a
```

To view the currently configured trust anchors, run `get-tas`:

```
$ rec_control get-tas
Configured Trust Anchors:
.       63149 13 1 a59da3f5c1b97fcd5fa2b3b2b0ac91d38a60d33a
net.    2574 13 1 a5c5acb889a7ba9b5aa5bef2b0ac9fe1565ddaab
```

To remove a trust anchor, run `clear-ta`:

```
$ rec_control clear-ta domain.example
Removed Trust Anchor for subdomain.example
```

Note: The root trust anchor cannot be removed in this manner.

6.2.2 Negative Trust Anchors

Negative trust anchors (defined in **RFC 7646**) can be used to temporarily disable DNSSEC validation for a part of the DNS-tree. This can be done when e.g. a TLD or high-traffic zone goes bogus. Note that it is good practice to verify that this is indeed the case and not because of malicious actions.

To configure a negative trust anchor, use the `addNTA()` function in the *lua-config-file* and restart the recursor. This function requires the name of the zone and an optional reason:

```
addNTA('example.', "Someone messed up the delegation")
addNTA('powerdns.com') -- No reason given
```

Runtime Configuration of Negative Trust Anchors

The *rec_control* command can be used to manage the negative trust anchors of a running instance. These runtime settings are lost when restarting the recursor, more permanent NTAs should be added to the *lua-config-file* with `addNTA()`.

Adding a negative trust anchor is done with the `add-nta` command (that optionally accepts a reason):

```
$ rec_control add-nta domain.example botched keyroll
Added Negative Trust Anchor for domain.example. with reason 'botched keyroll'
```

To view the currently configured negative trust anchors, run `get-ntas`:

```
$ rec_control get-ntas
Configured Negative Trust Anchors:
subdomain.example.      Operator failed key-roll
otherdomain.example.    DS in parent, no DNSKEY in zone
```

To remove negative trust anchor(s), run `clear-nta`:

```
$ rec_control clear-nta subdomain.example
Removed Negative Trust Anchors for subdomain.example
```

`clear-nta` accepts multiple domain-names and accepts '*' (beware the shell quoting) to remove all negative trust anchors.

CHAPTER

SEVEN

LUA CONFIGURATION

Since version 4.0.0, the PowerDNS Recursor supports additional configuration options that have to be loaded through *lua-config-file*.

7.1 Managing DNSSEC Trust Anchors in the Lua Configuration

The DNSSEC Trust Anchors and Negative Trust Anchors must be stored in the Lua Configuration file. See the *DNSSEC in the PowerDNS Recursor* for all information about DNSSEC in the PowerDNS Recursor. This page only documents the Lua functions for DNSSEC configuration

addDS(*name*, *dscontent*)
Adds a DS record (Trust Anchor) to the configuration

Parameters

- **name** (*str*) – The name in the DNS tree from where this Trust Anchor should be used
- **dsrecord** (*str*) – The DS Record content associated with `name`

addNTA(*name*[, *reason*])
Adds a Negative Trust Anchor for `name` to the configuration. Please read *Negative Trust Anchors* for operational information on NTAs.

Parameters

- **name** (*str*) – The name in the DNS tree from where this NTA should be used
- **reason** (*str*) – An optional comment to add to this NTA

7.2 Using Sortlist

Sortlist is a complicated feature which allows for the ordering of A and AAAA records in answers to be modified, optionally dependently on who is asking. Since clients frequently connect to the 'first' IP address they see, this can effectively allow you to make sure that user from, say 10.0.0.0/8 also preferably connect to servers in 10.0.0.0/8.

The syntax consists of a netmask for which this ordering instruction applies, followed by a set of netmask (groups) which describe the desired ordering. So an ordering instruction of "1.0.0.0/8", "2.0.0.0/8" will put anything within 1/8 first, and anything in 2/8 second. Other IP addresses would follow behind the addresses sorted earlier.

If netmasks are grouped, this means these get equal ordering.

7.2.1 addSortList

`addSortList()` is used in the *lua-config-file* and is intended to exactly mirror the semantics of the BIND sortlist option, but the syntax is slightly different.

As an example, the following BIND sortlist:

```
{ 17.50.0.0/16; {17.238.240.0/24; 17.138.149.200;
{17.218.242.254; 17.218.252.254;}; 17.38.42.80;
17.208.240.100; }; };
```

Gets transformed into:

```
addSortList("17.50.0.0/16", {"17.238.240.0/24", "17.138.149.200",
{"17.218.242.254", "17.218.252.254"}, "17.38.42.80",
"17.208.240.100" })
```

In other words: each IP address is put within quotes, and are separated by commas instead of semicolons. For the rest everything is identical.

7.3 Logging DNS messages with Protocol Buffers

The PowerDNS Recursor has the ability to emit a stream of protocol buffers messages over TCP, containing information about queries, answers and policy decisions.

Messages contain the IP address of the client initiating the query, the one on which the message was received, whether it was received over UDP or TCP, a timestamp and the qname, qtype and qclass of the question. In addition, messages related to responses contain the name, type, class and rdata of A, AAAA and CNAME records present in the response, as well as the response code.

Finally, if a RPZ or custom Lua policy has been applied, response messages also contain the applied policy name and some tags. This is particularly useful to detect and act on infected hosts.

7.3.1 Configuring Protocol Buffer logs

Protobuf export to a server is enabled using the `protobufServer()` directive:

protobufServer(*server*[[[[[[[[, *timeout=2*], *maxQueuedEntries=100*], *reconnectWaitTime=1*], *maskV4=32*], *maskV6=128*], *asyncConnect=false*], *taggedOnly=false*])

Parameters

- **server** (*string*) – The IP and port to connect to
- **timeout** (*int*) – Time in seconds to wait when sending a message
- **maxQueuedEntries** (*int*) – How many entries will be kept in memory if the server becomes unreachable
- **reconnectWaitTime** (*int*) – How long to wait, in seconds, between two reconnection attempts
- **maskV4** (*int*) – network mask to apply to the client IPv4 addresses, for anonymization purposes. The default of 32 means no anonymization.
- **maskV6** (*int*) – Same as maskV4, but for IPv6. Defaults to 128.
- **taggedOnly** (*bool*) – Only entries with a policy or a policy tag set will be sent.
- **asyncConnect** (*bool*) – When set to false (default) the first connection to the server during startup will block up to `timeout` seconds, otherwise the connection is done in a separate thread, after the first message has been queued..

7.3.2 Logging outgoing queries and responses

While *protobufServer()* only exports the queries sent to the recursor from clients, with the corresponding responses, `outgoingProtobufServer()` can be used to export outgoing queries sent by the recursor to authoritative servers, along with the corresponding responses.

outgoingProtobufServer(*server*[[[[, *timeout=2*], *maxQueuedEntries=100*], *reconnectWaitTime=1*], *asyncConnect=false*])

Parameters

- **server** (*string*) – The IP and port to connect to
- **timeout** (*int*) – Time in seconds to wait when sending a message
- **maxQueuedEntries** (*int*) – How many entries will be kept in memory if the server becomes unreachable
- **reconnectWaitTime** (*int*) – How long to wait, in seconds, between two reconnection attempts
- **asyncConnect** (*bool*) – When set to false (default) the first connection to the server during startup will block up to `timeout` seconds, otherwise the connection is done in a separate thread, after the first message has been queued..

7.3.3 Protobol Buffers Definition

The protocol buffers message types can be found in the dnsmessage.proto file and is included here:

```
/*
 * This file is part of PowerDNS or dnsdist.
 * Copyright -- PowerDNS.COM B.V. and its contributors
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of version 2 of the GNU General Public License as
 * published by the Free Software Foundation.
 *
 * In addition, for the avoidance of any doubt, permission is granted to
 * link this program with OpenSSL and to (re)distribute the binaries
 * produced as the result of such linking.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
 */
syntax = "proto2";

message PBDNSMessage {
  enum Type {
    DNSQueryType = 1;
    DNSResponseType = 2;
    DNSOutgoingQueryType = 3;
    DNSIncomingResponseType = 4;
  }
  enum SocketFamily {
    INET = 1;                                   // IPv4 (RFC 791)
    INET6 = 2;                                  // IPv6 (RFC 2460)
  }
  enum SocketProtocol {
    UDP = 1;                                    // User Datagram Protocol (RFC 768)
    TCP = 2;                                    // Transmission Control Protocol␣
↪(RFC 793)
  }
  enum PolicyType {
```

(continues on next page)

(continued from previous page)

```
    UNKNOWN = 1;                                  // No policy applied, or unknown␣
↪type
    QNAME = 2;                                    // Policy matched on the QName
    CLIENTIP = 3;                                 // Policy matched on the client IP
    RESPONSEIP = 4;                               // Policy matched on one of the␣
↪IPs contained in the answer
    NSDNAME = 5;                                  // Policy matched on the name of␣
↪one nameserver involved
    NSIP = 6;                                     // Policy matched on the IP of one␣
↪nameserver involved
  }
  required Type type = 1;
  optional bytes messageId = 2;                   // UUID, shared by the query and␣
↪the response
  optional bytes serverIdentity = 3;              // UUID of the server emitting the␣
↪protobuf message
  optional SocketFamily socketFamily = 4;
  optional SocketProtocol socketProtocol = 5;
  optional bytes from = 6;                        // DNS requestor (client)
  optional bytes to = 7;                          // DNS responder (server)
  optional uint64 inBytes = 8;                    // Size of the query or response␣
↪on the wire
  optional uint32 timeSec = 9;                    // Time of message reception␣
↪(seconds since epoch)
  optional uint32 timeUsec = 10;                  // Time of message reception␣
↪(additional micro-seconds)
  optional uint32 id = 11;                        // ID of the query/response as␣
↪found in the DNS header

  message DNSQuestion {
    optional string qName = 1;
    optional uint32 qType = 2;
    optional uint32 qClass = 3;
  }
  optional DNSQuestion question = 12;

  message DNSResponse {
    message DNSRR {
      optional string name = 1;
      optional uint32 type = 2;
      optional uint32 class = 3;
      optional uint32 ttl = 4;
      optional bytes rdata = 5;
    }
    optional uint32 rcode = 1;
    repeated DNSRR rrs = 2;
    optional string appliedPolicy = 3;            // Filtering policy (RPZ or Lua)␣
↪applied
    repeated string tags = 4;                     // Additional tags
    optional uint32 queryTimeSec = 5;             // Time of the corresponding query␣
↪reception (seconds since epoch)
    optional uint32 queryTimeUsec = 6;            // Time of the corresponding query␣
↪reception (additional micro-seconds)
    optional PolicyType appliedPolicyType = 7;    // Type of the filtering policy␣
↪(RPZ or Lua) applied
  }

  optional DNSResponse response = 13;
  optional bytes originalRequestorSubnet = 14;    // EDNS Client Subnet value
  optional string requestorId = 15;               // Username of the requestor
  optional bytes initialRequestId = 16;           // UUID of the incoming query that␣
↪initiated this outgoing query or incoming response
```

(continues on next page)

(continued from previous page)

```
  optional bytes deviceId = 17;                        // Device ID of the requestor␣
↪(could be mac address IP address or e.g. IMEI)
}
```

7.4 Response Policy Zones (RPZ)

Response Policy Zone is an open standard developed by Paul Vixie (ISC and Farsight) and Vernon Schryver (Rhyolite), to modify DNS responses based on a policy loaded via a zonefile.

Frequently, Response Policy Zones get to be very large and change quickly, so it is customary to update them over IXFR. It allows the use of third-party feeds, and near real-time policy updates.

7.4.1 Configuring RPZ

An RPZ can be loaded from file or slaved from a master. To load from file, use for example:

```
rpzFile("dblfilename", {defpol=Policy.Custom, defcontent="badserver.example.com"})
```

To slave from a master and start IXFR to get updates, use for example:

```
rpzMaster("192.0.2.4", "policy.rpz", {defpol=Policy.Drop})
```

In this example, 'policy.rpz' denotes the name of the zone to query for.

rpzFile (*filename*, *settings*)
Load an RPZ from disk.

Parameters

- **filename** (*str*) – The filename to load
- **settings** (*{}*) – A table to settings, see below

rpzMaster (*address*, *name*, *settings*)
Changed in version 4.2.0:: The first parameter can be a list of addresses.

Load an RPZ from AXFR and keep retrieving with IXFR.

Parameters

- **address** (*str*) – The IP address to transfer the RPZ from. Also accepts a list of addresses since 4.2.0 in which case they will be tried one after another in the submitted order until a response is obtained
- **name** (*str*) – The name of this RPZ
- **settings** (*{}*) – A table to settings, see below

7.4.2 RPZ settings

These options can be set in the `settings` of both *rpzMaster()* and *rpzFile()*.

defpol

Default policy: *Policy.Custom*, *Policy.Drop*, *Policy.NXDOMAIN*, *Policy.NODATA*, *Policy.Truncate*, *Policy.NoAction*.

defcontent

CNAME field to return in case of defpol=Policy.Custom

defttl

the TTL of the CNAME field to be synthesized for the default policy. The default is to use the zone's TTL,

maxTTL

The maximum TTL value of the synthesized records, overriding a higher value from `defttl` or the zone. Default is unlimited.

policyName

The name logged as 'appliedPolicy' in *protobuf* messages when this policy is applied.

zoneSizeHint

An indication of the number of expected entries in the zone, speeding up the loading of huge zones by reserving space in advance.

7.4.3 Extra settings for rpzMaster

In addition to the settings above the settings for *`rpzMaster()`* may contain:

tsigname

The name of the TSIG key to authenticate to the server. When this is set, *tsigalgo* and *tsigsecret* must also be set.

tsigalgo

The name of the TSIG algorithm (like 'hmac-md5') used

tsigsecret

Base64 encoded TSIG secret

refresh

An integer describing the interval between checks for updates. By default, the RPZ zone's default is used

maxReceivedMBytes

The maximum size in megabytes of an AXFR/IXFR update, to prevent resource exhaustion. The default value of 0 means no restriction.

localAddress

The source IP address to use when transferring the RPZ. When unset, *query-local-address* and *query-local-address6* are used.

axfrTimeout

New in version 4.1.2: Before 4.1.2, the timeout was fixed on 10 seconds.

The timeout in seconds of the total initial AXFR transaction. 20 by default.

dumpFile

New in version 4.2.0.

A path to a file where the recursor will dump the latest version of the RPZ zone after each successful update. This can be used to keep track of changes in the RPZ zone, or to speed up the initial loading of the zone via the *seedFile* parameter. The format of the generated zone file is the same than the one used with `rpzFile()`, and can also be generated via:

> rec_control dump-rpz *zone-name output-file*

seedFile

New in version 4.2.0.

A path to a file containing an existing dump of the RPZ zone. The recursor will try to load the zone from this file on startup, then immediately do an IXFR to retrieve any updates. If the file does not exist or is not valid, the normal process of doing a full AXFR will be used instead. This option allows a faster startup by loading an existing zone from a file instead of retrieving it from the network, then retrieving only the needed updates via IXFR. The format of the zone file is the same than the one used with `rpzFile()`, and can for example be generated via:

> rec_control dump-rpz *zone-name output-file*

It is also possible to use the *dumpFile* parameter in order to dump the latest version of the RPZ zone after each update.

7.4.4 Policy Actions

If no settings are included, the RPZ is taken literally with no overrides applied. Several Policy Actions exist

Policy.Custom

Will return a NoError, CNAME answer with the value specified with `defcontent`, when looking up the result of this CNAME, RPZ is not taken into account.

Policy.Drop

Will simply cause the query to be dropped.

Policy.NoAction

Will continue normal processing of the query.

Policy.NODATA

Will return a NoError response with no value in the answer section.

Policy.NXDOMAIN

Will return a response with a NXDomain rcode.

Policy.Truncate

will return a NoError, no answer, truncated response over UDP. Normal processing will continue over TCP

CHAPTER

EIGHT

SCRIPTING THE RECURSOR

In the PowerDNS recursor, it is possible to modify resolving behaviour using simple scripts written in the Lua programming language.

Note: This describes the Lua scripts as supported by 4.x. They are very different than the ones from 3.x, but tend to be faster and more correct.

These scripts can be used to quickly override dangerous domains, fix things that are wrong, for load balancing or for legal or commercial purposes. The scripts can also protect you or your users from malicious traffic.

Lua is extremely fast and lightweight, easily supporting hundreds of thousands of queries per second. The Lua language is explained very well in the excellent book Programming in Lua. If you already have programming experience, Learn Lua in 15 Minutes is a great primer.

For extra performance, a Just In Time compiled version of Lua called LuaJIT is supported.

8.1 Configuring Lua scripts

In order to load scripts, the PowerDNS Recursor must have Lua support built in. The packages distributed from the PowerDNS website have this language enabled, other distributions may differ. By default, the Recursor's configure script will attempt to detect if Lua is available.

note: Only one script can be loaded at the same time. If you load a different script, the current one will be replaced (safely)!

If Lua support is available, a script can be configured either via the configuration file, or at runtime via the `rec_control` tool. Scripts can be reloaded or unloaded at runtime with no interruption in operations. If a new script contains syntax errors, the old script remains in force.

On the command line, or in the configuration file, the setting *lua-dns-script* can be used to supply a full path to the Lua script.

At runtime, `rec_control reload-lua-script` can be used to either reload the script from its current location, or, when passed a new filename, load one from a new location. A failure to parse the new script will leave the old script in working order.

Note: It is also possible to precompile scripts using `luac`, and have PowerDNS load the result. This means that switching scripts is faster, and also that you'll be informed about syntax errors at compile time.

Finally, `rec_control unload-lua-script` can be used to remove the currently installed script, and revert to unmodified behaviour.

8.2 The DNSQuestion (`dq`) object

Apart from the *`ipfilter()`*-function, all functions work on a `dq` (DNSQuestion) object. This object contains details about the current state of the question. This state can be modified from the various hooks.

The DNSQuestion object contains at least the following fields:

class DNSQuestion

An object that contains everything about the current query. This object has the following attributes:

qname
DNSName of the name this query is for.

qtype
Type this query is for as an integer, can be compared against `pdns.A`, `pdns.AAAA`.

rcode
current DNS Result Code, which can be overridden, including to several magical values The rcode can be set to pdns.DROP to drop the query. Other statuses are normal DNS return codes, like `pdns.NOERROR`, `pdns.NXDOMAIN` etc.

isTcp
Boolean whether the query have been received over TCP.

remoteaddr
ComboAddress of the requestor.

localaddr
ComboAddress where this query was received on.

variable
Boolean which, if set, indicates the recursor should not packet cache this answer. Honored even when returning false from a hook! Important when providing answers that vary over time or based on sender details.

followupFunction
String that signals the nameserver to take one an additional action:

- followCNAMERecords: When adding a CNAME to the answer, this tells the recursor to follow that CNAME. See *CNAME Chain Resolution*
- getFakeAAAARecords: Get a fake AAAA record, see *DNS64*
- getFakePTRRecords: Get a fake PTR record, see *DNS64*
- udpQueryResponse: Do a UDP query and call a handler, see *UDP Query Response*

appliedPolicy
The decision that was made by the policy engine, see *Modifying Policy Decisions*.

`appliedPolicy.`**policyName**
A string with the name of the policy. Set by *policyName* in the *rpzFile()* and *rpzMaster()* configuration items. It is advised to overwrite this when modifying the *DNSQuestion.appliedPolicy.policyKind*

`appliedPolicy.`**policyAction**
The action taken by the engine

`appliedPolicy.`**policyCustom**
The CNAME content for the `pdns.policyactions.Custom` response, a string

`appliedPolicy.`**policyKind**
The kind of policy response, there are several policy kinds:
- `pdns.policykinds.Custom` will return a NoError, CNAME answer with the value specified in *DNSQuestion.appliedPolicy.policyCustom*
- `pdns.policykinds.Drop` will simply cause the query to be dropped
- `pdns.policykinds.NoAction` will continue normal processing of the query
- `pdns.policykinds.NODATA` will return a NoError response with no value in the answer section
- `pdns.policykinds.NXDOMAIN` will return a response with a NXDomain rcode
- `pdns.policykinds.Truncate` will return a NoError, no answer, truncated response over UDP. Normal processing will continue over TCP

appliedPolicy.**policyTTL**
The TTL in seconds for the pdns.policyactions.Custom response

wantsRPZ
A boolean that indicates the use of the Policy Engine. Can be set to false in prerpz to disable RPZ for this query.

data
A Lua object reference that is persistent throughout the lifetime of the *DNSQuestion* object for a single query. It can be used to store custom data. Most scripts initialise this to an empty table early on so they can store multiple items.

requestorId
New in version 4.1.0.

A string that will be used to set the requestorId field in *protobuf* messages.

deviceId
New in version 4.1.0.

A string that will be used to set the deviceId field in *protobuf* messages.

udpAnswer
Answer to the *udpQuery* when when using the udpQueryResponse *followupFunction*. Only filled when the call-back function is invoked.

udpQueryDest
Destination IP address to send the UDP packet to when using the udpQueryResponse *followupFunction*

udpQuery
The content of the UDP payload when using the udpQueryResponse *followupFunction*

udpCallback
The name of the callback function that is called when using the udpQueryResponse *followupFunction* when an answer is received.

validationState
New in version 4.1.0.

The result of the DNSSEC validation, accessible from the postresolve, nxdomain and nodata hooks. Possible states are pdns.validationstates.Indeterminate, pdns.validationstates.Bogus, pdns.validationstates.Insecure and pdns.validationstates.Secure. The result will always be pdns.validationstates.Indeterminate is validation is disabled or was not requested.

It also supports the following methods:

:addAnswer(*type*, *content*[, *ttl*, *name*])
Add an answer to the record of type with content.

Parameters

- **type** (*int*) – The type of record to add, can be pdns.AAAA etc.
- **content** (*str*) – The content of the record, will be parsed into wireformat based on the type
- **ttl** (*int*) – The TTL in seconds for this record
- **name** (DNSName) – The name of this record, defaults to *DNSQuestion.qname*

:addPolicyTag(*tag*)
Add a policy tag.

Parameters **tag** (*str*) – The tag to add

:discardPolicy(*policyname*)
Skip the filtering policy (for example RPZ) named `policyname` for this query. This is mostly useful in the `prerpz` hook.

Parameters **policyname** (*str*) – The name of the policy to ignore.

:getDH() → DNSHeader
Returns the *DNSHeader* of the query or nil.

:getPolicyTags() → {str}
Get the current policy tags as a table of strings.

:getRecords() → {DNSRecord}
Get a table of DNS Records in this DNS Question (or answer by now).

:setPolicyTags(*tags*)
Set the policy tags to `tags`, overwriting any existing policy tags.

Parameters **tags** (*{str}*) – The policy tags

:setRecords(*records*)
After your edits, update the answers of this question

Parameters **records** (*{DNSRecord}*) – The records to put in the packet

:getEDNSFlag(*name*) → bool
Returns true if the EDNS flag with `name` is set in the query.

Parameters **name** (*string*) – Name of the flag.

:getEDNSFlags() → {str}
Returns a list of strings with all the EDNS flag mnemonics in the query.

:getEDNSOption(*num*) → str
Get the EDNS Option with number `num` as a bytestring.

:getEDNSOptions() → {str: str}
Get a map of all EDNS Options

:getEDNSSubnet() → Netmask
Returns the *Netmask* specified in the EDNSSubnet option, or empty if there was none.

:addPolicyTag(*tag*)
Add policyTag `tag` to the list of policyTags

Parameters **tag** (*str*) – The tag to add

:getPolicyTags() → {str}
Get a list the policyTags for this message.

8.3 DNSHeader Object

The DNS header as returned by *DNSQuestion:getDH()* represents a header of a DNS message.

class DNSHeader

represents a header of a DNS message.

:getRD() → bool
The value of the Recursion Desired bit.

:getAA() → bool
The value of the Authoritative Answer bit.

:getAD() → bool
The value of the Authenticated Data bit.

:getCD() → bool
The value of the Checking Disabled bit.

:getTC() → bool
The value of the Truncation bit.

:getRCODE() → int
The Response Code of the query

:getOPCODE() → int
The Operation Code of the query

:getID() → int
The ID of the query

8.4 The EDNSOptionView Class

class EDNSOptionView
An object that represents the values of a single EDNS option

:count()
.. versionadded:: 4.2.0
The number of values for this EDNS option.

:getValues()
.. versionadded:: 4.2.0
Return a table of NULL-safe strings values for this EDNS option.

size
The size in bytes of the first value of this EDNS option.

:getContent()
Returns a NULL-safe string object of the first value of this EDNS option.

8.5 DNS names and comparing them

The PowerDNS Recursor uses a native format for the names it handles. This native format is exposed to Lua as well.

8.5.1 The DNSName object

The PowerDNS Recursor's Lua engine has the notion of a *DNSName*, an object that represents a name in the DNS. It is returned by several functions and has several functions to programmatically interact with it. *DNSNames* can be compared agains each other using the *:equal* function or the == operator. As names in the DNS are case-insensitive, `www.powerdns.com` is equal to `Www.PowerDNS.cOM`.

Creating a *DNSName* is done with *newDN()*. The PowerDNS Recursor will complain loudly if the name is invalid (e.g. too long, dot in the wrong place).

A small example of the functionality of a *DNSName* is shown below:

```
myname = newDN("www.example.com")
print(myname:countLabels()) -- prints "3"
print(myname:wirelength()) -- prints "17"
name2 = newDN(myname)
name2:chopoff() -- returns true, as 'www' was stripped
print(name2:countLabels()) -- prints "2"
if myname:isPartOf(name2) then -- prints "it is"
```

(continues on next page)

(continued from previous page)

```
  print('it is')
end
```

newDN(*name*) → DNSName
Returns the *DNSName* object of `name`.

Parameters **name** (*string*) – The name to create a DNSName for

class DNSName
A `DNSName` object represents a name in the DNS.

:chopOff() → bool
Removes the left-most label and returns `true`. `false` is returned if no label was removed

:countLabels() → int
Returns the number of DNSLabels in the name

:equal(*name*) → bool
Returns true when both names are equal for the DNS, i.e case insensitive.

Parameters **name** (`DNSName`) – The name to compare against.

:isPartOf(*name*) → bool
Returns true if the DNSName is part of the DNS tree of `name`.

```
newDN("www.powerdns.com"):isPartOf(newDN("CoM.")) -- true
```

Parameters **name** (`DNSName`) – The name to check against

:toString() → str
:toStringNoDot() → str
Returns a human-readable form of the DNSName. With or without trailing dot.

:wirelength() → int
Returns the length in bytes of the DNSName as it would be on the wire.

8.5.2 DNS Suffix Match Groups

The *newDS()* function creates a "Suffix Match group" that allows fast checking if a *DNSName* is part of a group. This could e.g. be used to answer questions for known malware domains. To check e.g. the `dq.qname` against a list:

```
m = newDS()
m:add({'example.com', 'example.net'})
m:check(dq.qname) -- Would be true is dq.qname is a name in example.com or example.
↪net
```

newDS() → DNSSuffixMatchGroup
Creates a new DNS Suffix Match Group.

class DNSSuffixMatchGroup
This class represents a group of DNS names that can be used to quickly compare a single *DNSName* against.

:add(*domain*)
:add(*domains*)
Add one or more domains to the Suffix Match Group.

Parameters

- **domain** (*{str}*) – A domain name to add
- **domain** – A list of Domains to add

:check(*domain*) → bool
Check `domain` against the Suffix Match Group. Returns true if it is matched, false otherwise.

Parameters **domain** (`DNSName`) – The domain name to check

:toString() → str
Returns a string of the set of suffixes matched by the Suffix Match Group

8.6 DNS Record

DNS record objects are returned by *DNSQuestion:getRecords()*.

class DNSRecord
Represents a single DNS record. It has these attributes:

name
The name of the record. A *DNSName*.

place
The place where the record is located,

- 0 for the question section
- 1 for the answer section
- 2 for the authority section
- 3 for the additional section

ttl
The TTL of the record

type
The type of the record (as an integer). Can for example be compared to `pdns.A`.

And the following methods:

:changeContent(*newcontent*)
Replace the record content with `newcontent`. The type and class cannot be changed.

Parameters **newcontent** (*str*) – The replacing content

:getCA() → ComboAddress
If the record type is A or AAAA, a *ComboAddress* representing the content is returned, nil otherwise.

:getContent() → str
Return a string representation of the record content.

8.7 The `ComboAddress` class

IP addresses are moved around in a native format, called ComboAddress within PowerDNS. ComboAddresses can be IPv4 or IPv6, and unless you want to know, you don't need to.

Make a *ComboAddress* with:

```
newCA("::1")
```

A *ComboAddress* can be compared against a NetmaskGroup with the *NetMaskGroup:match()* function.

To compare the address (so not the port) of two ComboAddresses, use `:equal`:

```
a = newCA("[::1]:56")
b = newCA("[::1]:53")
a == b      -- false, port mismatch
a:equal(b) -- true
```

To convert an address to human-friendly representation, use *:toString* or *:toStringWithPort*. To get only the port number, use *:getPort()*.

NewCA (*address*) → ComboAddress
Creates a *ComboAddress*.

Parameters **address** (*string*) – The address to convert

class ComboAddress
An object representing an IP address and port tuple.

:getPort () → int
The portnumber.

:getRaw () → str
A bytestring representing the address.

:isIPv4 () → bool
True if the address is an IPv4 address.

:isIPv6 () → bool
True if the address is an IPv6 address.

:isMappedIPv4 () → bool
True if the address is an IPv4 address mapped into an IPv6 one.

:mapToIPv4 () → ComboAddress
If the address is an IPv4 mapped into an IPv6 one, return the corresponding IPv4 *ComboAddress*.

:toString () → str
Returns the IP address without the port number as a string.

:toStringWithPort () → str
Returns the IP address with the port number as a string.

:truncate (*bits*)
Truncate to the supplied number of bits

Parameters **bits** (*int*) – The number of bits to truncate to

8.8 Netmasks and NetMaskGroups

There are two classes in the PowerDNS Recursor that can be used to match IP addresses against.

8.8.1 Netmask class

The *Netmask* class represents an IP netmask.

```
mask = newNetmask("192.0.2.1/24")
mask:isIPv4() -- true
mask:match("192.0.2.8") -- true
```

newNetmask (*mask*) → Netmask
Creates a new *Netmask*.

Parameters **mask** (*str*) – The mask to convert.

class Netmask
Represents a netmask.

:empty() → bool
True if the netmask doesn't contain a valid address.

:getBits() → int
The number of bits in the address.

:getNetwork() → ComboAddress
Returns a *ComboAddress* representing the network (no mask applied).

:getMaskedNetwork() → ComboAddress
Returns a *ComboAddress* representing the network (truncating according to the mask).

:isIpv4() → bool
True if the netmask is an IPv4 netmask.

:isIpv6() → bool
True if the netmask is an IPv6 netmask.

:match(*address*) → bool
True if the address passed in address matches

Parameters **address** (*str*) – IP Address to match against.

:toString() → str
Returns a human-friendly representation.

8.8.2 NetMaskGroup class

NetMaskGroups are more powerful than plain Netmasks. They can be matched against netmasks objects:

```
nmg = newNMG()
nmg:addMask("127.0.0.0/8")
nmg:addMasks({"213.244.168.0/24", "130.161.0.0/16"})
nmg:addMasks(dofile("bad.ips")) -- contains return {"ip1","ip2"..}

if nmg:match(dq.remoteaddr) then
  print("Intercepting query from ", dq.remoteaddr)
end
```

Prefixing a mask with ! excludes that mask from matching.

newNMG() → NetMaskGroup
Returns a new, empty *NetMaskGroup*.

class NetMaskGroup
IP addresses are passed to Lua in native format.

:addMask(*mask*)
Adds `mask` to the NetMaskGroup.

Parameters **mask** (*str*) – The mask to add.

:addMasks(*masks*)
Adds `masks` to the NetMaskGroup.

Parameters **mask** (*{str}*) – The masks to add.

:match(*address*) → bool
Returns true if `address` matches any of the masks in the group.

Parameters **address** (ComboAddress) – The IP addres to match the netmasks against.

8.9 Lua Scripting and Statistics

The Lua engine can generate metrics and can

8.9.1 Generating Metrics

Custom metrics can be added which will be shown in the output of 'rec_control get-all' and sent to the metrics server over the Carbon protocol. They will also appear in the JSON HTTP API.

Create a custom metric with:

```
myMetric=getMetric("myspecialmetric")
```

getMetric(*name*) → Metric
Returns the *Metric* object with the name `name`, creating the metric if it does not exist.

Parameters **name** (*str*) – The metric to retrieve

class Metric
Represents a custom metric

Metric::inc()
Increase metric by 1

Metric::incBy(amount)
Increase metric by amount

Parameters **amount** (*int*) –

Metric::set(to)
Set metric to value `to`

Parameters **to** (*int*) –

Metric::get() -> int
Get value of metric

Metrics are shared across all of PowerDNS and are fully atomic and high performance. A *Metric* object is effectively a pointer to an atomic value.

Note that metrics live in the same namespace as 'system' metrics. So if you generate one that overlaps with a PowerDNS stock metric, you will get double output and weird results.

8.9.2 Looking at Statistics

New in version 4.1.0.

Statistics can be retrieved from Lua using the *getStat()* call.

getStat(*name*) → int
Returns the value of a statistic.

Parameters **name** (*string*) – The name of the statistic.

For example, to retrieve the number of cache misses:

```
cacheMisses = getStat("cache-misses")
```

Please be aware that retrieving statistics is a relatively costly operation, and as such should for example not be done for every query.

8.10 Logging from the Lua scripts

To log messages with the main PowerDNS Recursor process, use *pdnslog()*. *pdnslog()* can also write out to a syslog loglevel if specified. Use `pdnslog(message, pdns.loglevels.LEVEL)` with the correct pdns.loglevels entry. Entries are listed in the following table:

pdnslog(*message*)
pdnslog(*message*, *level*)
Log `message` at the Info level if ``level`` is not set.

Parameters

- **msg** (*str*) – The message to log
- **level** (*int*) – The log level to log at, see below.

- All - `pdns.loglevels.All`
- Alert - `pdns.loglevels.Alert`
- Critical - `pdns.loglevels.Critical`
- Error - `pdns.loglevels.Error`
- Warning - `pdns.loglevels.Warning`
- Notice - `pdns.loglevels.Notice`
- Info - `pdns.loglevels.Info`
- Debug - `pdns.loglevels.Debug`
- None - `pdns.loglevels.None`

8.11 Intercepting queries with Lua

To get a quick start, we have supplied a sample script that showcases all functionality described below. Please find it here.

Queries can be intercepted in many places:

- before any packet parsing begins (*ipfilter()*)
- before any filtering policy have been applied (*prerpz()*)
- before the resolving logic starts to work (*preresolve()*)
- after the resolving process failed to find a correct answer for a domain (*nodata()*, *nxdomain()*)
- after the whole process is done and an answer is ready for the client (*postresolve()*)
- before an outgoing query is made to an authoritative server (*preoutquery()*)

8.11.1 Writing Lua PowerDNS Recursor scripts

Addresses and DNS Names are not passed as strings but as native objects. This allows for easy checking against Netmasks and domain sets. It also means that to print such names, the `:toString` method must be used (or even `:toStringWithPort` for addresses).

Once a script is loaded, PowerDNS looks for several functions in the loaded script. All of these functions are optional.

If a function returns true, it will indicate that it handled a query. If it returns false, the Recursor will continue processing unchanged (with one minor exception).

8.11.2 Interception Functions

ipfilter(*remoteip*, *localip*, *dh*) → bool

This hook gets queried immediately after consulting the packet cache, but before parsing the DNS packet. If this hook returns something else than false, the packet is dropped. However, because this check is after the packet cache, the IP address might still receive answers that require no packet parsing.

With this hook, undesired traffic can be dropped rapidly before using precious CPU cycles for parsing. As an example, to filter all queries coming from 1.2.3.0/24, or with the AD bit set:

```
badips = newNMG()
badips:addMask("1.2.3.0/24")

function ipfilter(rem, loc, dh)
    return badips:match(rem) or dh:getAD()
end
```

This hook does not get the full *DNSQuestion* object, since filling out the fields would require packet parsing, which is what we are trying to prevent with this function.

Parameters

- **remoteip** (ComboAddress) – The IP(v6) address of the requestor
- **localip** (ComboAddress) – The address on which the query arrived.
- **dh** (DNSHeader) – The DNS Header of the query.

gettag(*remote*, *ednssubnet*, *localip*, *qname*, *qtype*, *ednsoptions*, *tcp*) → int
gettag(*remote*, *ednssubnet*, *localip*, *qname*, *qtype*, *ednsoptions*) → int

Changed in version 4.1.0: The `tcp` parameter was added.

The `gettag` function is invoked when the Recursor attempts to discover in which packetcache an answer is available.

This function must return an integer, which is the tag number of the packetcache. In addition to this integer, this function can return a table of policy tags. The resulting tag number can be accessed via `dq.tag` in the *preresolve()* hook, and the policy tags via *dq:getPolicyTags()* in every hook.

New in version 4.1.0: It can also return a table whose keys and values are strings to fill the *DNSQuestion.data* table, as well as a `requestorId` value to fill the *DNSQuestion.requestorId* field and a `deviceId` value to fill the *DNSQuestion.deviceId* field.

The tagged packetcache can e.g. be used to answer queries from cache that have e.g. been filtered for certain IPs (this logic should be implemented in *gettag()*). This ensure that queries are answered quickly compared to setting *dq.variable* to true. In the latter case, repeated queries will pass through the entire Lua script.

Parameters

- **remote** (ComboAddress) – The sender's IP address
- **ednssubnet** (Netmask) – The EDNS Client subnet that was extracted from the packet
- **localip** (ComboAddress) – The IP address the query was received on
- **qname** (DNSName) – The domain name the query is for
- **qtype** (*int*) – The query type of the query
- **ednsoptions** – A table whose keys are EDNS option codes and values are *EDNSOptionView* objects. This table is empty unless the *gettag-needs-edns-options* option is set.
- **tcp** (*bool*) – Added in 4.1.0, a boolean indicating whether the query was received over UDP (false) or TCP (true).

prerpz(*dq*)
This hook is called before any filtering policy have been applied, making it possible to completely disable filtering by setting *dq.wantsRPZ* to false. Using the *dq:discardPolicy()* function, it is also possible to selectively disable one or more filtering policy, for example RPZ zones, based on the content of the dq object.

As an example, to disable the "malware" policy for example.com queries:

```
function prerpz(dq)
  -- disable the RPZ policy named 'malware' for example.com
  if dq.qname:equal('example.com') then
    dq:discardPolicy('malware')
  end
  return false
end
```

Parameters **dq** (DNSQuestion) – The DNS question to handle

preresolve(*dq*)
This function is called before any DNS resolution is attempted, and if this function indicates it, it can supply a direct answer to the DNS query, overriding the internet. This is useful to combat botnets, or to disable domains unacceptable to an organization for whatever reason.

Parameters **dq** (DNSQuestion) – The DNS question to handle

postresolve(*dq*)
is called right before returning a response to a client (and, unless *dq.variable* is set, to the packet cache too). It allows inspection and modification of almost any detail in the return packet.

Parameters **dq** (DNSQuestion) – The DNS question to handle

nxdomain(*dq*)
is called after the DNS resolution process has run its course, but ended in an 'NXDOMAIN' situation, indicating that the domain does not exist. Works entirely like *postresolve()*, but saves a trip through Lua for answers which are not NXDOMAIN.

Parameters **dq** (DNSQuestion) – The DNS question to handle

nodata(*dq*)
is just like *nxdomain()*, except it gets called when a domain exists, but the requested type does not. This is where one would implement *DNS64*.

Parameters **dq** (DNSQuestion) – The DNS question to handle

preoutquery(*dq*)
This hook is not called in response to a client packet, but fires when the Recursor wants to talk to an authoritative server. When this hook sets the special result code -3, the whole DNS client query causing this outquery gets dropped.

However, this function can also return records like *preresolve()*.

Parameters **dq** (DNSQuestion) – The DNS question to handle

Semantics

The functions must return true if they have taken over the query and wish that the nameserver should not proceed with its regular query-processing. When a function returns false, the nameserver will process the query normally until a new function is called.

If a function has taken over a request, it should set an rcode (usually 0), and specify a table with records to be put in the answer section of a packet. An interesting rcode is NXDOMAIN (3, or pdns.NXDOMAIN), which specifies the non-existence of a domain.

The *ipfilter()* and *preoutquery()* hooks are different, in that *ipfilter()* can only return a true of false value, and that *preoutquery()* can also set rcode -3 to signify that the whole query should be terminated.

A minimal sample script:

```
function nxdomain(dq)
    print("Intercepting NXDOMAIN for: ",dq.qname:toString())
    if dq.qtype == pdns.A
    then
        dq.rcode=0 -- make it a normal answer
        dq:addAnswer(pdns.A, "192.168.1.1")
        return true
    end
    return false
end
```

Warning: Please do NOT use the above sample script in production! Responsible NXDomain redirection requires more attention to detail.

Useful 'rcodes' include 0 for "no error", `pdns.NXDOMAIN` for "NXDOMAIN", `pdns.DROP` to drop the question from further processing. Such a drop is accounted in the 'policy-drops' metric.

8.11.3 DNS64

The `getFakeAAAARecords` and `getFakePTRRecords` followupFunctions can be used to implement DNS64. See DNS64 support in the PowerDNS Recursor for more information.

To get fake AAAA records for DNS64 usage, set dq.followupFunction to `getFakeAAAARecords`, dq.followupPrefix to e.g. "64:ff9b::" and dq.followupName to the name you want to synthesize an IPv6 address for.

For fake reverse (PTR) records, set dq.followupFunction to `getFakePTRRecords` and set dq.followupName to the name to look up and dq.followupPrefix to the same prefix as used with `getFakeAAAARecords`.

8.11.4 Follow up actions

When modifying queries, it might be needed that the Recursor does some extra work after the function returns. The *dq.followupFunction* can be set in this case.

CNAME chain resolution

It may be useful to return a CNAME record for Lua, and then have the PowerDNS Recursor continue resolving that CNAME. This can be achieved by setting dq.followupFunction to `followCNAMERecords` and dq.followupDomain to "www.powerdns.com". PowerDNS will do the rest.

UDP Query Response

The `udpQueryResponse` *dq.followupFunction* allows you to query a simple key-value store over UDP asynchronously.

Several dq variables can be set:

- *dq.udpQueryDest*: destination IP address to send the UDP packet to
- *dq.udpQuery*: The content of the UDP payload
- *dq.udpCallback*: The name of the callback function that is called when an answer is received

The callback function must accept the `dq` object and can find the response to the UDP query in *dq.udpAnswer*.

In this callback function, *dq.followupFunction* can be set again to any of the available functions for further processing.

This example script queries a simple key/value store over UDP to decide on whether or not to filter a query:

```
--[[
This implements a two-step domain filtering solution where the status of an IP␣
↪address
and a domain name need to be looked up.
To do so, we use the udpQuestionResponse answers which generically allows us to do␣
↪asynchronous
lookups via UDP.
Such lookups can be slow, but they won't block PowerDNS while we wait for them.

To benefit from this hook,
..

To test, use the 'kvresp' example program provided.
--]]

function preresolve (dq)
        print ("prereesolve handler called for: "..dq.remoteaddr:toString().. ",␣
↪local: ".. dq.localaddr:toString()..", ".. dq.qname:toString()..", ".. dq.qtype)
        dq.followupFunction="udpQueryResponse"
        dq.udpCallback="gotdomaindetails"
        dq.udpQueryDest=newCA("127.0.0.1:5555")
        dq.udpQuery = "DOMAIN "..dq.qname:toString()
        return true;
end

function gotdomaindetails(dq)
        print("gotdomaindetails called, got: "..dq.udpAnswer)
        if(dq.udpAnswer == "0")
        then
                print("This domain needs no filtering, not looking up this domain")
                dq.followupFunction=""
                return false
        end
        print("Domain might need filtering for some users")
        dq.variable = true -- disable packet cache
        local data={}
        data["domaindetails"]= dq.udpAnswer
        dq.data=data
        dq.udpQuery="IP "..dq.remoteaddr:toString()
        dq.udpCallback="gotipdetails"
        print("returning true in gotipdetails")
        return true
end

function gotipdetails(dq)
        dq.followupFunction=""
        print("So status of IP is "..dq.udpAnswer.." and status of domain is "..dq.
↪data.domaindetails)
        if(dq.data.domaindetails=="1" and dq.udpAnswer=="1")
        then
                print("IP wants filtering and domain is of the filtered kind")
                dq:addAnswer(pdns.CNAME, "blocked.powerdns.com")
                return true
        else
                print("Returning false (normal resolution should proceed, for this␣
↪user)")
```

(continues on next page)

(continued from previous page)

```
                return false
        end
end
```

8.11.5 Example Script

```
pdnslog("pdns-recursor Lua script starting!", pdns.loglevels.Warning)

blockset = newDS()
blockset:add{"powerdns.org", "xxx"}

dropset = newDS();
dropset:add("123.cn")

malwareset = newDS()
malwareset:add("nl")

magic2 = newDN("www.magic2.com")


magicMetric = getMetric("magic")

-- shows the various ways of blocking, dropping, changing questions
-- return false to say you did not take over the question, but we'll still listen
↪to 'variable'
-- to selectively disable the cache
function preresolve(dq)
        print("Got question for "..dq.qname:toString().." from "..dq.
↪remoteaddr:toString().." to "..dq.localaddr:toString())

        local ednssubnet=dq:getEDNSSubnet()
        if(ednssubnet) then
                print("Packet EDNS subnet source: "..ednssubnet:toString(), ".."
↪ednssubnet:getNetwork():toString())
        end


        local a=dq:getEDNSOption(3)
        if(a) then
                print("There is an EDNS option 3 present: "..a)
        end

        loc = newCA("127.0.0.1")
        if(dq.remoteaddr:equal(loc))
        then
                print("Query from loopback")
        end

        -- note that the comparisons below are CaSe InSensiTivE and you don't have
↪to worry about trailing dots
        if(dq.qname:equal("magic.com"))
        then
                magicMetric:inc()
                print("Magic!")
        else
                print("not magic..")
        end

        if(dq.qname:__eq(magic2)) -- we hope to improve this syntax
```

(continues on next page)

(continued from previous page)

```
        then
                print("Faster magic") -- compares against existing DNSName
        end                           -- sadly, dq.qname == magic2 won't work yet

        if blockset:check(dq.qname) then
                dq.variable = true  -- disable packet cache in any case
                if dq.qtype == pdns.A then
                        dq:addAnswer(pdns.A, "1.2.3.4")
                        dq:addAnswer(pdns.TXT, "\"Hello!\"", 3601) -- ttl
↪
                        return true;
                end
        end

        if dropset:check(dq.qname) then
                dq.rcode = pdns.DROP
                return true;
        end



        if malwareset:check(dq.qname) then
                dq:addAnswer(pdns.CNAME, "xs.powerdns.com.")
                dq.rcode = 0
                dq.followupFunction="followCNAMERecords"    -- this makes PowerDNS
↪lookup your CNAME
                return true;
        end

        return false;
end


-- this implements DNS64

function nodata(dq)
        if dq.qtype == pdns.AAAA then
                dq.followupFunction="getFakeAAAARecords"
                dq.followupName=dq.qname
                dq.followupPrefix="fe80::"
                return true
        end

        if dq.qtype == pdns.PTR then
                dq.followupFunction="getFakePTRRecords"
                dq.followupName=dq.qname
                dq.followupPrefix="fe80::"
                return true
        end
        return false
end


badips = newNMG()
badips:addMask("127.1.0.0/16")

-- this check is applied before any packet parsing is done
function ipfilter(rem, loc, dh)
        print("ipfilter called, rem: ", rem:toStringWithPort(), "loc: ",
↪loc:toStringWithPort(),"match:", badips:match(rem))
        print("id: ",dh:getID(), "aa: ", dh:getAA(), "ad: ", dh:getAD(), "arcount:
↪", dh:getARCOUNT())
```

(continues on next page)

(continued from previous page)

```
        print("ports: ",rem:getPort(),loc:getPort())
        return badips:match(rem)
end

-- postresolve runs after the packet has been answered, and can be used to change␣
↪things
-- or still drop
function postresolve(dq)
        print("postresolve called for ",dq.qname:toString())
        local records = dq:getRecords()
        for k,v in pairs(records) do
                print(k, v.name:toString(), v:getContent())
                if v.type == pdns.A and v:getContent() == "185.31.17.73"
                then
                        print("Changing content!")
                        v:changeContent("130.161.252.29")
                        v.ttl=1
                end
        end
        dq:setRecords(records)
        return true
end

nxdomainsuffix=newDN("com")

function nxdomain(dq)
        print("Hooking: ",dq.qname:toString())
        if dq.qname:isPartOf(nxdomainsuffix)
        then
                dq.rcode=0 -- make it a normal answer
                dq:addAnswer(pdns.CNAME, "ourhelpfulservice.com")
                dq:addAnswer(pdns.A, "1.2.3.4", 60, "ourhelpfulservice.com")
                return true
        end
        return false
end
```

Dropping all traffic from botnet-infected users

Frequently, DoS attacks are performed where specific IP addresses are attacked, often by queries coming in from open resolvers. These queries then lead to a lot of queries to 'authoritative servers' which actually often aren't nameservers at all, but just targets of attack.

The following script will add a requestor's IP address to a blocking set if they've sent a query that caused PowerDNS to attempt to talk to a certain subnet.

This specific script is, as of January 2015, useful to prevent traffic to ezdns.it related traffic from creating CPU load. This script requires PowerDNS Recursor 4.x or later.

```
lethalgroup=newNMG()
lethalgroup:addMask("192.121.121.0/24") -- touch these nameservers and you die

function preoutquery(dq)
    print("pdns wants to ask "..dq.remoteaddr:toString().." about "..dq.
↪qname:toString().." "..dq.qtype.." on behalf of requestor "..dq.
↪localaddr:toString())
    if(lethalgroup:match(dq.remoteaddr))
    then
        print("We matched the group "..lethalgroup:tostring().."!", "killing query␣
↪dead & adding requestor "..dq.localaddr:toString().." to block list")
```

(continues on next page)

(continued from previous page)

```
        dq.rcode = -3 -- "kill"
        return true
    end
    return false
end
```

8.11.6 Modifying Policy Decisions

The PowerDNS Recursor has a *policy engine based on Response Policy Zones (RPZ)*. Starting with version 4.0.1 of the recursor, it is possible to alter this decision inside the Lua hooks.

If the decision is modified in a Lua hook, `false` should be returned, as the query is not actually handled by Lua so the decision is picked up by the Recursor. The result of the policy decision is checked after *`preresolve()`* and *`postresolve()`*.

For example, if a decision is set to `pdns.policykinds.NODATA` by the policy engine and is unchanged in *`preresolve()`*, the query is replied to with a NODATA response immediately after *`preresolve()`*.

Example script

```
-- Dont ever block my own domain and IPs
myDomain = newDN("example.com")

myNetblock = newNMG()
myNetblock:addMasks({"192.0.2.0/24"})

function preresolve(dq)
  if dq.qname:isPartOf(myDomain) and dq.appliedPolicy.policyKind ~= pdns.
↪policykinds.NoAction then
    pdnslog("Not blocking our own domain!")
    dq.appliedPolicy.policyKind = pdns.policykinds.NoAction
  end
  return false
end

function postresolve(dq)
  if dq.appliedPolicy.policyKind ~= pdns.policykinds.NoAction then
    local records = dq:getRecords()
    for k,v in pairs(records) do
      if v.type == pdns.A then
        local blockedIP = newCA(v:getContent())
        if myNetblock:match(blockedIP) then
          pdnslog("Not blocking our IP space")
          dq.appliedPolicy.policyKind = pdns.policykinds.NoAction
        end
      end
    end
  end
  return false
end
```

8.11.7 SNMP Traps

PowerDNS Recursor, when compiled with SNMP support, has the ability to act as a SNMP agent to provide SNMP statistics and to be able to send traps from Lua.

For example, to send a custom SNMP trap containing the qname from the `preresolve` hook:

```
function preresolve(dq)
  sendCustomSNMPTrap('Trap from preresolve, qname is '..dq.qname:toString())
  return false
end
```

MIB

```
-- -*- snmpv2 -*-
-- ---------------------------------------------------------------------
-- MIB file for PowerDNS Recursor
-- ---------------------------------------------------------------------

PDNSRECURSOR-MIB DEFINITIONS ::= BEGIN

IMPORTS
    OBJECT-TYPE, MODULE-IDENTITY, enterprises,
    Counter64, NOTIFICATION-TYPE
        FROM SNMPv2-SMI
    CounterBasedGauge64
        FROM HCNUM-TC
    OBJECT-GROUP, MODULE-COMPLIANCE, NOTIFICATION-GROUP
        FROM SNMPv2-CONF;

rec MODULE-IDENTITY
    LAST-UPDATED "201611290000Z"
    ORGANIZATION "PowerDNS BV"
    CONTACT-INFO "support@powerdns.com"
    DESCRIPTION
       "This MIB module describes information gathered through PowerDNS Recursor."

    REVISION "201611290000Z"
    DESCRIPTION "Initial revision."

    ::= { powerdns 2 }

powerdns                OBJECT IDENTIFIER ::= { enterprises 43315 }

stats OBJECT IDENTIFIER ::= { rec 1 }

questions OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of questions"
    ::= { stats 1 }

ipv6Questions OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 questions"
    ::= { stats 2 }

tcpQuestions OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
```

(continues on next page)

(continued from previous page)

```
        "Number of TCP questions"
    ::= { stats 3 }

cacheHits OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of cache hits"
    ::= { stats 4 }

cacheMisses OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of cache misses"
    ::= { stats 5 }

cacheEntries OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of cache entries"
    ::= { stats 6 }

cacheBytes OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Size of the cache in bytes"
    ::= { stats 7 }

packetcacheHits OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of packetcache hits"
    ::= { stats 8 }

packetcacheMisses OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of packetcache misses"
    ::= { stats 9 }

packetcacheEntries OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of packetcache entries"
    ::= { stats 10 }

packetcacheBytes OBJECT-TYPE
    SYNTAX CounterBasedGauge64
```

(continues on next page)

(continued from previous page)

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Size of the packetcache in bytes"
    ::= { stats 11 }

mallocBytes OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of bytes allocated by malloc"
    ::= { stats 12 }

servfailAnswers OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of servfail answers"
    ::= { stats 13 }

nxdomainAnswers OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of nxdomain answers"
    ::= { stats 14 }

noerrorAnswers OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of noerror answers"
    ::= { stats 15 }

unauthorizedUdp OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of unauthorized UDP queries"
    ::= { stats 16 }

unauthorizedTcp OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of unauthorized TCP queries"
    ::= { stats 17 }

tcpClientOverflow OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of TCP client connections refused because of too many connections"
    ::= { stats 18 }
```

(continues on next page)

(continued from previous page)

```
clientParseErrors OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of client parse errors"
    ::= { stats 19 }

serverParseErrors OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of server parse errors"
    ::= { stats 20 }

tooOldDrops OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries dropped because of a timeout"
    ::= { stats 21 }

answers01 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries answered in less than 1 ms"
    ::= { stats 22 }

answers110 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries answered in 1-10 ms"
    ::= { stats 23 }

answers10100 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries answered in 10-100 ms"
    ::= { stats 24 }

answers1001000 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries answered in 100-1000 ms"
    ::= { stats 25 }

answersSlow OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
```

(continues on next page)

(continued from previous page)

```
    DESCRIPTION
        "Number of queries answered in more than 1000 ms"
    ::= { stats 26 }

auth4Answers01 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv4 queries answered in less than 1 ms"
    ::= { stats 27 }

auth4Answers110 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv4 queries answered in 1-10 ms"
    ::= { stats 28 }

auth4Answers10100 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv4 queries answered in 10-100 ms"
    ::= { stats 29 }

auth4Answers1001000 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv4 queries answered in 100-1000 ms"
    ::= { stats 30 }

auth4Answersslow OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv4 queries answered in more than 1000 ms"
    ::= { stats 31 }

auth6Answers01 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 queries answered in less than 1 ms"
    ::= { stats 32 }

auth6Answers110 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 queries answered in 1-10 ms"
    ::= { stats 33 }

auth6Answers10100 OBJECT-TYPE
```

(continues on next page)

(continued from previous page)

```
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 queries answered in 10-100 ms"
    ::= { stats 34 }

auth6Answers1001000 OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 queries answered in 100-1000 ms"
    ::= { stats 35 }

auth6AnswersSlow OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 queries answered in more than 1000 ms"
    ::= { stats 36 }

qaLatency OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Average latency in microseconds"
    ::= { stats 37 }

unexpectedPackets OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of unexpected packets"
    ::= { stats 38 }

caseMismatches OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of case mismatches"
    ::= { stats 39 }

spoofPrevents OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of spoof prevents"
    ::= { stats 40 }

nssetInvalidations OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of nsset invalidations"
```

(continues on next page)

(continued from previous page)

```
    ::= { stats 41 }

resourceLimits OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of resolution aborted because of a local resource limit"
    ::= { stats 42 }

overCapacityDrops OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries dropped because the threads limit was reached"
    ::= { stats 43 }

policyDrops OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries dropped because of a policy"
    ::= { stats 44 }

noPacketError OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of calls to recvmsg() that returned no packet even though the␣
↪socket was ready"
    ::= { stats 45 }

dlgOnlyDrops OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of records dropped because of they belonged to a delegation-only␣
↪zone"
    ::= { stats 46 }

ignoredPackets OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of ignored packets"
    ::= { stats 47 }

maxMthreadStack OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Maximum size of the Mthread stack"
    ::= { stats 48 }

negcacheEntries OBJECT-TYPE
```

(continues on next page)

(continued from previous page)

```
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of negcache entries"
    ::= { stats 49 }

throttleEntries OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of throttle entries"
    ::= { stats 50 }

nsspeedsEntries OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of nsspeeds entries"
    ::= { stats 51 }

failedHostEntries OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of failed host entries"
    ::= { stats 52 }

concurrentQueries OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of concurrent queries"
    ::= { stats 53 }

securityStatus OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Current security status"
    ::= { stats 54 }

outgoingTimeouts OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of outgoing timeouts"
    ::= { stats 55 }

outgoing4Timeouts OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv4 outgoing timeouts"
```

(continues on next page)

(continued from previous page)

```
    ::= { stats 56 }

outgoing6Timeouts OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 outgoing timeouts"
    ::= { stats 57 }

tcpOutqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of outgoing TCP queries sent"
    ::= { stats 58 }

allOutqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of outgoing queries sent"
    ::= { stats 59 }

ipv6Outqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of IPv6 outgoing queries sent"
    ::= { stats 60 }

throttledOutqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of throttled outgoing queries"
    ::= { stats 61 }

dontOutqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of outgoing queries not sent because of a 'dont-query' setting"
    ::= { stats 62 }

unreachables OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of errors due to an unreachable server"
    ::= { stats 63 }

chainResends OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
```

(continues on next page)

(continued from previous page)

```
    STATUS current
    DESCRIPTION
        "Number of chain resends"
    ::= { stats 64 }

tcpClients OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of TCP clients"
    ::= { stats 65 }

udpRecvbufErrors OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of UDP recvbuf errors (Linux only)"
    ::= { stats 66 }

udpSndbufErrors OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of UDP sndbuf errors (Linux only)"
    ::= { stats 67 }

udpNoportErrors OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of UDP noport errors (Linux only)"
    ::= { stats 68 }

udpinErrors OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of UDP in errors (Linux only)"
    ::= { stats 69 }

ednsPingMatches OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of EDNS Ping matches"
    ::= { stats 70 }

ednsPingMismatches OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of EDNS Ping mismatches"
    ::= { stats 71 }
```

(continues on next page)

(continued from previous page)

```
dnssecQueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC queries"
    ::= { stats 72 }

nopingOutqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of outgoing queries w/o ping"
    ::= { stats 73 }

noednsOutqueries OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of outgoing queries w/o EDNS"
    ::= { stats 74 }

uptime OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Process uptime in seconds"
    ::= { stats 75 }

realMemoryUsage OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Memory usage"
    ::= { stats 76 }

fdUsage OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "File descriptors usage"
    ::= { stats 77 }

userMsec OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "CPU usage (user) in ms"
    ::= { stats 78 }

sysMsec OBJECT-TYPE
    SYNTAX CounterBasedGauge64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
```

(continues on next page)

(continued from previous page)

```
        "CPU usage (system) in ms"
    ::= { stats 79 }

dnssecValidations OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC validations"
    ::= { stats 80 }

dnssecResultInsecure OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC insecure results"
    ::= { stats 81 }

dnssecResultSecure OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC secure results"
    ::= { stats 82 }

dnssecResultBogus OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC bogus results"
    ::= { stats 83 }

dnssecResultIndeterminate OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC indeterminate results"
    ::= { stats 84 }

dnssecResultNta OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of DNSSEC NTA results"
    ::= { stats 85 }

policyResultNoaction OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of policy-mandated no-action results"
    ::= { stats 86 }

policyResultDrop OBJECT-TYPE
    SYNTAX Counter64
```

(continues on next page)

(continued from previous page)

```
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of policy-mandated drops"
    ::= { stats 87 }

policyResultNxdomain OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of policy-mandated NXdomain results"
    ::= { stats 88 }

policyResultNodata OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of policy-mandated nodata results"
    ::= { stats 89 }

policyResultTruncate OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of policy-mandated truncate results"
    ::= { stats 90 }

policyResultCustom OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of policy-mandated custom results"
    ::= { stats 91 }

queryPipeFullDrops OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries dropped because the query distribution pipe was full"
    ::= { stats 92 }

truncatedDrops OBJECT-TYPE
    SYNTAX Counter64
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Number of queries dropped because they were larger than 512 bytes"
    ::= { stats 93 }

---
--- Traps / Notifications
---

trap OBJECT IDENTIFIER ::= { rec 10 }
traps OBJECT IDENTIFIER ::= { trap 0 } --- reverse-mappable
trapObjects OBJECT IDENTIFIER ::= { rec 11 }
```

(continues on next page)

(continued from previous page)

```
trapReason OBJECT-TYPE
    SYNTAX OCTET STRING
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "Reason for this trap"
    ::= { trapObjects 1 }

customTrap NOTIFICATION-TYPE
    OBJECTS {
        trapReason
    }
    STATUS current
    DESCRIPTION "Trap sent by sendCustomTrap"
    ::= { traps 1 }

---
--- Conformance
---

recConformance OBJECT IDENTIFIER ::= { rec 100 }

recCompliances MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION "PowerDNS Recursor compliance statement"
    MODULE
    MANDATORY-GROUPS {
        recGroup,
        recTrapsGroup
    }
    ::= { recConformance 1 }

recGroup OBJECT-GROUP
    OBJECTS {
        questions,
        ipv6Questions,
        tcpQuestions,
        cacheHits,
        cacheMisses,
        cacheEntries,
        cacheBytes,
        packetcacheHits,
        packetcacheMisses,
        packetcacheEntries,
        packetcacheBytes,
        mallocBytes,
        servfailAnswers,
        nxdomainAnswers,
        noerrorAnswers,
        unauthorizedUdp,
        unauthorizedTcp,
        tcpClientOverflow,
        clientParseErrors,
        serverParseErrors,
        tooOldDrops,
        answers01,
        answers110,
        answers10100,
        answers1001000,
        answersSlow,
```

(continues on next page)

(continued from previous page)

```
        auth4Answers01,
        auth4Answers110,
        auth4Answers10100,
        auth4Answers1001000,
        auth4Answersslow,
        auth6Answers01,
        auth6Answers110,
        auth6Answers10100,
        auth6Answers1001000,
        auth6AnswersSlow,
        qaLatency,
        unexpectedPackets,
        caseMismatches,
        spoofPrevents,
        nssetInvalidations,
        resourceLimits,
        overCapacityDrops,
        policyDrops,
        noPacketError,
        dlgOnlyDrops,
        ignoredPackets,
        maxMthreadStack,
        negcacheEntries,
        throttleEntries,
        nsspeedsEntries,
        failedHostEntries,
        concurrentQueries,
        securityStatus,
        outgoingTimeouts,
        outgoing4Timeouts,
        outgoing6Timeouts,
        tcpOutqueries,
        allOutqueries,
        ipv6Outqueries,
        throttledOutqueries,
        dontOutqueries,
        unreachables,
        chainResends,
        tcpClients,
        udpRecvbufErrors,
        udpSndbufErrors,
        udpNoportErrors,
        udpinErrors,
        ednsPingMatches,
        ednsPingMismatches,
        dnssecQueries,
        nopingOutqueries,
        noednsOutqueries,
        uptime,
        realMemoryUsage,
        fdUsage,
        userMsec,
        sysMsec,
        dnssecValidations,
        dnssecResultInsecure,
        dnssecResultSecure,
        dnssecResultBogus,
        dnssecResultIndeterminate,
        dnssecResultNta,
        policyResultNoaction,
        policyResultDrop,
```

(continues on next page)

(continued from previous page)

```
        policyResultNxdomain,
        policyResultNodata,
        policyResultTruncate,
        policyResultCustom,
        queryPipeFullDrops,
        truncatedDrops,
        trapReason
    }
    STATUS current
    DESCRIPTION "Objects conformance group for PowerDNS Recursor"
    ::= { recConformance 2 }

recTrapsGroup NOTIFICATION-GROUP
    NOTIFICATIONS {
        customTrap
    }
    STATUS current
    DESCRIPTION "Traps conformance group for PowerDNS Recursor"
    ::= { recConformance 3 }

END
```

8.11.8 Maintenance callback

Starting with version 4.1.4 of the recursor, it is possible to define a *maintenance()* callback function that will be called periodically. This function expects no argument and doesn't return any value

```
function maintenance()
    -- This would be called every second
    -- Perform here your maintenance
end
```

The interval can be configured through the *lua-maintenance-interval* setting.

8.12 Other functions

These are some functions that don't really have a place in one of the other categories.

getregisteredname(*name*) → str
Returns the shortest domain name based on Mozilla's Public Suffix List. In general it will tell you the 'registered domain' for a given name.

For example `getregisteredname('www.powerdns.com')` returns "powerdns.com"

Parameters **name** (*str*) – The name to check for.

getRecursorThreadId() → int
returns an unsigned integer identifying the thread handling the current request.

CHAPTER

NINE

BUILT-IN WEBSERVER AND HTTP API

The PowerDNS Recursor features a built-in built-in webserver that exposes a JSON/REST API. This API allows for controlling several functions and reading statistics.

The following documents contain the information for the PowerDNS API:

9.1 Data format

The API accepts and emits JSON. The `Accept:` header determines the output format. An unknown value or `*/*` will cause a `400 Bad Request`.

All text is UTF-8 and HTTP headers will reflect this.

Data types:

- empty fields: `null` but present
- Regex: implementation defined
- Dates: ISO 8601

9.1.1 General Collections Interface

Collections generally support `GET` and `POST` with these meanings:

GET

Retrieve a list of all entries.

The special `type` and `url` fields are included in the response objects:

- `type`: name of the resource type
- `url`: url to the object

Response format:

```
[
  obj1
  [, further objs]
]
```

Example:

```
[
  {
    "type": "AType",
    "id": "anid",
```

(continues on next page)

(continued from previous page)

```
    "url": "/atype/anid",
    "a_field": "a_value"
  },
  {
    "type": "AType",
    "id": "anotherid",
    "url": "/atype/anotherid",
    "a_field": "another_value"
  }
]
```

POST

Create a new entry. The client has to supply the entry in the request body, in JSON format. `application/x-www-form-urlencoded` data MUST NOT be sent.

Clients SHOULD not send the 'url' field.

Client body:

```
obj1
```

Example:

```
{
  "type": "AType",
  "id": "anewid",
  "a_field": "anew_value"
}
```

9.1.2 REST

- GET: List/Retrieve. Success reply: `200 OK`
- POST: Create. Success reply: `201 Created`, with new object as body.
- PUT: Update. Success reply: `200 OK`, with modified object as body. For some operations, `204 No Content` is returned instead (and the modified object is not given in the body).
- DELETE: Delete. Success reply: `200 OK`, no body.

9.1.3 not-so-REST

For interactions that do not directly map onto CRUD, we use these:

- GET: Query. Success reply: `200 OK`
- PUT: Action/Execute. Success reply: `200 OK`

Action/Execute methods return a JSON body of this format:

```
{
  "message": "result message"
}
```

9.1.4 Authentication

The PowerDNS daemons accept a static API Key, configured with the *api-key* option, which has to be sent in the `X-API-Key` header.

9.1.5 Errors

Response code `4xx` or `5xx`, depending on the situation. Never return `2xx` for an error!

- Invalid JSON body from client: `400 Bad Request`
- JSON body from client not a hash: `400 Bad Request`
- Input validation failed: `422 Unprocessable Entity`

Error responses have a JSON body of this format:

```
{
  "error": "short error message",
  "errors": [
    {  },
  ]
}
```

Where `errors` is optional, and the contents are error-specific.

Common Error Causes

400 Bad Request

1. The client body was not a JSON document, or it could not be parsed, or the root element of the JSON document was not a hash.
2. The client did not send an `Accept:` header, or it was set to `*/*`.
3. For requests that operate on a zone, the `zone_id` URL part was invalid. To get a valid `zone_id`, list the zones with the `/api/v1/servers/:server_id/zones` endpoint.

9.2 Server

`Server`
An object representing a single PowerDNS server. In the built-in API, only one Server exists (called "localhost").

pdnsmgrd and pdnscontrol MUST NOT return "localhost", but SHOULD return other servers.

Object Properties

- **`type`** (*`string`*) – Set to "Server"
- **`id`** (*`string`*) – The id of the server, "localhost"
- **`daemon_type`** (*`string`*) – "recursor" for the PowerDNS Recursor and "authoritative" for the Authoritative Server
- **`version`** (*`string`*) – The version of the server software
- **`url`** (*`string`*) – The API endpoint for this server
- **`config_url`** (*`string`*) – The API endpoint for this server's configuration
- **`zones_url`** (*`string`*) – The API endpoint for this server's zones

Example:

```
{
  "type": "Server",
  "id": "localhost",
  "url": "/api/v1/servers/localhost",
```

(continues on next page)

(continued from previous page)

```
  "daemon_type": "recursor",
  "version": "4.1.0",
  "config_url": "/api/v1/servers/localhost/config{/config_setting}",
  "zones_url": "/api/v1/servers/localhost/zones{/zone}",
}
```

Note: the servers collection is read-only, and the only allowed returned server is read-only as well. On a pdnscontrol server, the servers collection is read-write, and the returned server resources are read-write as well. Write permissions may depend on the credentials you have supplied.

9.3 Zones

9.3.1 Zone

A Zone object represents an authoritative DNS Zone.

A Resource Record Set (below as "RRset") are all records for a given name and type.

Comments are per-RRset.

Zone

Represents a configured zone in the PowerDNS server.

Object Properties

- **id** (*string*) – Opaque zone id (string), assigned by the server, should not be interpreted by the application. Guaranteed to be safe for embedding in URLs.
- **name** (*string*) – Name of the zone (e.g. "example.com.") MUST have a trailing dot
- **type** (*string*) – Set to "Zone"
- **url** (*string*) – API endpoint for this zone
- **kind** (*string*) – Zone kind, one of "Native", "Forwarded".
- **rrsets** (*[RRSet]*) – RRSets in this zone
- **servers** (*[str]*) – For zones of type "Forwarded", addresses to send the queries to
- **recursion_desired** (*bool*) – For zones of type "Forwarded", Whether or not the RD bit should be set in the query

9.3.2 RRSet

RRSet

This represents a Resource Record set (all record with the same name and type).

Object Properties

- **name** (*string*) – Name for record set (e.g. "www.powerdns.com.")
- **type** (*string*) – Type of this record (e.g. "A", "PTR", "MX")
- **ttl** (*integer*) – DNS TTL of the records, in seconds. MUST NOT be included when `changetype` is set to "DELETE".
- **changetype** (*string*) – MUST be added when updating the RRSet. Must be `REPLACE` or `DELETE`. With `DELETE`, all existing RRs matching `name` and `type` will be deleted, including all comments. With `REPLACE`: when `records` is present, all existing RRs matching `name` and `type` will be deleted, and then new records given in `records` will be created. If no records are left, any existing comments will be

deleted as well. When `comments` is present, all existing comments for the RRs matching `name` and `type` will be deleted, and then new comments given in `comments` will be created.

- **records** (*[RREntry]*) – All records in this RRSet. When updating records, this is the list of new records (replacing the old ones). Must be empty when `changetype` is set to `DELETE`. An empty list results in deletion of all records (and comments).
- **comments** (*[Comment]*) – List of *Comment*. Must be empty when `changetype` is set to `DELETE`. An empty list results in deletion of all comments. `modified_at` is optional and defaults to the current server time.

9.3.3 RREntry

RREntry

The RREntry object represents a single record in an *RRSet*.

Object Properties

- **content** (*string*) – The content of this record
- **disabled** (*bool*) – Whether or not this record is disabled
- **set-ptr** (*bool*) – If set to true, the server will find the matching reverse zone and create a PTR there. Existing PTR records are replaced. If no matching reverse *Zone*, an error is thrown. Only valid in client bodies, only valid for A and AAAA types. Not returned by the server.

9.3.4 Comment

Comment

Object Properties

- **content** (*string*) – The actual comment
- **account** (*string*) – Name of an account that added the comment
- **modified_at** (*integer*) – Timestamp of the last change to the comment

9.4 ConfigSetting

ConfigSetting

Represents a configuration item (as found in :doc:'../settings')

Object Properties

- **type** (*string*) – set to "ConfigSetting"
- **name** (*string*) – The name of this setting (e.g. 'webserver-port')
- **value** (*string*) – The value of setting `name`

Example:

```
{
  "name": "webserver-port",
  "type": "ConfigSetting",
  "value": "8081"
}
```

9.5 StatisticItem

StatisticItem
Represents a single statistic item (as found in *Gathered Information*)

Object Properties

- **type** (*string*) – set to "StatisticItem"
- **name** (*string*) – The name of this item
- **value** (*string*) – The value of this item

9.6 Webserver

To launch the internal webserver, add a *webserver* to the configuration file. This will instruct PowerDNS to start a webserver on localhost at port 8081, without password protection. By default the webserver listens on localhost, meaning only local users (on the same host) will be able to access the webserver. Since the default ACL before 4.1.0 allows access from everywhere if *webserver-address* is set to a different value, we strongly advise the use of a password protection. The webserver lists a lot of potentially sensitive information about the PowerDNS process, including frequent queries, frequently failing queries, lists of remote hosts sending queries, hosts sending corrupt queries etc. The webserver does not allow remote management. The following webserver related configuration items are available:

- *webserver*: If set to anything but 'no', a webserver is launched.
- *webserver-address*: Address to bind the webserver to. Defaults to 127.0.0.1, which implies that only the local computer is able to connect to the nameserver! To allow remote hosts to connect, change to 0.0.0.0 or the physical IP address of your nameserver.
- *webserver-password*: If set, viewers will have to enter this plaintext password in order to gain access to the statistics.
- *webserver-port*: Port to bind the webserver to.
- *webserver-allow-from*: Netmasks that are allowed to connect to the webserver

9.7 Enabling the API

To enable the API, the webserver and the HTTP API need to be enabled. Add these lines to the `recursor.conf`:

```
webserver=yes
webserver-port=8082
api-key=changeme
```

And restart `pdns_recursor`, the following examples should start working:

```
curl -v -H 'X-API-Key: changeme' http://127.0.0.1:8082/api/v1/servers/localhost |␣
↪jq .
curl -v -H 'X-API-Key: changeme' http://127.0.0.1:8082/api/v1/servers/localhost/
↪zones | jq .
```

9.8 URL Endpoints

All API endpoints for the PowerDNS Recursor are documented here:

9.8.1 API root endpoints

GET /api
Version discovery endpoint.

Example response:

```
[
  {
    "url": "/api/v1",
    "version": 1
  }
]
```

GET /api/v1
APIv1 root endpoint. Gives some information about the current API.

Not yet implemented:

- `api_features`
- `servers_modifiable`
- `oauth`

Example response:

```
{
  "server_url": "/api/v1/servers{/server}",
  "api_features": []
}
```

9.8.2 Server endpoint

GET /api/v1/servers
Server collection access.

GET /api/v1/servers/:server_id
Returns a single *Server*

Parameters

- **server_id** – The name of the server.

9.8.3 Configuration endpoint

9.8.4 Configuration endpoint

GET /api/v1/servers/:server_id/config
Returns all *ConfigSetting* for a single server

Parameters

- **server_id** – The name of the server

POST /api/v1/servers/:server_id/config

Note: Not implemented

Creates a new config setting. This is useful for creating configuration for new backends.

Parameters

- **server_id** – The name of the server

GET /api/v1/servers/:server_id/config/:config_setting_name
Retrieve a single setting

Parameters

- **server_id** – The name of the server
- **config_setting_name** – The name of the setting to retrieve

PUT /api/v1/servers/:server_id/config/:config_setting_name
Change a single setting

Note: Only *allow-from* can be set.

Parameters

- **server_id** – The name of the server
- **config_setting_name** – The name of the setting to change

Example request

```
PUT /api/v1/servers/localhost/config/allow-from HTTP/1.1
Host: localhost:8082
User-Agent: curl/7.54.1
Accept: application/json
X-Api-Key: secret
Content-Type: application/json
Content-Length: 48

{ "name": "allow-from", "value": ["127.0.0.0/8"] }
```

Example response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: close
Content-Length: 48
Content-Security-Policy: default-src 'self'; style-src 'self' 'unsafe-inline'
Content-Type: application/json
Server: PowerDNS/0.0.g00799130f
X-Content-Type-Options: nosniff
X-Frame-Options: deny
X-Permitted-Cross-Domain-Policies: none
X-Xss-Protection: 1; mode=block

{"name": "allow-from", "value": ["127.0.0.0/8"]}
```

9.8.5 Statistics endpoint

GET /api/v1/servers/:server_id/statistics
Query PowerDNS internal statistics. Returns a list of *StatisticItem* elements.

The names and meaning of these items are described *here*.

Parameters

- **server_id** – The name of the server

Example response:


```
[{"name": "all-outqueries", "type": "StatisticItem", "value": "341"}, {"name":
↪"answers-slow", "type": "StatisticItem", "value": "0"}, {"name": "answers0-1
↪", "type": "StatisticItem", "value": "0"}, {"name": "answers1-10", "type":
↪"StatisticItem", "value": "0"}, {"name": "answers10-100", "type":
↪"StatisticItem", "value": "0"}, {"name": "answers100-1000", "type":
↪"StatisticItem", "value": "0"}, {"name": "auth4-answers-slow", "type":
↪"StatisticItem", "value": "200"}, {"name": "auth4-answers0-1", "type":
↪"StatisticItem", "value": "13"}, {"name": "auth4-answers1-10", "type":
↪"StatisticItem", "value": "1"}, {"name": "auth4-answers10-100", "type":
↪"StatisticItem", "value": "68"}, {"name": "auth4-answers100-1000", "type":
↪"StatisticItem", "value": "19"}, {"name": "auth6-answers-slow", "type":
↪"StatisticItem", "value": "0"}, {"name": "auth6-answers0-1", "type":
↪"StatisticItem", "value": "0"}, {"name": "auth6-answers1-10", "type":
↪"StatisticItem", "value": "0"}, {"name": "auth6-answers10-100", "type":
↪"StatisticItem", "value": "0"}, {"name": "auth6-answers100-1000", "type":
↪"StatisticItem", "value": "0"}, {"name": "cache-entries", "type":
↪"StatisticItem", "value": "124"}, {"name": "cache-hits", "type":
↪"StatisticItem", "value": "0"}, {"name": "cache-misses", "type":
↪"StatisticItem", "value": "0"}, {"name": "case-mismatches", "type":
↪"StatisticItem", "value": "0"}, {"name": "chain-resends", "type":
↪"StatisticItem", "value": "0"}, {"name": "client-parse-errors", "type":
↪"StatisticItem", "value": "0"}, {"name": "concurrent-queries", "type":
↪"StatisticItem", "value": "1"}, {"name": "dlg-only-drops", "type":
↪"StatisticItem", "value": "0"}, {"name": "dnssec-queries", "type":
↪"StatisticItem", "value": "0"}, {"name": "dnssec-result-bogus", "type":
↪"StatisticItem", "value": "0"}, {"name": "dnssec-result-indeterminate", "type
↪": "StatisticItem", "value": "0"}, {"name": "dnssec-result-insecure", "type
↪": "StatisticItem", "value": "0"}, {"name": "dnssec-result-nta", "type":
↪"StatisticItem", "value": "0"}, {"name": "dnssec-result-secure", "type":
↪"StatisticItem", "value": "9"}, {"name": "dnssec-validations", "type":
↪"StatisticItem", "value": "9"}, {"name": "dont-outqueries", "type":
↪"StatisticItem", "value": "0"}, {"name": "edns-ping-matches", "type":
↪"StatisticItem", "value": "0"}, {"name": "edns-ping-mismatches", "type":
↪"StatisticItem", "value": "0"}, {"name": "failed-host-entries", "type":
↪"StatisticItem", "value": "0"}, {"name": "fd-usage", "type": "StatisticItem",
↪ "value": "25"}, {"name": "ignored-packets", "type": "StatisticItem", "value
↪": "0"}, {"name": "ipv6-outqueries", "type": "StatisticItem", "value": "0"},
↪{"name": "ipv6-questions", "type": "StatisticItem", "value": "0"}, {"name":
↪"malloc-bytes", "type": "StatisticItem", "value": "0"}, {"name": "max-
↪mthread-stack", "type": "StatisticItem", "value": "0"}, {"name": "negcache-
↪entries", "type": "StatisticItem", "value": "1"}, {"name": "no-packet-error",
↪ "type": "StatisticItem", "value": "0"}, {"name": "noedns-outqueries", "type
↪": "StatisticItem", "value": "0"}, {"name": "noerror-answers", "type":
↪"StatisticItem", "value": "0"}, {"name": "noping-outqueries", "type":
↪"StatisticItem", "value": "0"}, {"name": "nsset-invalidations", "type":
↪"StatisticItem", "value": "0"}, {"name": "nsspeeds-entries", "type":
↪"StatisticItem", "value": "26"}, {"name": "nxdomain-answers", "type":
↪"StatisticItem", "value": "0"}, {"name": "outgoing-timeouts", "type":
↪"StatisticItem", "value": "200"}, {"name": "outgoing4-timeouts", "type":
↪"StatisticItem", "value": "200"}, {"name": "outgoing6-timeouts", "type":
↪"StatisticItem", "value": "0"}, {"name": "over-capacity-drops", "type":
↪"StatisticItem", "value": "0"}, {"name": "packetcache-entries", "type":
↪"StatisticItem", "value": "0"}, {"name": "packetcache-hits", "type":
↪"StatisticItem", "value": "0"}, {"name": "packetcache-misses", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-drops", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-result-custom", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-result-drop", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-result-noaction", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-result-nodata", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-result-nxdomain", "type":
↪"StatisticItem", "value": "0"}, {"name": "policy-result-truncate", "type":
↪"StatisticItem", "value": "0"}, {"name": "qa-latency", "type": "StatisticItem
↪", "value": "0"}, {"name": "questions", "type": "StatisticItem", "value": "0
↪"}, {"name": "real-memory-usage", "type": "StatisticItem", "value": "3502080
↪"}, {"name": "resource-limits", "type": "StatisticItem", "value": "13"}, {
↪"name": "security-status", "type": "StatisticItem", "value": "0"}, {"name":
↪"server-parse-errors", "type": "StatisticItem", "value": "0"}, {"name":
↪"servfail-answers", "type": "StatisticItem", "value": "0"}, {"name": "spoof-
```


(continues on next page)

(continued from previous page)

9.8.6 Zones endpoint

GET /api/v1/servers/:server_id/zones
Get all zones from the server.

Query Parameters

- **server_id** – The name of the server

POST /api/v1/servers/:server_id/zones
Creates a new domain. The client body must contain a *Zone*.

Query Parameters

- **server_id** – The name of the server

GET /api/v1/servers/:server_id/zones/:zone_id
Returns zone information.

Query Parameters

- **server_id** – The name of the server
- **zone_id** – The id number of the *Zone*

DELETE /api/v1/servers/:server_id/zones/:zone_id
Deletes this zone, all attached metadata and rrsets.

Query Parameters

- **server_id** – The name of the server
- **zone_id** – The id number of the *Zone*

9.8.7 Query Tracing endpoint

Note: Not yet implemented

PUT /api/v1/servers/:server_id/trace
Configure query tracing.

Query Parameters

- **server_id** – The name of the server

Client body:

```
{
  "domains": "<regex_string>"
}
```

Set `domains` to `null` to turn off tracing.

GET /api/v1/servers/:server_id/trace
Retrieve query tracing log and current config.

Query Parameters

- **server_id** – The name of the server

Response Body:

```
{
  "domains": "<Regex>",
  "log": [
    "<log_line>"
  ]
}
```

9.8.8 Cache manipulation endpoint

PUT /api/v1/servers/:server_id/cache/flush?domain=:domain
Flush the positive, negative and packet cache for a given domain name.

Query Parameters

- **server_id** – The name of the server
- **domain** – The domainname to flush for

New in version 4.1.3.

Query Parameters

- **subtree** – If set to *true*, also flush the whole subtree (default = *false*)

Example Response:

```
{
  "count": 10,
  "result": "Flushed cache."
}
```

9.8.9 Log endpoint

GET /api/v1/servers/:server_id/search-log?q=:search_term
Query the log, filtered by `search_term`. Returns a single JSON object with a single array of strings.

Parameters

- **server_id** – The name of the server
- **search_term** – The string to search for

9.8.10 Failure logging endpoint

Note: Not yet implemented

PUT /api/v1/servers/:server_id/failure
Configure query failure logging.

Query Parameters

- **server_id** – The name of the server

Example client body:

```
{
  "top-domains": 100,
  "domains": ".*\.example\.com$",
}
```

Property int top-domains Number of top resolved domains that are automatically monitored for failures.

Property string domains A Regex of domains that are additionally monitored for resolve failures.

GET /api/v1/servers/:server_id/failure

Note: Not yet implemented

Retrieve query failure logging and current config.

Example response body:

```
{
  "top-domains": 100,
  "domains": ".*\.example\.com$",
  "log": [
    {
      "first_occurred": 1234567890,
      "domain": "www.example.net",
      "qtype": "A",
      "failure": "dnssec-parent-validation-failed",
      "failed_parent": "example.com",
      "details": "foo bar",
      "queried_servers": [
          {
            "name": "ns1.example.net",
            "address": "192.0.2.53"
          },
      ]
    },
  ]
}
```

Property string failed_parent The parent domain, this is generally OPTIONAL.

Property string failure_code Reason of failure.

- `dnssec-validation-failed`: DNSSEC Validation failed for this domain.
- `dnssec-parent-validation-failed`: DNSSEC Validation failed for one of the parent domains. Response MUST contain `failed_parent`.
- `nxdomain`: This domain was not present on the authoritative nameservers.
- `nodata`: ???
- `all-servers-unreachable`: All auth nameservers that have been tried did not respond.
- `parent-unresolvable`: Response MUST contain `failed_parent`.
- `refused`: All auth nameservers that have been tried responded with REFUSED.
- `servfail`: All auth nameservers that have been tried responded with SERVFAIL.

Property string domain The domain queried

9.8.11 RPZ Statistics endpoint

New in version 4.1.2.

GET /api/v1/servers/:server_id/rpzstatistics
Query PowerDNS for *Response Policy Zones* statistics.

Statistics are mapped per configured RPZ zone. The statistics are:

Last_update UNIX timestamp when the latest update was received

Records Number of records in the RPZ

Serial Current SOA serial of the RPZ zone

Transfers_failed Number of times a transfer failed

Transfers_full Number of times an AXFR succeeded

Transfers_success Number of times an AXFR or IXFR succeeded

Example response:

```
{
  "myRPZ": {
    "last_update": 1521798212,
    "records": 1343149,
    "serial": 5489,
    "transfers_failed": 0,
    "transfers_full": 3,
    "transfers_success": 478
  }
}
```

CHAPTER

TEN

DNS64 SUPPORT

DNS64, described in **RFC 6147** is a technology to allow IPv6-only clients to receive special IPv6 addresses that are proxied to IPv4 addresses. This proxy service is then called NAT64.

As an example, let's say an IPv6 only client would want to connect to `www.example.com`, it would request the AAAA records for that name. However, if `example.com` does not actually have an IPv6 address, what we do is 'fake up' an IPv6 address. We do this by retrieving the A records for `www.example.com`, and translating them to AAAA records. Elsewhere, a NAT64 device listens on these IPv6 addresses, and extracts the IPv4 address from each packet, and proxies it on.

For maximum flexibility, DNS64 support is included in the *Scripting The Recursor*. This allows for example to hand out custom IPv6 gateway ranges depending on the location of the requestor, enabling the use of NAT64 services close to the user.

Apart from faking AAAA records, it is also possible to also generate the associated PTR records. This makes sure that reverse lookup of DNS64-generated IPv6 addresses generate the right name. The procedure is similar, a request for an IPv6 PTR is converted into one for the corresponding IPv4 address.

To setup DNS64, with both forward and reverse records, create the following Lua script and save it to a file called `dns64.lua`

```
-- this small script implements dns64 without any specials or customization
prefix = "fe80::21b:77ff:0:0"

function nodata ( dq )
  if dq.qtype ~= pdns.AAAA then
    return false
  end  --  only AAAA records

  -- don't fake AAAA records if DNSSEC validation failed
  if dq.validationState == pdns.validationstates.Bogus then
     return false
  end

  dq.followupFunction = "getFakeAAAARecords"
  dq.followupPrefix = prefix
  dq.followupName = dq.qname
  return true
end

-- the ip6.arpa address is the reverse of the prefix address above
function preresolve ( dq )
  if dq.qtype == pdns.PTR and dq.qname:isPartOf(newDN("f.f.7.7.b.1.2.0.0.0.0.0.0.0.
↪0.0.0.0.0.0.0.8.e.f.ip6.arpa.")) then
    dq.followupFunction = "getFakePTRRecords"
    dq.followupPrefix = prefix
    dq.followupName = dq.qname
    return true
  end
```

(continues on next page)

(continued from previous page)

```
  return false
end
```

Where fe80::21b::77ff:0:0 is your "Pref64" translation prefix and the "ip6.arpa" string is the reversed form of this Pref64 address. Now ensure your script gets loaded by specifying it with *lua-dns-script=dns64.lua*.

To enhance DNS64, see the *Scripting The Recursor* documentation.

CHAPTER

ELEVEN

SECURITY OF THE POWERDNS RECURSOR

For Security Advisories, see the *dedicated page*.

11.1 PowerDNS Security Policy

If you have a security problem to report, please email us at both security@powerdns.com and ahu@ds9a.nl. Please do not mail security issues to public lists, nor file a ticket, unless we do not get back to you in a timely manner. We fully credit reporters of security issues, and respond quickly, but please allow us a reasonable timeframe to coordinate a response.

We remind PowerDNS users that under the terms of the GNU General Public License, PowerDNS comes with ABSOLUTELY NO WARRANTY. This license is included in this documentation.

As of the 9th of September 2016, no actual security problems with PowerDNS Authoritative Server 3.4.10, Recursor 3.6.3, Recursor 3.7.2, or later are known about. This page will be updated with all bugs which are deemed to be security problems, or could conceivably lead to those. Any such notifications will also be sent to all PowerDNS mailing lists.

11.1.1 HackerOne

Security issues can also be reported on our HackerOne page and might fetch a bounty. Do note that only the PowerDNS software is in scope for the HackerOne program, not our websites or other infrastructure.

11.1.2 Disclosure Policy

- Let us know as soon as possible upon discovery of a potential security issue, and we'll make every effort to quickly resolve the issue.
- Provide us a reasonable amount of time to resolve the issue before any disclosure to the public or a third-party.
- We will always credit researchers in our *Security Advisories*.

11.2 Anti-spoofing

The PowerDNS Recursor uses a fresh UDP source port for each outgoing query, making spoofing around 64000 times harder. This raises the bar from 'easily doable given some time' to 'very hard'. Under some circumstances, 'some time' has been measured at 2 seconds. This technique was first used by `dnscache` by Dan J. Bernstein and is standardized in **RFC 5452**

In addition, PowerDNS detects when it is being sent too many unexpected answers, and mistrusts a proper answer if found within a clutch of unexpected ones.

This behaviour can be tuned using the *spoof-nearmiss-max*.

11.3 Throttling

PowerDNS implements a very simple but effective nameserver. Care has been taken not to overload remote servers in case of overly active clients.

This is implemented using the 'throttle'. This accounts all recent traffic and prevents queries that have been sent out recently from going out again.

There are three levels of throttling.

- If a remote server indicates that it is lame for a zone, the exact question won't be repeated in the next 60 seconds.
- After 4 ServFail responses in 60 seconds, the query gets throttled too.
- 5 timeouts in 20 seconds also lead to query suppression.

11.4 Security Polling

PowerDNS products can poll the security status of their respective versions. This polling, naturally, happens over DNS. If the result is that a given version has a security problem, the software will report this at level 'Error' during startup, and repeatedly during operations.

By default, security polling happens on the domain 'secpoll.powerdns.com', but this can be changed with the *security-poll-suffix*. If this setting is made empty, no polling will take place. Organizations wanting to host their own security zones can do so by changing this setting to a domain name under their control.

To make this easier, the zone used to host secpoll.powerdns.com is available here.

To enable distributors of PowerDNS to signal that they have backported versions, the PACKAGEVERSION compilation-time macro can be used to set a distributor suffix.

11.4.1 Details

PowerDNS software sadly sometimes has critical security bugs. Even though we send out notifications of these via all channels available, we find that not everybody actually find out about our security releases.

To solve this, PowerDNS software will start polling for security notifications, and log these periodically. Secondly, the security status of the software will be reported using the built-in metrics. This allows operators to poll for the PowerDNS security status and alert on it.

In the implementation of this idea, we have taken the unique role of operating system distributors into account. Specifically, we can deal with backported security fixes.

Finally, this feature can be disabled, or operators can have the automated queries point at their own status service.

Implementation

PowerDNS software periodically tries to resolve 'auth-x.y.z.security-status.secpoll.powerdns.com|TXT' or 'recursor-x.y.z.security-status.secpoll.powerdns.com'.

The data returned is in one of the following forms:

- NXDOMAIN or resolution failure -> 0
- "1 Ok" -> 1
- "2 Upgrade recommended for security reasons, see . . . " -> 2
- "3 Upgrade mandatory for security reasons, see . . . " -> 3

In cases 2 or 3, periodic logging commences. The metric security-status is set to 2 or 3 respectively. If at a later date, resolution fails, the security-status is not reset to 1. It could be lowered however if we discover the security status is less urgent than we thought.

If resolution fails, and the previous security-status was 1, the new security-status becomes 0 ('no data'). If the security-status was higher than 1, it will remain that way, and not get set to 0.

In this way, security-status of 0 really means 'no data', and can not mask a known problem.

Distributions

Distributions frequently backport security fixes to the PowerDNS versions they ship. This might lead to a version number that is known to us to be insecure to be secure in reality.

To solve this issue, PowerDNS can be compiled with a distribution setting which will move the security polls from: 'auth-x.y.z.security-status.secpoll.powerdns.com' to 'auth-x.y.z-n.debian.security-status.secpoll.powerdns.com.

Note two things, one, there is a separate namespace for debian, and secondly, we use the package version of this release. This allows us to know that 4.0.1-1 (say) is insecure, but that 4.0.1-2 is not.

Configuration Details

The configuration setting *security-poll-suffix* is by default set to 'secpoll.powerdns.com'. If empty, nothing is polled. This can be moved to 'secpoll.yourorganization.com'.

If compiled with PACKAGEVERSION=3.1.6-abcde.debian, queries will be sent to "auth-3.1.6-abcde.debian.security-status.security-poll-suffix".

Delegation

If a distribution wants to host its own file with version information, we can delegate dist.security-status.secpoll.powerdns.com to their nameservers directly.

CHAPTER

TWELVE

POWERDNS RECURSOR SETTINGS

Each setting can appear on the command line, prefixed by '–', or in the configuration file. The command line overrides the configuration file.

Note: Settings marked as 'Boolean' can either be set to an empty value, which means on, or to 'no' or 'off' which means off. Anything else means on.

As an example:

- `serve-rfc1918` on its own means: do serve those zones.
- `serve-rfc1918=off` or `serve-rfc1918=no` means: do not serve those zones.
- Anything else means: do serve those zones.

12.1 aaaa-additional-processing

- Boolean
- Default: No

If turned on, the recursor will attempt to add AAAA IPv6 records to questions for MX records and NS records. Can be quite slow as absence of these records in earlier answers does not guarantee their non-existence. Can double the amount of queries needed.

12.2 allow-from

- IP ranges, separated by commas
- Default: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

Netmasks (both IPv4 and IPv6) that are allowed to use the server. The default allows access only from **RFC 1918** private IP addresses. Due to the aggressive nature of the internet these days, it is highly recommended to not open up the recursor for the entire internet. Questions from IP addresses not listed here are ignored and do not get an answer.

12.3 allow-from-file

- Path

Like *allow-from*, except reading from file. Overrides the *allow-from* setting. To use this feature, supply one netmask per line, with optional comments preceded by a "#".

12.4 any-to-tcp

- Boolean
- Default: no

Answer questions for the ANY type on UDP with a truncated packet that refers the remote server to TCP. Useful for mitigating ANY reflection attacks.

12.5 api-config-dir

New in version 4.0.0.

- Path
- Default: unset

Directory where the REST API stores its configuration and zones.

12.6 api-key

New in version 4.0.0.

- String
- Default: unset

Static pre-shared authentication key for access to the REST API.

12.7 api-readonly

New in version 4.0.0.

- Boolean
- Default: no

Disallow data modification through the REST API when set.

12.8 api-logfile

New in version 4.0.0.

- Path
- Default: unset

Location of the server logfile (used by the REST API).

12.9 auth-can-lower-ttl

- Boolean
- Default: no

Authoritative zones can transmit a TTL value that is lower than that specified in the parent zone. This is called a 'delegation inconsistency'. To follow **RFC 2181 section 5.2** and **5.4** to the letter, enable this feature. This will mean a slight deterioration of performance, and it will not solve any problems, but does make the recursor more standards compliant. Not recommended unless you have to tick an 'RFC 2181 compliant' box.

12.10 auth-zones

- Comma separated list of 'zonename=filename' pairs

Zones read from these files (in BIND format) are served authoritatively. DNSSEC is not supported. Example:

```
auth-zones=example.org=/var/zones/example.org, powerdns.com=/var/zones/powerdns.com
```

12.11 carbon-interval

- Integer
- Default: 30

If sending carbon updates, this is the interval between them in seconds. See *Metrics and Statistics*.

12.12 carbon-ourname

- String

If sending carbon updates, if set, this will override our hostname. Be careful not to include any dots in this setting, unless you know what you are doing. See *Sending metrics to Graphite/Metronome over Carbon*.

12.13 carbon-server

- IP address

If set to an IP or IPv6 address, will send all available metrics to this server via the carbon protocol, which is used by graphite and metronome. You may specify an alternate port by appending :port, for example: `127.0.0.1:2004`. See *Metrics and Statistics*.

12.14 chroot

- Path to a Directory

If set, chroot to this directory for more security. See *Security of the PowerDNS Recursor*

Make sure that `/dev/log` is available from within the chroot. Logging will silently fail over time otherwise (on logrotate).

When using `chroot`, all other paths (except for *config-dir*) set in the configuration are relative to the new root.

When using `chroot` and the API (*webserver*), *api-readonly* **must** be set and *api-config-dir* unset.

When running on a system where systemd manages services, `chroot` does not work out of the box, as PowerDNS cannot use the `NOTIFY_SOCKET`. Either do not `chroot` on these systems or set the 'Type' of this service to 'simple' instead of 'notify' (refer to the systemd documentation on how to modify unit-files).

12.15 client-tcp-timeout

- Integer
- Default: 2

Time to wait for data from TCP clients.

12.16 config-dir

- Path

Location of configuration directory (`recursor.conf`). Usually `/etc/powerdns`, but this depends on `SYSCONFDIR` during compile-time.

12.17 config-name

- String
- Default: unset

When running multiple recursors on the same server, read settings from `recursor-name.conf`, this will also rename the binary image.

12.18 cpu-map

New in version 4.1.0.

- String
- Default: unset

Set CPU affinity for worker threads, asking the scheduler to run those threads on a single CPU, or a set of CPUs. This parameter accepts a space separated list of thread-id=cpu-id, or thread-id=cpu-id-1,cpu-id-2,...,cpu-id-N. For example, to make the worker thread 0 run on CPU id 0 and the worker thread 1 on CPUs 1 and 2:

```
cpu-map=0=0 1=1,2
```

The number of worker threads is determined by the *threads* setting. If *pdns-distributes-queries* is set, an additional thread is started, assigned the id 0, and is the only one listening on client sockets and accepting queries, distributing them to the other worker threads afterwards.

This parameter is only available on OS that provides the *pthread_setaffinity_np()* function.

12.19 daemon

- Boolean
- Default: no

Changed in version 4.0.0: Default is now "no", was "yes" before.

Operate in the background.

12.20 delegation-only

- Domains, comma separated

Which domains we only accept delegations from (a Verisign special).

12.21 disable-packetcache

- Boolean
- Default: no

Turn off the packet cache. Useful when running with Lua scripts that can not be cached.

12.22 disable-syslog

- Boolean
- Default: no

Do not log to syslog, only to stdout. Use this setting when running inside a supervisor that handles logging (like systemd). **Note**: do not use this setting in combination with *daemon* as all logging will disappear.

12.23 dnssec

New in version 4.0.0.

- One of `off`, `process-no-validate`, `process`, `log-fail`, `validate`, String
- Default: `process-no-validate`

Set the mode for DNSSEC processing:

12.23.1 off

No DNSSEC processing whatsoever. Ignore DO-bits in queries, don't request any DNSSEC information from authoritative servers. This behaviour is similar to PowerDNS Recursor pre-4.0.

12.23.2 process-no-validate

Respond with DNSSEC records to clients that ask for it, set the DO bit on all outgoing queries. Don't do any validation.

12.23.3 process

Respond with DNSSEC records to clients that ask for it, set the DO bit on all outgoing queries. Do validation for clients that request it (by means of the AD- bit or DO-bit in the query).

12.23.4 log-fail

Similar behaviour to `process`, but validate RRSIGs on responses and log bogus responses.

12.23.5 validate

Full blown DNSSEC validation. Send SERVFAIL to clients on bogus responses.

12.24 dnssec-log-bogus

- Boolean
- Default: no

Log every DNSSEC validation failure. **Note**: This is not logged per-query but every time records are validated as Bogus.

12.25 dont-query

- Netmasks, comma separated
- Default: 127.0.0.0/8, 10.0.0.0/8, 100.64.0.0/10, 169.254.0.0/16, 192.168.0.0/16, 172.16.0.0/12, ::1/128, fc00::/7, fe80::/10, 0.0.0.0/8, 192.0.0.0/24, 192.0.2.0/24, 198.51.100.0/24, 203.0.113.0/24, 240.0.0.0/4, ::/96, ::ffff:0:0/96, 100::/64, 2001:db8::/32

The DNS is a public database, but sometimes contains delegations to private IP addresses, like for example 127.0.0.1. This can have odd effects, depending on your network, and may even be a security risk. Therefore, the PowerDNS Recursor by default does not query private space IP addresses. This setting can be used to expand or reduce the limitations.

Queries to addresses for zones as configured in any of the settings *forward-zones*, *forward-zones-file* or *forward-zones-recurse* are performed regardless of these limitations.

12.26 ecs-add-for

New in version 4.2.0.

- Comma separated list of netmasks
- Default: 0.0.0.0/0, ::, !127.0.0.0/8, !10.0.0.0/8, !100.64.0.0/10, !169.254.0.0/16, !192.168.0.0/16, !172.16.0.0/12, !::1/128, !fc00::/7, !fe80::/10

List of requestor netmasks for which the requestor IP Address should be used as the **EDNS Client Subnet** for outgoing queries. Outgoing queries for requestors that do not match this list will use the *ecs-scope-zero-address* instead. Valid incoming ECS values from *use-incoming-edns-subnet* are not replaced.

Regardless of the value of this setting, ECS values are only sent for outgoing queries matching the conditions in the *edns-subnet-whitelist* setting. This setting only controls the actual value being sent.

This defaults to not using the requestor address inside RFC1918 and similar "private" IP address spaces.

12.27 ecs-ipv4-bits

New in version 4.1.0.

- Integer
- Default: 24

Number of bits of client IPv4 address to pass when sending EDNS Client Subnet address information.

12.28 ecs-ipv6-bits

New in version 4.1.0.

- Integer
- Default: 56

Number of bits of client IPv6 address to pass when sending EDNS Client Subnet address information.

12.29 ecs-scope-zero-address

New in version 4.1.0.

- IPv4 or IPv6 Address
- Default: empty

The IP address sent via EDNS Client Subnet to authoritative servers listed in *edns-subnet-whitelist* when *use-incoming-edns-subnet* is set and the query has an ECS source prefix-length set to 0. The default is to look for the first usable (not an `any` one) address in *query-local-address* then *query-local-address6*. If no suitable address is found, the recursor fallbacks to sending 127.0.0.1.

12.30 edns-outgoing-bufsize

- Integer
- Default: 1680

This is the value set for the EDNS0 buffer size in outgoing packets. Lower this if you experience timeouts.

12.31 edns-subnet-whitelist

- Comma separated list of domain names and netmasks
- Default: (none)

List of netmasks and domains that **EDNS Client Subnet** should be enabled for in outgoing queries.

For example, an EDNS Client Subnet option containing the address of the initial requestor (but see *ecs-add-for*) will be added to an outgoing query sent to server 192.0.2.1 for domain X if 192.0.2.1 matches one of the supplied netmasks, or if X matches one of the supplied domains. The initial requestor address will be truncated to 24 bits for IPv4 (see *ecs-ipv4-bits*) and to 56 bits for IPv6 (see *ecs-ipv6-bits*), as recommended in the privacy section of RFC 7871.

By default, this option is empty, meaning no EDNS Client Subnet information is sent.

12.32 entropy-source

- Path
- Default: /dev/urandom

PowerDNS can read entropy from a (hardware) source. This is used for generating random numbers which are very hard to predict. Generally on UNIX platforms, this source will be `/dev/urandom`, which will always supply random numbers, even if entropy is lacking. Change to `/dev/random` if PowerDNS should block waiting for enough entropy to arrive.

12.33 etc-hosts-file

- Path
- Default: /etc/hosts

The path to the /etc/hosts file, or equivalent. This file can be used to serve data authoritatively using *export-etc-hosts*.

12.34 export-etc-hosts

- Boolean
- Default: no

If set, this flag will export the host names and IP addresses mentioned in `/etc/hosts`.

12.35 export-etc-hosts-search-suffix

- String

If set, all hostnames in the *export-etc-hosts* file are loaded in canonical form, based on this suffix, unless the name contains a '.', in which case the name is unchanged. So an entry called 'pc' with `export-etc-hosts-search-suffix='home.com'` will lead to the generation of 'pc.home.com' within the recursor. An entry called 'server1.home' will be stored as 'server1.home', regardless of this setting.

12.36 forward-zones

- 'zonename=IP' pairs, comma separated

Queries for zones listed here will be forwarded to the IP address listed. i.e.

```
forward-zones=example.org=203.0.113.210, powerdns.com=2001:DB8::BEEF:5
```

Multiple IP addresses can be specified and port numbers other than 53 can be configured:

```
forward-zones=example.org=203.0.113.210:5300;127.0.0.1, powerdns.com=127.0.0.1;198.
↪51.100.10:530;[2001:DB8::1:3]:5300
```

Forwarded queries have the 'recursion desired' bit set to 0, meaning that this setting is intended to forward queries to authoritative servers.

IMPORTANT: When using DNSSEC validation (which is default), forwards to non-delegated (e.g. internal) zones that have a DNSSEC signed parent zone will validate as Bogus. To prevent this, add a Negative Trust Anchor (NTA) for this zone in the *lua-config-file* with `addNTA("your.zone", "A comment")`. If this forwarded zone is signed, instead of adding NTA, add the DS record to the *lua-config-file*. See the *DNSSEC in the PowerDNS Recursor* information.

12.37 forward-zones-file

- Path

Same as *forward-zones*, parsed from a file. Only 1 zone is allowed per line, specified as follows:

```
example.org=203.0.113.210, 192.0.2.4:5300
```

Zones prefixed with a '+' are forwarded with the recursion-desired bit set, for which see *forward-zones-recurse*. Default behaviour without '+' is as with *forward-zones*.

Changed in version 4.0.0: Comments are allowed, everything behind '#' is ignored.

The DNSSEC notes from *forward-zones* apply here as well.

12.38 forward-zones-recurse

- 'zonename=IP' pairs, comma separated

Like regular *forward-zones*, but forwarded queries have the 'recursion desired' bit set to 1, meaning that this setting is intended to forward queries to other recursive servers.

The DNSSEC notes from *forward-zones* apply here as well.

12.39 gettag-needs-edns-options

New in version 4.1.0.

- Boolean
- Default: no

If set, EDNS options in incoming queries are extracted and passed to the *gettag()* hook in the `ednsoptions` table.

12.40 hint-file

- Path

If set, the root-hints are read from this file. If unset, default root hints are used.

12.41 include-dir

- Path

Directory to scan for additional config files. All files that end with .conf are loaded in order using `POSIX` as locale.

12.42 latency-statistic-size

- Integer
- Default: 10000

Indication of how many queries will be averaged to get the average latency reported by the 'qa-latency' metric.

12.43 local-address

- IP addresses, comma separated
- Default: 127.0.0.1

Local IPv4 or IPv6 addresses to bind to. Addresses can also contain port numbers, for IPv4 specify like this: `192.0.2.4:5300`, for IPv6: `[::1]:5300`.

Warning: When binding to wildcard addresses, UNIX semantics mean that answers may not be sent from the address a query was received on. It is highly recommended to bind to explicit addresses.

12.44 local-port

- Integer
- Default: 53

Local port to bind to. If an address in *local-address* does not have an explicit port, this port is used.

12.45 log-timestamp

New in version 4.1.0.

- Bool
- Default: yes

When printing log lines to stdout, prefix them with timestamps. Disable this if the process supervisor timestamps these lines already.

Note: The systemd unit file supplied with the source code already disables timestamp printing

12.46 non-local-bind

- Boolean
- Default: no

Bind to addresses even if one or more of the *local-address*'s do not exist on this server. Setting this option will enable the needed socket options to allow binding to non-local addresses. This feature is intended to facilitate ip-failover setups, but it may also mask configuration issues and for this reason it is disabled by default.

12.47 loglevel

- Integer between 0 and 9
- Default: 6

Amount of logging. Higher is more, more logging may destroy performance. It is recommended not to set this below 3.

12.48 log-common-errors

- Boolean
- Default: no

Some DNS errors occur rather frequently and are no cause for alarm.

12.49 log-rpz-changes

New in version 4.1.0.

- Boolean
- Default: no

Log additions and removals to RPZ zones at Info (6) level instead of Debug (7).

12.50 logging-facility

- Integer

If set to a digit, logging is performed under this LOCAL facility. See *Logging*. Do not pass names like 'local0'!

12.51 lowercase-outgoing

- Boolean
- Default: no

Set to true to lowercase the outgoing queries. When set to 'no' (the default) a query from a client using mixed case in the DNS labels (such as a user entering mixed-case names or draft-vixie-dnsext-dns0x20-00), PowerDNS preserves the case of the query. Broken authoritative servers might give a wrong or broken answer on this encoding. Setting `lowercase-outgoing` to 'yes' makes the PowerDNS Recursor lowercase all the labels in the query to the authoritative servers, but still return the proper case to the client requesting.

12.52 lua-config-file

- Filename

If set, and Lua support is compiled in, this will load an additional configuration file for newer features and more complicated setups. See *Lua Configuration* for the options that can be set in this file.

12.53 lua-dns-script

- Path
- Default: unset

Path to a lua file to manipulate the Recursor's answers. See *Scripting The Recursor* for more information.

12.54 lua-maintenance-interval

New in version 4.1.4.

- Integer
- Default: 1

The interval between calls to the Lua user defined *maintenance()* function in seconds. See *Maintenance callback*

12.55 max-cache-entries

- Integer
- Default: 1000000

Maximum number of DNS cache entries. 1 million per thread will generally suffice for most installations.

12.56 max-cache-ttl

- Integer
- Default: 86400

Maximum number of seconds to cache an item in the DNS cache, no matter what the original TTL specified.

Changed in version 4.1.0: The minimum value of this setting is 15. i.e. setting this to lower than 15 will make this value 15.

12.57 max-mthreads

- Integer
- Default: 2048

Maximum number of simultaneous MTasker threads.

12.58 max-packetcache-entries

- Integer
- Default: 500000

Maximum number of Packet Cache entries. 1 million per thread will generally suffice for most installations.

12.59 max-qperq

- Integer
- Default: 50

The maximum number of outgoing queries that will be sent out during the resolution of a single client query. This is used to limit endlessly chasing CNAME redirections.

12.60 max-negative-ttl

- Integer
- Default: 3600

A query for which there is authoritatively no answer is cached to quickly deny a record's existence later on, without putting a heavy load on the remote server. In practice, caches can become saturated with hundreds of thousands of hosts which are tried only once. This setting, which defaults to 3600 seconds, puts a maximum on the amount of time negative entries are cached.

12.61 max-recursion-depth

- Integer
- Default: 40

Total maximum number of internal recursion calls the server may use to answer a single query. 0 means unlimited. The value of *stack-size* should be increased together with this one to prevent the stack from overflowing.

Changed in version 4.1.0: Before 4.1.0, this settings was unlimited.

12.62 max-tcp-clients

- Integer
- Default: 128

Maximum number of simultaneous incoming TCP connections allowed.

12.63 max-tcp-per-client

- Integer
- Default: 0 (unlimited)

Maximum number of simultaneous incoming TCP connections allowed per client (remote IP address).

12.64 max-tcp-queries-per-connection

New in version 4.1.0.

- Integer
- Default: 0 (unlimited)

Maximum number of DNS queries in a TCP connection.

12.65 max-total-msec

- Integer
- Default: 7000

Total maximum number of milliseconds of wallclock time the server may use to answer a single query.

12.66 max-udp-queries-per-round

New in version 4.2.0.

- Integer
- Default: 10000

Under heavy load the recursor might be busy processing incoming UDP queries for a long while before there is no more of these, and might therefore neglect scheduling new `mthreads`, handling responses from authoritative servers or responding to *rec_control* requests. This setting caps the maximum number of incoming UDP DNS queries processed in a single round of looping on `recvmsg()` after being woken up by the multiplexer, before returning back to normal processing and handling other events.

12.67 minimum-ttl-override

- Integer
- Default: 0 (disabled)

This setting artificially raises all TTLs to be at least this long. While this is a gross hack, and violates RFCs, under conditions of DoS, it may enable you to continue serving your customers. Can be set at runtime using `rec_control set-minimum-ttl 3600`.

12.68 network-timeout

- Integer
- Default: 1500

Number of milliseconds to wait for a remote authoritative server to respond.

12.69 nsec3-max-iterations

New in version 4.1.0.

- Integer
- Default: 2500

Maximum number of iterations allowed for an NSEC3 record. If an answer containing an NSEC3 record with more iterations is received, its DNSSEC validation status is treated as Insecure.

12.70 packetcache-ttl

- Integer
- Default: 3600

Maximum number of seconds to cache an item in the packet cache, no matter what the original TTL specified.

12.71 packetcache-servfail-ttl

- Integer
- Default: 60

Maximum number of seconds to cache a 'server failure' answer in the packet cache.

Changed in version 4.0.0: This setting's maximum is capped to *packetcache-ttl*. i.e. setting `packetcache-ttl=15` and keeping `packetcache-servfail-ttl` at the default will lower `packetcache-servfail-ttl` to `15`.

12.72 pdns-distributes-queries

- Boolean
- Default: yes

If set, PowerDNS will have only 1 thread listening on client sockets, and distribute work by itself over threads. Improves performance on Linux.

12.73 query-local-address

- IPv4 Address, comma separated
- Default: 0.0.0.0

Send out local queries from this address, or addresses, by adding multiple addresses, increased spoofing resilience is achieved.

12.74 query-local-address6

- IPv6 addresses, comma separated
- Default: unset

Send out local IPv6 queries from this address or addresses. Disabled by default, which also disables outgoing IPv6 support.

12.75 quiet

- Boolean
- Default: yes

Don't log queries.

12.76 reuseport

- Boolean
- Default: no

If `SO_REUSEPORT` support is available, allows multiple processes to open a listening socket on the same port.

Since 4.1.0, when `pdns-distributes-queries` is set to false and `reuseport` is enabled, every thread will open a separate listening socket to let the kernel distribute the incoming queries, avoiding any thundering herd issue as well as the distributor thread being a bottleneck, thus leading to much higher performance on multi-core boxes.

12.77 rng

- String
- Default: auto

Specify which random number generator to use. Permissible choises are

- auto - choose automatically
- sodium - Use libsodium `randombytes_uniform`
- openssl - Use libcrypto `RAND_bytes`
- getrandom - Use libc getrandom, falls back to urandom if it does not really work
- arc4random - Use BSD `arc4random_uniform`
- urandom - Use `/dev/urandom`
- kiss - Use simple settable deterministic RNG. **FOR TESTING PURPOSES ONLY!**

Note: Not all choises are available on all systems.

12.78 root-nx-trust

- Boolean
- Default: yes

If set, an NXDOMAIN from the root-servers will serve as a blanket NXDOMAIN for the entire TLD the query belonged to. The effect of this is far fewer queries to the root-servers.

Changed in version 4.0.0: Default is 'yes' now, was 'no' before 4.0.0

12.79 security-poll-suffix

- String
- Default: secpoll.powerdns.com.

Domain name from which to query security update notifications. Setting this to an empty string disables secpoll.

12.80 serve-rfc1918

- Boolean
- Default: yes

This makes the server authoritatively aware of: `10.in-addr.arpa`, `168.192.in-addr.arpa`, `16-31.172.in-addr.arpa`, which saves load on the AS112 servers. Individual parts of these zones can still be loaded or forwarded.

12.81 server-down-max-fails

- Integer
- Default: 64

If a server has not responded in any way this many times in a row, no longer send it any queries for *server-down-throttle-time* seconds. Afterwards, we will try a new packet, and if that also gets no response at all, we again throttle for *server-down-throttle-time* seconds. Even a single response packet will drop the block.

12.82 server-down-throttle-time

- Integer
- Default: 60

Throttle a server that has failed to respond *server-down-max-fails* times for this many seconds.

12.83 server-id

- String
- Default: The hostname of the server

The reply given by The PowerDNS recursor to a query for 'id.server' with its hostname, useful for in clusters. When a query contains the **:rfc:'NSID EDNS0 Option <5001>'__**, this value is returned in the response as the NSID value.

This setting can be used to override the answer given to these queries. Set to "disabled" to disable NSID and 'id.server' answers.

Query example (where 192.0.2.14 is your server):

```
dig @192.0.2.14 CHAOS TXT id.server.
dig @192.0.2.14 example.com IN A +nsid
```

12.84 setgid, setuid

- String
- Default: unset

PowerDNS can change its user and group id after binding to its socket. Can be used for better *security*.

12.85 single-socket

- Boolean
- Default: no

Use only a single socket for outgoing queries.

12.86 snmp-agent

New in version 4.1.0.

- Boolean
- Default: no

If set to true and PowerDNS has been compiled with SNMP support, it will register as an SNMP agent to provide statistics and be able to send traps.

12.87 snmp-master-socket

New in version 4.1.0.

- String
- Default: empty

If not empty and `snmp-agent` is set to true, indicates how PowerDNS should contact the SNMP master to register as an SNMP agent.

12.88 socket-dir

- Path

Where to store the control socket and pidfile. The default depends on `LOCALSTATEDIR` during compile-time (usually `/var/run` or `/run`).

When using *chroot* the default becomes to `/`.

12.89 socket-owner, socket-group, socket-mode

Owner, group and mode of the controlsocket. Owner and group can be specified by name, mode is in octal.

12.90 spoof-nearmiss-max

- Integer
- Default: 20

If set to non-zero, PowerDNS will assume it is being spoofed after seeing this many answers with the wrong id.

12.91 stack-size

- Integer
- Default: 200000

Size of the stack per thread.

12.92 statistics-interval

New in version 4.1.0.

- Integer
- Default: 1800

Interval between logging statistical summary on recursor performance. Use 0 to disable.

12.93 stats-ringbuffer-entries

- Integer
- Default: 10000

Number of entries in the remotes ringbuffer, which keeps statistics on who is querying your server. Can be read out using `rec_control top-remotes`.

12.94 tcp-fast-open

New in version 4.1.0.

- Integer
- Default: 0 (Disabled)

Enable TCP Fast Open support, if available, on the listening sockets. The numerical value supplied is used as the queue size, 0 meaning disabled.

12.95 threads

- Integer
- Default: 2

Spawn this number of threads on startup.

12.96 trace

- Boolean
- Default: no

If turned on, output impressive heaps of logging. May destroy performance under load.

12.97 udp-source-port-min

New in version 4.2.0.

- Integer
- Default: 1024

This option sets the low limit of UDP port number to bind on.

In combination with *udp-source-port-max* it configures the UDP port range to use. Port numbers are randomized within this range on initialization, and exceptions can be configured with *udp-source-port-avoid*

12.98 udp-source-port-max

New in version 4.2.0.

- Integer
- Default: 65535

This option sets the maximum limit of UDP port number to bind on.

See *udp-source-port-min*.

12.99 udp-source-port-avoid

New in version 4.2.0.

- String
- Default: 11211

A list of comma-separated UDP port numbers to avoid when binding. Ex: *5300,11211*

See *udp-source-port-min*.

12.100 udp-truncation-threshold

- Integer
- Default: 1680

EDNS0 allows for large UDP response datagrams, which can potentially raise performance. Large responses however also have downsides in terms of reflection attacks. This setting limits the accepted size. Maximum value is 65535, but values above 4096 should probably not be attempted.

12.101 use-incoming-edns-subnet

- Boolean
- Default: no

Whether to process and pass along a received EDNS Client Subnet to authoritative servers. The ECS information will only be sent for netmasks and domains listed in *edns-subnet-whitelist* and will be truncated if the received scope exceeds *ecs-ipv4-bits* for IPv4 or *ecs-ipv6-bits* for IPv6.

12.102 version

Print version of this binary. Useful for checking which version of the PowerDNS recursor is installed on a system.

12.103 version-string

- String
- Default: PowerDNS Recursor version number

By default, PowerDNS replies to the 'version.bind' query with its version number. Security conscious users may wish to override the reply PowerDNS issues.

12.104 webserver

- Boolean
- Default: no

Start the webserver (for REST API).

12.105 webserver-address

- IP Address
- Default: 127.0.0.1

IP address for the webserver to listen on.

12.106 webserver-allow-from

- IP addresses, comma separated
- Default: 127.0.0.1,::1

Changed in version 4.1.0: Default is now 127.0.0.1,::1, was 0.0.0.0,::/0 before.

These subnets are allowed to access the webserver.

12.107 webserver-password

- String
- Default: unset

Password required to access the webserver.

12.108 webserver-port

- Integer
- Default: 8082

TCP port where the webserver should listen on.

12.109 write-pid

- Boolean
- Default: yes

If a PID file should be written to *socket-dir*

12.110 xpf-allow-from

New in version 4.2.0.

- IP ranges, separated by commas
- Default: empty

Note: This is an experimental implementation of draft-bellis-dnsop-xpf.

The server will trust XPF records found in queries sent from those netmasks (both IPv4 and IPv6), and will adjust queries' source and destination accordingly. This is especially useful when the recursor is placed behind a proxy like dnsdist. Note that the ref:*setting-allow-from* setting is still applied to the original source address, and thus access restriction should be done on the proxy.

12.111 `xpf-rr-code`

New in version 4.2.0.

- Integer
- Default: 0

Note: This is an experimental implementation of draft-bellis-dnsop-xpf.

This option sets the resource record code to use for XPF records, as long as an official code has not been assigned to it. 0 means that XPF is disabled.

CHAPTER

THIRTEEN

MANUAL PAGES

13.1 pdns_recursor

13.1.1 Synopsis

pdns_recursor [*OPTION*]...

13.1.2 Description

pdns_recursor is a high performance, simple and secure recursing nameserver. It currently powers hundreds of millions internet connections.

The recursor is configured via a configuration file, but each item in that file can be overridden on the command line.

This manpage lists the core set of features needed to get the PowerDNS Recursor working, for full and up to date details head to https://doc.powerdns.com/.

13.1.3 Examples

To listen on 192.0.2.53 and allow the 192.0.2.0/24 subnet to recurse, and run as in the background, execute:

```
# pdns_recursor --local-address=192.0.2.53 --allow-from=192.0.2.0/24 --daemon
```

To stop the recursor by hand, run:

```
# rec_control quit
```

However, the recommended way of starting and stopping the recursor is to use the init.d script or *systemctl(1)*.

13.1.4 Options

For authoritative listing of options, consult the online documentation at <*https://doc.powerdns.com/*>

--allow-from=<networks> If set, only allow these comma separated *networks*, with network mask to recurse. For example: 192.0.2.0/24,203.0.113.128/25.

--auth-zones=<authzones> Where *authzone* is <zonename>=<filename>. Serve *zonename* from *filename* authoritatively. For example: ds9a.nl=/var/zones/ds9a.nl,powerdns.com=/var/zones/powerdns.com.

--chroot=<directory> chroot the process to *directory*.

--client-tcp-timeout=<num> Timeout in seconds when talking to TCP clients.

--config-dir=<directory> Location of configuration directory (recursor.conf), the default depends on the SYSCONFDIR option at build-time, which is usually /etc/powerdns. The default can be found with `pdns_recursor --config | grep 'config-dir='`.

--daemon Operate as a daemon.

--delegation-only Which domains we only accept delegations from (a Verisign special).

--entropy-source=<file> Read new entropy from *file*, defaults to /dev/urandom.

--export-etc-hosts If set, this flag will export the hostnames and IP addresses mentioned in /etc/hosts.

--forward-zones=<forwardzones> Where *forwardzone* is <zonename>=<address>. Queries for *zonename* will be forwarded to *address*. *address* should be an IP address, not a hostname (to prevent chicken and egg problems). Example: forward-zones= ds9a.nl=213.244.168.210, powerdns.com=127.0.0.1.

--forward-zones-file=<filename> Similar to *–forward-zones*, but read the options from *filename*. *filename* should contain one zone per line, like: ds9a.nl=213.244.168.210.

--help Show a summary of options.

--hint-file=<filename> Load root hints from this *filename*

--local-address=<address> Listen on *address*, separated by spaces or commas.

--local-port=<port> Listen on *port*.

--log-common-errors If we should log rather common errors.

--max-cache-entries=<num> Maximum number of entries in the main cache.

--max-negative-ttl=<num> maximum number of seconds to keep a negative cached entry in memory.

--max-tcp-clients=<num> Maximum number of simultaneous TCP clients.

--max-tcp-per-client=<num> If set, maximum number of TCP sessions per client (IP address).

--query-local-address=<address> Use *address* as Source IP address when sending queries.

--query-local-address6=<address> Send out local IPv6 queries from *address*. Disabled by default, which also disables outgoing IPv6 support. A useful setting is '::0'.

--quiet Suppress logging of questions and answers.

--server-id=<text> Return *text* WHen queried for 'id.server' TXT, defaults to hostname.

--serve-rfc1918 On by default, this makes the server authoritatively aware of: 10.in-addr.arpa, 168.192.in-addr.arpa and 16-31.172.in-addr.arpa, which saves load on the AS112 servers. Individual parts of these zones can still be loaded or forwarded.

--setgid=<gid> If set, change group id to *gid* for more security.

--setuid=<uid> If set, change user id to *uid* for more security.

--single-socket If set, only use a single socket for outgoing queries.

--socket-dir=<directory> The controlsocket will live in *directory*.

--spoof-nearmiss-max=<num> If non-zero, assume spoofing after this many near misses.

--trace if we should output heaps of logging.

--version-string=<text> *text* WILL be reported on version.pdns or version.bind queries.

13.1.5 See also

rec_control(1) systemctl(1)

13.2 rec_control

13.2.1 Synopsis

rec_control [*OPTION*]... *COMMAND* [*COMMAND-OPTION*]...

13.2.2 Description

rec_control allows the operator to query and control a running instance of the PowerDNS Recursor.

rec_control talks to the recursor via a the 'controlsocket'. Which is usually located in `/var/run` . The *–socket-dir* or the *–config-dir* and *–config-name* switches control to which process **rec_control** connects.

13.2.3 Examples

To see if the Recursor is alive, run:

```
# rec_control ping
```

To stop the recursor by hand, run:

```
# rec_control quit
```

To dump the cache to disk, execute:

```
# rec_control dump-cache /tmp/the-cache
```

13.2.4 Options

--help provide this helpful message.

--config-dir=<path> Directory where the recursor.conf lives.

--config-name=<name> Name of the virtual configuration.

--socket-dir=<path> Where the controlsocket will live, please use **–config-dir** instead.

--socket-pid=<pid> When running in SMP mode, pid of **pdns_recursor** to control.

--timeout=<num> Number of seconds to wait for the remote PowerDNS Recursor to respond. Set to 0 for infinite.

13.2.5 Commands

add-nta *DOMAIN* [*REASON*] Add a Negative Trust Anchor for *DOMAIN*, suffixed optionally with *REASON*.

add-ta *DOMAIN DSRECORD* Add a Trust Anchor for *DOMAIN* with DS record data *DSRECORD*. This adds the new Trust Anchor to the existing set of Trust Anchors for *DOMAIN*.

current-queries Shows the currently active queries.

clear-nta *DOMAIN...* Remove Negative Trust Anchor for one or more *DOMAIN*s. Set domain to '*' to remove all NTA's.

clear-ta [*DOMAIN*]... Remove Trust Anchor for one or more *DOMAIN*s. Note that removing the root trust anchor is not possible.

dump-cache *FILENAME* Dumps the entire cache to *FILENAME*. This file should not exist already, PowerDNS will refuse to overwrite it. While dumping, the recursor will not answer questions.

Typical PowerDNS Recursors run multiple threads, therefore you'll see duplicate, different entries for the same domains. The negative cache is also dumped to the same file. The per-thread positive and negative cache dumps are separated with an appropriate comment.

Note: **`pdns_recursor`** often runs in a chroot. You can retrieve the file using:

```
rec_control dump-cache /tmp/file
mv /proc/$(pidof pdns_recursor)/root/tmp/file /tmp/filename
```

---

dump-edns *FILENAME* Dumps the EDNS status to the filename mentioned. This file should not exist already, PowerDNS will refuse to overwrite it. While dumping, the recursor will not answer questions.

Note: **`pdns_recursor`** often runs in a chroot. You can retrieve the file using:

```
rec_control dump-edns /tmp/file
mv /proc/$(pidof pdns_recursor)/root/tmp/file /tmp/filename
```

---

dump-nsspeeds *FILENAME* Dumps the nameserver speed statistics to the *FILENAME* mentioned. This file should not exist already, PowerDNS will refuse to overwrite it. While dumping, the recursor will not answer questions. Statistics are kept per thread, and the dumps end up in the same file.

Note: **`pdns_recursor`** often runs in a chroot. You can retrieve the file using:

```
rec_control dump-nsspeeds /tmp/file
mv /proc/$(pidof pdns_recursor)/root/tmp/file /tmp/filename
```

---

dump-rpz *ZONE NAME FILE NAME* Dumps the content of the RPZ zone named *ZONE NAME* to the *FILENAME* mentioned. This file should not exist already, PowerDNS will refuse to overwrite it otherwise. While dumping, the recursor will not answer questions.

Note: **`pdns_recursor`** often runs in a chroot. You can retrieve the file using:

```
rec_control dump-rpz ZONE_NAME /tmp/file
mv /proc/$(pidof pdns_recursor)/root/tmp/file /tmp/filename
```

---

get *STATISTIC* [*STATISTIC*]... Retrieve a statistic. For items that can be queried, see *Metrics and Statistics*

get-all Retrieve all known statistics.

get-ntas Get a list of the currently configured Negative Trust Anchors.

get-tas Get a list of the currently configured Trust Anchors.

get-parameter *KEY* [*KEY*]... Retrieves the specified configuration parameter(s).

get-qtypelist Retrieves QType statistics. Queries from cache aren't being counted yet.

help Shows a list of supported commands understood by the running **`pdns_recursor`**

ping Check if server is alive.

quit Request shutdown of the recursor.

quit-nicely Request nice shutdown of the recursor.

reload-acls Reloads ACLs.

reload-lua-script [*FILENAME*] (Re)loads Lua script *FILENAME*. If *FILENAME* is empty, attempt to reload the currently loaded script. This replaces the script currently loaded.

reload-lua-config [*FILENAME*] (Re)loads Lua configuration *FILENAME*. If *FILENAME* is empty, attempt to reload the currently loaded file. Note that *FILENAME* will be fully executed, any settings changed at runtime that are not modified in this file, will still be active. Reloading RPZ, especially by AXFR, can take some time; during which the recursor will not answer questions.

reload-zones Reload authoritative and forward zones. Retains current configuration in case of errors.

set-carbon-server *CARBON SERVER* [*CARBON OURNAME*] Set the carbon-server setting to *CARBON SERVER*. If *CARBON OURNAME* is not empty, also set the carbon-ourname setting to *CARBON OURNAME*.

set-dnssec-log-bogus *SETTING* Set dnssec-log-bogus setting to *SETTING*. Set to 'on' or 'yes' to log DNSSEC validation failures and to 'no' or 'off' to disable logging these failures.

set-max-cache-entries *NUM* Change the maximum number of entries in the DNS cache. If reduced, the cache size will start shrinking to this number as part of the normal cache purging process, which might take a while.

set-max-packetcache-entries *NUM* Change the maximum number of entries in the packet cache. If reduced, the cache size will start shrinking to this number as part of the normal cache purging process, which might take a while.

set-minimum-ttl *NUM* Set minimum-ttl-override to *NUM*.

top-queries Shows the top-20 queries. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-pub-queries Shows the top-20 queries grouped by public suffix list. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-largeanswer-remotes Shows the top-20 remote hosts causing large answers. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-remotes Shows the top-20 most active remote hosts. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-servfail-queries Shows the top-20 queries causing servfail responses. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-bogus-queries Shows the top-20 queries causing bogus responses. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-pub-servfail-queries Shows the top-20 queries causing servfail responses grouped by public suffix list. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-pub-bogus-queries Shows the top-20 queries causing bogus responses grouped by public suffix list. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-servfail-remotes Shows the top-20 most active remote hosts causing servfail responses. Statistics are over the last 'stats-ringbuffer-entries' queries.

top-bogus-remotes Shows the top-20 most active remote hosts causing bogus responses. Statistics are over the last 'stats-ringbuffer-entries' queries.

trace-regex *REGEX* Emit resolution trace for matching queries. Empty regex to disable trace.

Queries matching this regular expression will generate voluminous tracing output. Be aware that matches from the packet cache will still not generate tracing. To unset the regex, pass **trace-regex** without a new regex.

The regular expression is matched against domain queries terminated with a '.'. For example the regex 'powerdns.com$' will not match a query for 'www.powerdns.com', since the attempted match will be with 'www.powerdns.com.'.

In addition, since this is a regular expression, to exclusively match queries for 'www.powerdns.com', one should escape the dots: '^www.powerdns.com.$'.

Multiple matches can be chained with the 'l' operator. For example, to match all queries for Dutch (.nl) and German (.de) domain names, use: '.nl.$l.de.$'.

unload-lua-script Unloads Lua script if one was loaded.

version Report running version.

wipe-cache ***DOMAIN*** **[*DOMAIN*] [...]** Wipe entries for *DOMAIN* (exact name match) from the cache. This is useful if, for example, an important server has a new IP address, but the TTL has not yet expired. Multiple domain names can be passed. *DOMAIN* can be suffixed with a '$'. to delete the whole tree from the cache. i.e. 'powerdns.com$' will remove all cached entries under and including the powerdns.com name.

Note: this command also wipes the negative cache.

Warning: Don't just wipe "www.somedomain.com", its NS records or CNAME target may still be undesired, so wipe "somedomain.com" as well.

13.2.6 See also

pdns_recursor(1)

CHAPTER

FOURTEEN

SECURITY ADVISORIES

All security advisories for the PowerDNS Recursor are listed here.

14.1 PowerDNS Security Advisory 2006-01: Malformed TCP queries can lead to a buffer overflow which might be exploitable

- CVE: CVE-2006-4251
- Date: 13th of November 2006
- Affects: PowerDNS Recursor versions 3.1.3 and earlier, on all operating systems.
- Not affected: No versions of the PowerDNS Authoritative Server ('pdns_server') are affected.
- Severity: Critical
- Impact: Potential remote system compromise.
- Exploit: As far as we know, no exploit is available as of 11th of November 2006.
- Solution: Upgrade to PowerDNS Recursor 3.1.4, or apply the patches referred below and recompile
- Workaround: Disable TCP access to the Recursor. This will have slight operational impact, but it is likely that this will not lead to meaningful degradation of service. Disabling access is best performed at packet level, either by configuring a firewall, or instructing the host operating system to drop TCP connections to port 53. Additionally, exposure can be limited by configuring the `allow-from` setting so only trusted users can query your nameserver.

PowerDNS Recursor 3.1.3 and previous miscalculate the length of incoming TCP DNS queries, and will attempt to read up to 4 gigabytes of query into a 65535 byte buffer.

We have not verified if this problem might actually lead to a system compromise, but are acting on the assumption that it might.

For distributors, a minimal patch is available on the PowerDNS wiki. Additionally, those shipping very old versions of the PowerDNS Recursor might benefit from this patch.

The impact of these and other security problems can be lessened by considering the advice in FIXME: security-settings.

14.2 PowerDNS Security Advisory 2006-02: Zero second CNAME TTLs can make PowerDNS exhaust allocated stack space, and crash

- CVE: CVE-2006-4252

- Date: 13th of November 2006
- Affects: PowerDNS Recursor versions 3.1.3 and earlier, on all operating systems.
- Not affected: No versions of the PowerDNS Authoritative Server ('pdns_server') are affected.
- Severity: Moderate
- Impact: Denial of service
- Exploit: This problem can be triggered by sending queries for specifically configured domains
- Solution: Upgrade to PowerDNS Recursor 3.1.4, or apply commit 919.
- Workaround: None known. Exposure can be limited by configuring the **allow-from** setting so only trusted users can query your nameserver.

PowerDNS would recurse endlessly on encountering a CNAME loop consisting entirely of zero second CNAME records, eventually exceeding resources and crashing.

14.3 PowerDNS Security Advisory 2008-01: System random generator can be predicted, leading to the potential to 'spoof' PowerDNS Recursor

- CVE: Not yet assigned
- Date: 31st of March 2008
- Affects: PowerDNS Recursor versions 3.1.4 and earlier, on most operating systems
- Not affected: No versions of the PowerDNS Authoritative Server ('pdns_server') are affected.
- Severity:Moderate
- Impact: Data manipulation; client redirection
- Exploit: This problem can be triggered by sending queries for specifically configured domains, sending spoofed answer packets immediately afterwards.
- Solution: Upgrade to PowerDNS Recursor 3.1.5, or apply changesets 1159, 1160 and 1164.
- Workaround: None known. Exposure can be limited by configuring the **allow-from** setting so only trusted users can query your nameserver.

We would like to thank Amit Klein of Trusteer for bringing a serious vulnerability to our attention which would enable a smart attacker to 'spoof' previous versions of the PowerDNS Recursor into accepting possibly malicious data.

Details can be found on this Trusteer page.

This security problem was announced in this email message.

It is recommended that all users of the PowerDNS Recursor upgrade to 3.1.5 as soon as practicable, while we simultaneously note that busy servers are less susceptible to the attack, but not immune.

The vulnerability is present on all operating systems where the behaviour of the libc random() function can be predicted based on its past output. This includes at least all known versions of Linux, as well as Microsoft Windows, and probably FreeBSD and Solaris.

The magnitude of this vulnerability depends on internal details of the system random() generator. For Linux, the mathematics of the random generator are complex, but well understood and Amit Klein has written and published a proof of concept that can successfully predict its output after uninterrupted observation of 40-50 DNS queries.

Because the observation needs to be uninterrupted, busy PowerDNS Recursor instances are harder to subvert - other data is highly likely to be interleaved with traffic generated by an attacker.

Nevertheless, operators are urged to update at their earliest convenience.

14.4 PowerDNS Security Advisory 2010-01: PowerDNS Recursor up to and including 3.1.7.1 can be brought down and probably exploited

- CVE: CVE-2009-4009
- Date: 6th of January 2010
- Affects: PowerDNS Recursor 3.1.7.1 and earlier
- Not affected: No versions of the PowerDNS Authoritative ('pdns_server') are affected.
- Severity: Critical
- Impact: Denial of Service, possible full system compromise
- Exploit: Withheld
- Solution: Upgrade to PowerDNS Recursor 3.1.7.2 or higher
- Workaround: None. The risk of exploitation or denial of service can be decreased slightly by using the `allow-from` setting to only provide service to known users. The risk of a full system compromise can be reduced by running with a suitable reduced privilege user and group settings, and possibly chroot environment.

Using specially crafted packets, it is possible to force a buffer overflow in the PowerDNS Recursor, leading to a crash.

This vulnerability was discovered by a third party that (for now) prefers not to be named. PowerDNS is very grateful however for their help in improving PowerDNS security.

14.5 PowerDNS Security Advisory 2010-02: PowerDNS Recursor up to and including 3.1.7.1 can be spoofed into accepting bogus data

- CVE: CVE-2009-4010
- Date: 6th of January 2010
- Affects: PowerDNS Recursor 3.1.7.1 and earlier
- Not affected: No versions of the PowerDNS Authoritative ('pdns_server') are affected.
- Severity: High
- Impact: Using smart techniques, it is possible to fool the PowerDNS Recursor into accepting unauthorized data
- Exploit: Withheld
- Solution: Upgrade to PowerDNS Recursor 3.1.7.2 or higher
- Workaround: None.

Using specially crafted zones, it is possible to fool the PowerDNS Recursor into accepting bogus data. This data might be harmful to your users. An attacker would be able to divert data from, say, bigbank.com to an IP address of his choosing.

This vulnerability was discovered by a third party that (for now) prefers not to be named. PowerDNS is very grateful however for their help in improving PowerDNS security.

14.6 PowerDNS Security Advisory 2014-01: PowerDNS Recursor 3.6.0 can be crashed remotely

- CVE: CVE-2014-3614
- Date: 10th of September 2014
- Credit: Dedicated PowerDNS users willing to study a crash that happens once every few months (thanks)
- Affects: Only PowerDNS Recursor version 3.6.0.
- Not affected: No other versions of PowerDNS Recursor, no versions of PowerDNS Authoritative Server
- Severity: High
- Impact: Crash
- Exploit: The sequence of packets required is known
- Risk of system compromise: No
- Solution: Upgrade to PowerDNS Recursor 3.6.1
- Workaround: Restrict service using `allow-from <../recursor/settings.md#allow-from>‘__, install script that restarts PowerDNS

Recently, we've discovered that PowerDNS Recursor 3.6.0 (but NOT earlier) can crash when exposed to a specific sequence of malformed packets. This sequence happened spontaneously with one of our largest deployments, and the packets did not appear to have a malicious origin.

Yet, this crash can be triggered remotely, leading to a denial of service attack. There appears to be no way to use this crash for system compromise or stack overflow.

Upgrading to 3.6.1 solves the issue.

In addition, if you want to apply a minimal fix to your own tree, it can be found here

As for workarounds, only clients in allow-from are able to trigger the crash, so this should be limited to your userbase. Secondly, this and this can be used to enable Upstart and Systemd to restart the PowerDNS Recursor automatically.

14.7 PowerDNS Security Advisory 2014-02: PowerDNS Recursor 3.6.1 and earlier can be made to provide bad service

- CVE: CVE-2014-8601
- Date: 8th of December 2014
- Credit: Florian Maury (ANSSI)
- Affects: PowerDNS Recursor versions 3.6.1 and earlier
- Not affected: PowerDNS Recursor 3.6.2; no versions of PowerDNS Authoritative Server
- Severity: High
- Impact: Degraded service
- Exploit: This problem can be triggered by sending queries for specifically configured domains
- Risk of system compromise: No
- Solution: Upgrade to PowerDNS Recursor 3.6.2
- Workaround: None known. Exposure can be limited by configuring the **allow-from** setting so only trusted users can query your nameserver.

Recently we released PowerDNS Recursor 3.6.2 with a new feature that strictly limits the amount of work we'll perform to resolve a single query. This feature was inspired by performance degradations noted when resolving domains hosted by 'ezdns.it', which can require thousands of queries to resolve.

During the 3.6.2 release process, we were contacted by a government security agency with news that they had found that all major caching nameservers, including PowerDNS, could be negatively impacted by specially configured, hard to resolve domain names. With their permission, we continued the 3.6.2 release process with the fix for the issue already in there.

We recommend that all users upgrade to 3.6.2 if at all possible. Alternatively, if you want to apply a minimal fix to your own tree, it can be found here, including patches for older versions.

As for workarounds, only clients in allow-from are able to trigger the degraded service, so this should be limited to your userbase.

14.8 PowerDNS Security Advisory 2015-01: Label decompression bug can cause crashes or CPU spikes

- CVE: CVE-2015-1868 (original), CVE-2015-5470 (update)
- Date: 23rd of April 2015, updated 7th of July 2015
- Credit: Aki Tuomi, Toshifumi Sakaguchi
- Affects: PowerDNS Recursor versions 3.5 and up; Authoritative Server 3.2 and up
- Not affected: Recursor 3.6.4; Recursor 3.7.3; Auth 3.3.3; Auth 3.4.5
- Severity: High
- Impact: Degraded service
- Exploit: This problem can be triggered by sending queries for specifically configured domains, or by sending specially crafted query packets
- Risk of system compromise: No
- Solution: Upgrade to any of the non-affected versions
- Workaround: Run your Recursor under a supervisor. Exposure can be limited by configuring the `allow-from <../recursor/settings.md#allow-from>`__ setting so only trusted users can query your nameserver. There is no workaround for the Authoritative server.

A bug was discovered in our label decompression code, making it possible for names to refer to themselves, thus causing a loop during decompression. On some platforms, this bug can be abused to cause crashes. On all platforms, this bug can be abused to cause service-affecting CPU spikes.

We recommend that all users upgrade to a corrected version if at all possible. Alternatively, if you want to apply a minimal fix to your own tree, please find patches here.

As for workarounds, for the Recursor: only clients in allow-from are able to trigger the degraded service, so this should be limited to your userbase; further, we recommend running your critical services under supervision such as systemd, supervisord, daemontools, etc.

There is no workaround for the Authoritative Server.

We want to thank Aki Tuomi for noticing this in production, and then digging until he got to the absolute bottom of what at the time appeared to be a random and spurious failure.

We want to thank Toshifumi Sakaguchi for further investigation into the issue after the initial announcement, and for demonstrating to us quite clearly the CPU spike issues.

Update 7th of July 2015: Toshifumi Sakaguchi discovered that the original fix was insufficient in some cases. Updated versions of the Authoritative Server and Recursor were released on the 9th of June. Minimal patches are available. The insufficient fix was assigned CVE-2015-5470.

14.9 PowerDNS Security Advisory 2016-02: Crafted queries can cause abnormal CPU usage

- CVE: CVE-2016-7068
- Date: December 15th 2016
- Credit: Florian Heinz and Martin Kluge
- Affects: PowerDNS Authoritative Server up to and including 3.4.10, 4.0.1, PowerDNS Recursor up to and including 3.7.3, 4.0.3
- Not affected: PowerDNS Authoritative Server 3.4.11, 4.0.2 and PowerDNS Recursor 3.7.4, 4.0.4
- Severity: Medium
- Impact: Degraded service or Denial of service
- Exploit: This issue can be triggered by sending specially crafted query packets
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: Run dnsdist with the rules provided below in front of potentially affected servers.

An issue has been found in PowerDNS allowing a remote, unauthenticated attacker to cause an abnormal CPU usage load on the PowerDNS server by sending crafted DNS queries, which might result in a partial denial of service if the system becomes overloaded. This issue is based on the fact that the PowerDNS server parses all records present in a query regardless of whether they are needed or even legitimate. A specially crafted query containing a large number of records can be used to take advantage of that behaviour. This issue has been assigned CVE-2016-7068.

PowerDNS Authoritative Server up to and including 3.4.10 and 4.0.1 are affected. PowerDNS Recursor up to and including 3.7.3 and 4.0.3 are affected.

dnsdist can be used to block crafted queries, using `RecordsCountRule()` and `RecordsTypeCountRule()` to block queries with crafted records.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Florian Heinz and Martin Kluge for finding and subsequently reporting this issue.

14.10 PowerDNS Security Advisory 2016-04: Insufficient validation of TSIG signatures

- CVE: CVE-2016-7073 CVE-2016-7074
- Date: December 15th 2016
- Credit: Mongo
- Affects: PowerDNS Authoritative Server up to and including 3.4.10, 4.0.1, PowerDNS Recursor from 4.0.0 and up to and including 4.0.3
- Not affected: PowerDNS Authoritative Server 3.4.11, 4.0.2, PowerDNS Recursor < 4.0.0, 4.0.4
- Severity: Medium
- Impact: Zone content alteration
- Exploit: This problem can be triggered by an attacker in position of man-in-the-middle
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

Two issues have been found in PowerDNS Authoritative Server allowing an attacker in position of man-in-the-middle to alter the content of an AXFR because of insufficient validation of TSIG signatures. The first issue is a missing check of the TSIG time and fudge values in `AXFRRetriever`, leading to a possible replay attack. This issue has been assigned CVE-2016-7073. The second issue is a missing check that the TSIG record is the last one, leading to the possibility of parsing records that are not covered by the TSIG signature. This issue has been assigned CVE-2016-7074.

PowerDNS Authoritative Server up to and including 3.4.10 and 4.0.1 are affected. PowerDNS Recursor from 4.0.0 up to and including 4.0.3 are affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Mongo for finding and subsequently reporting this issue.

14.11 PowerDNS Security Advisory 2017-03: Insufficient validation of DNSSEC signatures

- CVE: CVE-2017-15090
- Date: November 27th 2017
- Credit: Kees Monshouwer
- Affects: PowerDNS Recursor from 4.0.0 and up to and including 4.0.6
- Not affected: PowerDNS Recursor < 4.0.0, 4.0.7
- Severity: Medium
- Impact: Records manipulation
- Exploit: This problem can be triggered by an attacker in position of man-in-the-middle
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in the DNSSEC validation component of PowerDNS Recursor, where the signatures might have been accepted as valid even if the signed data was not in bailiwick of the DNSKEY used to sign it. This allows an attacker in position of man-in-the-middle to alter the content of records by issuing a valid signature for the crafted records. This issue has been assigned CVE-2017-15090.

PowerDNS Recursor from 4.0.0 up to and including 4.0.6 are affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Kees Monshouwer for finding and subsequently reporting this issue.

14.12 PowerDNS Security Advisory 2017-05: Cross-Site Scripting in the web interface

- CVE: CVE-2017-15092
- Date: November 27th 2017
- Credit: Nixu, Chris Navarrete of Fortinet's Fortiguard Labs
- Affects: PowerDNS Recursor from 4.0.0 up to and including 4.0.6
- Not affected: PowerDNS Recursor 4.0.7, 3.7.x
- Severity: Medium
- Impact: Alteration and denial of service of the web interface

- Exploit: This problem can be triggered by an attacker sending DNS queries to the server
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in the web interface of PowerDNS Recursor, where the qname of DNS queries was displayed without any escaping, allowing a remote attacker to inject HTML and Javascript code into the web interface, altering the content. This issue has been assigned CVE-2017-15092.

PowerDNS Recursor from 4.0.0 up to and including 4.0.6 are affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Nixu and Chris Navarrete of Fortinet's Fortiguard Labs for independently finding and reporting this issue.

14.13 PowerDNS Security Advisory 2017-06: Configuration file injection in the API

- CVE: CVE-2017-15093
- Date: November 27th 2017
- Credit: Nixu
- Affects: PowerDNS Recursor up to and including 4.0.6, 3.7.4
- Not affected: PowerDNS Recursor 4.0.7
- Severity: Medium
- Impact: Alteration of configuration by an API user
- Exploit: This problem can be triggered by an attacker with valid API credentials
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: Disable the ability to alter the configuration via the API by setting *api-config-dir* to an empty value (default), or set the API read-only via the *api-readonly* setting.

An issue has been found in the API of PowerDNS Recursor during a source code audit by Nixu. When *api-config-dir* is set to a non-empty value, which is not the case by default, the API allows an authorized user to update the Recursor's ACL by adding and removing netmasks, and to configure forward zones. It was discovered that the new netmask and IP addresses of forwarded zones were not sufficiently validated, allowing an authenticated user to inject new configuration directives into the Recursor's configuration. This issue has been assigned CVE-2017-15093.

PowerDNS Recursor up to and including 4.0.6 and 3.7.4 are affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Nixu for finding and subsequently reporting this issue.

14.14 PowerDNS Security Advisory 2017-07: Memory leak in DNSSEC parsing

- CVE: CVE-2017-15094
- Date: November 27th 2017
- Credit: Nixu

- Affects: PowerDNS Recursor from 4.0.0 up to and including 4.0.6
- Not affected: PowerDNS Recursor 4.0.7
- Severity: Medium
- Impact: Denial of service
- Exploit: This problem can be triggered by an authoritative server sending crafted ECDSA DNSSEC keys to the Recursor.
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: Disable DNSSEC validation by setting the *dnssec* parameter to *off* or *process-no-validate* (default).

An issue has been found in the DNSSEC parsing code of PowerDNS Recursor during a code audit by Nixu, leading to a memory leak when parsing specially crafted DNSSEC ECDSA keys. These keys are only parsed when validation is enabled by setting *dnssec* to a value other than *off* or *process-no-validate* (default). This issue has been assigned CVE-2017-15094.

PowerDNS Recursor from 4.0.0 up to and including 4.0.6 are affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Nixu for finding and subsequently reporting this issue.

14.15 PowerDNS Security Advisory 2017-08: Crafted CNAME answer can cause a denial of service

- CVE: CVE-2017-15120
- Date: December 11th 2017
- Credit: Toshifumi Sakaguchi
- Affects: PowerDNS Recursor from 4.0.0 up to and including 4.0.7
- Not affected: PowerDNS Recursor 3.7.4, 4.0.8, 4.1.0
- Severity: High
- Impact: Denial of service
- Exploit: This problem can be triggered by an authoritative server sending a crafted CNAME answer with a class other than IN to the Recursor.
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version
- Workaround: run the process inside a supervisor like supervisord or systemd

An issue has been found in the parsing of authoritative answers in PowerDNS Recursor, leading to a NULL pointer dereference when parsing a specially crafted answer containing a CNAME of a different class than IN. This issue has been assigned CVE-2017-15120.

When the PowerDNS Recursor is run inside a supervisor like supervisord or systemd, it will be automatically restarted, limiting the impact to somewhat degraded service.

PowerDNS Recursor from 4.0.0 up to and including 4.0.7 are affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank Toshifumi Sakaguchi for finding and subsequently reporting this issue.

14.16 PowerDNS Security Advisory 2018-01: Insufficient validation of DNSSEC signatures

- CVE: CVE-2018-1000003
- Date: January 22nd 2018
- Credit: CZ.NIC
- Affects: PowerDNS Recursor 4.1.0
- Not affected: PowerDNS Recursor < 4.1.0, 4.1.1
- Severity: Low
- Impact: Denial of existence spoofing
- Exploit: This problem can be triggered by an attacker in position of man-in-the-middle
- Risk of system compromise: No
- Solution: Upgrade to a non-affected version

An issue has been found in the DNSSEC validation component of PowerDNS Recursor, allowing an ancestor delegation NSEC or NSEC3 record to be used to wrongfully prove the non-existence of a RR below the owner name of that record. This would allow an attacker in position of man-in-the-middle to send a NXDOMAIN answer for a name that does exist. This issue has been assigned CVE-2018-1000003.

PowerDNS Recursor 4.1.0 is affected.

For those unable to upgrade to a new version, a minimal patch is available

We would like to thank CZ.NIC for finding and subsequently reporting this issue! Please also see https://lists.nic.cz/pipermail/knot-dns-users/2018-January/001309.html

14.17 Older security advisories

Version 3.0 of the PowerDNS recursor contains a denial of service bug which can be exploited remotely. This bug, which we believe to only lead to a crash, has been fixed in 3.0.1. There are no guarantees however, so an upgrade from 3.0 is highly recommended.

All versions of PowerDNS before 2.9.21.1 do not respond to certain queries. This in itself is not a problem, but since the discovery by Dan Kaminsky of a new spoofing technique, this silence for queries PowerDNS considers invalid, within a valid domain, allows attackers more chances to feed *other* resolvers bad data.

All versions of PowerDNS before 2.9.18 contain the following two bugs, which only apply to installations running with the LDAP backend, or installations providing recursion to a limited range of IP addresses. If any of these apply to you, an upgrade is highly advised:

- The LDAP backend did not properly escape all queries, allowing it to fail and not answer questions. We have not investigated further risks involved, but we advise LDAP users to update as quickly as possible (Norbert Sendetzky, Jan de Groot)
- Questions from clients denied recursion could blank out answers to clients who are allowed recursion services, temporarily. Reported by Wilco Baan. This would've made it possible for outsiders to blank out a domain temporarily to your users. Luckily PowerDNS would send out SERVFAIL or Refused, and not a denial of a domain's existence.

All versions of PowerDNS before 2.9.17 are known to suffer from remote denial of service problems which can disrupt operation. Please upgrade to 2.9.17 as this page will only contain detailed security information from 2.9.17 onwards.

CHAPTER

FIFTEEN

UPGRADE GUIDE

Before upgrading, it is advised to read the *Changelogs*. When upgrading several versions, please read **all** notes applying to the upgrade.

15.1 4.1.x to 4.2.0 or master

Two new settings have been added:

- *xpf-allow-from* can contain a list of IP addresses ranges from which XPF (X-Proxied-For) records will be trusted.
- *xpf-rr-code* should list the number of the XPF record to use (in lieu of an assigned code).

15.2 4.0.x to 4.1.0

loglevel defaulted to 4 but was always overridden to 6 during the startup. The issue has been fixed and the default value set to 6 to keep the behavior consistent.

The `--enable-libsodium` configure flag has changed from 'no' to 'auto'. This means that if libsodium and its development header are installed, it will be linked in.

15.3 4.0.3 to 4.0.4

One setting has been added to limit the risk of overflowing the stack:

- *max-recursion-depth*: defaults to 40 and was unlimited before

15.4 4.0.0 to 4.0.1

Two settings have changed defaults, these new defaults decrease CPU usage:

- *root-nx-trust* changed from "no" to "yes"
- *log-common-errors* changed from "yes" to "no"

CHAPTER

SIXTEEN

CHANGELOGS

The changelogs for the recursor are split between release trains.

16.1 Changelogs for 4.1.x

16.1.1 4.1.3

Released: 22nd of May 2018

This release improves the stability and resiliency of the RPZ implementation, prevents metrics gathering from slowing down the processing of DNS queries and fixes an issue related to the cleaning of EDNS Client Subnet entries from the cache.

Improvements

- Move carbon/webserver/control/stats handling to a separate thread. ¶ References: pull request 6567
- Use a separate, non-blocking pipe to distribute queries. ¶ References: pull request 6566
- Add a subtree option to the *API* cache flush endpoint. ¶ References: #6550, pull request 6562
- Update copyright years to 2018 (Matt Nordhoff). ¶ References: #6610, #6130, pull request 6611
- Fix a warning on botan >= 2.5.0. ¶ References: #6474, pull request 6596, pull request 6478
- Add `_raw` versions for `QName` / `ComboAddresses` to the `FFI` API. ¶ References: pull request 6583

Bug Fixes

- Respect the `AXFR` timeout while connecting to the `RPZ` server. ¶ References: pull request 6469
- Don't increase the `DNSSEC` validations counters when running with `process-no-validate`. ¶ References: pull request 6467
- Count a lookup into an internal auth zone as a cache miss. ¶ References: pull request 6313
- Delay the loading of `RPZ` zones until the parsing is done, fixing a race condition. ¶ References: #6237, pull request 6588
- Reorder includes to avoid boost `L` conflict. ¶ References: #6517, #6516, #6358, #6542, pull request 6595
- Use canonical ordering in the `ECS` index.

  ¶ References: #6505, pull request 6586
- Add `-rdynamic` to `C{,XX}FLAGS` when we build with `LuaJIT`. ¶ References: pull request 6514, pull request 6630
- Increase `MTasker` stacksize to avoid crash in exception unwinding (Chris Hofstaedtler). ¶ References: #6179, pull request 6418

- Use the SyncRes time in our unit tests when checking cache validity (Chris Hofstaedtler). ¶ References: #6086, pull request 6419

16.1.2 4.1.2

Released: 29th of March 2018

This release improves the stability and resiliency of the RPZ implementation and fixes several issues related to EDNS Client Subnet.

New Features

- Add FFI version of `gettag()`. ¶ References: pull request 6344

Improvements

- Add the option to set the AXFR timeout for RPZs. ¶ References: pull request 6290, pull request 6268, pull request 6303, pull request 6298
- IXFR: correct behavior of dealing with DNS Name with multiple records and speed up IXFR transaction (Leon Xu). ¶ References: pull request 6172
- Add *RPZ statistics endpoint* to the *API*. ¶ References: #6225, pull request 6379

Bug Fixes

- Retry loading RPZ zones from server when they fail initially. ¶ References: #6238, pull request 6336, pull request 6237, pull request 6293
- Fix ECS-based cache entry refresh code. ¶ References: pull request 6300
- Fix ECS-specific NS AAAA not being returned from the cache. ¶ References: #6319, pull request 6320

16.1.3 4.1.1

Released: 22nd of January 2018

This is the second release in the 4.1 train.

This release fixes PowerDNS Security Advisory *2018-01*.

The full release notes can be read on the blog.

This is a release on the stable branch, containing a fix for the abovementioned security issue and several bug fixes from the development branch.

Improvements

- Don't process records for another class than IN. We don't use records of another class than IN, but we used to store some of them in the cache which is useless. Just skip them. ¶ References: #6198, pull request 6085

Bug Fixes

- Correctly handle ancestor delegation NSEC{,3} for children. Fixes the DNSSEC validation issue found in Knot Resolver, where a NSEC{3} ancestor delegation is wrongly use to prove the non-existence of a RR below the delegation. We already had the correct check for the exact owner name, but not for RRs below the delegation. (Security Advisory *2018-01*) ¶ References: pull request 6215

- Fix the computation of the closest encloser for positive answers. When the positive answer is expanded from a wildcard with NSEC3, the closest encloser is not always parent of the qname, depending on the number of labels in the initial wildcard. ¶ References: #6199, pull request 6092
- Pass the correct buffer size to `arecvfrom()`. The incorrect size could possibly cause DNSSEC failures. ¶ References: #6200, pull request 6095
- Fix to make `primeHints` threadsafe, otherwise there's a small chance on startup that the root-server IPs will be incorrect. ¶ References: #6212, pull request 6209
- Don't validate signature for "glue" CNAME, since anything else than the initial CNAME can't be considered authoritative. ¶ References: #6201, pull request 6137

16.1.4 4.1.0

Released: 4th of December 2017

This is the first release in the 4.1 train.

The full release notes can be read on the blog.

This is a major release containing significant speedups (both in throughput and latency), enhanced capabilities and a highly conformant and robust DNSSEC validation implementation that is ready for heavy production use. In addition, our EDNS Client Subnet implementation now scales effortlessly to networks needing very fine grained scopes (as used by some 'country sized' service providers).

- Improved DNSSEC support,
- Improved documentation,
- Improved RPZ support,
- Improved EDNS Client Subnet support,
- Support for Botan 2.x (and removal of support for Botan 1.10),
- SNMP support,
- Lua engine has gained access to more parts of the recursor,
- CPU affinity can now be specified,
- TCP Fast Open support,
- New performance metrics.

Changes since 4.1.0-rc3:

Bug Fixes

- Dump the validation status of negcache entries, fix DNSSEC type. ¶ References: pull request 5972
- Fix DNSSEC validation of DS denial from the negative cache. ¶ References: pull request 5978
- Store additional records as non-auth, even on AA=1 answers. ¶ References: pull request 5997
- Don't leak when the loading a public ECDSA key fails. ¶ References: pull request 6008
- When validating DNSKeys, the zone should be part of the signer. ¶ References: pull request 6009
- Cache Secure validation state when inserting negcache entries. ¶ References: pull request 5980

16.1.5 4.1.0-rc3

Released: 17th of November 2017

The third Release Candidate adds support for Botan 2.x (and removes support for Botan 1.10!), has a lot of DNSSEC fixes, features a cleaned up web UI and has miscellaneous minor improvements.

Improvements

- Add the DNSSEC validation state to the `DNSQuestion` Lua object (although the ability to update the validation state from these hooks is postponed to after 4.1.0). ¶ References: #5888, pull request 5895
- Add support for Botan 2.x and remove support for Botan 1.10. ¶ References: #5797, #2250, pull request 5498
- Print more details of trust anchors. In addition, the *trace* output that mentions if data from authoritative servers gets accepted now also prints the TTL and clarifies the 'place' number previously printed. ¶ References: pull request 5876
- Better support for deleting entries in `NetmaskTree` and `NetmaskGroup`. ¶ References: pull request 5616

Bug Fixes

- Prevent possible downgrade attacks in the recursor. ¶ References: pull request 5889
- Split NODATA / NXDOMAIN NSEC wildcard denial proof of existence. Otherwise there is a very real risk that a NSEC will cover a more specific wildcard and we end up with what looks like a NXDOMAIN proof but is a NODATA one. ¶ References: #5882, pull request 5885
- Fix incomplete validation of cached entries. ¶ References: pull request 5904
- Fix going Insecure on NSEC3 hashes with too many iterations, since we could have gone Bogus on a positive answer synthetized from a wildcard if the corresponding NSEC3 had more iterations that we were willing to accept, while the correct result is Insecure. ¶ References: pull request 5912
- Sort NS addresses by speed and remove old ones. ¶ References: #1066, pull request 5877
- Purge `nsSpeeds` entries even if we get less than 2 new entries. ¶ References: pull request 5896
- Add EDNS to truncated, servfail answers. ¶ References: #5618, pull request 5881
- Use `_exit()` when we really really want to exit, for example after a fatal error. This stops us dying while we die. A call to `exit()` will trigger destructors, which may paradoxically stop the process from exiting, taking down only one thread, but harming the rest of the process. ¶ References: pull request 5917
- In the recursor secpoll code, we assumed the TXT record would be the first record first record we received. Sometimes it was the RRSIG, leading to a silent error, and no secpoll check. Fixed the assumption, added an error. ¶ References: pull request 5930
- Don't crash when asked to run with zero threads. ¶ References: pull request 5938
- Only accept types not matching the query if we asked for ANY. Even from forward-recurse servers. ¶ References: #5934, pull request 5939
- Allow the use of a 'self-resolving' NS if cached A / AAAA exists. Before this, we could skip a perfectly valid NS for which we had retrieved the A and / or AAAA entries, for example via a glue. ¶ References: #2758, pull request 5937
- Add the config-name argument to the definition of configname. There was a bug where the config-name parameter was not used to change the path of the config file. This meant that some commands via rec_control (e.g. reload-acls) would fail when run against a recursor which had config-name defined. The correct behaviour was present in some, but not all, definitions of configname. (@jake2184) ¶ References: pull request 5961

16.1.6 4.1.0-rc2

Released: 30th of October 2017

The second Release Candidate contains several correctness fixes for DNSSEC, mostly in the area of verifying negative responses.

Improvements

- Don't directly store NSEC3 records in the positive cache. ¶ References: pull request 5834
- Improve logging for the built-in *webserver* and the *Carbon* sender. ¶ References: pull request 5805
- New b.root ipv4 address (Kees Monshouwer). ¶ References: #5663, pull request 5824
- Add *experimental metrics* that track the time spent inside PowerDNS per query. These metrics ignore time spent waiting for the network. ¶ References: pull request 5774
- Add *log-timestamp* setting. This option can be used to disable printing timestamps to stdout, this is useful when using `systemd-journald` or another supervisor that timestamps output by itself. ¶ References: pull request 5842

Bug Fixes

- Check that the NSEC covers an empty non-terminal when looking for NODATA. ¶ References: pull request 5808
- Disable validation for infrastructure queries (e.g. when recursing for a name). Also validate entries from the Negative cache if they were not validated before. ¶ References: #5827, pull request 5835
- Fix DNSSEC validation for denial of wildcards in negative answers and denial of existence proofs in wildcard-expanded positive responses. ¶ References: #5861, pull request 5868
- Fix DNSSEC validation when using `-flto`. ¶ References: pull request 5873
- Lowercase all outgoing qnames when *lowercase-outgoing* is set. ¶ References: pull request 5740
- Create *socket-dir* from the init-script. ¶ References: #5439, pull request 5762
- Fix crashes with uncaught exceptions in MThreads. ¶ References: pull request 5803

16.1.7 4.1.0-rc1

Released: 9th of October 2017

The RC1 release features many fixes to the DNSSEC validation code, reported by different users. Other improvements include: logging, RPZ and the Remote Logger.

While not specifically mentioned in the ChangeLog, also thanks to Winfried Angele for bringing a documentation issue to our attention!

Improvements

- Improve `--quiet=false` output to include DNSSEC and more timing details. ¶ References: pull request 5756
- Add DNSSEC test vectors for RSA, ECDSA, ed25519 and GOST. ¶ References: pull request 5733
- Wrap the webserver's and Resolver::tryGetSOASerial objects into smart pointers (also thanks to Christian Hofstaedtler for reviewing!) ¶ References: pull request 5543
- Add more unit tests for the NetmaskTree and ECS cache index. ¶ References: pull request 5545
- Switch the default webserver's ACL to `127.0.0.1, ::1`. ¶ References: pull request 5588

- Add help text on autodetecting systemd support. (Ruben Kerkhof thanks for reporting!) ¶ References: #5524, pull request 5598
- Add `log-rpz-changes` to log RPZ additions and removals. ¶ References: pull request 5622
- Log the policy type (QName, Client IP, NS IP...) over protobuf. ¶ References: pull request 5621
- Remove unused SortList compare operator for ComboAddress. ¶ References: pull request 5637
- Add support for dumping the in-memory RPZ zones to a file. ¶ References: pull request 5620
- Support for identifying devices by id such as mac address. ¶ References: pull request 5646
- Implement dynamic cache sizeing. ¶ References: pull request 5699
- Improve dnsbulktest experience in Travis for more robustness. ¶ References: pull request 5755
- Set `TC=1` if we had to omit part of the AUTHORITY section. ¶ References: pull request 5772
- autoconf: set `--enable-libsodium` to `auto`. ¶ References: pull request 5764

Bug Fixes

- Don't fetch the DNSKEY of a zone to validate the DS of the same zone. ¶ References: pull request 5569
- Improve DNSSEC debug logging, ¶ References: pull request 5614
- Add NSEC records on nx-trust cache hits. ¶ References: #5649, pull request 5672
- Handle NSEC wrap-around. ¶ References: #5650, pull request 5671
- Fix erroneous check for section 4.1 of rfc6840. ¶ References: #5648, #5651, pull request 5670
- Handle direct NSEC queries. ¶ References: #5705, pull request 5715
- Detect zone cuts by asking for DS instead of NS. ¶ References: #5681, pull request 5716
- Do not allow direct queries for RRSIG or NSEC3. ¶ References: #5735, pull request 5738
- The target zone being insecure doesn't mean that the denial of the DS is too, if the parent zone is Secure.. ¶ References: pull request 5771
- Add a missing header for PRId64 in the negative cache, required on EL5/EL6. ¶ References: pull request 5530
- Prevent an infinite loop if we need auth and the best match is not. ¶ References: pull request 5549
- Be more careful about the validation of negative answers. ¶ References: pull request 5570
- Fix libatomic detection on ppc64. (Sander Hoentjen) ¶ References: #5456, pull request 5599
- Fix sortlist in the presence of CNAME. (Benoit Perroud thanks for reporting this issue!) ¶ References: #5357, pull request 5615
- Fix cache handling of ECS queries with a source length of 0. ¶ References: pull request 5515
- Handle SNMP alarms so we can reconnect to the master. ¶ References: #5327, pull request 5328
- Fix Recursor 4.1.0 alpha 1 compilation on FreeBSD. (@RvdE) ¶ References: pull request 5662
- Remove pdns.PASS and pdns.TRUNCATE. ¶ References: pull request 5739
- Fix a crash when getting a public GOST key if the private one is not set. ¶ References: pull request 5734
- Don't negcache entries for longer than their RRSIG validity. ¶ References: pull request 5773
- Gracefully handle Socket::accept() returning a null pointer on EAGAIN. ¶ References: pull request 5792

16.1.8 4.1.0-alpha1

Released: 18th of July 2017

This is the first release of the PowerDNS Recursor in the 4.1 release train. This release contains several performance and correctness improvements in the EDNS Client subnet area, as well as better DNSSEC processing.

New Features

- Add support for RPZ wildcarded target names. ¶ References: #5237, pull request 5265
- Add server-side TCP Fast Open support. This adds a new option *tcp-fast-open*. ¶ References: #5128, pull request 5138
- Pass `tcp` to `gettag()` to allow a script to take different actions whether a query came in over TCP or UDP. ¶ References: pull request 4569
- Allow setting the requestor ID field in the `DNSQuestion` from all hooks. ¶ References: pull request 4569
- Implement CNAME wildcards in recursor authoritative component. ¶ References: #2818, pull request 5063
- Allow returning the `DNSQuestion.data` table from `gettag()`. ¶ References: #4981, pull request 4982
- Add *SNMP* support. ¶ References: pull request 4990, pull request 5404
- Allow access to EDNS options from the `gettag()` hook. ¶ References: #5195, pull request 5198
- Pass `tcp` to `gettag()`, allow setting the requestor ID from hooks. ¶ References: pull request 4569
- Allow retrieving stats from Lua via the `getStat()` call. ¶ References: pull request 5293
- Add ECS metrics. ¶ References: pull request 5409
- Add a *cpu-map* directive to set CPU affinity per thread. ¶ References: pull request 5482

Improvements

- Implement "on-the-fly" DNSSEC processing. This places the DNSSEC processing alongside the regular recursion, reducing possible cornercases, adding unit tests and making the code better maintainable. ¶ References: #4490, #4362, #4254, #4994, pull request 5486, pull request 5223, pull request 5528, pull request 5463
- Use ECS when updating the validation state if needed. ¶ References: pull request 5484
- Use the RPZ zone's TTL and add a new *maxTTL* setting. ¶ References: pull request 5057
- RPZ updates are done zone by zone, zones are now shared pointers. ¶ References: #5236, #5231, pull request 5275, pull request 5307
- Split SyncRes::doResolveAt, add const and static whenever possible. Possibly improving performance while making the code easier to maintain. ¶ References: pull request 5106
- Packet cache speedup and cleanup. ¶ References: pull request 5102
- Make Lua mandatory for recursor builds. ¶ References: pull request 5146
- Use one listening socket per thread when reuseport is enabled. ¶ References: pull request 5103, pull request 5487
- Stop (de)serializing `DNSQuestion.data`. ¶ References: pull request 5141
- Refactor the negative cache into a class. ¶ References: pull request 5226
- Only check the netmask for subnet specific cache entries. ¶ References: pull request 5319

- Refactor and split `SyncRes::doResolveAt()`, making it easier to understand. Get rid of `SyncRes::d_nocache`, makes sure we can't get into a root refresh loop. Limit the use of global variables in SyncRes, to make it easier to understand the interaction between components ¶ References: pull request 5236
- Add an ECS index to the cache ¶ References: pull request 5472, pull request 5461
- When dumping the cache, also dump RRSIGs. ¶ References: pull request 5511
- Don't always override *loglevel* to 6. ¶ References: pull request 5485
- Make more specific Netmasks < to less specific ones. ¶ References: pull request 5530, pull request 5406

Bug Fixes

- Fix validation at the exact RRSIG inception or expiration time. ¶ References: pull request 5525
- Fix `remote`/`local` inversion in *`preoutquery()`*. ¶ References: #4969, pull request 4984
- Show a useful error when an invalid *lua-config-file* is configured. ¶ References: #4939, #5075, pull request 5078
- Fix *`DNSQuestion`* members alterations from Lua not being taken into account. ¶ References: pull request 4860
- Ensure locks can not be copied. ¶ References: pull request 5209
- Only apply *root-nx-trust* if the received SOA is ".". ¶ References: #5246, pull request 5252
- Don't throw an exception when logging to protobuf without a question set. ¶ References: pull request 5312
- Correctly truncate EDNS Client Subnetmasks. ¶ References: pull request 5320
- Clean up auth/recursor code mismatches in the API (Christian Hofstaedtler). ¶ References: #5398, pull request 5466
- Only increase `no-packet-error` on the first read. ¶ References: #5474, pull request 5474

16.2 Changelogs for 4.0.x

This page has all the changelogs for the PowerDNS Recursor 4.0 release train.

16.2.1 PowerDNS Recursor 4.0.8

Released 11th of December 2017

This release fixes PowerDNS Security Advisory *2017-08*.

Bug fixes

- #5930: Don't assume TXT record is first record for secpoll
- #6082: Don't add non-IN records to the cache

16.2.2 PowerDNS Recursor 4.0.7

Released 27th of November 2017

This release fixes PowerDNS Security Advisories *2017-03*, *2017-05*, *2017-06* and *2017-07*.

Bug fixes

- #4561: Update rec_control manpage (Winfried Angele)
- #4824: Check in the detected OpenSSL/libcrypto for ECDSA
- #5406: Make more specific Netmasks < to less specific ones
- #5525: Fix validation at the exact RRSIG inception or expiration time
- #5740: Lowercase all outgoing qnames when lowercase-outgoing is set
- #5599: Fix libatomic detection on ppc64
- #5961: Edit confignname definition to include the 'config-name' argument (Jake Reynolds)
- #5995: Security Advisories *2017-03*, *2017-05*, *2017-06* and *2017-07*.

Improvements

- #4646: Extract nested exception from Luawrapper
- #4960: Use explicit yes for default-enabled settings (Christian Hofstaedtler)
- #5078: Throw an error when lua-conf-file can't be loaded
- #5261: get-remote-ring's "other" report should only have two items. (Patrick Cloke)
- #5320: PowerDNS sdig does not truncate trailing bits of EDNS Client Subnet mask
- #5488: Only increase *no-packet-error* on the first read
- #5498: Add support for Botan 2.x
- #5511: Add more information to recursor cache dumps
- #5523: Fix typo in two log messages (Ruben Kerkhof)
- #5598: Add help text on autodetecting systemd support
- #5726: Be more resilient with broken auths
- #5739: Remove pdns.PASS and pdns.TRUNCATE
- #5755: Improve dnsbulktest experience in travis for more robustness
- #5762: Create socket-dir from init-script
- #5843: b.root renumbering, effective 2017-10-24
- #5921: Don't retry security polling too often when it fails

16.2.3 PowerDNS Recursor 4.0.6

Released 6th of July 2017

This release features a fix for the ed25519 verifier. This verifier hashed the message before verifying, resulting in unverifiable signatures. Also on the Elliptic Curve front, support was added for ED448 (DNSSEC algorithm 16) by using libdecaf.

Besides that, this release features massive improvements to our edns-client-subnet handling, and some IXFR fixes. Note that this release changes *use-incoming-edns-subnet* to disabled by default.

Bug fixes

- **commit c24288b87:** Use the incoming ECS for cache lookup if *use-incoming-edns-subnet* is set
- **commit b91dc6e92:** when making a netmask from a comboaddress, we neglected to zero the port. This could lead to a proliferation of netmasks.
- **commit 261591b6f:** Don't take the initial ECS source for a scope one if EDNS is off
- **commit 66f894b7a:** also set `d_requestor` without Lua: the ECS logic needs it
- **commit c2086f265:** Fix IXFR skipping the additions part of the last sequence
- **commit a5c9534d0:** Treat requestor's payload size lower than 512 as equal to 512
- **commit 61b1ea2f4:** make URI integers 16 bits, fixes ticket #5443
- **commit 27f9da3c2:** unbreak quoting; fixes ticket #5401

Improvements

- **commit 2325010e6:** with this, EDNS Client Subnet becomes compatible with the packet cache, using the existing variable answer facility.
- **commit 2ec8d8148:** Remove just enough entries from the cache, not one more than asked
- **commit 71df15677:** Move expired cache entries to the front so they are expunged
- **commit d84834c4c:** changed IPv6 addr of b.root-servers.net (Arsen Stasic)
- **commit bcce047bc:** e.root-servers.net has IPv6 now (phonedph1)
- **commit cef8ec7c2:** hello decaf signers (ED25519 and ED448) Testing algorithm 15: 'Decaf ED25519' ->'Decaf ED25519' -> 'Decaf ED25519' Signature & verify ok, signature 68usec, verify 93usec Testing algorithm 16: 'Decaf ED448' ->'Decaf ED448' -> 'Decaf ED448' Signature & verify ok, signature 163usec, verify 252usec (Kees Monshouwer)
- **commit 68490a4b5:** don't use the libdecaf ed25519 signer when libsodium is enabled (Kees Monshouwer)
- **commit 5a88a8ed5:** do not hash the message in the ed25519 signer (Kees Monshouwer)
- **commit 0e7893bf4:** Disable use-incoming-edns-subnet by default

16.2.4 PowerDNS Recursor 4.0.5

Released 13th of June 2017

This release adds ed25519 (algorithm 15) support for DNSSEC and adds the 2017 DNSSEC root key. If you do DNSSEC validation, this upgrade is **mandatory** to continue validating after October 2017.

Bug fixes

- commit af76224: Correctly lowercase the TSIG algorithm name in hash computation, fixes #4942
- commit 86c4ed0: Clear the RPZ NS IP table when clearing the policy, this prevents false positives
- commit 5e660e9: Fix cache-only queries against a forward-zone, fixes #5211
- commit 2875033: Only delegate if NSes are below apex in auth-zones, fixes #4771
- commit e7c183d: Remove hardcoding of port 53 for TCP/IP forwarded zones in recursor, fixes #4799
- commit 5bec36e: Make sure `labelsToAdd` is not empty in `getZoneCuts()`
- commit 0f59e05: Wait until after daemonizing to start the outgoing protobuf thread, prevents hangs when the protobuf server is not available

- commit 233e144: Ensure (re)priming the root never fails
- commit 3642cb3: Don't age the root, fixes a regression from 3.x
- commit 83f9226: Fix exception when sending a protobuf message for an empty question
- commit ffdd813: LuaWrapper: Allow embedded NULs in strings received from Lua
- commit c5ffd90: Fix coredumps on illumos/SmartOS, fixes #4579 (Roman Dayneko)
- commit 651c0e9: StateHolder: Allocate (and copy if needed) before taking the lock
- commit 547d68f: SuffixMatchNode: Fix insertion issue for an existing node
- commit 3ada4e2: Fix negative port detection for IPv6 addresses on 32-bit systems

Additions and Enhancements

- commit 7705e1c: Add support for RPZ wildcarded target names. Fixes #5237
- #5165: Speed up RPZ zone loading and add a `zoneSizeHint` parameter to `rpzFile` and `rpzMaster` for faster reloads
- #4794: Make the RPZ summary consistent (Fixes #4342) and log additions/removals at debug level, not info
- commit 1909556: Add the 2017 root key
- commit abfe671 and commit 7abbb2c: Update Ed25519 algorithm number and mnemonic and hook up to the Recursor (Kees Monshouwer)
- #5355: Add `use-incoming-edns-subnet` option to process and pass along ECS and fix some ECS bugs in the process
- commit dff1a11: Refuse to start with chroot set in a systemd env (Fixes #4848)
- commit 5a38a56: Handle exceptions raised by `closesocket()` to prevent process termination
- #4619: Document missing `top-pub-queries` and `top-pub-servfail-queries` commands for `rec_control` (phonedph1)
- commit 502a850: IPv6 address for g.root-servers.net added (Kevin Otte)
- commit 7a2a645: Log outgoing queries / incoming responses via protobuf

16.2.5 PowerDNS Recursor 4.0.4

Released January 13th 2017

The 4.0.4 version of the PowerDNS Recursor fixes PowerDNS Security Advisories *2016-02* and *2016-04*.

Bug fixes

- commit 658d9e4: Check TSIG signature on IXFR (Security Advisory *2016-04*)
- commit 91acd82: Don't parse spurious RRs in queries when we don't need them (Security Advisory *2016-02*)
- commit 400e28d: Fix incorrect length check in `DNSName` when extracting qtype or qclass
- commit 2168188: rec: Wait until after daemonizing to start the RPZ and protobuf threads
- commit 3beb3b2: On (re-)priming, fetch the root NS records
- commit cfeb109: rec: Fix src/dest inversion in the protobuf message for TCP queries
- commit 46a6666: NSEC3 optout and Bogus insecure forward fixes

- commit bb437d4: On RPZ customPolicy, follow the resulting CNAME
- commit 6b5a8f3: DNSSEC: don't go bogus on zero configured DSs
- commit 1fa6e1b: Don't crash on an empty query ring
- commit bfb7e5d: Set the result to NoError before calling `preresolve`

Additions and Enhancements

- commit 7c3398a: Add `max-recursion-depth` to limit the number of internal recursion
- commit 3d59c6f: Fix building with ECDSA support disabled in libcrypto
- commit 0170a3b: Add requestorId and some comments to the protobuf definition file
- commit d8cd67b: Make the negcache forwarded zones aware
- commit 46ccbd6: Cache records for zones that were delegated to from a forwarded zone
- commit 5aa64e6, commit 5f4242e and commit 0f707cd: DNSSEC: Implement keysearch based on zone-cuts
- commit ddf6fa5: rec: Add support for boost::context >= 1.61
- commit bb6bd6e: Add `getRecursorThreadId()` to Lua, identifying the current thread
- commit d8baf17: Handle CNAMEs at the apex of secure zones to other secure zones

16.2.6 PowerDNS Recursor 4.0.3

Released September 6th 2016

The 4.0.3 version of the PowerDNS Recursor features many improvements to the Policy Engine (RPZ) and the Lua bindings to it. We would like to thank Wim (42wim) for testing and reporting on the RPZ module.

Bug fixes

- #4350: Call `gettag()` for TCP queries
- #4376: Fix the use of an uninitialized filtering policy
- #4381: Parse query-local-address before lua-config-file
- #4383: Fix accessing an empty policyCustom, policyName from Lua
- #4387: ComboAddress: don't allow invalid ports
- #4388: Fix RPZ default policy not being applied over IXFR
- #4391: DNSSEC: Actually follow RFC 7646 §2.1
- #4396: Add boost context ldflags so freebsd builds can find the libs
- #4402: Ignore NS records in a RPZ zone received over IXFR
- #4403: Fix build with OpenSSL 1.1.0 final
- #4404: Don't validate when a Lua hook took the query
- #4425: Fix a protobuf regression (requestor/responder mix-up)

Additions and Enhancements

- #4394: Support Boost 1.61+ fcontext
- #4402: Add Lua binding for DNSRecord::d_place

16.2.7 PowerDNS Recursor 4.0.2

Released August 26th 2016

This release fixes a regression in 4.x where CNAME records for DNSSEC signed domains were not sorted before the final answers, leading to some clients (notably some versions of Chrome) not being able to extract the required answer from the packet. This happened exclusively for DNSSEC signed domains, but the problem happens even for clients not requesting DNSSEC validation.

Further fixes and changes can be found below:

Bug fixes

- #4264: Set `dq.rcode` before calling postresolve
- #4294: Honor PIE flags.
- #4310: Fix build with LibreSSL, for which OPENSSL_VERSION_NUMBER is irrelevant
- #4340: Don't shuffle CNAME records.
- #4354: Fix delegation-only

Additions and enhancements

- #4288: Respect the timeout when connecting to a protobuf server
- #4300: allow newDN to take a DNSName in; document missing methods
- #4301: expose SMN toString to lua
- #4318: Anonymize the protobuf ECS value as well
- #4324: Allow Lua access to the result of the Policy Engine decision, skip RPZ, finish RPZ implementation
- #4349: Remove unused `DNSPacket::d_qlen`
- #4351: RPZ: Use query-local-address(6) by default
- #4357: Move the root DNSSEC data to a header file

16.2.8 PowerDNS Recursor 4.0.1

Released July 29th 2016

This release has several improvements with regards to DNSSEC validation and it improves interoperability with DNSSEC clients that expect an AD-bit on validated data when they query with only the DO-bit set.

Bug fixes

- #4119 Improve DNSSEC record skipping for non dnssec queries (Kees Monshouwer)
- #4162 Don't validate zones from the local auth store, go one level down while validating when there is a CNAME
- #4187:
- Don't go bogus on islands of security
- Check all possible chains for Insecures
- Don't go Bogus on a CNAME at the apex
- #4215 RPZ: default policy should also override local data RRs

- #4243 Fix a crash when the next name in a chained query is empty and `rec_control current-queries` is invoked

Improvements

- #4056 OpenSSL 1.1.0 support (Christian Hofstaedtler)
- #4133 Add limits to the size of received {A,I}XFR (CVE-2016-6172)
- #4140 Fix warnings with gcc on musl-libc (James Taylor)
- #4160 Also validate on +DO
- #4164 Fail to start when the lua-dns-script does not exist
- #4168 Add more Netmask methods for Lua (Aki Tuomi)
- #4210 Validate DNSSEC for security polling
- #4217 Turn on root-nx-trust by default and log-common-errors=off
- #4207 Allow for multiple trust anchors per zone
- #4242 Fix compilation warning when building without Protobuf

16.2.9 PowerDNS Recursor 4.0.0

Released July 11th 2016

PowerDNS Recursor 4.0.0 is part of the great 4.x "Spring Cleaning" of PowerDNS which lasted through the end of 2015.

As part of the general cleanup, we did the following:

- Moved to C++ 2011, a cleaner more powerful version of C++ that has allowed us to improve the quality of implementation in many places.
- Implemented dedicated infrastructure for dealing with DNS names that is fully "DNS Native" and needs less escaping and unescaping
- Switched to binary storage of DNS records in all places
- Moved ACLs to a dedicated Netmask Tree
- Implemented a version of RCU for configuration changes
- Instrumented our use of the memory allocator, reduced number of malloc calls substantially.
- The Lua hook infrastructure was redone using LuaWrapper; old scripts will no longer work, but new scripts are easier to write under the new interface.

In addition to this cleanup, which has many internal benefits and solves longstanding issues with escaped domain names, 4.0.0 brings the following major new features:

- RPZ aka Response Policy Zone support
- IXFR slaving in the PowerDNS Recursor for RPZ
- DNSSEC processing in Recursor (Authoritative has had this for years)
- DNSSEC validation (without NSEC(3) proof validation)
- EDNS Client Subnet support in PowerDNS Recursor (Authoritative has had this for years)
- Lua asynchronous queries for per-IP/per-domain status
- Caches that can now be wiped per whole zone instead of per name
- Statistics on authoritative server response times (split for IPv4 and IPv6)
- APIs are no longer marked as 'experimental' and had one final URL change

- New metric: tcp-answer-bytes to measure DNS TCP/IP bandwidth, and many other new metrics

Please be aware that beyond the items listed here, there have been heaps of tiny changes. As always, please carefully test a new release before deploying it.

This release features the following fixes compared to rc1:

- #3989 Fix usage of std::distance() in DNSName::isPartOf() (signed/unsigned comparisons)
- #4017 Fix building without Lua. Add `isTcp` to `dq`.
- #4023 Actually log on dnssec=log-fail
- #4028 DNSSEC fixes (NSEC casing, send DO-bit over TCP, DNSSEC trace additions)
- #4052 Don't fail configure on missing fcontext.hpp
- #4096 Don't call `commit()` if we skipped all the records

It has the following improvements:

- #3400 Enable building on OpenIndiana
- #4016 Log protobuf messages for cache hits. Add policy tags in gettag()
- #4040 Allow DNSSEC validation when chrooted
- #4094 Sort included html files for improved reproducibility (Christian Hofstaedtler)

And these additions:

- #3981 Import JavaScript sources for libs shipped with Recursor (Christian Hofstaedtler)
- #4012 add tags support to ProtobufLogger.py
- #4032 Set the existing policy tags in `dq` for `{pre,post}resolve`
- #4077 Add DNSSEC validation statistics
- #4090 Allow reloading the lua-config-file at runtime
- #4097 Allow logging DNSSEC bogus in any mode
- #4125 Add protobuf fields for the query's time in the response

PowerDNS Recursor 4.0.0-rc1

Released June 9th 2016

This first (and hopefully last) Release Candidate contains the finishing touches to the experimental DNSSEC support by adding (Negative) Trust Anchor support and fixing a possible issue with DNSSEC and forwarded domains:

- #3910 Add (Negative) Trust Anchor management
- #3926 Set +CD on forwarded recursive queries

Other changes:

- #3941 Ensure delegations from local auth zones are followed
- #3924 Add a virtual hosting unit-file
- #3929 Set the FDs in the unit file to a sane value

Bug fixes:

- #3961 Fix building on EL6 i386
- #3957 Add error reporting when parsing forward-zones(-recurse) (Aki Tuomi)

PowerDNS Recursor 4.0.0-beta1

Released May 27th 2016

This release fixes a bug in the DNSSEC implementation where a name would we validated as bogus when talking to non-compliant authoritative servers:

- #3875 Disable DNSSEC for domain where the auth responds with FORMERR or NOTIMP

Improvements

- #3866 Increase max FDs in systemd unit file
- #3905 Add a dnssec=process-no-validate option and make it default

Bug fixes

- #3881 Fix the `noEdnsOutQueries` counter
- #3892 support `clock_gettime` for platforms that require -lrt

PowerDNS Recursor 4.0.0-alpha3

Released May 10th 2016

This release features several leaps in the correctness and stability of the DNSSEC implementation.

Notable changes are:

- #3752 Correct handling of query flags in conformance with RFC 6840

Bug fixes

- #3804 Fix a memory leak in DNSSEC validation
- #3785 and #3390 Correctly validate insecure delegations
- #3606 Various DNSSEC fixes, disabling DNSSEC on forward-zones
- #3681 Catch exception with a malformed DNSName in `rec_control wipe-cache`
- #3779, #3768, #3766, #3783 and #3789 DNSName and other hardening improvements

Improvements

- #3801 Add missing Lua rcodes bindings
- #3587 Update L-Root addresses

PowerDNS Recursor 4.0.0-alpha2

Released March 9th 2016

Note that the DNSSEC implementation has several bugs in this release, it is advised to set `dnssec=off` in your recursor.conf.

This release features many low-level performance fixes. Other notable changes since 4.0.0-alpha1 are:

- #3259, #3280 The PowerDNS Recursor now properly uses GNU autoconf and autotools for building and installing
- OpenSSL crypto primitives are now used for DNSSEC validation

- #3313 Implement the logic we need to generate EDNS MAC fields in dnsdist & read them in recursor (blogpost
- #3350 Add lowercase-outgoing feature to Recursor
- #3410 Recuweb is now built-in to the daemon
- #3230 API: drop JSONP, add web security headers (Christian Hofstaedtler)
- #3485 Allow multiple carbon-servers
- #3427, #3479, #3472 MTasker modernization (Andrew Nelless)

Bug fixes

- #3444, #3442 RPZ IXFR fixes
- #3448 Remove edns-subnet-whitelist whitelist pointing to powerdns.com (Christian Hofstaedtler)
- #3293 make asynchronous UDP Lua queries work again in 4.x
- #3365 Apply rcode set in UDPQueryResponse callback (Jan Broers)
- #3244 Fix the forward zones in the recursor
- #3135 Use 56 bits instead of 64 in EDNS Client Subnet option (Winfried Angele)
- #3527 Make the recursor counters atomic

Improvements

- #3435 Add `toStringNoDot` and `chopOff` functions to Lua
- #3437 Add `pdns.now` timeval struct to recursor Lua
- #3352 Cache improvements
- #3502 Make second argument to pdnslog optional (Thiago Farina)
- #3520 Reduce log level of periodic statistics to notice (Jan Broers)

16.2.10 PowerDNS Recursor 4.0.0-alpha1

Released December 24th 2015

16.3 Changelogs for all pre 4.0 releases

Note: Beyond PowerDNS 2.9.20, the Authoritative Server and Recursor are released separately. Hence, this changelog starts at version 3.0.

16.3.1 PowerDNS Recursor 3.6.4

Released 9th of June 2015

This is a security release fixing *Security Advisory 2015-01*

Bug fixes:

- commit bccd068: Limit the maximum length of a qname

16.3.2 PowerDNS Recursor 3.7.3

Released 9th of June 2015

Bug fixes:

- commit 92f7b2b: Limit the maximum length of a qname

This is a security release fixing *Security Advisory 2015-01*

Improvements:

- commit 46366a5, commit f318a7d: pdnssec: check for glue and delegations in parent zones (Kees Monshouwer)

16.3.3 PowerDNS Recursor 3.7.2

Released 23rd of April, 2015

Among other bug fixes and improvements (as listed below), this release incorporates a fix for CVE-2015-1868, as detailed in *PowerDNS Security Advisory 2015-01*

Bug fixes:

- commit adb10be commit 3ec3e0f commit dc02ebf Fix handling of forward references in label compressed packets; fixes CVE-2015-1868
- commit a7be3f1: make sure we never call sendmsg with msg_control!=NULL && msg_controllen>0. Fixes ticket #2227
- commit 9d835ed: Improve robustness of root-nx-trust.

Improvements:

- commit 99c595b: Silence warnings that always occur on FreeBSD (Ruben Kerkhof)

16.3.4 PowerDNS Recursor 3.6.3

Released 23rd of April, 2015

The only difference between Recursor 3.6.2 and 3.6.3 is a fix for CVE-2015-1868, as detailed in *PowerDNS Security Advisory 2015-01*

16.3.5 PowerDNS Recursor 3.7.0

Unreleased, please see the 3.7.1 changelog below.

16.3.6 PowerDNS Recursor 3.7.1

Released February 12th, 2015.

This version contains a mix of speedups and improvements, the combined effect of which is vastly improved resilience against traffic spikes and malicious query overloads.

Of further note is the massive community contribution, mostly over Christmas. Especially Ruben Kerkhof, Pieter Lexis, Kees Monshouwer and Aki Tuomi delivered a lot of love. Thanks!

Minor changes:

- Removal of dead code here and there 04dc6d618734fc630122de4c56dff641ebaf0988
- Per-qtype response counters are now 64 bit 297bb6acf7902068693a4aae1443c424d0e8dd52 on 64 bit systems

- Add IPv6 addresses for b and c.root-servers.net hints efc2595423c9a1be6f2d8f4da25445198ceb8b57
- Add IP address to logging about terminated queries 37aa9904d1cc967ba4b5d5e17dbe41485f8cdece
- Improve qtype name logging fab3ed3453e15ae88e29a0e4071b214eb19caad9 (Aki Tuomi)
- Redefine 'BAD_NETS' for dont-query based on newer IANA guidance 12cd44ee0fcde5893f85dccc499bfc35152c5fff (lochiiconnectivity)
- Add documentation links to systemd unit eb154adfdffa5c78624e2ea98e938d7b5787119e (Ruben Kerkhof)

Improvements:

- Upgrade embedded PolarSSL to 1.3.9: d330a2ea1a93d7675ef680311f8aa0306aeefcf1
- yahttp upgrade c290975778942ed1082ca66918695a5bd2d6bac4 c65a57e888ee48eaa948e590c90c51420bffa847 (Aki Tuomi)
- Replace . in hostnames by - for Carbon so as not to confuse Metronome 46541751ed1c3bc051d78217543d5fc76733e212
- Manpages got a lot of love and are now built from Markdown (Pieter Lexis)
- Move to PolarSSL base64 488360551009784ab35c43ee4580e773a2a8a227 (Kees Monshouwer)
- The quiet=no query logging is now more informative 461df9d20c560d240285f772c09b3beb89d46daa
- We can finally bind to 0.0.0.0 and :: and guarantee answers from the correct source b71b60ee73ef3c86f80a2179981eda2e61c4363f
- We use per-packet timestamps to drop ancient traffic in case of overload b71b60ee73ef3c86f80a2179981eda2e61c4363f, non-Linux portability in d63f0d83631c41eff203d30b0b7c475a88f1db59
- Builtin webserver can be queried with the API key in the URL again c89f8cd022c4a9409b95d22ffa3b03e4e98dc400
- Ringbuffers are now available via API c89f8cd022c4a9409b95d22ffa3b03e4e98dc400
- Lua 5.3 compatibility 59c6fc3e3931ca87d484337daee512e716bc4cf4 (Kees Monshouwer)
- No longer leave a stale UNIX domain socket around from rec_control if the recursor was down 524e4f4d81f4ed9eb218715cbc8a59f0b9868234, ticket #2061
- Running with 'quiet=no' would strangely actually prevent debug messages from being logged f48d7b657ec32517f8bfcada3bfe6353ca313314
- Webserver now implements CORS for the API ea89a97e864c43c1cb03f2959ad04c4ebe7580ad, fixing ticket #1984
- Houskeeping thread would sometimes run multiple times simultaneously, which worked, but was odd cc59bce675e62e2b9657b42614ce8be3312cae82

New features:

- New `root-nx-trust` flag makes PowerDNS generalize NXDOMAIN responses from the root-servers 01402d56846a3a61811ebd4e6bc97e53f908e568
- `getregisteredname()` for Lua, which turns 'www.bbc.co.uk' into 'bbc.co.uk' 8cd4851beb78bc6ab320926fb5cb6a09282016b1
- Lua preoutquery filter 3457a2a0ec41d3b3aff7640f30008788e1228a6e
- Lua IP-based filter (ipfilter) before parsing packets 4ea949413c495254acb0bd19335142761c1efc0c
- `iputils` class for Lua, to quickly process IP addresses and netmasks in their native format
- `getregisteredname` function for Lua, to find the registered domain for a given name
- Various new ringbuffers: top-servfail-remotes, top-largeanswer-remotes, top-servfail-queries

Speedups:

- Remove unneeded malloc traffic 93d4a89096e64d53740790f58fadec56f6a0af14 8682c32bc45b6ffa7c0f6da778e1b223ae7f03ce a903b39cfe7364c56324038264d3db50b8cece87
- Our nameserver-loop detection carried around a lot of baggage for complex domain names, plus did not differentiate IPv4 and IPv6 well enough 891fbf888ccac074e3edc38864641ca774f2f03c
- Prioritize new queries over nameserver responses, improving latency under query bursts bf3b0cec366c090af000b066267b6f6bbb3a512a
- Remove escaping in case there was nothing to escape 83b746fd1d94c8742d8bd87a44beb44c154230c7
- Our logging infrastructure had a lot of locking d1449e4d073595e1e1581804f121fc90e37158bf
- Reduce logging level of certain common messages, which locked up synchronously logging systems 854d44e31c76aa650520e6d462dd3a02b5936f7a
- Add limit on total wall-clock time spent on a query 9de3e0340fa066d4c59449e1643a1de8c343f8f2
- Packet cache is now case-insensitive, which increases hitrate 90974597aadaf1096e3fd0dc450be7422ea591a5

Security relevant:

- Check for PIE, RELRO and stack protector during configure 8d0354b189c12e1e14f5309d3b49935c17f9eeb0 (Aki Tuomi)
- Testing for support of PIE etc was improved in b2053c28ccb9609e2ce7bcb6beda83f98a062aa3 and beyond, fixes #2125 (Ruben Kerkhof)
- Max query-per-query limit (max-qperq) is now configurable 173d790ead08f67733010ca4c6fc404a040fe699

Bugs fixed:

- IPv6 outgoing queries had a disproportionate effect on our query load. Fixed in 76f190f2a0877cd79ede2994124c1a58dc69ae49 and beyond.
- rec_control gave incorrect output on a timeout 12997e9d800734da51b808767e1e2477244c30eb
- When using the webserver AND having an error in the Lua script, recursor could crash during startup 62f0ae62984adadab687c23fe1b287c1f219b2cb
- Hugely long version strings would trip up security polling 18b7333828a1275ae5f5574a9c8330290d8557ff (Kees Monshouwer)
- The 'remotes' ringbuffer was sized incorrectly f8f243b01215d6adcb59389f09ef494f1309041f
- Cache sizes had an off-by-one scaling problem, with the wrong number of entries allocated per thread f8f243b01215d6adcb59389f09ef494f1309041f
- Our automatic file descriptor limit raising was attempted *after* setuid, which made it a lot less effective. Found and fixed by Aki Tuomi a6414fdce9b0ec32c340d1f2eea2254f3fedc1c1
- Timestamps used for dropping packets were occasionally wrong 183eb8774e4bc2569f06d5894fec65740f4b70b6 and 4c4765c104bacc146533217bcc843efb244a8086 (RC2) with thanks to Winfried for debugging.
- In RC1, our new DoS protection measures would crash the Recursor if too many root servers were unreachable. 6a6fb05ad81c519b4002ed1db00f3ed9b7bce6b4. Debugging and testing by Fusl.

Various other documentation changes by Christian Hofstaedtler and Ruben Kerkhof. Lots of improvements all over the place by Kees Monshouwer.

16.3.7 PowerDNS Recursor 3.6.2

Note: Version 3.6.2 is a bugfix update to 3.6.1. Released on the 30th of October 2014.

Official download page

A list of changes since 3.6.1 follows.

- commit ab14b4f: expedite servfail generation for ezdns-like failures (fully abort query resolving if we hit more than 50 outqueries). This also prevents the issue documented in *PowerDNS Security Advisory 2014-02* (CVE-2014-8601)
- commit 42025be: PowerDNS now polls the security status of a release at startup and periodically. More detail on this feature, and how to turn it off, can be found in Security polling.
- commit 5027429: We did not transmit the right 'local' socket address to Lua for TCP/IP queries in the recursor. In addition, we would attempt to lookup a filedescriptor that wasn't there in an unlocked map which could conceivably lead to crashes. Closes ticket 1828, thanks Winfried for reporting
- commit 752756c: Sync embedded yahttp copy. API: Replace HTTP Basic auth with static key in custom header
- commit 6fdd40d: add missing `#include <pthread.h>` to rec-channel.hh (this fixes building on OS X).

16.3.8 PowerDNS Recursor 3.6.1

Warning: Version 3.6.1 is a mandatory security upgrade to 3.6.0! Released on the 10th of September 2014.

PowerDNS Recursor 3.6.0 could crash with a specific sequence of packets. For more details, see the advisory. PowerDNS Recursor 3.6.1 was very well tested, and is in full production already, so it should be a safe upgrade.

Downloads

- Official download page

In addition to various fixes related to this potential crash, 3.6.1 fixes a few minor issues and adds a debugging feature:

- We could not encode IPv6 AAAA records that mapped to IPv4 addresses in some cases (:ffff.1.2.3.4). Fixed in commit c90fcbd , closing ticket 1663.
- Improve systemd startup timing with respect to network availability (commit cf86c6a), thanks to Morten Stevens.
- Realtime telemetry can now be enabled at runtime, for example with 'rec_control carbon-server 82.94.213.34 ourname1234'. This ties in to our existing carbon-server and carbon-ourname settings, but now at runtime. This specific invocation will make your stats appear automatically on our public telemetry server.

16.3.9 PowerDNS Recursor version 3.6.0

This is a performance, feature and bugfix update to 3.5/3.5.3. It contains important fixes for slightly broken domain names, which your users expect to work anyhow. It also brings robust resilience against certain classes of attacks.

Downloads

- Official download page
- native RHEL5/6 packages from Kees Monshouwer

Changes between RC1 and release

- commit 30b13ef: do not apply some of our filters to root and gtlds, plus remove some useless {}
- commit cc81d90: fix yahttp copy in dist-recursor for BSD cp
- commit b798618: define __APPLE_USE_RFC_3542 during recursor build on Darwin, fixes ticket 1449

- commit 1d7f863: Merge pull request ticket 1443 from zeha/recursor-nostrip
- commit 5cdeede: remove (non-working) [aaaa-]additional-processing flags from the recursor. Closes ticket 1448
- commit 984d747: Support building recursor on kFreeBSD and Hurd
- commit 79240f1: Allow not stripping of binaries in recursor's make install
- commit e9c2ad3: document pdns.DROP for recursor, add policy-drops metric for it

New features

- commit aadceba: Implement minimum-ttl-override config setting, plus runtime configurability via 'rec_control set-minimum-ttl'.
- Lots of work on the JSON API, which is exposed via Aki Tuomi's 'yahttp'. Massive thanks to Christian Hofstaedtler for delivering this exciting new functionality. Documentation & demo forthcoming, but code to use it is available on GitHub.
- Lua modules can now use 'pdnslog(INFO..'), as described in ticket 1074, implemented in commit 674a305
- Adopt any-to-tcp feature to the recursor. Based on a patch by Winfried Angele. Closes ticket 836, commit 56b4d21 and commit e661a20.
- commit 2c78bd5: implement built-in statistics dumper using the 'carbon' protocol, which is also understood by metronome (our mini-graphite). Use 'carbon-server', 'carbon-ourname' and 'carbon-interval' settings.
- New setting 'udp-truncation-threshold' to configure from how many bytes we should truncate. commit a09a8ce.
- Proper support for CHaos class for CHAOS TXT queries. commit c86e1f2, addition for lua in commit f94c53d, some warnings in commit 438db54 however.
- Added support for Lua scripts to drop queries w/o further processing. commit 0478c54.
- Kevin Holly added qtype statistics to recursor and rec_control (get-qtypelist) (commit 79332bf)
- Add support for include-files in configuration, also reload ACLs and zones defined in them (commit 829849d, commit 242b90e, commit 302df81).
- Paulo Anes contributed server-down-max-fails which helps combat Recursive DNS based amplification attacks. Described in this post. Also comes with new metric 'failed-host-entries' in commit 406f46f.
- commit 21e7976: Implement "followCNAMERecords" feature in the Lua hooks.

Improvements

- commit 06ea901: make pdns-distributes-queries use a hash so related queries get sent to the same thread. Original idea by Winfried Angele. Astoundingly effective, approximately halves CPU usage!
- commit b13e737: –help now writes to stdout instead of stderr. Thanks Winfried Angele.
- To aid in limiting DoS attacks, when truncating a response, we actually truncate all the way so only the question remains. Suggested in ticket 1092, code in commit add935a.
- No longer experimental, the switch 'pdns-distributes-queries' can improve multi-threaded performance on Linux (various cleanup commits).
- Update to embedded PolarSSL, plus remove previous AES implementation and shift to PolarSSL (commit e22d9b4, commit 990ad9a)
- commit 92c0733 moves various Lua magic constants into an enum namespace.
- set group and supplementary groups before chroot (commit 6ee50ce, ticket 1198).
- commit 4e9a20e: raise our socket buffer setting so it no longer generates a warning about lowering it.

- commit 4e9a20e: warn about Linux suboptimal IPv6 settings if we detect them.
- SIGUSR2 turns on a 'trace' of all DNS traffic, a second SIGUSR2 now turns it off again. commit 4f217ce.
- Various fixes for Lua 5.2.
- commit 81859ba: No longer attempt to answer questions coming in from port 0, reply would not reach them anyhow. Thanks to Niels Bakker and 'sid3windr' for insight & debugging. Closes ticket 844.
- commit b1a2d6c: now, I'm not one to get OCD over things, but that log message about stats based on 1801 seconds got to me. 1800 now.

Fixes

- 0c9de4fc: stay away from getaddrinfo unless we really can't help it for ascii ipv6 conversions to binary
- commit 08f3f63: fix average latency calculation, closing ticket 424.
- commit 75ba907: Some of our counters were still 32 bits, now 64.
- commit 2f22827: Fix statistics and stability when running with pdns-distributes-queries.
- commit 6196f90: avoid merging old and new additional data, fixes an issue caused by weird (but probably legal) Akamai behaviour
- commit 3a8a4d6: make sure we don't exceed the number of available filedescriptors for mthreads. Raises performance in case of DoS. See this post for further details.
- commit 7313fe6: implement indexed packet cache wiping for recursor, orders of magnitude faster. Important when reloading all zones, which causes massive cache cleaning.
- rec_control get-all would include 'cache-bytes' and 'packetcache-bytes', which were expensive operations, too expensive for frequent polling. Removed in commit 8e42d27.
- All old workarounds for supporting Windows of the XP era have been removed.
- Fix issues on S390X based systems which have unsigned characters (commit 916a0fd)

16.3.10 PowerDNS Recursor version 3.5.3

Released September 17th, 2013

This is a bugfix and performance update to 3.5.2. It brings serious performance improvements for dual stack users.

Downloads

- Official download page
- native RHEL5/6 packages from Kees Monshouwer

Changes since 3.5.2

- 3.5 replaced our ANY query with A+AAAA for users with IPv6 enabled. Extensive measurements by Darren Gamble showed that this change had a non-trivial performance impact. We now do the ANY query like before, but fall back to the individual A+AAAA queries when necessary. Change in commit 1147a8b.
- The IPv6 address for d.root-servers.net was added in commit 66cf384, thanks Ralf van der Enden.
- We now drop packets with a non-zero opcode (i.e. special packets like DNS UPDATE) earlier on. If the experimental pdns-distributes-queries flag is enabled, this fix avoids a crash. Normal setups were never susceptible to this crash. Code in commit 35bc40d, closes ticket 945.
- TXT handling was somewhat improved in commit 4b57460, closing ticket 795.

16.3.11 PowerDNS Recursor version 3.5.2

Released June 7th, 2013

This is a stability and bugfix update to 3.5.1. It contains important fixes that improve operation for certain domains.

Downloads

- Official download page
- native RHEL5/6 packages from Kees Monshouwer

Changes since 3.5.1

- Responses without the QR bit set now get matched up to an outstanding query, so that resolution can be aborted early instead of waiting for a timeout. Code in commit ee90f02.
- The depth limiter changes in 3.5.1 broke some legal domains with lots of indirection. Improved in commit d393c2d.
- Slightly improved logging to aid debugging. Code in commit 437824d and commit 182005e.

16.3.12 PowerDNS Recursor version 3.5.1

Released May 3rd, 2013

This is a stability and bugfix update to 3.5. It contains important fixes that improve operation for certain domains.

Downloads

- Official download page
- native RHEL5/6 packages from Kees Monshouwer

Changes since 3.5

- We now abort earlier while following endless glue or CNAME chains. Fix in commit 02d1742.
- Some unused code would crash certain gcc versions on ARM. Reported by Morten Stevens, fixed in commit 5b188e8.
- The 3.5 fix for ticket 731 was too strict, causing trouble with at least one domain. Reported by Aki Tuomi, check slightly relaxed in commit 4134690.
- Automake/autoconf now use non-deprecated syntax. Reported by Morten Stevens, change in commit ca17ef2.

16.3.13 PowerDNS Recursor version 3.5

Released April 15th, 2013

This is a stability, security and bugfix update to 3.3/3.3.1. It contains important fixes for slightly broken domain names, which your users expect to work anyhow. **Note**: Because a semi-sanctioned 3.4-pre was distributed for a long time, and people have come to call that 3.4, we are skipping an actual 3.4 release to avoid confusion.

Downloads

- Official download page
- native RHEL5/6 packages from Kees Monshouwer

Changes between RC5 and the final 3.5 release

- Winfried Angele reported that restarting a very busy recursor could lead to crashes. Fixed in r3153, closing ticket 735.

Changes between RC4 and RC5

- Bernd-René Predota of Liberty Global reported that Recursor 3.3 would treat empty non-AA NOERROR responses as authoritative NXDATA responses. This bug turned out to be in 3.5-RC4 too. Fixed in commit 3146, related to ticket 731.

Changes between RC3 (unreleased) and RC4

- Winfried Angele spotted, even before release, that commit 3132 in RC3 broke outgoing IPv6 queries. We are grateful for his attention to detail! Fixed in commit 3141. Changes between RC2 and RC3 (unreleased)
- Use private temp dir when running under systemd, thanks Morten Stevens and Ruben Kerkhof. Change in commit 3105.
- NSD mistakenly compresses labels for RP and other types, violating a MUST in RFC 3597. Recursor does not decompress these labels, violating a SHOULD in RFC3597. We now decompress these labels, and reportedly NSD will stop compressing them. Reported by Jan-Piet Mens, fixed in commit 3109.
- When forwarding to another recursor, we would handle responses to ANY queries incorrectly. Spotted by Jan-Piet Mens, fixed in commit 3116, closes ticket 704.
- Our local-nets definition (used as a default for some settings) now includes the networks from RFC 3927 and RFC 6598. Reported by Maik Zumstrull, fixed in commit 3122.
- The RC1 change to stop using ANY queries to get A+AAAA for name servers in one go had a 5% performance impact. This impact is corrected in commit 3132. Thanks to Winfried Angele for measuring and reporting this. Closes ticket 710.
- New command 'rec_control dump-nsspeeds' will dump our NS speeds (latency) cache. Code in commit 3131.

Changes between RC1 and RC2

- While Recursor 3.3 was not vulnerable to the specific attack noted in 'Ghost Domain Names: Revoked Yet Still Resolvable' (more information at A New DNS Exploitation Technique: Ghost Domain Names), further investigation showed that a variant of the attack could work. This was fixed in commit 3085. This should also close the slightly bogus CVE-2012-1193. Closes ticket 668.
- The auth-can-lower-ttl flag was removed, as it did not have any effect in most situations, and thus did not operate as advertised. We now always comply with the related parts of RFC 2181. Change in commit 3092, closing ticket 88.

New features

- The local zone server now understands wildcards, code in commit 2062.
- The Lua postresolve and nodata hooks, that had been distributed as a '3.3-hooks' snapshot earlier, have been merged. Code in commit 2309.

- A new feature, rec_control trace-regex allows the tracing of lookups for specific names. Code in commit 3044, commit 3073.
- A new setting, export-etc-hosts-search-suffix, adds a configurable suffix to names imported from /etc/hosts. Code in commit 2544, commit 2545.

Improvements

- We now throttle queries that don't work less aggressively, code in commit 1766.
- Various improvements in tolerance against broken auths, code in commit 1996, commit 2188, commit 3074 (thanks Winfried).
- Additional processing is now optional, and disabled by default. Presumably this yields a performance improvement. Change in commit 2542.
- rec_control reload-lua-script now reports errors. Code in commit 2627, closing ticket 278.
- rec_control help now lists commands. Code in commit 2628.
- rec_control wipe-cache now also wipes the recursor's packet cache. Code in commit 2880 from ticket 333.
- Morten Stevens contributed a systemd file. Import in commit 2966, now part of the recursor tarball.
- commit 2990 updates the address of D.root-servers.net.
- Winfried Angele implemented and documented the ipv6-questions metric. Merge in commit 3034, closing ticket 619.
- We no longer use ANY to get A+AAAA for nameservers, because some auth operators have decided to break ANY lookups. As a bonus, we now track v4 and v6 latency separately. Change in commit 3064.

Bugs fixed

- Some unaligned memory access was corrected, code in commit 2060, commit 2122, commit 2123, which would cause problems on UltraSPARC.
- Garbage encountered during reload-acls could cause crashes. Fixed in commit 2323, closing ticket 330.
- The recursor would lose its root hints in a very rare situation. Corrected in commit 2380.
- We did not always drop supplemental groups while dropping privileges. Reported by David Black of Atlassian, fixed in commit 2524.
- Cache aging would sometimes get confused when we had a mix of expired and non-expired records in cache. Spotted and fixed by Winfried Angele in commit 3068, closing ticket 438.
- rec_control reload-acl no longer ignores arguments. Fix in commit 3037, closing ticket 490.
- Since we re-parse our commandline in rec_control we've been doubling the commands on the commandline, causing weird output. Reported by Winfried Angele. Fixed in commit 2992, closing ticket 618. This issue was not present in any officially released versions.
- commit 2879 drops some spurious stderr logging from Lua scripts, and makes sure 'place' is always valid.
- We would sometimes refuse to resolve domains with just one nameserver living at the apex. Fixed in commit 2817.
- We would sometimes stick RRs in the wrong parts of response packets. Fixed in commit 2625.
- The ACL parser was too liberal, sometimes causing recursors to be very open. Fixed in commit 2629, closing ticket 331.
- rec_control now honours socket-dir from recursor.conf. Fixed in commit 2630.
- When traversing CNAME chains, sometimes we would end up with multiple SOAs in the result. Fixed in commit 2633.

16.3.14 Recursor version 3.3.1

Warning:Unreleased

Version 3.3.1 contains a small number of important fixes, adds some memory usage statistics, but no new features.

- Discovered by John J and Robin J, the PowerDNS Recursor did not process packets that were truncated in mid-record, and also did not act on the 'truncated' (TC) flag in that case. This broke a very small number of domains, most of them served by very old versions of the PowerDNS Authoritative Server. Fix in commit 1740.
- PowerDNS emitted a harmless, but irritating, error message on receiving certain very short packets. Discovered by Winfried A and John J, fix in commit 1729.
- PowerDNS could crash on startup if configured to provide service on malformed IPv6 addresses on FreeBSD, or in case when the FreeBSD kernel was compiled without any form of IPv6 support. Debugged by Bryan Seitz, fix in commit 1727.
- Add max-mthread-stack metric to debug rare crashes. Could be used to save memory on constrained systems. Implemented in commit 1745.
- Add cache-bytes and packetcache-bytes metrics to measure our 'pre-malloc' memory utilization. Implemented in commit 1750.

16.3.15 Recursor version 3.3

Released on the 22nd of September 2010.

Warning: Version 3.3 fixes a number of small but persistent issues, rounds off our IPv6 %link-level support and adds an important feature for many users of the Lua scripts.

In addition, scalability on Solaris 10 is improved.

Bug fixes

- 'dist-recursor' script was not compatible with pure POSIX /bin/sh, discovered by Simon Kirby. Fix in commit 1545.
- Simon Bedford, Brad Dameron and Laurient Papier discovered relatively high TCP/IP loads could cause TCP/IP service to shut down over time. Addressed in commits 1546, 1640, 1652, 1685, 1698. Additional information provided by Zwane Mwaikambo, Nicholas Miell and Jeff Roberson. Testing by Christian Hofstaedtler and Michael Renner.
- The PowerDNS Recursor could not read the 'root zone' (this is something else than the root hints) because of an unquoted TXT record. This has now been addressed, allowing operators to hardcode the root zone. This can improve security if the root zone used is kept up to date. Change in commit 1547.
- A return of an old bug, when a domain gets new nameservers, but the old nameservers continue to contain a copy of the domain, PowerDNS could get 'stuck' with the old servers. Fixed in commit 1548.
- Discovered & reported by Alexander Gall of SWITCH, the Recursor used to try to resolve 'AXFR' records over UDP. Fix in commit 1619.
- The Recursor embedded authoritative server messed up parsing a record like '@ IN MX 15 @'. Spotted by Aki Tuomi, fix in commit 1621.
- The Recursor embedded authoritative server messed up parsing really really long lines. Spotted by Marco Davids, fix in commit 1624, commit 1625.
- Packet cache was not DNS class correct. Spotted by "Robin", fix in commit 1688.
- The packet cache would cache some NXDOMAINs for too long. Solving this bug exposed an underlying oddity where the initial NXDOMAIN response had an overly long (untruncated) TTL, whereas all the next ones would be ok. Solved in commit 1679, closing ticket 281. Especially important for RBL operators. Fixed after some nagging by Alex Broens (thanks).

Improvements

- The priming of the root now uses more IPv6 addresses. Change in commit 1550, closes ticket 287. Also, the IPv6 address of I.ROOT-SERVERS.NET was added in commit 1650.
- The `rec_control dump-cache` command now also dumps the 'negative query' cache. Code in commit 1713.
- PowerDNS Recursor can now bind to fe80 IPv6 space with '%eth0' link selection. Suggested by Darren Gamble, implemented with help from Niels Bakker. Change in commit 1620.
- Solaris on x86 has a long standing bug in port_getn(), which we now work around. Spotted by 'Dirk' and 'AS'. Solution suggested by the Apache runtime library, update in commit 1622.
- New runtime statistic: 'tcp-clients' which lists the number of currently active TCP/IP clients. Code in commit 1623.
- Deal better with UltraDNS style CNAME redirects containing SOA records. Spotted by Andy Fletcher from UKDedicated in ticket 303, fix in commit 1628.
- The packet cache, which has 'ready to use' packets containing answers, now artificially ages the ready to use packets. Code in commit 1630.
- Lua scripts can now indicate that certain queries will have 'variable' answers, which means that the packet cache will not touch these answers. This is great for overriding some domains for some users, but not all of them. Use setvariable() in Lua to indicate such domains. Code in commit 1636.
- Add query statistic called 'dont-outqueries', plus add IPv6 address :: and IPv4 address 0.0.0.0 to the default "dont-query" set, preventing the Recursor from talking to itself. Code in commit 1637.
- Work around a gcc 4.1 bug, still in wide use on common platforms. Code in commit 1653.
- Add 'ARCHFLAGS' to PowerDNS Recursor Makefile, easing 64 bit compilation on mainly 32 bit platforms (and vice versa).
- Under rare circumstances, querying the Recursor for statistics under very high load could lead to a crash (although this has never been observed). Bad code removed & good code unified in commit 1675.
- Spotted by Jeff Sipek, the rec_control manpage did not list the new get-all command. commit 1677.
- On some platforms, it may be better to have PowerDNS itself distribute queries over threads (instead of leaving it up to the kernel). This experimental feature can be enabled with the 'pdns-distributes-queries' setting. Code in commit 1678 and beyond. Speeds up Solaris measurably.
- Cache cleaning code was cleaned up, unified and expanded to cover the 'negative cache', which used to be cleaned rather bluntly. Code in commit 1702, further tweaks in commit 1712, spotted by Darren Gamble, Imre Gergely and Christian Kovacic.

Changes between RC1, RC2 and RC3.

- RC2: Fixed linking on RHEL5/CentOS5, which both ship with a gcc compiler that claims to support atomic operations, but doesn't. Code in commit 1714. Spotted by 'Bas' and Imre Gergely.
- RC2: Negative query cache was configured to grow too large, and was not cleaned efficiently. Code in commit 1712, spotted by Imre Gergely.
- RC3: Root failed to be renewed automatically, relied on fallback to make this happen. Code in commit 1716, spotted by Detlef Peeters.

16.3.16 Recursor version 3.2

Released on the 7th of March 2010.

Warning: Lua scripts from version 3.1.7.* are fully compatible with version 3.2. However, scripts written for development snapshot releases, are NOT. Please see Scripting for details!

The 3.2 release is the first major release of the PowerDNS Recursor in a long time. Partly this is because 3.1.7.* functioned very well, and delivered satisfying performance, partly this is because in order to really move forward, some heavy lifting had to be done.

As always, we are grateful for the large PowerDNS community that is actively involved in improving the quality of our software, be it by submitting patches, by testing development versions of our software or helping debug interesting issues. We specifically want to thank Stefan Schmidt and Florian Weimer, who both over the years have helped tremendously in keeping PowerDNS fast, stable and secure.

This version of the PowerDNS Recursor contains a rather novel form of lock-free multithreading, a situation that comes close to the old '–fork' trick, but allows the Recursor to fully utilize multiple CPUs, while delivering unified statistics and operational control.

In effect, this delivers the best of both worlds: near linear scaling, with almost no administrative overhead.

Compared to 'regular multithreading', whereby threads cooperate more closely, more memory is used, since each thread maintains its own DNS cache. However, given the economics, and the relatively limited total amount of memory needed for high performance, this price is well worth it.

In practical numbers, over 40,000 queries/second sustained performance has now been measured by a third party, with a 100.0% packet response rate. This means that the needs of around 400,000 residential connections can now be met by a single commodity server.

In addition to the above, the PowerDNS Recursor is now providing resolver service for many more Internet users than ever before. This has brought with it 24/7 Service Level Agreements, and 24/7 operational monitoring by networking personnel at some of the largest telecommunications companies in the world.

In order to facilitate such operation, more statistics are now provided that allow the visual verification of proper PowerDNS Recursor operation. As an example of this there are now graphs that plot how many queries were dropped by the operating system because of a CPU overload, plus statistics that can be monitored to determine if the PowerDNS deployment is under a spoofing attack. All in all, this is a large and important PowerDNS Release, paving the way for further innovation.

Note: This release removes support for the 'fork' multi-processor option. In addition, the default is now to spawn two threads. This has been done in such a way that total memory usage will remain identical, so each thread will use half of the allocated maximum number of cache entries.

Changes between RC2 and -release

- 'Make install' when an existing configuration file contained a 'fork' statement has been fixed. Spotted by Darren Gamble, code in commit 1534.
- Reloading a non-existent allow-from-file caused the control thread to stop working. Spotted by Imre Gergely, code in commit 1532.
- Parser got confused by reading en empty line in auth-forward-zones. Spotted by Imre Gergely, code in commit 1533.
- David Gavarret discovered undocumented and not-working settings to set the owner, group and access modes of the control socket. Code by Aki Tuomi and documentation in commit 1535. Fixup in commit 1536 for FreeBSD as found by Ralf van der Enden.
- Tiny improvement possibly solving an issue on Solaris 10's completion port event multiplexer (commit 1537).

Changes between RC1 and RC2

- Compilation on Solaris 10 has been fixed (various patchlevels had different issues), code in commit 1522.
- Compatibility with CentOS4/RHEL4 has been restored, the gcc and glibc versions shipped with this distribution contain a Thread Local Storage bug which we now work around. Thanks to Darren Gamble and Imre Gergely for debugging this issue, code in commit 1527.

- A failed setuid operation, because of misconfiguration, would result in a crash instead of an error message. Fixed in commit 1523.
- Imre Gergely discovered that PowerDNS was doing spurious root repriming when invalidating nssets. Fixed in commit 1531.
- Imre Gergely discovered our rrd graphs had not been changed for the new multithreaded world, and did not allow scaling beyond 200% cpu use. In addition, CPU usage graphs did not add up correctly. Implemented in commit 1524.
- Andreas Jakum discovered the description of 'max-packetcache-entries' and 'forward-zones-recurse' was wrong in the output of '–help' and '–config'. In addition, some stray backup files made it into the RC1 release. Addressed in commit 1529. Full release notes follow, including some overlap with the incremental release notes above. Improvements
- Multithreading, allowing near linear scaling to multiple CPUs or cores. Configured using 'threads=' (many commits). This also deprecates the '–fork' option.
- Added ability to read a configuration item of a running PowerDNS Recursor using 'rec_control get-parameter' (commit 1243), suggested by Wouter de Jong.
- Added ability to read all statistics in one go of a running PowerDNS Recursor using 'rec_control get-all' (commit 1496), suggested by Michael Renner.
- Speedups in packet generation (Commits 1258, 1259, 1262)
- TCP deferred accept() filter is turned on again for slight DoS protection. Code in commit 1414.
- PowerDNS Recursor can now do TCP/IP queries to remote IPv6 addresses (commit 1412).
- Solaris 9 '/dev/poll' support added, Solaris 8 now deprecated. Changes in commit 1421, commit 1422, commit 1424, commit 1413.
- Lua functions can now also see the address _to_ which a question was sent, using getlocaladdress(). Implemented in commit 1309 and commit 1315.
- Maximum cache sizes now default to a sensible value. Suggested by Roel van der Made, implemented in commit 1354.
- Domains can now be forwarded to IPv6 addresses too, using either ::1 syntax or [::1]:25. Thanks to Wijnand Modderman for discovering this issue, fixed in commit 1349.
- Lua scripts can now load libraries at runtime, for example to calculate md5 hashes. Code by Winfried Angele in commit 1405.
- Periodic statistics output now includes average queries per second, as well as packet cache numbers (commit 1493).
- New metrics are available for graphing, plus added to the default graphs (commit 1495, commit 1498, commit 1503)
- Fix errors/crashes on more recent versions of Solaris 10, where the ports functions could return ENOENT under some circumstances. Reported and debugged by Jan Gyselinck, fixed in commit 1372.

New features

- Add pdnslog() function for Lua scripts, so errors or other messages can be logged properly.
- New settings to set the owner, group and access modes of the control socket (socket-owner, socket-group, socket-mode). Code by Aki Tuomi and documentation in commit 1535. Fixup in commit 1536 for FreeBSD as found by Ralf van der Enden.
- rec_control now accepts a –timeout parameter, which can be useful when reloading huge Lua scripts. Implemented in commit 1366.

- Domains can now be forwarded with the 'recursion-desired' bit on or off, using either **forward-zones-recurse** or by prefixing the name of a zone with a '+' in **forward-zones-file**. Feature suggested by Darren Gamble, implemented in commit 1451.
- Access control lists can now be reloaded at runtime (implemented in commit 1457).
- PowerDNS Recursor can now use a pool of query-local-addresses to further increase resilience against spoofing. Suggested by Ad Spelt, implemented in commit 1426.
- PowerDNS Recursor now also has a packet cache, greatly speeding up operations. Implemented in commit 1426, commit 1433 and further.
- Cache can be limited in how long it maximally stores records, for BIND compatibility (TTL limiting), by setting **max-cache-ttl**.Idea by Winfried Angele, implemented in commit 1438.
- Cache cleaning turned out to be scanning more of the cache than necessary for cache maintenance. In addition, far more frequent but smaller cache cleanups improve responsiveness. Thanks to Winfried Angele for discovering this issue. (commits 1501, 1507)
- Performance graphs enhanced with separate CPU load and cache effectiveness plots, plus display of various overload situations (commits 1503)

Compiler/Operating system/Library updates

- PowerDNS Recursor can now compile against newer versions of Boost (verified up to and including 1.42.0). Reported & fixed by Darix in commit 1274. Further fixes in commit 1275, commit 1276, commit 1277, commit 1283.
- Fix compatibility with newer versions of GCC (closes ticket ticket 227, spotted by Ruben Kerkhof, code in commit 1345, more fixes in commit 1394, 1416, 1440).
- Rrdtool update graph is now compatible with FreeBSD out of the box. Thanks to Bryan Seitz (commit 1517).
- Fix up Makefile for older versions of Make (commit 1229).
- Solaris compilation improvements (out of the box, no handwork needed).
- Solaris 9 MTasker compilation fixes, as suggested by John Levon. Changes in commit 1431.

Bug fixes

- Under rare circumstances, the recursor could crash on 64 bit Linux systems running glibc 2.7, as found in Debian Lenny. These circumstances became a lot less rare for the 3.2 release. Discovered by Andreas Jakum and debugged by #powerdns, fix in commit 1519.
- Imre Gergely discovered that PowerDNS was doing spurious root repriming when invalidating nssets. Fixed in commit 1531.
- Configuration parser is now resistant against trailing tabs and other whitespace (commit 1242)
- Fix typo in a Lua error message. Close ticket 210, as reported by Stefan Schmidt (commit 1319).
- Profiled-build instructions were broken, discovered & fixes suggested by Stefan Schmidt. ticket 239, fix in commit 1462.
- Fix up duplicate SOA from a remote authoritative server from showing up in our output (commit 1475).
- All security fixes from 3.1.7.2 are included.
- Under highly exceptional circumstances on FreeBSD the PowerDNS Recursor could crash because of a TCP/IP error. Reported and fixed by Andrei Poelov in ticket 192, fixed in commit 1280.
- PowerDNS Recursor can be a root-server again. Error spotted by the ever vigilant Darren Gamble (ticket 229), fix in commit 1458.

- Rare TCP/IP errors no longer lead to PowerDNS Recursor logging errors or becoming confused. Debugged by Josh Berry of Plusnet PLC. Code in commit 1457.
- Do not hammer parent servers in case child zones are misconfigured, requery at most once every 10 seconds. Reported & investigated by Stefan Schmidt and Andreas Jakum, fixed in commit 1265.
- Properly process answers from remote authoritative servers that send error answers without including the original question (commit 1329, commit 1327).
- No longer spontaneously turn on 'export-etc-hosts' after reloading zones. Discovered by Paul Cairney, reported in ticket 225, addressed in commit 1348.
- Very abrupt server failure of large numbers of high-volume authoritative servers could trigger an out of memory situation. Addressed in commit 1505.
- Make timeouts for queries to remote authoritative servers configurable with millisecond granularity. In addition, the old code turned out to consider the timeout expired when the integral number of seconds since 1970 increased by 1 - which *on average* is after 500ms. This might have caused spurious timeouts! New default timeout is 1500ms. See **network-timeout** setting for more details. Code in commit 1402.

16.3.17 Recursor version 3.1.7.2

Released on the 6th of January 2010.

This release consist of a number of vital security updates. These updates address issues that can in all likelihood lead to a full system compromise. In addition, it is possible for third parties to pollute your cache with dangerous data, exposing your users to possible harm.

This version has been well tested, and at the time of this release is already powering millions of internet connections, and should therefore be a risk-free upgrade from 3.1.7.1 or any earlier version of the PowerDNS Recursor.

All known versions of the PowerDNS Recursor are impacted to a greater or lesser extent, so an immediate update is advised.

These vulnerabilities were discovered by a third party that can't yet be named, but who we thank for their contribution to a more secure PowerDNS Recursor.

For more information, see *PowerDNS Security Advisory 2010-01* and *PowerDNS Security Advisory 2010-02*.

16.3.18 Recursor version 3.1.7.1

Released on the 2nd of August 2009.

This release consists entirely of fixes for tiny bugs that have been reported over the past year. In addition, compatibility has been restored with the latest versions of the gcc compiler and the 'boost' libraries.

No features have been added, but some debugging code that very slightly impacted performance (and polluted the console when operating in the foreground) has been removed.

FreeBSD users may want to upgrade because of a very remote chance of 3.1.7 and previous crashing once every few years. For other operators not currently experiencing problems, there is no reason to upgrade.

- Improved error messages when parsing zones for authoritative serving (commit 1235).
- Better resilience against whitespace in configuration (changesets 1237, 1240, 1242)
- Slight performance increase (commit 1378)
- Fix rare case where timeouts were not being reported to the right query-thread (commit 1260)
- Fix compilation against newer versions of the Boost C++ libraries (commit 1381)
- Close very rare issue with TCP/IP close reporting ECONNRESET on FreeBSD. Reported by Andrei Poelov in ticket 192.
- Silence debugging output (commit 1286).

- Fix compilation against newer versions of gcc (commit 1384)
- No longer set export-etc-hosts to 'on' on reload-zones. Discovered by Paul Cairney, closes ticket 225.
- Sane default for the maximum cache size in the Recursor, suggested by Roel van der Made (commit 1354).
- No longer exit because of the changed behaviour of the Solaris 'completion ports' in more recent versions of Solaris. Fix in commit 1372, reported by Jan Gyselinck.

16.3.19 Recursor version 3.1.7

Released the 25th of June 2008.

This version contains powerful scripting abilities, allowing operators to modify DNS responses in many interesting ways. Among other things, these abilities can be used to filter out malware domains, to perform load balancing, to comply with legal and other requirements and finally, to implement 'NXDOMAIN' redirection.

It is hoped that the addition of Lua scripting will enable responsible DNS modification for those that need it.

For more details about the Lua scripting, which can be modified, loaded and unloaded at runtime, see Scripting. Many thanks are due to the #lua irc channel, for excellent near-realtime Lua support. In addition, a number of PowerDNS users have been enthusiastically testing prereleases of the scripting support, and have found and solved many issues.

In addition, 3.1.7 fixes a number of bugs

- In 3.1.5 and 3.1.6, an authoritative server could continue to renew its authority, even though a domain had been delegated to other servers in the meantime.

  In the rare cases where this happened, and the old servers were not shut down, the observed effect is that users were fed outdated data. Bug spotted and analysed by Darren Gamble, fix in commit 1182 and commit 1183.
- Thanks to long time PowerDNS contributor Stefan Arentz, for the first time, Mac OS X 10.5 users can compile and run the PowerDNS Recursor! Patch in commit 1185.
- Sten Spans spotted that for outgoing TCP/IP queries, the **query-local-address** setting was not honored. Fixed in commit 1190.
- **rec_control wipe-cache** now also wipes domains from the negative cache, hurrying up the expiry of negatively cached records. Suggested by Simon Kirby, implemented in commit 1204.
- When a forwarder server is configured for a domain, using the **forward-zones** setting, this server IP address was filtered using the **dont-query** setting, which is generally not what is desired: the server to which queries are forwarded will often live in private IP space, and the operator should be trusted to know what he is doing. Reported and argued by Simon Kirby, fix in commit 1211.
- Marcus Rueckert of OpenSUSE reported that very recent gcc versions emitted a (correct) warning on an overly complicated line in syncres.cc, fixed in commit 1189.
- Stefan Schmidt discovered that the netmask matching code, used by the new Lua scripts, but also by all other parts of PowerDNS, had problems with explicit '/32' matches. Fixed in commit 1205.

16.3.20 Recursor version 3.1.6

Released on the 1st of May 2008.

This version fixes two important problems, each on its own important enough to justify a quick upgrade.

- Version 3.1.5 had problems resolving several slightly misconfigured domains, including for a time 'juniper.net'. Nameserver timeouts were not being processed correctly, leading PowerDNS to not update the internal clock, which in turn meant that any queries immediately following an error would time out as well. Because of retries, this would usually not be a problem except on very busy servers, for domains with different nameservers at different levels of the DNS-hierarchy, like 'juniper.net'.

This issue was fixed rapidly because of the help of XS4ALL (Eric Veldhuyzen, Kai Storbeck), Brad Dameron and Kees Monshouwer. Fix in commit 1178.

- The new high-quality random generator was not used for all random numbers, especially in source port selection. This means that 3.1.5 is still a lot more secure than 3.1.4 was, and its algorithms more secure than most other nameservers, but it also means 3.1.5 is not as secure as it could be. A quick upgrade is recommended. Discovered by Thomas Biege of Novell (SUSE), fixed in commit 1179.

16.3.21 Recursor version 3.1.5

Released on the 31st of March 2008.

Much like 3.1.4, this release does not add a lot of major features. Instead, performance has been improved significantly (estimated at around 20%), and many rare and not so rare issues were addressed. Multi-part TXT records now work as expected - the only significant functional bug found in 15 months. One of the oldest feature requests was fulfilled: version 3.1.5 can finally forward queries for designated domains to multiple servers, on differing port numbers if needed. Previously only one forwarder address was supported. This lack held back a number of migrations to PowerDNS.

We would like to thank Amit Klein of Trusteer for bringing a serious vulnerability to our attention which would enable a smart attacker to 'spoof' previous versions of the PowerDNS Recursor into accepting possibly malicious data.

Details can be found on this Trusteer page.

It is recommended that all users of the PowerDNS Recursor upgrade to 3.1.5 as soon as practicable, while we simultaneously note that busy servers are less susceptible to the attack, but not immune.

The PowerDNS Security Advisory can be found in *PowerDNS Security Advisory 2008-01*.

This version can properly benefit from all IPv4 and IPv6 addresses in use at the root-servers as of early February 2008. In order to implement this, changes were made to how the Recursor deals internally with A and AAAA queries for nameservers, see below for more details.

Additionally, newer releases of the G++ compiler required some fixes (see ticket 173).

This release was made possible by the help of Wichert Akkerman, Winfried Angele, Arnoud Bakker (Fox-IT), Niels Bakker (no relation!), Leo Baltus (Nederlandse Publieke Omroep), Marco Davids (SIDN), David Gavarret (Neuf Cegetel), Peter Gervai, Marcus Goller (UPC), Matti Hiljanen (Saunalahti/Elisa), Ruben Kerkhof, Alex Kiernan, Amit Klein (Trusteer), Kenneth Marshall (Rice University), Thomas Rietz, Marcus Rueckert (OpenSUSE), Augie Schwer (Sonix), Sten Spans (Bit), Stefan Schmidt (Freenet), Kai Storbeck (xs4all), Alex Trull, Andrew Turnbull (No Wires) and Aaron Thompson, and many more who filed bugs anonymously, or who we forgot to mention.

Security related issues

- Amit Klein has informed us that System random generator output can be predicted based on its past behaviour, allowing a smart attacker to 'spoof' our nameserver. Full details in *PowerDNS Security Advisory 2008-01*.
- The Recursor will by default no longer query private-space nameservers. This closes a slight security risk and simultaneously improves performance and stability. For more information, see **dont-query** in pdns_recursor settings. Implemented in commit 923.
- Applied fix for ticket 110 ('PowerDNS should change directory to '/' in chroot), implemented in commit 944.

Performance

- The DNS packet writing and parsing infrastructure performance was improved in several ways, see commits 925, 926, 928, 931, 1021, 1050.

- Remove multithreading overhead from the Recursor (commit 999).

Bug fixes

- Built-in authoritative server now properly derives the TTL from the SOA record if not specified. Implemented in commit 1165. Additionally, even when TTL was specified for the built-in authoritative server, it was ignored. Reported by Stefan Schmidt, closing ticket 147.
- Empty TXT record components can now be served. Implemented in commit 1166, closing ticket 178. Spotted by Matti Hiljanen.
- The Recursor would not properly override old data with new, sometimes serving old and new data concurrently. Fixed in commit 1137.
- SOA records with embedded carriage-return characters are now parsed correctly. Implemented in commit 1167, closing ticket 162.
- Some routing conditions could cause UDP connected sockets to generate an error which PowerDNS did not deal with properly, leading to a leaked file descriptor. As these run out over time, the recursor could crash. This would also happen for IPv6 queries on a host with no IPv6 connectivity. Thanks to Kai of xs4all and Wichert Akkerman for reporting this issue. Fix in commit 1133.
- Empty unknown record types can now be stored without generating a scary error (commit 1129)
- Applied fix for ticket 111, ticket 112 and ticket 153 - large (multipart) TXT records are now retrieved and served properly. Fix in commit 996.
- Solaris compilation instructions in Recursor documentation were wrong, leading to an instant crash on startup. Luckily nobody reads the documentation, except for Marcus Goller who found the error. Fixed in commit 1124.
- On Solaris, finally fix the issue where queries get distributed strangely over CPUs, or not get distributed at all. Much debugging and analysing performed by Alex Kiernan, who also supplied fixes. Implemented in commit 1091, commit 1093.
- Various fixes for modern G++ versions, most spotted by Marcus Rueckert (commits 964, 965, 1028, 1052), and Ruben Kerkhof (commit 1136, closing ticket 175).
- Recursor would not properly clean up pidfile and control socket, closing ticket 120, code in commit 988, commit 1098 (part of fix by Matti Hiljanen, spotted by Leo Baltus)
- Recursor can now serve multi-line records from its limited authoritative server (commit 1014).
- When parsing zones, the 'm' time specification stands for minutes, not months! Closing Debian bug 406462 (commit 1026)
- Authoritative zone parser did not support '@' in the content of records. Spotted by Marco Davids, fixed in commit 1030.
- Authoritative zone parser could be confused by trailing TABs on record lines (commit 1062).
- EINTR error code could block entire server if received at the wrong time. Spotted by Arnoud Bakker, fix in commit 1059.
- Fix crash on NetBSD on Alpha CPUs, might improve startup behaviour on empty caches on other architectures as well (commit 1061).
- Outbound TCP queries were being performed sub-optimally because of an interaction with the 'MPlexer'. Fixes in commit 1115, commit 1116.

New features

- Implemented **rec_control** command **get uptime**, as suggested by Niels Bakker (commit 935). Added to default rrdtool scripts in commit 940.

- The Recursor Authoritative component, meant for having the Recursor serve some zones authoritatively, now supports $INCLUDE and $GENERATE. Implemented in commit 951 and commit 952, commit 967 (discovered by Thomas Rietz),
- Implemented **forward-zones-file** option in order to support larger amounts of zones which should be forwarded to another nameserver (commit 963).
- Both **forward-zones** and **forward-zones-file** can now specify multiple forwarders per domain, implemented in commit 1168, closing ticket 81. Additionally, both these settings can also specify non-standard port numbers, as suggested in ticket ticket 122. Patch authored by Aaron Thompson, with additional work by Augie Schwer.
- Sten Spans contributed **allow-from-file**, implemented in commit 1150. This feature allows the Recursor to read access rules from a (large) file.

General improvements

- Ruben Kerkhof fixed up weird permission bits as well as our SGML documentation code in commit 936 and commit 937.
- Full IPv6 parity. If configured to use IPv6 for outgoing queries (using **query-local-address6=::0** for example), IPv6 and IPv4 addresses are finally treated 100% identically, instead of 'mostly'. This feature is implemented using 'ANY' queries to find A and AAAA addresses in one query, which is a new approach. Treat with caution.
- Now perform EDNS0 root refreshing queries, so as to benefit from all returned addresses. Relevant since early February 2008 when the root-servers started to respond with IPv6 addresses, which made the default non-EDNS0 maximum packet length reply no longer contain all records. Implemented in commit 1130. Thanks to dns-operations AT mail.oarc.isc.org for quick suggestions on how to deal with this change.
- **rec_control** now has a timeout in case the Recursor does not respond. Implemented in commit 945.
- (Error) messages are now logged with saner priorities (commit 955).
- Outbound query IP interface stemmed from 1997 (!) and was in dire need of a cleanup (commit 1117).
- L.ROOT-SERVERS.NET moved (commit 1118).

16.3.22 Recursor version 3.1.4

Released the 13th of November 2006.

This release contains almost no new features, but consists mostly of minor and major bug fixes. It also addresses two major security issues, which makes this release a highly recommended upgrade.

Security issues

- Large TCP questions followed by garbage could cause the recursor to crash. This critical security issue has been assigned CVE-2006-4251, and is fixed in commit 915. More information can be found in *"PowerDNS Security Advisory 2006-01: Malformed TCP queries can lead to a buffer overflow which might be exploitable"*.
- CNAME loops with zero second TTLs could cause crashes in some conditions. These loops could be constructed by malicious parties, making this issue a potential denial of service attack. This security issue has been assigned CVE-2006-4252 and is fixed by commit 919. More information can be found in *"PowerDNS Security Advisory 2006-02: Zero second CNAME TTLs can make PowerDNS exhaust allocated stack space, and crash"*. Many thanks to David Gavarret for helping pin down this problem.

Bugs

- On certain error conditions, PowerDNS would neglect to close a socket, which might therefore eventually run out. Spotted by Stefan Schmidt, fixed in commits 892, 897, 899.
- Some nameservers (including PowerDNS in rare circumstances) emit a SOA record in the authority section. The recursor mistakenly interpreted this as an authoritative "NXRRSET". Spotted by Bryan Seitz, fixed in commit 893.
- In some circumstances, PowerDNS could end up with a useless (not working, or no longer working) set of nameserver records for a domain. This release contains logic to invalidate such broken NSSETs, without overloading authoritative servers. This problem had previously been spotted by Bryan Seitz, 'Cerb' and Darren Gamble. Invalidations of NSSETs can be plotted using the "nsset-invalidations" metric, available through **rec_control get**. Implemented in commit 896 and commit 901.
- PowerDNS could crash while dumping the cache using **rec_control dump-cache**. Reported by Wouter of WideXS and Stefan Schmidt and many others, fixed in commit 900.
- Under rare circumstances (depleted TCP buffers), PowerDNS might send out incomplete questions to remote servers. Additionally, on big-endian systems (non-Intel and non-AMD generally), sending out large TCP answers questions would not work at all, and possibly crash. Brought to our attention by David Gavaret, fixed in commit 903.
- The recursor contained the potential for a dead-lock processing an invalid domain name. It is not known how this might be triggered, but it has been observed by 'Cerb' on #powerdns. Several dead-locks where PowerDNS consumed all CPU, but did not answer questions, have been reported in the past few months. These might be fixed by commit 904.
- IPv6 'allow-from' matching had problems with the least significant bits, sometimes allowing disallowed addresses, but mostly disallowing allowed addresses. Spotted by Wouter from WideXS, fixed in commit 916.

Improvements

- PowerDNS has support to drop answers from so called 'delegation only' zones. A statistic ("dlg-only-drops") is now available to plot how often this happens. Implemented in commit 890.
- Hint-file parameter was mistakenly named "hints-file" in the documentation. Spotted by my Marco Davids, fixed in commit 898.
- **rec_control quit** should be near instantaneous now, as it no longer meticulously cleans up memory before exiting. Problem spotted by Darren Gamble, fixed in commit 914, closing ticket 84.
- init.d script no longer refers to the Recursor as the Authoritative Server. Spotted by Wouter of WideXS, fixed in commit 913.
- A potentially serious warning for users of the GNU C Library version 2.5 was fixed. Spotted by Marcus Rueckert, fixed in commit 920.

16.3.23 Recursor version 3.1.3

Released the 12th of September 2006.

Compared to 3.1.2, this release again consists of a number of mostly minor bug fixes, and some slight improvements.

Many thanks are again due to Darren Gamble who together with his team has discovered many misconfigured domains that do work with some other name servers. DNS has long been tolerant of misconfigurations, PowerDNS intends to uphold that tradition. Almost all of the domains found by Darren now work as well in PowerDNS as in other name server implementations.

Thanks to some recent migrations, this release, or something very close to it, is powering over 40 million internet connections that we know of. We appreciate hearing about successful as well as unsuccessful migrations, please feel free to notify pdns.bd@powerdns.com of your experiences, good or bad.

Bug-fixes

- The MThread default stack size was too small, which led to problems, mostly on 64-bit platforms. This stack size is now configurable using the **stack-size** setting should our estimate be off. Discovered by Darren Gamble, Sten Spans and a number of others. Fixed in commit 868.
- Plug a small memory leak discovered by Kai and Darren Gamble, fixed in commit 870.
- Switch from the excellent nedmalloc to dlmalloc, based on advice by the nedmalloc author. Nedmalloc is optimised for multithreaded operation, whereas the PowerDNS recursor is single threaded. The version of nedmalloc shipped contained a number of possible bugs, which are probably resolved by moving to dlmalloc. Some reported crashes on hitting 2G of allocated memory on 64 bit systems might be solved by this switch, which should also increase performance. See commit 873 for details.

Improvements

- The cache is now explicitly aware of the difference between authoritative and unauthoritative data, allowing it to deal with some domains that have different data in the parent zone than in the authoritative zone. Patch in commit 867.
- No longer try to parse DNS updates as if they were queries. Discovered and fixed by Jan Gyselinck, fix in commit 871.
- Rebalance logging priorities for less log cluttering and add IP address to a remote server error message. Noticed and fixed by Jan Gyselinck (commit 877).
- Add **logging-facility** setting, allowing syslog to send PowerDNS logging to a separate file. Added in commit 871.

16.3.24 Recursor version 3.1.2

Released Monday 26th of June 2006.

Compared to 3.1.1, this release consists almost exclusively of bug-fixes and speedups. A quick update is recommended, as some of the bugs impact operators of authoritative zones on the internet. This version has been tested by some of the largest internet providers on the planet, and is expected to perform well for everybody.

Many thanks are due to Darren Gamble, Stefan Schmidt and Bryan Seitz who all provided excellent feedback based on their large-scale tests of the recursor.

Bug-fixes

- Internal authoritative server did not differentiate between 'NXDOMAIN' and 'NXRRSET', in other words, it would answer 'no such host' when an AAAA query came in for a domain that did exist, but did not have an AAAA record. This only affects users with **auth-zones** configured. Discovered by Bryan Seitz, fixed in commit 848.
- ANY queries for hosts where nothing was present in the cache would not work. This did not cause real problems as ANY queries are not reliable (by design) for anything other than debugging, but did slow down the nameserver and cause unnecessary load on remote nameservers. Fixed in commit 854.
- When exceeding the configured maximum amount of TCP sessions, TCP support would break and the nameserver would waste CPU trying to accept TCP connections on UDP ports. Noted by Bryan Seitz, fixed in commit 849.

- DNS queries come in two flavours: recursion desired and non-recursion desired. The latter is not very useful for a recursor, but is sometimes (erroneously) used by monitoring software or load balancers to detect nameserver availability. A non-rd query would not only not recurse, but also not query authoritative zones, which is confusing. Fixed in commit 847.
- Non-standard DNS TCP queries, that did occur however, could drive the recursor to 100% CPU usage for extended periods of time. This did not disrupt service immediately, but does waste a lot of CPU, possibly exhausting resources. Discovered by Bryan Seitz, fixed in commit 858, which is post-3.1.2-rc1.
- The PowerDNS recursor did not honour the rare but standardised 'ANY' query class (normally 'ANY' refers to the query type, not class), upsetting the Wildfire Jabber server. Discovered and debugged by Daniel Nauck, fixed in commit 859, which is post-3.1.2-rc1.
- Everybody's favorite, when starting up under high load, a bogus line of statistics was sometimes logged. Fixed in commit 851.
- Remove some spurious debugging output on dropping a packet by an unauthorized host. Discovered by Kai. Fixed in commit 854.

Improvements

- Misconfigured domains, with a broken nameserver in the parent zone, should now work better. Changes motivated and suggested by Darren Gamble. This makes PowerDNS more compliant with RFC 2181 by making it prefer authoritative data over non-authoritative data. Implemented in commit 856.
- PowerDNS can now listen on multiple ports, using the **local-address** setting. Added in commit 845.
- A number of speedups which should have a noticeable impact, implemented in commits 850, 852, 853, 855
- The recursor now works around an issue with the Linux kernel 2.6.8, as shipped by Debian. Fixed by Christof Meerwald in commit 860, which is post 3.1.2-rc1.

16.3.25 Recursor version 3.1.1

Released on the 23rd of May 2006.

Warning: 3.1.1 is identical to 3.1 except for a bug in the packet chaining code which would mainly manifest itself for IPv6 enabled Konqueror users with very fast connections to their PowerDNS installation. However, all 3.1 users are urged to upgrade to 3.1.1. Many thanks to Alessandro Bono for his quick aid in solving this problem.

Many thanks are due to the operators of some of the largest internet access providers in the world, each having many millions of customers, who have tested the various 3.1 pre-releases for suitability. They have uncovered and helped fix bugs that could impact us all, but are only (quickly) noticeable with such vast amounts of DNS traffic.

After version 3.0.1 has proved to hold up very well under tremendous loads, 3.1 adds important new features

- Ability to serve authoritative data from 'BIND' style zone files (using **auth-zones** statement).
- Ability to forward domains so configured to external servers (using **forward-zones**).
- Possibility of 'serving' the contents of `/etc/hosts` over DNS, which is very well suited to simple domestic router/DNS setups. Enabled using **export-etc-hosts**.
- As recommended by recent standards documents, the PowerDNS recursor is now authoritative for RFC-1918 private IP space zones by default (suggested by Paul Vixie).
- Full outgoing IPv6 support (off by default) with IPv6 servers getting equal treatment with IPv4, nameserver addresses are chosen based on average response speed, irrespective of protocol.
- Initial Windows support, including running as a service ('NET START "POWERDNS RECURSOR"'). **rec_channel** is still missing, the rest should work. Performance appears to be below that of the UNIX versions, this situation is expected to improve.

Bug fixes

- No longer send out SRV and MX record priorities as zero on big-endian platforms (UltraSPARC). Discovered by Eric Sproul, fixed in commit 773.
- SRV records need additional processing, especially in an Active Directory setting. Reported by Kenneth Marshall, fixed in commit 774.
- The root-records were not being refreshed, which could lead to problems under inconceivable conditions. Fixed in commit 780.
- Fix resolving domain names for nameservers with multiple IP addresses, with one of these addresses being lame. Other nameserver implementations were also unable to resolve these domains, so not a big bug. Fixed in commit 780.
- For a period of 5 minutes after expiring a negative cache entry, the domain would not be re-cached negatively, leading to a lot of duplicate outgoing queries for this short period. This fix has raised the average cache hit rate of the recursor by a few percent. Fixed in commit 783.
- Query throttling was not aggressive enough and not all sorts of queries were throttled. Implemented in commit 786.
- Fix possible crash during startup when parsing empty configuration lines (commit 807).
- Fix possible crash when the first query after wiping a cache entry was for the just deleted entry. Rare in production servers. Fixed in commit 820.
- Recursor would send out differing TTLs when receiving a misconfigured, standards violating, RRSET with different TTLs. Implement fix as mandated by RFC 2181, paragraph 5.2. Reported by Stephen Harker (commit 819).
- The **top-remotes** would list remotes more than once, once per source port. Discovered by Jorn Ekkelenkamp, fixed in commit 827, which is post 3.1-pre1.
- Default **allow-from** allowed queries from fe80::/16, corrected to fe80::/10. Spotted by Niels Bakker, fixed in commit 829, which is post 3.1-pre1.
- While PowerDNS blocks failing queries quickly, multiple packets could briefly be in flight for the same domain and nameserver. This situation is now explicitly detected and queries are chained to identical queries already in flight. Fixed in commit 833 and commit 834, post 3.1-pre1.

Improvements

- ANY queries are now implemented as in other nameserver implementations, leading to a decrease in outgoing queries. The RFCs are not very clear on desired behaviour, what is implemented now saves bandwidth and CPU and brings us in line with existing practice. Previously ANY queries were not cached by the PowerDNS recursor. Implemented in commit 784.
- **rec_control** was very sparse in its error reporting, and user unfriendly as well. Reported by Erik Bos, fixed in commit 818 and commit 820.
- IPv6 addresses were printed in a non-standard way, fixed in commit 788.
- TTLs of records are now capped at two weeks, commit 820.
- **allow-from** IPv4 netmasks now automatically work for IP4-to-IPv6 mapper IPv4 addresses, which appear when running on the wildcard **::** IPv6 address. Lack of feature noted by Marcus 'darix' Rueckert. Fixed in commit 826, which is post 3.1-pre1.
- Errors before daemonizing are now also sent to syslog. Suggested by Marcus 'darix' Rueckert. Fixed in commit 825, which is post 3.1-pre1.
- When launching without any form of configured network connectivity, all root-servers would be cached as 'down' for some time. Detect this special case and treat it as a resource-constraint, which is not accounted against specific nameservers. Spotted by Seth Arnold, fixed in commit 835, which is post 3.1-pre1.

- The recursor now does not allow authoritative servers to keep supplying its own NS records into perpetuity, which causes problems when a domain is redelegated but the old authoritative servers are not updated to this effect. Noticed and explained at length by Darren Gamble of Shaw Communications, addressed by commit 837, which is post 3.1-pre2.
- Some operators may want to follow RFC 2181 paragraph 5.2 and 5.4. This harms performance and does not solve any real problem, but does make PowerDNS more compliant. If you want this, enable **auth-can-lower-ttl**. Implemented in commit 838, which is post 3.1-pre2.

16.3.26 Recursor version 3.0.1

Released 25th of April 2006, download.

This release consists of nothing but tiny fixes to 3.0, including one with security implications. An upgrade is highly recommended.

- Compilation used both `cc` and `gcc`, leading to the possibility of compiling with different compiler versions (commit 766).
- **rec_control** would leave files named `lsockXXXXXX` around in the configured socket-dir. Operators may wish to remove these files from their socket-dir (often `/var/run`), quite a few might have accumulated already (commit 767).
- Certain malformed packets could crash the recursor. As far as we can determine these packets could only lead to a crash, but as always, there are no guarantees. A quick upgrade is highly recommended (commits 760, 761). Reported by David Gavarret.
- Recursor would not distinguish between NXDOMAIN and NXRRSET (commit 756). Reported and debugged by Jorn Ekkelenkamp.
- Some error messages and trace logging statements were improved (commits 756, 758, 759).
- stderr was closed during daemonizing, but not dupped to /dev/null, leading to slight chance of odd behaviour on reporting errors (commit 757)

Operating system specific fixes

- The stock Debian sarge Linux kernel, 2.6.8, claims to support epoll but fails at runtime. The epoll self-testing code has been improved, and PowerDNS will fall back to a select based multiplexer if needed (commit 758) Reported by Michiel van Es.
- Solaris 8 compilation and runtime issues were addressed. See the README for details (commit 765). Reported by Juergen Georgi and Kenneth Marshall.
- Solaris 10 x86_64 compilation issues were addressed (commit 755). Reported and debugged by Eric Sproul.

16.3.27 Recursor version 3.0

Released 20th of April 2006, download.

This is the first separate release of the PowerDNS Recursor. There are many reasons for this, one of the most important ones is that previously we could only do a release when both the recursor and the authoritative nameserver were fully tested and in good shape. The split allows us to release new versions when each part is ready.

Now for the real news. This version of the PowerDNS recursor powers the network access of over two million internet connections. Two large access providers have been running pre-releases of 3.0 for the past few weeks and results are good. Furthermore, the various pre-releases have been tested nearly non-stop with DNS traffic replayed at 3000 queries/second.

As expected, the 2 million households shook out some very rare bugs. But even a rare bug happens once in a while when there are this many users.

We consider this version of the PowerDNS recursor to be the most advanced resolver publicly available. Given current levels of spam, phishing and other forms of internet crime we think no recursor should offer less than the best in spoofing protection. We urge all operators of resolvers without proper spoofing countermeasures to consider PowerDNS, as it is a Better Internet Nameserver Daemon.

A good article on DNS spoofing can be found here. Some more information, based on a previous version of PowerDNS, can be found on the PowerDNS development blog.

Warning: Because of recent DNS based denial of service attacks, running an open recursor has become a security risk. Therefore, unless configured otherwise this version of PowerDNS will only listen on localhost, which means it does not resolve for hosts on your network. To fix, configure the **local-address** setting with all addresses you want to listen on. Additionally, by default service is restricted to RFC 1918 private IP addresses. Use **allow-from** to selectively open up the recursor for your own network. See pdns_recursor settings for details.

Important new features of the PowerDNS recursor 3.0

- Best spoofing protection and detection we know of. Not only is spoofing made harder by using a new network address for each query, PowerDNS detects when an attempt is made to spoof it, and temporarily ignores the data. For details, see Anti-spoofing.
- First nameserver to benefit from epoll/kqueue/Solaris completion ports event reporting framework, for stellar performance.
- Best statistics of any recursing nameserver we know of, see Statistics.
- Last-recently-used based cache cleanup algorithm, keeping the 'best' records in memory
- First class Solaris support, built on a 'try and buy' Sun CoolThreads T 2000.
- Full IPv6 support, implemented natively.
- Access filtering, both for IPv4 and IPv6.
- Experimental SMP support for nearly double performance. See PowerDNS Recursor performance.

Many people helped package and test this release. Jorn Ekkelenkamp of ISP-Services helped find the '8000 SOAs' bug and spotted many other oddities and XS4ALL internet funded a lot of the recent development. Joaquín M López Muñoz of the boost::multi_index_container was again of great help.

CHAPTER

SEVENTEEN

END OF LIFE STATEMENTS

The currently supported release train of the PowerDNS Recursor is 4.1.

PowerDNS Recursor 4.0 will only receive correctness, stability and security updates.

PowerDNS Recursor 3.x, and 2.x are end of life.

Note: Users with a commercial agreement with PowerDNS.COM BV or Open-Xchange can receive extended support for releases which are End Of Life. If you are such a user, these EOL statements do not apply to you.

CHAPTER

EIGHTEEN

COMPILING THE POWERDNS RECURSOR

As the PowerDNS Recursor is distributed with a configure script, compiling it is a matter of:

```
tar xf pdns-recursor-$VERSION.tar.bz2
cd pdns-recursor-$VERSION
./configure
make
make install
```

18.1 Getting the sources

There are 3 ways of getting the source.

If you want the bleeding edge, you can clone the repository at GitHub and run `autoreconf -vi` in the `pdns/recursordist` directory of the clone.

You can also download snapshot tarballs here.

You can also download releases on the website. These releases are PGP-signed with one of these key-ids:

- FBAE 0323 821C 7706 A5CA 151B DCF5 13FA 7EED 19F3
- 1628 90D0 689D D12D D33E 4696 1C5E E990 D2E7 1575
- B76C D467 1C09 68BA A87D E61C 5E50 715B F2FF E1A7
- 16E1 2866 B773 8C73 976A 5743 6FFC 3343 9B0D 04DF

There is a PGP keyblock with these keys available on https://www.powerdns.com/powerdns-keyblock.asc.

18.2 Dependencies

To build the PowerDNS Recursor, a C++ compiler with support for C++ 2011 is required. This means gcc 4.9 and newer and clang 3.5 and newer. Furthermore, the Makefiles require GNU make, not BSD make.

By default, the PowerDNS recursor requires the following libraries and headers:

- Boost 1.35 or newer
- Lua 5.1+ or LuaJit
- OpenSSL

18.3 Optional dependencies

Several options that can be passed to `./configure` can enable and disable different features. These will require additional dependencies

18.3.1 ed25519 support with libsodium

The PowerDNS Recursor can link with libsodium to support ed25519 (DNSSEC algorithm 15). To detect libsodium, use the `--enable-libsodium` configure option.

18.3.2 ed25519 and ed448 support with libdecaf

libdecaf is a library that allows the PowerDNS Recursor to support ed25519 and Ed448 (DNSSEC algorithms 15 and 16). To detect libdecaf, use the `--enable-libdecaf` configure option.

18.3.3 Protobuf to emit DNS logs

The PowerDNS Recursor can log DNS query information over *Protocol Buffers*. To enable this functionality, install the protobuf library and compiler. The configure script will automatically detect this and bump the Boost version depencency to 1.42.

To disable building this functionality, use `--without-protobuf`.

18.3.4 systemd notify support

During configure, `configure` will attempt to detect the availibility of systemd or systemd-daemon headers. To force the use of systemd (and failing configure if the headers do not exist), use `--enable-systemd`. To set the directory where the unit files should be installed, use `--with-systemd=/path/to/unit/dir`.

CHAPTER

NINETEEN

CRYPTOGRAPHIC SOFTWARE AND EXPORT CONTROL

In certain legal climates, PowerDNS might potentially require an export control status, particularly since PowerDNS software contains cryptographic primitives.

PowerDNS does not itself implement any cryptographic algorithms but relies on third party implementations of AES, RSA, ECDSA, GOST, MD5 and various SHA-based hashing algorithms.

Starting with 4.0.0, PowerDNS will link in hash and cryptographic primitives from the open source OpenSSL library.

Optionally, PowerDNS can link in a copy of the open source Botan cryptographic library.

Optionally, PowerDNS can link in a copy of the open source Sodium library.

19.1 Specific United States Export Control Notes

PowerDNS is not "US Origin" software. For re-export, like most open source, publicly available "mass market" projects, PowerDNS is considered to be governed by section 740.13(e) of the US EAR, "Unrestricted encryption source code", under which PowerDNS source code would be considered re-exportable from the US without an export license under License Exception TSU (Technology and Software - Unrestricted).

Like most open source projects containing some encryption, the ECCN that best fits PowerDNS software is 5D002.

The official link to the publicly available source code is *https://downloads.powerdns.com/releases*.

If absolute certainty is required, we recommend consulting an expert in US Export Control, or asking the BIS for confirmation.

CHAPTER

TWENTY

INTERNALS OF THE POWERDNS RECURSOR

Warning: This section is aimed at programmers wanting to contribute to the recursor, or to help fix bugs. It is not required reading for a PowerDNS operator, although it might prove interesting.

The PowerDNS Recursor consists of very little code, the core DNS logic is less than a thousand lines.

This smallness is achieved through the use of some fine infrastructure: MTasker, MOADNSParser, MPlexer and the C++ Standard Library/Boost. This page will explain the conceptual relation between these components, and the route of a packet through the program.

20.1 The PowerDNS Recursor

The Recursor started out as a tiny project, mostly a technology demonstration. These days it consists of the core plus 9000 lines of features. This combined with a need for very high performance has made the recursor code less accessible than it was. The page you are reading hopes to rectify this situation.

20.2 Synchronous code using MTasker

The original name of the program was **syncres**, which is still reflected in the file name `syncres.cc`, and the class SyncRes. This means that PowerDNS is written naively, with one thread of execution per query, synchronously waiting for packets, Normally this would lead to very bad performance (unless running on a computer with very fast threading, like possibly the Sun CoolThreads family), so PowerDNS employs MTasker for very fast userspace threading.

MTasker, which was developed separately from PowerDNS, does not provide a full multithreading system but restricts itself to those features a nameserver needs. It offers cooperative multitasking, which means there is no forced preemption of threads. This in turn means that no two **MThreads** ever really run at the same time.

This is both good and bad, but mostly good. It means PowerDNS does not have to think about locking. No two threads will ever be talking to the DNS cache at the same time, for example.

It also means that the recursor could block if any operation takes too long.

The core interaction with MTasker are the waitEvent() and sendEvent() functions. These pass around PacketID objects. Everything PowerDNS needs to wait for is described by a PacketID event, so the name is a bit misleading. Waiting for a TCP socket to have data available is also passed via a PacketID, for example.

The version of MTasker in PowerDNS is newer than that described at the MTasker site, with a vital difference being that the waitEvent() structure passes along a copy of the exact PacketID sendEvent() transmitted. Furthermore, threads can trawl through the list of events being waited for and modify the respective PacketIDs. This is used for example with **near miss** packets: packets that appear to answer questions we asked, but differ in the DNS id. On seeing such a packet, the recursor trawls through all PacketIDs and if it finds any nearmisses, it updates the PacketID::nearMisses counter. The actual PacketID thus lives inside MTasker while any thread is waiting for it.

20.3 MPlexer

The Recursor uses a separate socket per outgoing query. This has the important benefit of making spoofing 64000 times harder, and additionally means that ICMP errors are reported back to the program. In measurements this appears to happen to one in ten queries, which would otherwise take a two-second timeout before PowerDNS moves on to another nameserver.

However, this means that the program routinely needs to wait on hundreds or even thousands of sockets. Different operating systems offer various ways to monitor the state of sockets or more generally, file descriptors. To abstract out the differing strategies (`select`, `epoll`, `kqueue`, `completion ports`), PowerDNS contains **MPlexer** classes, all of which descend from the FDMultiplexer class.

This class is very simple and offers only five important methods: addReadFD(), addWriteFD(), removeReadFD(), removeWriteFD() and run.

The arguments to the **add** functions consist of an fd, a callback, and a boost::any variable that is passed as a reference to the callback.

This might remind you of the MTasker above, and it is indeed the same trick: state is stored within the MPlexer. As long as a file descriptor remains within either the Read or Write active list, its state will remain stored.

On arrival of a packet (or more generally, when an FD becomes readable or writable, which for example might mean a new TCP connection), the callback is called with the aforementioned reference to its parameter.

The callback is free to call removeReadFD() or removeWriteFD() to remove itself from the active list.

PowerDNS defines such callbacks as newUDPQuestion(), newTCPConnection(), handleRunningTCPConnection().

Finally, the run() method needs to be called whenever the program is ready for new data. This happens in the main loop in pdns_recursor.cc. This loop is what MTasker refers to as **the kernel**. In this loop, any packets or other MPlexer events get translated either into new MThreads within MTasker, or into calls to sendEvent(), which in turn wakes up other MThreads.

20.4 MOADNSParser

Yes, this does stand for **the Mother of All DNS Parsers**. And even that name does not do it justice! The MOADNSParser is the third attempt I've made at writing DNS packet parser and after two miserable failures, I think I've finally gotten it right.

Writing and parsing DNS packets, and the DNS records it contains, consists of four things:

1. Parsing a DNS record (from packet) into memory
2. Generating a DNS record from memory (to packet)
3. Writing out memory to user-readable zone format
4. Reading said zone format into memory

This gets tedious very quickly, as one needs to implement all four operations for each new record type, and there are dozens of them.

While writing the MOADNSParser, it was discovered there is a remarkable symmetry between these four transitions. DNS Records are nearly always laid out in the same order in memory as in their zone format representation. And reading is nothing but inverse writing.

So, the MOADNSParser is built around the notion of a **Conversion**, and we write all Conversion types once. So we have a Conversion from IP address in memory to an IP address in a DNS packet, and vice versa. And we have a Conversion from an IP address in zone format to memory, and vice versa.

This in turn means that the entire implementation of the ARecordContent is as follows (wait for it!)

```
conv.xfrIP(d_ip);
```

Through the use of the magic called `c++ Templates`, this one line does everything needed to perform the four operations mentioned above.

At one point, I got really obsessed with PowerDNS memory use. So, how do we store DNS data in the PowerDNS recursor? I mentioned **memory** above a lot - this means we could just store the DNSRecordContent objects. However, this would be wasteful.

For example, storing the following:

```
www.example.org  3600 IN   CNAME  outpost.example.org.
```

Would duplicate a lot of data. So, what is actually stored is a partial DNS packet. To store the CNAMEDNSRecordContent that corresponds to the above, we generate a DNS packet that has **www.example.org IN CNAME** as its question. Then we add **3600 IN CNAME outpost.example.org**. as its answer. Then we chop off the question part, and store the rest in the **www.example.org IN CNAME** key in our cache.

When we need to retrieve **www.example.org IN CNAME**, the inverse happens. We find the proper partial packet, prefix it with a question for **www.example.org IN CNAME**, and expand the resulting packet into the answer **3600 IN CNAME outpost.example.org.**.

Why do we go through all these motions? Because of DNS compression, which allows us to omit the whole **.example.org.** part, saving us 9 bytes. This is amplified when storing multiple MX records which all look more or less alike. This optimization is not performed yet though.

Even without compression, it makes sense as all records are automatically stored very compactly.

The PowerDNS recursor only parses a number of **well known record types** and passes all other information across verbatim - it doesn't have to know about the content it is serving.

20.5 The C++ Standard Library / Boost

C++ is a powerful language. Perhaps a bit too powerful at times, you can turn a program into a real freakshow if you so desire.

PowerDNS generally tries not to go overboard in this respect, but we do build upon a very advanced part of the Boost C++ library: boost::multi index container.

This container provides the equivalent of SQL indexes on multiple keys. It also implements compound keys, which PowerDNS uses as well.

The main DNS cache is implemented as a multi index container object, with a compound key on the name and type of a record. Furthermore, the cache is sequenced, each time a record is accessed it is moved to the end of the list. When cleanup is performed, we start at the beginning. New records also get inserted at the end. For DNS correctness, the sort order of the cache is case insensitive.

The multi index container appears in other parts of PowerDNS, and MTasker as well.

20.6 Actual DNS Algorithm

The DNS RFCs do define the DNS algorithm, but you can't actually implement it exactly that way, it was written in 1987.

Also, like what happened to HTML, it is expected that even non-standards conforming domains work, and a sizable fraction of them is misconfigured these days.

Everything begins with SyncRes::beginResolve(), which knows nothing about sockets, and needs to be passed a domain name, dns type and dns class which we are interested in. It returns a vector of DNSResourceRecord objects, ready for writing either into an answer packet, or for internal use.

After checking if the query is for any of the hardcoded domains (localhost, version.bind, id.server), the query is passed to SyncRes::doResolve, together with two vital parameters: the `depth` and `beenthere` set. As the word **recursor** implies, we will need to recurse for answers. The **depth** parameter documents how deep we've recursed already.

The `beenthere` set prevents loops. At each step, when a nameserver is queried, it is added to the `beenthere` set. No nameserver in the set will ever be queried again for the same question in the recursion process - we know for a fact it won't help us further. This prevents the process from getting stuck in loops.

SyncRes::doResolve first checks if there is a CNAME in cache, using SyncRes::doCNAMECacheCheck, for the domain name and type queried and if so, changes the query (which is passed by reference) to the domain the CNAME points to. This is the cause of many DNS problems, a CNAME record really means **start over with this query**.

This is followed by a call do SyncRes::doCacheCheck, which consults the cache for a straight answer to the question (as possibly rerouted by a CNAME). This function also consults the so called negative cache, but we won't go into that just yet.

If this function finds the correct answer, and the answer hasn't expired yet, it gets returned and we are (almost) done. This happens in 80 to 90% of all queries. Which is good, as what follows is a lot of work.

To recap:

1. beginResolve() - entry point, does checks for hardcoded domains
2. doResolve() - start of recursion process, gets passed `depth` of 0 and empty `beenthere` set
3. doCNAMECacheCheck() - check if there is a CNAME in cache which would reroute the query
4. doCacheCheck() - see if cache contains straight answer to possibly rerouted query.

If the data we were queried for was in the cache, we are almost done. One final step, which might as well be optional as nobody benefits from it, is SyncRes::addCruft. This function does additional processing, which means that if the query was for the MX record of a domain, we also add the IP address of the mail exchanger.

20.6.1 The non-cached case

This is where things get interesting, because we start out with a nearly empty cache and have to go out to the net to get answers to fill it.

The way DNS works, if you don't know the answer to a question, you find somebody who does. Initially you have no other place to go than the root servers. This is embodied in the SyncRes::getBestNSNamesFromCache method, which gets passed the domain we are interested in, as well as the `depth` and `beenthere` parameters mentioned earlier.

From now on, assume our query will be for **"www.powerdns.com."**. SyncRes::getBestNSNamesFromCache will first check if there are NS records in cache for `www.powerdns.com.`, but there won't be. It then checks `powerdns.com. NS`, and while these records do exist on the internet, the recursor doesn't know about them yet. So, we go on to check the cache for `com. NS`, for which the same holds. Finally we end up checking for `. NS`, and these we do know about: they are the root servers and were loaded into PowerDNS on startup.

So, SyncRes::getBestNSNamesFromCache fills out a set with the **names** of nameservers it knows about for the **"."** zone.

This set, together with the original query **"www.powerdns.com"** gets passed to SyncRes::doResolveAt. This function can't yet go to work immediately though, it only knows the names of nameservers it can try. This is like asking for directions and instead of hearing **take the third right** you are told **go to 123 Fifth Avenue, and take a right** - the answer doesn't help you further unless you know where 123 Fifth Avenue is.

SyncRes::doResolveAt first shuffles the nameservers both randomly and on performance order. If it knows a nameserver was fast in the past, it will get queried first. More about this later.

Ok, here is the part where things get a bit scary. How does SyncRes::doResolveAt find the IP address of a nameserver? Well, by calling SyncRes::getAs (**get A records**), which in turn calls.. SyncRes::doResolve. Hang on! That's where we came from! Massive potential for loops here. Well, it turns out that for any domain which

can be resolved, this loop terminates. We do pass the `beenthere` set again, which makes sure we don't keep on asking the same questions to the same nameservers.

Ok, SyncRes::getAs will give us the IP addresses of the chosen root-server, because these IP addresses were loaded on startup. We then ask these IP addresses (nameservers can have several) for its best answer for **''www.powerdns.com.''**. This is done using the LWRes class and specifically LWRes::asyncresolve, which gets passed domain name, type and IP address. This function interacts with MTasker and MPlexer above in ways which needn't concern us now. When it returns, the LWRes object contains the best answers the queried server had for our domain, which in this case means it tells us about the nameservers of `com.`, and their IP addresses.

All the relevant answers it gives are stored in the cache (or actually, merged), after which SyncRes::doResolveAt (which we are still in) evaluates what to do now.

There are 6 options:

1. The final answer is in, we are done, return to SyncRes::doResolve and SyncRes::beginResolve
2. The nameserver we queried tells us the domain we asked for authoritatively does not exist. In case of the root-servers, this happens when we query for *''www.powerdns.kom.''* for example, there is no *''kom.''*. Return to SyncRes::beginResolve, we are done.
3. A lesser form - it tells us it is authoritative for the query we asked about, but there is no record matching our type. This happens when querying for the IPv6 address of a host which only has an IPv4 address. Return to SyncRes::beginResolve, we are done.
4. The nameserver passed us a CNAME to another domain, and we need to reroute. Go to SyncRes::doResolve for the new domain.
5. The nameserver did not know about the domain, but does know who does, a *referral*. Stay within doResolveAt and loop to these new nameservers.
6. The nameserver replied saying *no idea*. This is called a *lame delegation*. Stay within SyncRes::doResolveAt and try the other nameservers we have for this domain.

When not redirected using a CNAME, this function will loop until it has exhausted all nameservers and all their IP addresses. DNS is surprisingly resilient that there is often only a single non-broken nameserver left to answer queries, and we need to be prepared for that.

This is the whole DNS algorithm in PowerDNS, all in less than 700 lines of code. It contains a lot of tricky bits though, related to the cache.

20.7 Some of the things we glossed over

Whenever a packet is sent to a remote nameserver, the response time is stored in the SyncRes::s_nsSpeeds map, using an exponentially weighted moving average. This EWMA averages out different response times, and also makes them decrease over time. This means that a nameserver that hasn't been queried recently gradually becomes **faster** in the eyes of PowerDNS, giving it a chance again.

A timeout is accounted as a 1s response time, which should take that server out of the running for a while.

Furthermore, queries are throttled. This means that each query to a nameserver that has failed is accounted in the `s_throttle` object. Before performing a new query, the query and the nameserver are looked up via shouldThrottle. If so, the query is assumed to have failed without even being performed. This saves a lot of network traffic and makes PowerDNS quick to respond to lame servers.

It also offers a modicum of protection against birthday attack powered spoofing attempts, as PowerDNS will not inundate a broken server with queries.

The negative query cache we mentioned earlier caches the cases 2 and 3 in the enumeration above. This data needs to be stored separately, as it represents **non-data**. Each negcache query entry is the name of the SOA record that was presented with the evidence of non-existence. This SOA record is then retrieved from the regular cache, but with the TTL that originally came with the NXDOMAIN (case 2) or NXRRSET (case 3).

20.8 The Recursor Cache

As mentioned before, the cache stores partial packets. It also stores not the **Time To Live** of records, but in fact the **Time To Die**. If the cache contains data, but it is expired, that data should not be deemed present. This bit of PowerDNS has proven tricky, leading to deadlocks in the past.

There are some other very tricky things to deal with. For example, through a process called **more details**, a domain might have more nameservers than listed in its parent zone. So, there might only be two nameservers for `powerdns.com.` in the **‘‘com.‘‘** zone, but the **‘‘powerdns.com‘‘** zone might list more.

This means that the cache should not, when talking to the **‘‘com.‘‘** servers later on, overwrite these four nameservers with only the two copies the **‘‘com.‘‘** servers pass us.

However, in other cases (like for example for SOA and CNAME records), new data should overwrite old data.

Note that PowerDNS deviates from RFC 2181 (section 5.4.1) in this respect.

20.9 Some small things

The server-side part of PowerDNS (`pdns_recursor.cc`), which listens to queries by end-users, is fully IPv6 capable using the ComboAddress class. This class is in fact a union of a `struct sockaddr_in` and a `struct sockaddr_in6`. As long as the `sin_family` (or `sin6_family`) and `sin_port` members are in the same place, this works just fine, allowing us to pass a ComboAddress*, cast to a `sockaddr*` to the socket functions. For convenience, the ComboAddress also offers a length() method which can be used to indicate the length - either sizeof(sockaddr_in) or sizeof(sockaddr_in6).

Access to the recursor is governed through the NetmaskGroup class, which internally contains Netmask, which in turn contain a ComboAddress.

HTTP ROUTING TABLE

/api

INDEX

O

P

Q

R

S

T

U

V

W