BIND 9

(Part 1 - Event based BIND 9 log- and system analysis)

Carsten Strotmann and the ISC Team
Welcome

Welcome to part one of our BIND 9 webinar series
In this Webinar

- best practice BIND 9 log template for authoritative server
- best practice BIND 9 log template for DNS resolver
- file system best practices for BIND 9 log files, transparent online compression
- searching through log-files with modern 'grep': ugrep, ripgrep, sack, sift …
- BIND 9 Log-Analysis with Linux Systemd-Journal
- Identifying hot-spots in BIND 9 logfiles with histograms
- Condensing information from large BIND 9 logfiles with the "logeater" tools
- Monitoring BIND 9 run-time information with atop
- Incident based DNSTAP and TCPDUMP traffic capture
BIND 9 logging
BIND 9 logging

- BIND 9 has one of the most flexible logging framework found in Unix/Linux server products
- categories define *what* should be written to the logs
- channel define *where* the log data should be written
- multiple categories can be written into the same channel
- one category can be written to multiple channels
BIND 9 logging

Channel (where to log)
BIND 9 logging

Channel (where to log)

- syslog
- file /var/named/query.log
- file /var/named/transfer.log
- null
- stderr

All content © 2021 Internet Systems Consortium, Inc.
BIND 9 logging

Channel (where to log)
- syslog
- file /var/named/query.log
- file /var/named/transfer.log
- null
- stderr

Category (what to log)
- security
- xfer-in / xfer-out
- queries
- dnssec
- default
BIND 9 logging

Channel (where to log)
- syslog
- file /var/named/query.log
- file /var/named/transfer.log
- null
- stderr

Category (what to log)
- security
- xfer-in / xfer-out
- queries
- dnssec
- default
Logging configuration

- separate log categories with many entries per day from categories with only a few entries a day
  - else it will be hard to spot the few entries and react on them
- combine categories with related information into the same channel
  - for example: xfer-in, xfer-out, notify or update and update-security
  - use the logging configuration statement print-category yes;
  - to be able to filter for the categories in combined log files
Logging severity

• BIND 9 supports the syslog *priorities* (debug, info, warning, error ...) also when writing to a log-file
  ▪ a lower priority for a channel will include all messages of the selected priority/severity and higher
• Using the logging statement `print-severity yes;`
  BIND 9 will print the logging severity in the log file
  ▪ this allows for later filtering of the log file based on the severity

2021-02-17T11:27:36.593 general: info: reloading configuration succeeded
2021-02-17T11:27:36.600 general: info: reloading zones succeeded
2021-02-17T11:27:37.006 general: notice: all zones loaded
2021-02-17T11:27:37.100 general: notice: running
2021-02-17T11:27:43.929 query-errors: info: client @0x5dbab4d4 ::1#63099 (fail05.dnssec.works): query f
2021-02-17T11:27:49.962 query-errors: info: client @0x5dbab4d4 ::1#59273 (fail05.dnssec.works): query f
2021-02-17T11:27:50.073 general: info: zone fail05.dnssec.works/IN: refresh: unexpected rcode (SERVFAIL
RPZ Logging

- The RPZ logging category gives information about the response policy zone activity on the server:

```
17-Feb-2021 11:56:55.311 client @0x7f4fb5f3cd40 2001:db8:ff:1a00:95e8:d30:c8a9:6ced#62678 (31b4bd31fg1x
17-Feb-2021 11:57:24.663 client @0x7f4fc2379b0 2001:db8:561:5:7935:d587:336b:436e#17126 (malware.com):
17-Feb-2021 11:57:24.840 client @0x7f4fc5a66560 2001:db8:4212:0:11da:b4a5:b7e:31dc#45577 (trojan.biz):
17-Feb-2021 11:57:41.814 client @0x7f4fc51668910 1002:db8:1121:5:7935:d587:336b:436e#4460 (badstuff.com)
```
Logging configuration template

- Please find a logging configuration template for BIND 9.16 in this document linked below:

  BIND 9 Logging templates
Query-Logging

- The BIND 9 log system allows to write a log entry for every incoming query
  - the log entries will list the IP address of the sender, the domain name of the query, the record type and information on the flags (DNSSEC OK, EDNS, Recursion Desired etc) of the query sender

```
2021-01-18T15:13:01.438 client @0x413ce4d4 192.0.2.4#13700 (blog.defaultroutes.de): \n  query: blog.defaultroutes.de IN AAAA - (5.45.109.212)
```
Performance issues with query logging

- Converting the query log information from binary data into human readable text format is CPU intensive
  - enabling query logging on an busy DNS server (> 500 Queries/Second) can reduce the performance of the DNS server and increase the latency of DNS answers
- Query-Logging should be disabled on (re-)start of the BIND 9 process and should only be enabled for a short period of time during troubleshooting:

```plaintext
options {
    directory "/var/named";
    querylog no;
    [...]
};
```
Alternatives to BIND 9 query logging

- BIND 9 query logging only gives information on the query, but not on the answer send out.
- Alternatives to query logging exist that capture queries and responses and have less impact on the DNS servers performance:
  - DNSTAP: Using DNSTAP with BIND [https://kb.isc.org/docs/aa-01342](https://kb.isc.org/docs/aa-01342)
  - PCAP/tcpdump based passive DNS data collection: for example DNSmezzo from AFNIC [https://github.com/AFNIC/DNSwitness](https://github.com/AFNIC/DNSwitness)
File-System and logging best practices
Partitions

- Modern Unix and Linux systems sometimes propose a single partition install
  - While this is usually good for Desktop/Laptop installs, it increases the risk of the operating system (root) partition running full
  - Bad things will happen once the root partition runs out of storage space
  - Recommendation: create a dedicated (large enough) partition for the log files (mount to `/var/log` or the place where your BIND 9 server writes the log files)
Log-file compression

- Tools like logrotate can compress logfiles to save space
  - However, disk space is seldom a problem today
  - Working with compressed log files is more difficult then with plain text files
  - Some file systems (BTRFS, ZFS) offer transparent online compression which can speed up I/O performance of logfile operations
Logging with Linux systemd-journald
Systemd Journal

- Modern Linux systems come with the `systemd` process management system
  - systemd includes a database of log information called *the journal*
  - the Systemd-Journal can be stored locally (on the same machine the service runs) or remote (on a central Journal-Server)
Benefits of the systemd journal

- Automatically correlates log entries from different sources (systemd process start up, network stack, BIND 9 server)
- Allows easy filtering of log information by time, service, keyword
- Output log entries as text (different flavors), JSON or binary
- On a central journal host, can merge log information from multiple hosts
- Can log remotely via push or pull configurations
- Support online compression
- Supports *Forward Secure Sealing* to detect tampering with written log data
Using the Systemd Journal with BIND 9

- BIND 9 needs to be started from a systemd service unit
- Systemd will capture all information written to the stdout and stderr channels of the BIND 9 process and store that information into the Journal
  - in the BIND 9 logging configuration, you need to configure the relevant log information to be send to stdout or stderr
Live Demo Systemd-Journal
Searching/filtering of logfiles
Modern Tools

- Traditional tools to search and filter through log-files are `grep`, `awk` and `perl`
- A few years ago, a number of new tools appeared that have pros compared with the traditional tools:
  - Multi-CPU core scalability
  - Faster regular expression engines
  - Interactive filter capabilities
  - Work transparently on compressed files
Modern "grep" alternatives

- Sack - https://github.com/sampson-chen/sack
- uGrep - https://github.com/Genivia/ugrep
- RipGrep - https://github.com/BurntSushi/ripgrep
- sift - https://sift-tool.org/
Live Demo - ugrep
Condensing information from large logfiles
Logeater

- from time to time I'm tasked with finding the root cause of an BIND 9 incident in large (> 1 GB) BIND 9 logfiles
- I've written the logeater tools to help with that task
- The source code (golang) can be found at https://github.com/cstrotm/logeater
- be careful: the logeater tools distribute the work on all available CPU cores, don't run the tools on a productions DNS server, the tools can *starve* the DNS server process
Logeater-queries

- logeater-queries will read a BIND 9 (9.10+) style query log file and will output statistics and aggregated information from the file:

```bash
$ ./logeater-queries -h
Usage of ./logeater-queries:
  -c list query network classes
  -d list domain names
  -i list query IP addresses
  -n no reverse IP lookup
  -s list statistics
  -t list query type
```
**Example use of logeater-queries**

- logeater-queries Example - printing network classes and query types of queries:

```
$ cat query.log | ./logeater-queries -c -t | column -t

Query-Network-Classes
21379 : IN
6 : CH

Query-Network-Types
8514 : A
4060 : AAAA
3079 : SOA
2372 : DNSKEY
927 : PTR
658 : MX
543 : NS
312 : DS
286 : TXT
186 : NSEC
129 : ANY
115 : CNAME
[...]
```
Example use of logeater-queries

- logeater-queries Example - printing the top ten query IP addresses with reverse name resolution:

```
$ cat query.log | ./logeater-queries -i | head | column -t
Query-IP-Addresses
1571   : 212.114.206.217                       [muc.example.de.]
794    : 72.13.58.112                        [dnsviz-db.verisignlabs.com.]
704    : 54.234.42.241                       [241.compute-1.amazonaws.com.]
682    : 2001:19f0:5001:df:76d7:5703:ba0a:e220 []
565    : 185.92.221.212                      [185.92.221.212.vultr.com.]
467    : 185.22.143.29                       [b9168f1d.cgn.dg-w.de.]
314    : 91.51.184.46                        [3b82e.dip0.t-ipconnet.de.]
```
Logeater-DNSSEC

- logeater-dnssec analyses the a log file with messages from the DNSSEC category and groups the error messages:

```
$ cat dnssec.log | ./logeater-dnssec | head
8727 : 0C9F6LG0E6NADAS8KGI1CLI9U09G7EIG.ad/NSEC3: no valid signature found
6953 : ad/SOA: no valid signature found
3976 : sat-tv.com/A: got insecure response; parent indicates it should be secure
1730 : mozilla.com/SOA: no valid signature found
1586 : stream.bestvideostreaming.is/A: no valid signature found
1577 : 8FC1DQ3C2Q3ERFD4UO40ENDBTSFME5J05.ad/NSEC3: no valid signature found
1576 : sat-tv.com/SOA: got insecure response; parent indicates it should be secure
1483 : 0c9f6lg0e6n13ad9i1c1ik9u09g7eig.ad/NSEC3: no valid signature found
968 : cbr.de/NSEC: no valid signature found
```
Logeater-Resolver

- logeater-resolver analyses the a log file with messages from the resolver category and groups the error messages:

```
$ cat resolvers.log | ./logeater-resolvers | head
42908 : s-cnc1.qq.com/AAAA: Name qq.com (SOA) not subdomain of zone ns-cnc1.qq.com -- invalid response
42713 : s-tel1.qq.com/AAAA: Name qq.com (SOA) not subdomain of zone ns-tel1.qq.com -- invalid response
42484 : s-os1.qq.com/AAAA: Name qq.com (SOA) not subdomain of zone ns-os1.qq.com -- invalid response
42297 : s-cmn1.qq.com/AAAA: Name qq.com (SOA) not subdomain of zone ns-cmn1.qq.com -- invalid response
20346 : mails.sonymusicfans.com/DS: invalid response
10920 : tpl.glb.nist.gov/DS: invalid response
9693 : media.netd.com.tr/AAAA for client 192.0.2.165#3347: Name netd.com.tr (SOA) not subdomain of zone
7932 : service.superc.net/AAAA for client 192.0.2.11#3073: Name superc.net (SOA) not subdomain of zone
4597 : brickleonavon.com/NS for client 192.0.2.46#3073: Name . (SOA) not subdomain of zone brickleonavon
```
Live-Demo Logeater Tools
Logfile histograms
Finding hot-spots or outliers

- When working with large BIND 9 log files (or tcpdump pcap files), I often generate histograms over the logfiles to find hot-spots or outliers
- Logfile Histograms can be created from any field in the logfile
  - Time based
  - Query name based
  - Query IP address based
  - Error message based
- Histogram creation is done with classic Unix tools in a shell pipeline
Time based histogram

- Example of a time based histogram from BIND 9 log files with DNSSEC issues

```
cat dnssec.log | grep "2021-02-17" | awk '{print substr($0,0,16)}' \
| uniq -c | awk '{printf("n%s ",$0) ; for (i = 0; i<$1/100 ; i++) \
{printf("*");}' | less
```
Time based histogram output

• This histogram shows an incident around 18:17 UTC on 27th April 2020:

<table>
<thead>
<tr>
<th>Time</th>
<th>Date</th>
<th>Event Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td>27-Apr-2020 18:09</td>
<td>**</td>
</tr>
<tr>
<td>159</td>
<td>27-Apr-2020 18:10</td>
<td>**</td>
</tr>
<tr>
<td>160</td>
<td>27-Apr-2020 18:11</td>
<td>**</td>
</tr>
<tr>
<td>169</td>
<td>27-Apr-2020 18:12</td>
<td>**</td>
</tr>
<tr>
<td>175</td>
<td>27-Apr-2020 18:13</td>
<td>**</td>
</tr>
<tr>
<td>166</td>
<td>27-Apr-2020 18:14</td>
<td>**</td>
</tr>
<tr>
<td>183</td>
<td>27-Apr-2020 18:15</td>
<td>**</td>
</tr>
<tr>
<td>224</td>
<td>27-Apr-2020 18:16</td>
<td>***</td>
</tr>
<tr>
<td>4082</td>
<td>27-Apr-2020 18:17</td>
<td>********************************************</td>
</tr>
<tr>
<td>4439</td>
<td>27-Apr-2020 18:18</td>
<td>********************************************</td>
</tr>
<tr>
<td>3476</td>
<td>27-Apr-2020 18:19</td>
<td>********************************************</td>
</tr>
<tr>
<td>3212</td>
<td>27-Apr-2020 18:20</td>
<td>********************************************</td>
</tr>
<tr>
<td>2484</td>
<td>27-Apr-2020 18:21</td>
<td>********************************************</td>
</tr>
<tr>
<td>2201</td>
<td>27-Apr-2020 18:22</td>
<td>********************************************</td>
</tr>
<tr>
<td>2036</td>
<td>27-Apr-2020 18:23</td>
<td>********************************************</td>
</tr>
<tr>
<td>2032</td>
<td>27-Apr-2020 18:24</td>
<td>********************************************</td>
</tr>
<tr>
<td>2059</td>
<td>27-Apr-2020 18:25</td>
<td>********************************************</td>
</tr>
<tr>
<td>1679</td>
<td>27-Apr-2020 18:26</td>
<td>********************************************</td>
</tr>
<tr>
<td>943</td>
<td>27-Apr-2020 18:27</td>
<td>********</td>
</tr>
<tr>
<td>639</td>
<td>27-Apr-2020 18:28</td>
<td>********</td>
</tr>
<tr>
<td>500</td>
<td>27-Apr-2020 18:29</td>
<td>*****</td>
</tr>
<tr>
<td>414</td>
<td>27-Apr-2020 18:30</td>
<td>*****</td>
</tr>
</tbody>
</table>
Example: Histogram by log messages

- This pipeline prints the log messages from security.log based on the number of recurring messages

```bash
cat security.log | grep "2021-02-14" | awk -F : '{print substr($4,0,60)}' | \
uniq -c | sort -nr | \
awk '{printf("%s ",$0) ; for (i = 0; i<$1/10 ; i++) {printf("
\n")};}'
```
Example: Histogram by log messages output

<table>
<thead>
<tr>
<th>Query Count</th>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>query (cache) 'www.google.com/A/IN'</td>
<td>denied **********</td>
</tr>
<tr>
<td>10</td>
<td>query (cache) 'peacecorps.gov/ANY/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>2</td>
<td>query (cache) 'com/ANY/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>2</td>
<td>query (cache) '67b.org/AAAA/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'www.qq.com/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'www.qq.com/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'researchscan541.eecs.umich.edu/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'pizzaseo.com/RRSIG/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'invalid.parrotdns.com/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'dnsscan.shadowserver.org/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'cpsc.gov/ANY/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'c.afekv.com/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) 'amazon.com/A/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) '5-45-109-212-60287611.spiderprobe.com/A/IN'</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) '37-120-183-122-60287611.spiderprobe.com/A/IN'</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) '212.109.45.5.in-addr.arpa/PTR/IN'</td>
<td>denied *</td>
</tr>
<tr>
<td>1</td>
<td>query (cache) '122.183.120.37.in-addr.arpa/PTR/IN'</td>
<td>denied *</td>
</tr>
</tbody>
</table>
System-Monitoring with atop
atop - a top alternative of 'steroids'

- atop is an real-time system monitor. It is an alternative to the original Unix/Linux top tool
- Advantages
  - Shows resource usage of ALL processes, even those that are closed/completed.
  - Monitors threads within processes & ignores processes that are unused.
  - Accumulates resource usage for all processes and users with the same name.
  - Highlights critical resources using colors (red).
  - Will add or remove columns as the size of the display window changes.
  - Includes disk I/O and network utilization.
  - Shows interrupt utilization per CPU and CPU frequencies
  - Uses netatop kernel module to monitor TCP & UDP and network bandwidth.
Atop Example
Atop background monitoring

- atop can be run as a daemon process in the background
  - in this example, atop collects the system information every 60 seconds into the (binary) file /var/log/atop/atop_20210215 (rotated daily)

```
$ systemctl status atop
● atop.service - Atop advanced performance monitor
  Loaded: loaded (/lib/systemd/system/atop.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2021-02-15 00:00:02 CET; 13h ago
    Docs: man:atop(1)
    Main PID: 10509 (atop)
    Tasks: 1 (limit: 4915)
    CGroup: /system.slice/atop.service
           └─10509 /usr/bin/atop -R -w /var/log/atop/atop_20210215 60

Feb 15 00:00:02 dns-resolver-12 systemd[1]: Started Atop advanced performance monitor.
```
Replay historical atop data

- atop can read the (binary) historical data
  - the tools then allows to move back and forth in the time to investigate past system events

```bash
atop -r /var/log/atop/atop_20210215
```
Live Demo of atop
Incident based traffic capture
DNS incidents and available data

- As it is hard to predict when a DNS or network incident will strike, often valuable detail data from the incident is missing
- Next we show an example of a system that measures critical system parameter (CPU load, UDP and TCP query volume) and will start a network capture with tcpdump once a threshold is reached
  - can be adapted to do other data collection (DNSTAP, wireshark/tshark, dnscap etc)
  - scripts should test the amount of free storage space before starting the data collection
  - the scripts should also raise an alarm, so that someone knows there is data to be examined
Systemd Timer Unit

- file /etc/systemd/system/tcpdump-starter.timer: this systemd unit starts the tcpdump-starter service every 60 seconds after the last termination of the same service

```ini
[Unit]
Description=TCPDUMP Starter Timer

[Timer]
OnUnitInactiveSec=60s

[Install]
WantedBy=timers.target
```
Systemd service unit

- The unit file in `/etc/systemd/system/tcpdump-starter.service` starts the `tcpdump-starter` script on the timer event.
  - Parameter `–c 12`: Start collecting detail information when the CPU load is > 12.
  - Parameter `–u 32000`: Start collecting detail information if the number of queries over UDP > 32.000.
  - Parameter `–t 1000`: Start collecting detail information if the number of queries over TCP > 1.000.

```
[Unit]
Description=TCPDUMP Starter Service

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/tcpdump-starter -c 12 -u 32000 -t 1000

[Install]
WantedBy=multi-user.target
```
Tcpcap-starter Script (1/4)

- The script sets some default values and processes the parameter to the script

```bash
#!/bin/sh
set -o nounset
cputhreshold=12
udpthreshold=32000
tcpthreshold=1000
diskthreshold=10000000
interface=eth0

while getopts ':t:u:c:' optname do
  case $optname in
    c) cputhreshold=${OPTARG};;
    t) tcpthreshold=${OPTARG};;
    u) udpthreshold=${OPTARG};;
    *) echo "$(basename $0): Internal error, unexpected option $OPTARG" >&2
       exit 3;;
  esac
done
shift $((OPTIND-1))
```
Tcpdump-starter Script (2/4)

- the 2nd part checks the CPU load

```bash
# test for high CPU load
avg=$(cat /proc/loadavg | cut -d '.' -f 1)
if [ $cputhreshold -le $avg ]; then
    echo "High CPU Load ${cputhreshold}"
    logger -t tcpdump "High CPU Load ${avg} - starting TCPDUMP"
    diskfree=$(df --output=avail / | tail -1)
    if [ $diskthreshold -ge $diskfree ]; then
        logger -t tcpdump "Not enough free disk space to start TCPDUMP."
    else
        tcpdump-run 5 $interface
    fi
fi
[...]
```
Tcddfump-starter Script (3/4)

- the 3rd part checks for the volume of UDP queries

```bash
[[...]
# test for high DNS UDP request rate
udpqr=$(/usr/sbin/rndc status | grep "recursive clients" | cut -d ':' -f 2 | cut -d '/' -f 1)
if [ $udpthreshold -le udpqr ]; then
    echo "High UDP DNS Load ${udpqr}";
    logger -t tcpdump "High UDP DNS Load ${udpqr} - starting TCPDUMP"
    if [ $diskthreshold -ge $diskfree ]; then
        logger -t tcpdump "Not enough free disk space to start TCPDUMP."
    else
        tcpdump-run 5 $interface
    fi
fi
[[...]
```
Tcddump-starter Script (4/4)

- the last part checks for the volume of TCP queries

```bash
[...]
# test for high DNS TCP request rate
tcpqr=($(/opt/local/bind/sbin/rndc status | grep "tcp clients" | cut -d ':' -f 2 | cut -d '/' -f 1)
if [ $tcpthreshold -le $tcpqr ]; then
    echo "High TCP DNS Load $tcpqr"
    logger -t tcddump "High TCP DNS Load $tcpqr" - starting TCPDUMP"
    if [ $diskthreshold -ge $diskfree ]; then
        logger -t tcddump "Not enough free disk space to start TCPDUMP."
    else
        tcpd dump-run 5 $interface
    fi
fi
```
Script tcpdump-run

- The script `tcpdump-run` collects network traffic for some time (5 minutes in this case)

```bash
#!/bin/sh
tcpdump -q -W $1 -G 60 -w /pcap/packettrace-%F-%T.pcap -i $2
```
References:

- BIND 9 Logging configuration templates:
  https://webinar.defaultroutes.de/webinar/bind9-logging-
- BIND 9 ARM - Logging:
  statement-definition-and-usage
DNS data collection

- DNS-OARC software tools [https://www.dns-oarc.net/oarc/software](https://www.dns-oarc.net/oarc/software)
- DNS witness [https://github.com/AFNIC/DNSWitness](https://github.com/AFNIC/DNSWitness)
- DNS Parse [https://github.com/pflarr/dns_parse](https://github.com/pflarr/dns_parse)
- Capture DNS - A simple program to capture and show DNS queries [https://github.com/lilydjwg/capture-dns](https://github.com/lilydjwg/capture-dns)
Systemd Journal

- Log Management with the Systemd Journal
  https://materials.rangeforce.com/tutorial/2020/04/16/Sys
  Journal/
- Forward secure sealing https://lwn.net/Articles/512895/
Atop

- atop home page https://www.atoptool.nl/
Upcoming Webinars

- March 17: Session 2. Long-term statistics monitoring and log analysis
- April 21: Session 3. Load balancing with DNSdist
- May 19: Session 4. Dynamic zones, pt1 - Basics
- June 16: Session 5. Dynamic zones, pt 2 - Advanced topics
Questions and Answers