

RSSAC028 Implementation study report

Consortium:

NLnet Labs

Stichting Internet Domeinregistratie Nederland (SIDN)

Authors: Willem Toorop (NLnet Labs), Yorgos Thessalonikefs (NLnet Labs), Benno Overeinder (NLnet Labs), Moritz Müller (SIDN), Marco Davids (SIDN).

Last modified: September 21, 2023

Table of Contents

Introduction	3
Naming schemes from RSSAC028	5
The Current Naming Scheme: 5.1	5
The Current Naming Scheme, with DNSSEC: 5.2	5
In-zone NS Names: 5.3	5
Shared Delegated TLD: 5.4	6
Names Delegated to Each Operator: 5.5	6
Single Shared Label for All Operators: 5.6	6
Survey of Root Server Operators	7
Survey results	7
Resolver testbed	10
Design	10
Virtual machines and operating systems	10
DNS software used	11
Virtual networking	11
Zone files for the naming schemes from RSSAC028	13
Orchestration	15
Simulating the proprietary authoritative name server software	15
Installation	16
Running the testbed	17
I.) Setup and Configuration	17
II.) Running the test queries and capturing traffic	18
III.) Processing the packet captures	19
Main Analysis	22
Acceptable response sizes	22
Support of response sizes according to Internet standards	22
Support of response sizes according to measurements in the wild	22
Summary	24
Reproducing RSSAC028 Appendix A results	25
Priming responses from all root server software	25
Most prevalent query parameters	25
Group I	27
Group II	29
Group III	29
Group IV	30
Group V	31
Group VI	31
Group VII	32
Group VIII	32

Priming responses properties	33
Testbed results	35
Naming Scheme 5.4 and Knot Resolver	37
Missing additional addresses of Group I and II name servers	38
Failed priming	39
Influence on UDP/TCP traffic	41
Avoiding TCP with naming scheme 5.3.1	41
Increase in TCP traffic	41
Failure scenarios	42
Unable to validate	42
Unable to send queries via TCP	43
Properties of resolvers	44
Impact of search lists on different naming schemes	46
Discussion and Conclusion	47
Observation per naming scheme	49
Acknowledgement	51
Appendix A: RSO Survey	52
Appendix B: Reproducing RSSAC028 Appendix A	54
Reproduced RSSAC028 Appendix A results	54
BIND 9.10.3-P4	54
NSD 4.1.13	54
Knot DNS 2.2.1	55
Knot DNS 2.3.0	55
Results for recent versions of BIND, NSD and Knot DNS	55
BIND 9.18.13	55
NSD 4.6.1	56
Knot DNS 3.2.5	56
Appendix C: Naming schemes cheat sheet	57

Introduction

The Root Server System Advisory Committee (RSSAC) conducted a comprehensive technical analysis of the naming scheme utilized for individual root servers, which has been published as RSSAC028¹. The objective was to offer guidance to the Internet Corporation for Assigned Names and Numbers (ICANN) Board of Directors and the Internet community more broadly regarding the operation, administration, security, and integrity of the Internet's root server system.

The current naming scheme was initially introduced in 1995 (see RSSAC023v2²) and has proven to be effective over the years. However, as part of its tasks to perform ongoing threat assessment and risk analysis of the root server system and to recommend any necessary audit activity to assess the current status of root servers and the root zone, the RSSAC undertook a thorough analysis of the naming scheme and carefully evaluated the consequences of implementing modifications.

RSSAC028 provides an extensive documentation of various alternative naming schemes for the root zone and associated root servers, including the existing naming scheme. Each scheme, ranging from the current naming scheme, the signed root-servers.net zone, in-zone NS names, shared delegated TLD, names delegated to each operator, to a single shared label for all operators, is elaborated upon in sections 5.1 to 5.6 of the RSSAC028 report. Furthermore, anticipated advantages and drawbacks are outlined for each scheme, accompanied by a comprehensive risk analysis.

The key recommendation stemming from RSSAC028 is that no immediate changes should be made to the current naming scheme until further studies have been conducted. Any modifications to the existing naming scheme should be based on the outcomes of these additional investigations.

This report seeks to address parts of the questions raised in RSSAC028. Its primary focus lies on the operational behavior of widely used open source DNS resolver software in relation to each naming scheme and root server deployment proposed in RSSAC028. Specifically, this report studies:

- The acceptable response size for priming queries
- The impact of naming schemes on priming responses
- The impact of missing glue records on recursive resolver implementations
- The impact of DNSSEC validation on signed priming responses
- The impact of search lists on different naming scheme

¹ Kumari, Warren, Joe Abley, John Bond, Brian Dickson, Paul Hoffman, Suresh Krishnaswamy, and Matt Larson. 2017. "RSSAC028-Technical Analysis of the Naming Scheme Used For Individual Root Servers.", <https://www.icann.org/en/system/files/files/rssac-028-03aug17-en.pdf>

² RSSAC. 2020. "RSSAC023v2: History of the Root Server System". <https://www.icann.org/en/system/files/files/rssac-023-17jun20-en.pdf>

The findings presented herein are the result of testbed simulations for each naming scheme and a literature study. With the help from input from the Root Server Operators (RSO), we created a [testbed](#) that represents the current configurations of the root server system. Furthermore, the testbed consists of a variety of recursive resolver software. By presenting the findings of the testbed simulations and their implications, this report aims to facilitate informed discussions and decision-making within the DNS community.

The remainder of this document is structured as follows: First, we reintroduce the [naming schemes from RSSAC028](#). Then, we discuss the [questionnaire](#) shared with the RSOs and list their responses. We take their responses as an input for configuring our testbed and the components and design choices are described in [Resolver testbed](#). We use the testbed to simulate the impact of the naming schemes on priming and describe the results of this analysis in [Main Analysis](#). We discuss our results in [Discussion and Conclusion](#). The appendices [Appendix A: RSO Survey](#) and [Appendix B: Reproducing RSSAC028 Appendix A](#) contain additional information about the survey and the testbed.

Naming schemes from RSSAC028

This section contains an overview of the naming schemes introduced in section 5 of RSSAC028. In this document, the individual schemes are referenced by the section number in RSSAC028 in which that scheme (5.1 to 5.6) is described. A short description for each scheme is taken from the section in RSSAC028 which introduces the scheme.

The Current Naming Scheme: 5.1

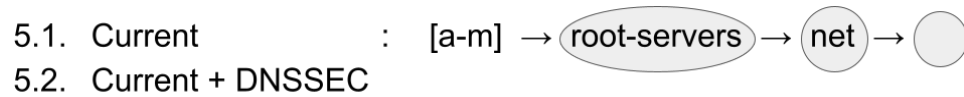


Figure 1: The current naming scheme

The authoritative servers for the root zone have the names '[a-m].root-servers.net'. The 'root-servers.net' zone is served by name servers that also serve the root zone. In this scheme, the 'root-servers.net' zone continues to be unsigned.

The Current Naming Scheme, with DNSSEC: 5.2

This is the same as the preceding scheme, but with 'root-servers.net' being signed by the zone's maintainer.

In-zone NS Names: 5.3

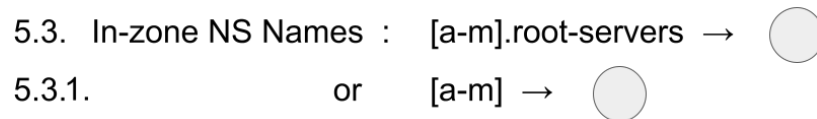


Figure 2: In-zone NS names

The root zone will have an NS RRset consisting of in-zone names with the A and AAAA records of the root servers. In our report 5.3 uses '[a-m].root-servers' as the name for the root servers. We also use a variant, 5.3.1, that uses a single letter '[a-m]' for the root servers.

Shared Delegated TLD: 5.4

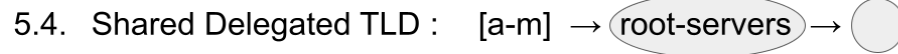


Figure 3: Shared delegated TLD

The root zone will have an NS RRset that consists of 13 domain names that share a new common delegated TLD (for example, the names 'a.root-servers', 'b.root-servers', and so on). There will be 13 records in the root zone's NS RRset pointing to the root server name server instances. The new shared TLD will be delegated to the same set of nameservers.

Names Delegated to Each Operator: 5.5

5.5. Names Delegated to Each Operator



Figure 4: Names delegated to each operator

A new domain will be delegated to each root server operator. The root zone will have an NS RRset consisting of server names that are managed by the corresponding root server operators. The names for this proposal can either have all records under a common label (for example, the names 'a.root-servers', 'b.root-servers', and so on) or can be short labels in the root zone (for example, the names 'a', 'b', and so on which in our study is referred to as scheme 5.5.1.). No other delegations are involved.

Single Shared Label for All Operators: 5.6

5.6. Single Shared Label for All Operators

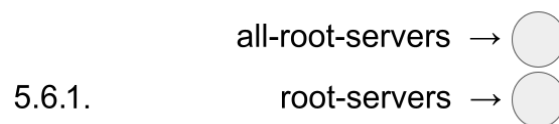


Figure 5: Single shared label for all operators

Instead of having individual names for each root server, the set of root servers could be given one name at the top level (such as 'all-root-servers.')

and that one name has the 13 IPv4 addresses and 13 IPv6 addresses of the root servers as two RRsets. We also use a variant, 5.6.1, that uses 'root-servers' as the TLD.

Survey of Root Server Operators

In order to test the naming schemes in a (realistic) simulation of the current root server system, we reached out to RSOs to identify authoritative server software, their configuration and other relevant parameters. We reached out to the RSO's via email, and asked the following questions (see [Appendix A](#) for the full e-mail):

- **Question 1:** What authoritative name server software are you using to serve the root zone or are you planning to use in the near future? Which version of the software do you use and on which platform? If different software is used in parallel or as a backup, please include them as well.
- **Question 2:** Is the name server software compiled with custom compile-time options and/or configuration for compilation? If so, are you able and willing to provide those options and/or configuration?
- **Question 3:** If the software used is open source, would you be willing to provide us with the configuration files? If yes, please attach the configuration files to this response.
- **Question 4:** If the software is proprietary, would you willing to share it with us or provide us with an installation we can use to perform our tests?
- **Question 5:** Do you have public facing load balancers or other infrastructure you consider important for this study?
- **Question 6:** Are there any other parameters of your name server setup relevant to this study? For example, do your name servers support DNS cookies? Which path MTU(s) do you have configured? Have you enabled minimal responses or similar functions?

Survey results

All RSO's answered our survey. [Table 1](#) and [Table 2](#) summarize the answers from the RSO's. The operators of the root letters A, B, F, I, J and M requested us not to publish all or some of their answers. Other operators provided only a limited set of answers. If provided and permitted, [Table 1](#) lists the configurations deployed by RSO's at the time of answering the survey. [Table 2](#) lists the configurations that RSO's plan to deploy in the future.

From the survey results that were indicated to be confidential, two proprietary authoritative name server software were reported. The providers of that software agreed to perform tests locally in their own testbeds and reported the results back to us. The tests consisted of zone files for each naming scheme and a test script. From the provided results, we recreated authoritative server replies that resemble the output of the RSO's as closely as possible. See [Simulating the proprietary authoritative name server software](#) for more details.

In the case of incomplete OS information we opted for the current latest version of the given version branch or Debian bullseye if the information was missing or only 'Linux' was mentioned. For RedHat derivatives we opted for Centos8 stream. In the case of an incomplete name server software version we opted for the current latest version of the given version branch. These choices are depicted with '→' in [Table 1](#) and [Table 2](#):

Letters	Operating System	Software	Version	Compile options	Other parameters
<i>A and J</i>				<i>Confidential</i>	
<i>B</i>				<i>Confidential</i>	
<i>C</i>	CentOS 7	BIND	9.16 →9.16.39	./configure --enable-dnstap --with-maxminddb --with-json-c --with-libidn2 --libdir=%{_prefix}/lib64	MTU: 1500
<i>D</i>	→Debian bullseye	NSD	4.1.20	--enable-root-server --enable-bind8-stats --enable-zone-stats --enable-ratelimit-default-is-off	EDNS: 1450
<i>E</i>	FreeBSD →FreeBSD13.x	BIND	9.16.x →9.16.39		DNS Cookies: Yes MTU: 1480
<i>F</i>	FreeBSD 12.x and 13.x	BIND	9.16.x →9.16.39		DNS Cookies: Partially Software config: keep-response-order { any; };
----- <i>Confidential</i> -----					
<i>G</i>	→Debian bullseye	BIND	9.16.29-S1		
<i>H</i>	Linux →Debian bullseye	NSD	4.5.0	--enable-root-server --enable-ipv6 --with-user=domain --enable-bind8-stats --with-pidfile=/etc/nsd/run/nsd.pid --enable-ratelimit --with-tcp-timeout=15 CFLAGS="-O2 -ffast-math -fomit-frame-pointer -fpeel-loops -fstack-protector-all -mtune=core2 -fPIE" LDFLAGS="-pie -Wl,-z,relro,-z,now"	MTU: 1500 MSS on loadbalancer: 1220 (for IPv6)
<i>I</i>				<i>Confidential</i>	
<i>K</i>	RedHat Linux derivative →Centos8 stream	BIND	9.16.x →9.16.39		DNS Cookies: No Minimal responses: yes MTU: 1500
		Knot DNS	3.1.x →3.1.9		
		NSD	4.x.x →4.6.1	--enable-root-server	

Letters	Operating System	Software	Version	Compile options	Other parameters
L	Ubuntu 18.04	Knot DNS	3.1.8		EDNS: 4096
		NSD	4.6.0		

M

Confidential

Table 1: Summary of answers to the RSO survey. This table contains information about software configured at the time of the survey.

Letters	Operating System	Software	Version	Compile options	Other parameters
C	AlmaLinux 9	Not mentioned, assume the same configuration		Not mentioned, assume the same configuration	
H	Linux →Debian bullseye	NSD	Regularly upgrade to recent NSD versions →4.6.1	Not mentioned, assume the same configuration	
K	RedHat Linux derivative →Centos8 stream	BIND	9.18.x →9.18.13	Not mentioned, assume the same configuration	
		Knot DNS	3.2.x →3.2.6		

Table 2: Summary of answers to the RSO survey. This table contains information about software that RSO's plan to deploy in the future.

In the remainder of the document we refer to the specific authoritative nameserver by { Letter, Operating System and Software } combination as follows: C Alma BIND 9.16.39, C CentOS BIND 9.16.39, D Linux NSD 4.1.20, D Linux NSD 4.6.1, E FreeBSD BIND 9.16.39, F FreeBSD BIND 9.16.39, G Linux BIND 9.16.29, H Linux NSD 4.5.0, K CentOS BIND 9.16.39, K CentOS BIND 9.18.13, K Knot 3.1.9, K Knot 3.2.6, K NSD 4.6.1, L Ubuntu Knot 3.1.8 and L Ubuntu NSD 4.6.0.

The five root server operators that request not to publish some or all of their software resulted in nine unique { Letter, OS, Software } combinations which we will refer to as X1 (conf), X2 (conf), X3 (conf), X4 (conf), X5 (conf), X6 (conf), X7 (conf), X8 (conf) and X9 (conf)

Resolver testbed

The testbed used for this report is based on a *resolver-testbed* developed and sponsored by ICANN. Our additional work that updates and automates the original *resolver-testbed* has been contributed back and is now available in ICANN's github repository for the testbed:

<https://github.com/icann/resolver-testbed> .

Complete documentation is provided in the code repository. Details relevant to this report are presented below.

Design

The original testbed used VirtualBox to contain the virtual machines (VMs) and virtual networks. It used manual steps for the setup and python scripts for provisioning and running the tests.

The current testbed still uses VirtualBox as the core virtualisation environment for VMs and virtual networking as it allows for a variety of host and guest operating systems. It is additionally complemented by Vagrant and Ansible for automated setup, provisioning and idempotency. These allow for trivial installations, easy reproducibility and provide guarantees about the state of the environment after several (re)configurations while testing the different naming schemes.

The current testbed also allows for local/hidden configuration. Local configuration files can be defined to augment already defined VMs and/or introduce new ones. The testbed is equipped to deal with such information that is not part of the public repository. This is crucial for this study as a number of root server operators have indicated that the operational information shared with the consortium is confidential.

The local configuration files that describe the root operators' setup are not part of the code repository and are only accessible by the consortium members.

Virtual machines and operating systems

The main virtual machines used in the testbed are the following:

gateway-vm. As the name suggests it functions as a gateway for both resolver and name server VMs also interconnecting their two distinct networks. It is thus the appropriate place for packet captures during measurements. There are no special requirements for the operating system so Debian bullseye was chosen.

servers-vm. A VM on the servers network used to host name server software. For this study, it is only used to serve the 'net.' zone (where appropriate based on naming scheme) and the check domains 'example.' and 'example2.'. The operating system chosen for this VM is FreeBSD 12.x. FreeBSD was already the choice of the original testbed and was kept for the convenience of firewall configuration.

resolvers-vm. A VM on the resolvers network used to host resolver software. For this study, it is used as the main host of all the tested resolver software. There are no special requirements for the operating system so debian bullseye was chosen.

root-servers. This is an Ansible group of VMs consisting of distinct VMs on selected OSes, hosting source compiled name server software that matches the root servers' operational environment as closely as possible. As with *servers-vm*, it resides on the servers network.

DNS software used

For the source compiled name servers on the *root-servers* VMs we refer the reader to the survey results from WP1. Additionally, a package from ports, *BIND 9.16.40* was used on *servers-vm* to serve the 'net.' zone and the check domains 'example.' and 'example2.'. Apart from facilitating the study, this name server's behavior is not relevant for the measurements.

For the *resolver software* the following resolvers and their versions were source compiled on resolvers-vm:

For BIND, versions 9.9.11, 9.10.8, 9.11.6, 9.12.4, 9.13.7, 9.14.10, 9.15.8, 9.16.41, 9.18.15 and 9.19.13. All versions are EOL except for 9.16.41 and 9.18.15 which are the current-stable, and version 9.19.13 which is the development one.

For Knot Resolver, versions 5.5.3 (5.5.0 released in March 2022) and 5.6.0 (released in January 2023).

For PowerDNS Recursor, versions 4.0.9, 4.1.15, 4.2.1, 4.7.5 and 4.8.4. All versions are EOL except for 4.7.5 (4.7.0 released in May 2022) and 4.8.4 (4.8.0 released in December 2022).

For Unbound, versions 1.5.10, 1.6.8, 1.7.3, 1.8.3, 1.9.6, 1.13.0, 1.14.0 and 1.17.1. All versions are EOL except for the latest one, 1.17.1 released in January 2023.

The old versions already available from the original testbed were kept since resolver software is often part of routers and middleboxes, machines once deployed and often forgotten. They can also reveal differences (if any) between old versions and the currently supported versions.

Virtual networking

The following distinct virtual networks were used in the environment:

Control NAT (10.0.2.0/24). This is the default NAT network in VirtualBox. It is used (and configured in the Vagrant boxes) by Vagrant to control and provision the VMs. It is configured as the first interface of every VM and is used for the default IPv4 route (if not configured otherwise) allowing access to the Internet. Vagrant also uses this interface to port forward the VMs local port 22 to a localhost port on the host for ssh access to the VM.

Servers network (172.21.0.0/16, fd00::21:0:0/96). This is the network where *servers-vm* and every VM in the root-servers group is connected to. One of the interfaces of *gateway-vm* is also connected to this network.

Resolvers network (172.20.0.0/24, fd00::20:0:0/96). This is the network where *resolvers-vm* is connected to. One of the interfaces of *gateway-vm* is also connected to this network.

The network is as described in the figure below.

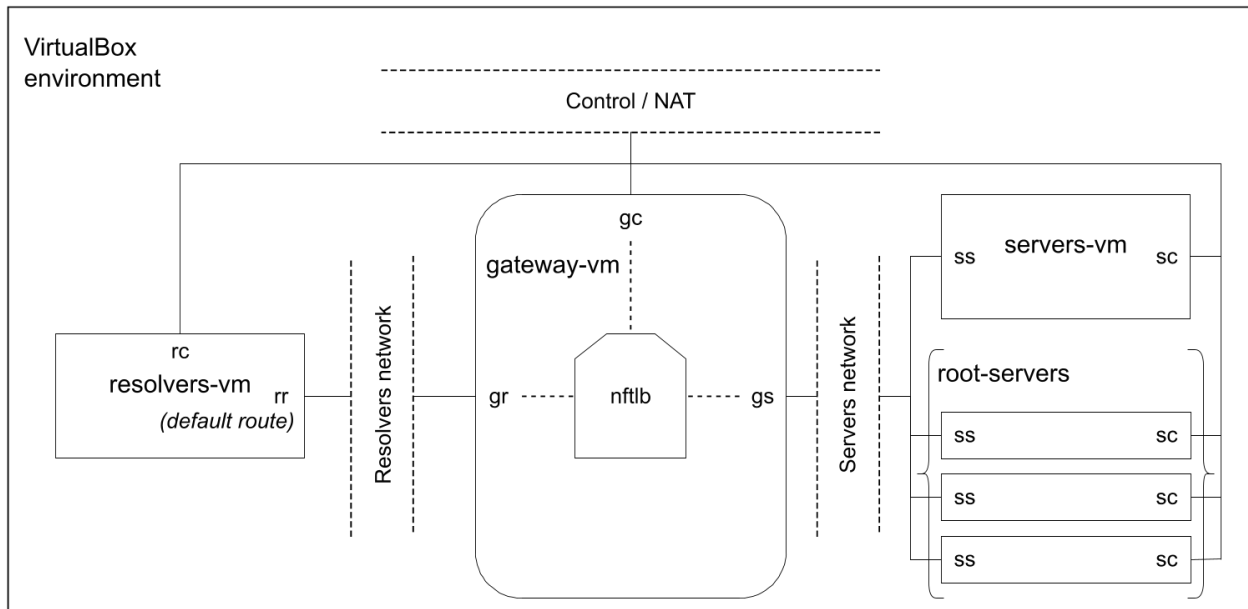


Figure 6: Testbed network layout

The network cards for each VM are the following:

- gateway-vm
 - gc for control NAT
 - gr on the resolvers network (172.20.0.1/24, fd00::20:0:1/96)
 - gs on the servers network (172.21.0.1/16, fd00::21:0:1/96)
- servers-vm
 - sc for control NAT
 - ss on the servers network (172.21.0.2/16, fd00::21:0:2/96)
- resolvers-vm
 - rc for control NAT
 - rr on the resolvers network (172.20.0.2/24, fd00::20:0:2/96)
- root-servers:
 - sc for control NAT
 - ss on the servers network (172.21.x.yz/16, fd00::21:x:yz/96)

The letters on the above root-servers addresses have the following IP numbering scheme:

- x encodes the root letter
- y encodes the OS
- z encodes the name server software, 0 reserved for the system itself

For example, 172.21.2.22 is the IPv4 address of the second OS and the second nameserver software for the B-root server.

There is an additional IP numbering scheme for resolver software on *resolvers-vm*. It is used for both outgoing traffic as well as for listening to queries in the resolver software configurations and helps identify each resolver in the packet captures. The scheme is 172.20.0.r and fd00::20:0:r where $r > 2$ and is encoded as:

- $3 \leq r < 20$, available range for Unbound resolver software
- $23 \leq r < 40$, available range for BIND resolver software
- $43 \leq r < 60$, available range for PowerDNS Recursor software
- $63 \leq r < 80$, available range for Knot Resolver software

This topology allows for all the relevant to the study inter-routing to flow through the *gateway-vm*, making it the ideal target for packet captures. *resolvers-vm* is further configured to have the default route via *gateway-vm*, thus capturing possible traffic that was not anticipated.

nftables load balancer (nftlb) is installed on the *gateway-vm* to further orchestrate the traffic. It is configured for Destination Network Address Translation (DNAT) from resolvers to name servers and, if needed, configured to target specific OSeS and name server software based on the naming scheme and setup under test.

One caveat of the testbed is that outgoing traffic from the virtual setup is only possible via the control NAT network. This is IPv4 only by default. Outgoing traffic is used for packages and source code installation. All the relevant traffic for the measurements stays confined in the virtual topology. As such, no further measures were taken to enable IPv6 for outgoing traffic.

Zone files for the naming schemes from RSSAC028

Generation of the zone files for the different naming schemes to be deployed on the root servers is automated via a Makefile. The Makefile uses syntax specific to the GNU implementation of make. The generated root and root servers zone files are based on a version of the root zone and the 'root-servers.net' zone that were downloaded from <https://www.internic.net/domain/>. Naming schemes 5.1 and 5.2 include an alternative version of the '.net' zone (to facilitate alternative 'root-servers.net.' zones) which is also generated by the Makefile. Finally two check domains are generated for performing test queries without leaving the testbed environment: 'example.' and 'example2.'

The TTL values for the SOA, the NS, the delegation NS and the DS RRsets in the generated root, the root servers and '.net' zones, are based on the values currently in use by those zones

on the Internet. Also the DNSSEC algorithms and key sizes are based on what is currently in use by those zones on the Internet. All those parameters are defined at the start of the Makefile and can be altered.

All generated zones are signed by `ldns-signzone` with inception date January 1, 2023 and expiration date January 1, 2033. Keys are generated on request with `ldns-keygen`. In case zones have the same name (such as with the different versions of the root zone), the same key is reused. The generated key for the root zones is also used to configure the trust-anchor for the resolver software on the `resolvers-vm`.

All downloads, generated DNSSEC keys and intermediary files are stored in the repository for reproducibility and also to be able to change only certain aspects of the generated zone files.

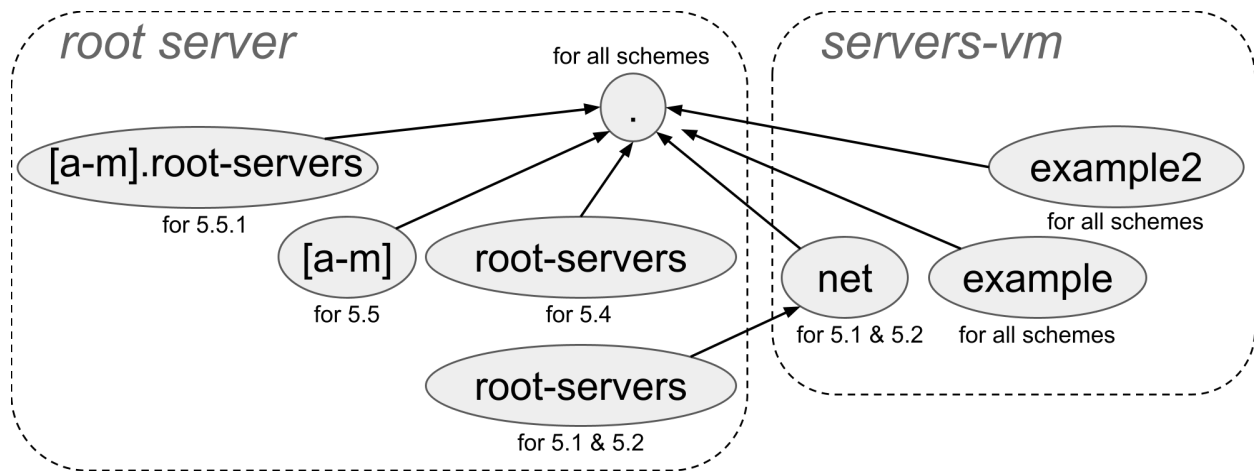


Figure 7: Zone files and their responsible servers

For all name schemes, the zone file for the root is provisioned on all the root servers. Naming schemes 5.1, 5.2 and 5.4 have a specific zone for the root servers, which are also provisioned on all the root servers. With naming schemes 5.5 and 5.5.1, each root server has its own zone, which are accordingly provisioned. Naming schemes 5.1 and 5.2 have an alternative version of '.net' which is provisioned on `servers-vm`. The two check domains to receive and answer the test query: 'example.' and 'example2.' are provisioned on `servers-vm`.

The IP addresses assigned to the individual root servers in the zone files are different from the addresses currently in use by the root servers on the internet. This allows us to determine whether a resolver succeeds priming. When priming succeeds, the resolver learns the new IP addresses from the priming response and will eventually target the different (new) IP addresses to reach the root.

Note that the [nftables load balancer \(nftlb\) on gateway-vm](#) directs both the IP addresses as they are currently used for the root servers on the internet, as well as the different (new) IP addresses to the appropriate { Letter, OS, Software }. The different IP addresses are *not* taken from the Special-Purpose Address registries, because many resolvers are protected to not target special purpose addresses as well as link and/or site local addresses.

The zone files are prefixed with the version number of the naming scheme (as defined in [Naming schemes from RSSAC028](#)) followed by a dash. For example 5.4-root is the file name used for the root zone for naming scheme 5.4.

For all naming schemes 5.1 to 5.6.1 root hints files are generated as well. The root hints **do** contain the IP addresses for the root servers as they are currently used on the internet.

Orchestration

Ansible playbooks are used extensively to orchestrate the virtual environment. After the initial network/system provisioning where the interfaces are set up and all required software is installed, different playbooks are used for provisioning before, during and after each measurement.

Simulating the proprietary authoritative name server software

Two proprietary authoritative softwares were reported in (confidential) survey results. We arranged with the providers of those software to perform tests locally in their own testbeds and report the results back to us. We used these results to recreate the behavior of the proprietary software in the testbed as close as possible.

The tests consisted of (for each naming scheme) a series of DiG queries targeted at a testbed instance of the software loaded with the zone files for the naming scheme. The parameters for the test queries are based on the analysis of what we have observed from resolvers in our testbed.

```
noedns : dig @<ipv4> +norec +ignore +noedns . NS
noedns : dig @<ipv6> +norec +ignore +noedns . NS
16384 : dig @<ipv4> +norec +ignore +bufsize=16384 +cookie . NS
16384 : dig @<ipv6> +norec +ignore +bufsize=16384 +cookie . NS
16384 DO : dig @<ipv4> +norec +ignore +bufsize=16384 +cookie +dnssec . NS
16384 DO : dig @<ipv6> +norec +ignore +bufsize=16384 +cookie +dnssec . NS
```

With one of the proprietary name servers, the responses never contained signatures for address records in the additional section and the largest response was lower than 1232 bytes. From this we deduced that TCP replies could always contain the full response. With the other proprietary name server, responses did contain signatures for address records in the additional section and the largest responses (for schemes 5.3 and 5.6) had a reduced set with the truncated (TC) flag set. We asked the provider to perform additional queries over TCP in order to simulate those responses too:

```
TCP DO : dig @<ipv4> +norec +bufsize=16384 +tcp +cookie . NS
TCP DO : dig @<ipv4> +norec +bufsize=16384 +tcp +cookie +dnssec . NS
```


The Idns software³ contains a tool to return crafted DNS messages in response to certain queries: Idns-testns. The tool reads a data file containing a series of entries with *match* directives followed by a detailed description of a reply, including what elements to take from the query such as query ID, query name and/or type.

Idns-testns has been extended to be able to match queries with a certain EDNS UDP Message size, as well as to be able to answer delegations. These changes are in the current develop branch of the Idns repository. Idns-testns has also been extended to listen on addresses configured in the data file. That last change allows us to treat Idns-testns as another name server software in the testbed, with the data file as configuration file. We considered that last change too specific to be included in mainline Idns. It is available in the rssac028 branch⁴.

The Idns-testns data files for the two proprietary name server softwares are based on

1. the responses returned from the software providers,
2. the query parameters that were chosen to allow us to deduce the properties of the responses for other parameters (see [Priming responses from all root server software](#) for and description of how these were chosen),
3. the queries we perceived coming from the resolvers to root servers in our testbed,
4. live responses perceived on the Internet for delegations and for direct DS, DNSKEY, A and AAAA responses. For example when queried directly for an address of a root letter, one of the proprietary name servers adds addresses of the other root servers to the additional section.

Installation

The testbed can be installed on a host running VirtualBox, Vagrant and Ansible. It has been tested on both Linux and FreeBSD hosts and the current measurements were carried out on a FreeBSD 12.3 host.

The steps to install the environment include:

1. Installation of the VMs. This is accomplished through Vagrant by grabbing defined Vagrant boxes and importing them to VirtualBox. Linked VM clones are used in this step that drastically reduce the import overhead (time and space) of VMs that share the same box (OS). Ansible is defined as the default provisioner for the VMs and Vagrant exports all the necessary connection information in an Ansible compatible format to be used by further Ansible invocations.
2. Initial network/system provisioning. This step is accomplished with Ansible playbooks that setup and configure the network interfaces and compile (if necessary) and install the required software and services.

For more details on the environment we refer the reader to the repository of the [testbed](#).

³ <https://nlnetlabs.nl/projects/ldns/about/>

⁴ <https://github.com/NLnetLabs/ldns/tree/rssac028>

Running the testbed

A testbed run is accomplished via Ansible playbooks, and scripts invoking Ansible playbooks. Collecting results for a set of configuration parameters consists of 13 steps performed in three stages: I.) Setup and configuration of the testbed, II.) Running the test queries and capturing traffic, and III.) Processing the packet captures, which we'll address individually in the following sections.

I.) Setup and Configuration

The following parameters can be configured:

- a. *naming scheme* : All name server software (all { Letter, OS, Software } combinations and *servers-vm*) is configured to load the zone files for the selected naming scheme to analyze.
- b. *root server IP addresses* : The zone files to be loaded for the naming scheme contain either IP addresses for the root servers as currently used on the internet, or alternative IP addresses. The latter allows to determine whether priming succeeds, in which case the new (alternative) IP addresses are targeted after priming.

The testbed results presented in this report all used alternative root server IP addresses in the zone files.
- c. *DNAT & load balancer* : Root server IP addresses (the ones used on the Internet as well as the alternative ones) are destination NATted (and load-balanced) to either the { OS, software } combinations for the associated letter, or are all targeted to a single { Letter, OS, Software } instance. The latter allows us to study more directly the effect of that combination on resolvers.
- d. *root hints* : The resolver software on *resolvers-vm* can be equipped with root hints (or not which means the resolver uses the build-in hints conforming with naming scheme 5.1). The IP addresses in the hints file can either be the ones as currently used on the Internet or alternative IP addresses.

The testbed results presented in this report all used root server IP addresses as currently used on the Internet in the hints.
- e. *current time* : The time on the *resolvers-vm* can be configured to be outside the period that DNSSEC signatures are valid.
- f. *TCP ability* : TCP access to the name servers can be blocked on *resolvers-vm*.

The testbed is setup and configured with these parameters in three steps:

1. All authoritative name server software in the root-servers group as well as the authoritative name server software on *servers-vm*, is configured to load the zone files based on the *a.) naming scheme* and *b.) root server IP addresses* parameters.

The testbed results presented in this report all used alternative root server IP addresses in the zone files.

The configured name server software is subsequently started and ready to answer queries.

2. Based on the value of the *c.) DNAT & load balancer* parameter, the nftables load balancer (nftlb) on *gateway-vm* is configured to either:
 - load balance all root server IP addresses for a letter to the { OS, Software } associated with that letter, or
 - let all root server IP addresses end up at a single { Letter, OS, Software } instance.

3. On *resolvers-vm*:

- The time is set to the current time, or a time outside of the validity period of the DNSSEC signatures in the zone files based on the *e.) current time* parameter.
- Outgoing TCP traffic targeted at the name servers (the root-servers and *servers-vm*) is blocked (or not) with iptables based on the *f.) TCP ability* parameter.
- All resolver software is configured with a root hints (or not) based on the value of the *d.) root hints* parameter.

The testbed results presented in this report all used root server IP addresses as currently used on the Internet in the hints files. (see [Zone files for the naming schemes from RSSAC028](#)).

The resolver software is **not** started after configuring.

II.) Running the test queries and capturing traffic

After setup and configuration, the following steps are performed (by an ansible playbook) to measure resolver behavior under the configured parameters:

4. Start *tcpdump* on *gateway-vm* listening on all interfaces and capturing all UDP and TCP traffic to port 53, writing the capture to disk.
5. Start all resolver software on *resolvers-vm*.
6. Wait (5 seconds) to allow capturing of possible (priming) traffic automatically initiated by resolvers.

7. Query all the resolver software on the [for that software designated IPv4 address](#) from *gateway-vm* for the first test with query name 'example.' and query type A.

Querying from *gateway-vm* makes the test queries show up in the packet captures which is useful in analysis to pinpoint the moment a resolver was queried in relation to the queries *from* the resolver.

8. Wait (5 seconds) after resolution to allow capturing of possible ongoing traffic from resolvers after replying to the client.
9. Query all the resolver software from *gateway-vm* for the second test with query name 'example2.' and query type AAAA to see if the success/failure of priming during the initial query has a different effect now.
10. Wait (5 seconds) after resolution to allow capturing of possible ongoing traffic from resolvers after replying to the client.
11. Stop and collect the *tcpdump* packet capture on *gateway-vm*.

III.) Processing the packet captures

The captured network traffic is processed for further analysis with two steps:

12. In the captures [each resolver software can be identified by IP address](#). For each capture, first the queries and responses are extracted per resolver software. Then the DNS messages from and to that resolver software are parsed and converted into textual representation of the message on a single line displaying:
 - The resolver software,
 - A relative timestamp in seconds starting from the first packet seen with that resolver
 - The IP version, the transport used and the direction of the message
 - The target or source of the message,
 - The query name and query type,
 - Between parentheses, for queries:
 - i. the EDNS UDP Message size, and whether the DO flag is set,and for responses:
 - ii. The size of the returned message,
 - iii. Whether or not the TC flag is set (i.e. the message was truncated),
 - iv. The number of RRs in the answer section / the number of RRsigns,
 - v. The number of RRs in the authority section / the number of RRsigns,
 - vi. The number of RRs in the additional section / the number of RRsigns,
 - vii. The EDNS UDP Message size, and whether the DO flag is set.

The test queries from and to the resolver are sent from *gateway-vm* and can be identified by IP address as well. They appear in the textual representation as *resolvers-vm* for target or source.

Below is an example result of this processing for bind-9.12.4 for naming scheme 5.1 (with alternative IP addresses for the root servers), with nftlb configured to load-balance the IP address of each letter over the authoritative software in use by that letter.

```
bind-9.12.4 0.000000 --4-> @resolvers-vm example. A (udp=4096)
bind-9.12.4 0.001737 --6-> @a example. A (udp=512,D0)
bind-9.12.4 0.001904 --6-> @a . NS (udp=512,D0)
bind-9.12.4 0.002290 <--6- @a . NS (sz=28,TC,udp=1232,D0)
bind-9.12.4 0.003173 -T6-> @a . NS (udp=4096,D0)
bind-9.12.4 0.003315 <-T6- @a . NS (sz=1097,an=13/1,ar=26,udp=4096,D0)
bind-9.12.4 0.003583 <--6- @a example. A (sz=432,ns=2/1,ar=2,udp=4096,D0)
bind-9.12.4 0.004027 --6-> @servers-vm example. A (udp=512,D0)
bind-9.12.4 0.004542 <--6- @servers-vm example. A (sz=279,an=1/1,udp=1232,D0)
bind-9.12.4 0.005688 <--4- @resolvers-vm example. A (sz=80,an=1,udp=4096)
bind-9.12.4 5.300018 --4-> @resolvers-vm example2. AAAA (udp=4096)
bind-9.12.4 5.300815 --6-> @New-K example2. AAAA (udp=512,D0)
bind-9.12.4 5.301407 <--6- @New-K example2. AAAA (sz=440,ns=2/1,ar=2,udp=1232,D0)
bind-9.12.4 5.301842 --6-> @servers-vm example2. AAAA (udp=4096,D0)
bind-9.12.4 5.302224 <--6- @servers-vm example2. AAAA (sz=517,an=1/1,ns=1/1,udp=1232,D0)
bind-9.12.4 5.302596 <--4- @resolvers-vm example2. AAAA (sz=93,an=1,udp=4096)
```

In this log we can see that bind-9.12 upon reception of the first test query, contacts a root server (letter A) at the old IPv6 address for *that* query, from which it learns the delegation at 0.003583. The priming query is sent only after the test query. The priming response is truncated (because it doesn't fit within the requested 512 bytes UDP Message size), which triggers a requery over TCP. The second test query is sent to the alternative IP address (for letter K), from which we can conclude that priming succeeded.

13. Those results are then further processed to extract metrics, such as:

- The moment of priming:
 - i. immediately after startup,
 - ii. after receiving the test query, but before starting work on the test query,
 - iii. after starting work on the test query.
- Whether or not priming succeeded
- Whether or not the test queries resolved
- The number of priming queries (qname: . , qtype: NS) sent to the root servers
- The number of follow up address queries (A and AAAA) sent to the root servers
- Transport used
- Size of the responses

A “test run” of the testbed consists of a series of the above described 13 steps in three stages for collecting testbed results, iterating the *a.) naming scheme* parameter over all naming schemes described in [Naming schemes from RSSAC028](#). The metrics resulting from a “test run” are then further organized in tables for visual inspection and analysis.

There are two kinds of test runs. In one, the *c.) DNAT & load balancer* parameter in the testbed is configured so that the IP addresses for each letter end up (load balanced) at the root server for that letter. All other parameters remain the same. These kinds of test runs reflect how resolvers would behave in the root server system as currently deployed on the internet and result in a table of metrics for naming schemes (as columns) by resolvers (as rows).

The other kind of “test run” is intended to analyze the behavior of resolver software in relation to the responses from each { Letter, OS, Software } combination. To do that, the *c.) DNAT & load balancer* parameter is configured to direct all root server IP addresses to a specific { Letter, OS, Software }; Then for each naming scheme, all the three stages are furthermore performed iterating over each unique { Letter, OS, Software } combination.

The second kind of test run results in metrics organized in a cube with three axes: Naming Schemes, Resolvers and { Letter, OS, Software } combinations. From this cube we extract three kind of tables for visual inspection and analysis:

1. { Letter, OS, Software } by Resolver
 - per Scheme
2. Scheme by { Letter, OS, Software }
 - per Resolver
3. Scheme by Resolver
 - per { Letter, OS, Software }

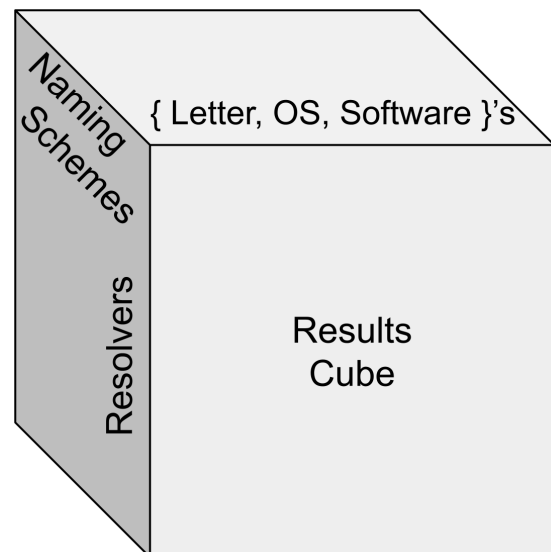


Figure 8: Organisation of measurement results

In [Testbed results](#) the kind of table is indicated alongside the presented metric tables.

To highlight the differences in the metrics displayed in the tables, equal rows and columns are collated. For the { Letter, OS, Software } combinations equal metrics often relate to groups of combinations that have equal properties for priming responses. Those groups are identified in [Priming responses from all root server software](#).

Main Analysis

Acceptable response sizes

In this section, we carry out a literature study with the goal to identify the typical maximum supported packet size on the Internet. The DNS community discussed packet size support extensively around the DNS Flag Day in 2020⁵. The DNS Flag Day 2020 recommended resolver and authoritative name server operators a maximum safe message size of 1232 bytes and required the support of the TCP fall-back mechanism.

The DNS Flag Day 2020 focussed on the communication between recursive resolvers and authoritative name servers. However, for this study, the supported packet size of validating stub resolvers is relevant as well. For this reason, we also include studies that measure packet size support between the stub resolver and the recursive resolver.

Also, DNS messages transmitted via TCP have the potential to transport larger messages than DNS messages transmitted via UDP. We include studies on DNS over TCP support as well.

Note, that the studies that we discuss here rely on different measurement methodologies and vantage points. For this reason they report varying supported message sizes and TCP support.

In general, we analyze studies that were published around 2020 and later.

Support of response sizes according to Internet standards

Originally, DNS messages over UDP were limited to 512 bytes. The EDNS(0) extension headers increased the maximum message size theoretically to 64 KB. However, the MTU most commonly found in the core of the Internet is around 1500 bytes. If messages exceed this size, the responding server can signal to the requesting resolver to fall back to TCP.

Support of response sizes according to measurements in the wild

Signaled supported message size: With EDNS(0), receivers of DNS responses can signal the largest possible DNS message they can receive via UDP. The DNS Flag Day 2020 recommends a supported buffer size of at least 1232 bytes.

In October 2020, Moura et al. have shown that 4.4% of resolvers announce an EDNS(0) buffer size of 1232 bytes to the authoritative name servers of the .nl ccTLD. 40% of resolvers in this study announce a size of 4069 bytes⁶. Six months later, a study by Fukuda et al. reported that

⁵ DNS OARC. 2020. "DNS flag day: 2020." <http://www.dnsflagday.net/2020/>.

⁶ Moura, Giovane C., Moritz Müller, Marco Davids, Maarten Wullink, and Cristian Hesselman. 2021. "Fragmentation, truncation, and timeouts: are large DNS messages falling to bits?"

20% of observed resolvers announce a buffer of 1232 bytes to the authoritative name servers of the .jp ccTLD⁷.

Huston et al. used an advertisement network to measure how many users rely on resolvers that comply with the Flag Day recommendations⁸. By December 2020, 5% of users rely on resolvers that announce a buffer size of 1232 bytes. 7% rely on resolvers that announce 1400 bytes and 80% of users rely on resolvers that announce a buffer size of 4096 bytes.

Additionally, they observed that some resolvers alter their behavior when resolving name server names, and thus might also do so when querying the root name servers. In some 30% of cases the EDNS(0) buffer size is either dropped from the query, or dropped below 1452 octets.

Kosek et al. used RIPE Atlas probes to study the announced buffer size of recursive resolvers towards their clients⁹. Here, 30% of probes relied on resolvers that announce a buffer size of 1232 bytes. 28% relied on resolvers announcing 4096 bytes.

Actually supported message size: An announced EDNS(0) buffer size does not necessarily mean that messages of this size can be reliably transported. In 2017, Huston employed measurements using an advertising network to test the transport of larger DNS messages between recursive resolvers and authoritative name servers on IPv6¹⁰. He reported a 5.01% drop rate for DNS messages of 1428 bytes and a 7.1% drop rate for DNS messages of 1,886 bytes.

Koolhaas et al. used RIPE Atlas probes for a similar study¹¹. They reported a failure rate of 0.93% for a message size of 1260 bytes and below in IPv4, which raises to a failure rate of 1.56% with a message size of 1400 bytes. For resolvers that rely on IPv6, drop rates stayed below 1% until a message size of 1460 byte. Message sizes of 1492 bytes caused drop rates of around 1.3%.

Passive and Active Measurement: 22nd International Conference, PAM 2021, 2021.
https://link.springer.com/chapter/10.1007/978-3-030-72582-2_27.

⁷ Kensuke, Kensuke, Yoshitaka Aharen, Shinta Sato, and Takeshi Mitamura. 2022. "Characterizing DNS query response sizes through active and passive measurements." *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*.

⁸ Huston, Geoff, and Joao L. Damas. 2021. "Measuring DNS Flag Day 2020." OARC 34.

⁹ Kosek, Mike, Trinh V. Doan, Simon Huber, and Vaibhav Bajpai. 2022. "Measuring DNS over TCP in the era of increasing DNS response sizes: a view from the edge." *ACM SIGCOMM Computer Communication Review* 52, no. 2.

¹⁰ Huston, Geoff. 2017. "Dealing with IPv6 fragmentation in the DNS." APNIC Blog.
<https://blog.apnic.net/2017/08/22/dealing-ipv6-fragmentation-dns/>.

¹¹ Koolhaas, Axel, and Tjeerd Slokker. 2020. "Defragmenting DNS: Determining the optimal maximum UDP response size for DNS."

In 2021, Huston et al. repeated and extended their study to also include resolvers that rely on IPv4. Overall, they concluded that “[t]here is a slight increase in drop rates above 512 octets to around 0.5%” but “no visible change in drop rates in payloads up to 1500 octets in size”.

Koolhaas et al. also studied the drop rates between stub and recursive resolver. Here, the drop rate via IPv4 stays below 2% as long as messages do not exceed 1448 bytes. For IPv6 connections, the drop rate stays below 2% with messages smaller/equal 1280 byte. Messages larger than that, but smaller than 1460 bytes have a failure rate of below 7%.

TCP support: When a message does not fit in the announced buffer, the server should signal to the client to retry the query via TCP.

In 2017, Huston reported a drop rate of 2.67% for messages of size 1428 bytes. When forcing resolvers to rely on TCP only, this drop rate grew to 5.01%. 1.97% of resolvers were unable to even transmit 169 byte small messages. This indicates that a small percentage of measured resolvers were unable to fall back to TCP. In their study four years later, they report a failure rate of DNS over TCP of between 1% and 2% across all transactions and message sizes.

In 2022, Mao et al. measured DNS over TCP support of public DNS resolvers¹². In order to understand whether these open resolvers are actively used, they used traffic traces of the authoritative name servers of major CDN to analyze how much DNS traffic these open resolvers were responsible for. They found that 2.7% to 4.8% of open resolvers were not able to fall back to TCP. These resolvers were responsible for 2.7% to 4.8% of all queries at the CDN.

In the same year, Kosek et al. relied on RIPE Atlas probes to study DNS over TCP support directly at the probes and at larger public DNS providers. Support of DNS over TCP at RIPE Atlas probes was poor, where only 25% were able to send queries over TCP reliably. In contrast, DNS over TCP support at public DNS services was overall on par with DNS over UDP support, with a few exceptions. Additionally, they also measured the performance of DNS over TCP and concluded that DNS over TCP to public DNS resolvers is 40% slower compared to DNS over UDP.

Summary

The maximum supported message size depends on whether messages are transported via UDP or TCP. Also the underlying IP version has an impact. Across the discussed studies, drop rates for messages smaller or equal than 1480 bytes stayed below 2%. Messages above this size are transported less reliably. This has likely to do with issues with TCP support, with failure rates between 1% and 5%. Failure rates for larger DNS messages and for DNS messages via TCP are worse when looking at the communication channel between stub and recursive resolver.

¹² Mao, Jiarun, Michael Rabinovich, and Kyle Schomp. 2022. “Assessing Support for DNS-over-TCP in the Wild.” *Passive and Active Measurement: 23rd International Conference PAM 2022*.

Reproducing RSSAC028 Appendix A results

As a baseline measurement for the “[acceptable response sizes for priming queries](#)” study goal, we have measured priming response sizes for the RSSAC028 naming schemes, with the authoritative software that was also used to create the table in Appendix A of the RSSAC028 report. This involved installation of older versions of BIND, NSD and Knot-DNS name servers. Reproduced sizes can be found in [Appendix B.1](#), as well as the sizes for the recent versions of the tested software (BIND, NSD and Knot-DNS).

The results of the older versions match those of RSSAC028, except for scheme 5.3 IPv4 DNSSEC with NSD which is 13 bytes smaller than in the report. The responses from the recent versions no longer produce large responses. All versions of BIND, NSD and Knot-DNS, released after October 1st 2020 comply with the recommendations of DNS Flag Day 2020 and have the EDNS buffer size configured default to 1232.

Priming responses from all root server software

To better understand resolver behavior in response to priming responses, we have looked at attributes of those responses for all the current and future root server software; that is all current and future { Letter, OS, Software } combinations. We observed several properties that priming responses may or may not have coming from the different software for the different schemes:

1. The number of IPv4 and IPv6 address records in the additional section,
2. The number of signatures in the packet in the answer and the additional section,
3. Whether or not the TC (truncate) flag is set in the response.

These properties vary in response to parameters of the priming query:

1. The transport used for the query,
2. Whether or not the query uses EDNS,
3. The maximum UDP Message size advertised in EDNS,
4. Whether or not the query had the DO flag set requesting DNSSEC resource records.

Most prevalent query parameters

We inventoried the most prevalent parameters for queries to the root from the DNS-OARC Day In The Life of the Internet (DITL) collection effort¹³. At the time of writing, packet captures from all A, D, F, G, H and J root letter anycast nodes were available. The captures were taken on 28 and 29 March 2023 and contained 84,796,581,713 queries. [Table 3](#) shows the most prevalent (Top-6) query parameters.

¹³ <https://www.dns-oarc.net/ditl/2011>

Percentage of queries	EDNS UDP message size	DO flag set
28.08%	4096	✓
25.27%	1232	✓
10.92%	No EDNS	
5.13%	512	✓
4.10%	4096	
3.55%	TCP	✓

Table 3: Most prevalent query parameters from A, D, F, G, H and J 2023 DITL data

The testbed resolvers always send EDNS with DO flag set and 4 distinct UDP Message sizes:

EDNS UDP message size	DO flag	Software
512	✓	: bind-9.10.8 ... 9.16.41
1232	✓	: bind-9.16.41 ... 9.19.13, knot-resolver-5.5.3 ... 5.6.0, pdns-recursor-4.2.1 ... 4.8.4, unbound-1.5.10, 1.13.0 ... 1.17.1
1680	✓	: pdns-recursor-4.0.9 ... 4.1.15
4096	✓	: bind-9.9.11, 9.11.6 ... 9.15.8, unbound-1.5.10 ... unbound-1.9.6
TCP	✓	: all resolver software

Table 4: Most prevalent query parameters from the resolver testbed

For the priming response analysis, we have chosen query parameters - based on the perceived most prevalent parameters - that allow us to deduce the aspects of the responses for other parameters. Those parameters are No EDNS, UDP message size 512 with DO flag, UDP message size 4096 with DO, and TCP with and without DO flag. The TCP responses will show the sizes without size constraints. The UDP message size 4096 is chosen over 1232, because many of the root-server authoritative name server software already restrict the maximum UDP message size to 1232. In this way we highlight the exceptions.

All priming queries have been sent with DiG 9.16.37. The following DiG commands have been issued to get the responses for the query parameters on both the IPv6 and IPv4 addresses of all the { Letter, OS, Software } combinations:

```

noedns: dig @<ip> +norec +ignore +noedns -c IN -q . -t NS
512 DO: dig @<ip> +norec +ignore +bufsize=512 +dnssec -c IN -q . -t NS
4096 DO: dig @<ip> +norec +ignore +bufsize=4096 +dnssec -c IN -q . -t NS
TCP: dig @<ip> +norec +tcp +nodnssec -c IN -q . -t NS
TCP DO: dig @<ip> +norec +tcp +dnssec -c IN -q . -t NS

```

These arguments to DiG will cause it to send a DNS Cookie option with EDNS. This will enable us to distinguish name servers that respond to DNS Cookies.

Several of the { Letter, OS, Software } combinations produce equal results. We have ordered those in eight groups and will treat them per group in the following sections.

Group I

The first group with equal priming responses consists primarily of BIND software. In all the priming response properties tables, the full list of { Letter, OS, Software } combinations for which those responses apply is in the table caption.

The top row in the table displays the query parameters:

- noedns : UDP transport without EDNS.
- 512 DO : UDP transport with an EDNS UDP message size of 512 and the DO flag set.
- 4096 DO : UDP transport with an EDNS UDP message size of 4096 and the DO flag set.
- TCP : TCP transport with EDNS. The row below the top row further distinguishes between TCP without the DO flag set and with the DO flag set.

The properties of the responses are in the column below the parameters for each naming scheme on each bordered row:

- size : The size of the response.
- NS : The number NS resource records in the answer section. An RRSIG for the NS RRset is indicated with "/ 1" appended to the value.
- A : The number of A resource records in the additional section. The number of signatures for those resource records is indicated with a number after a slash "/".
- AA : The number of AAAA resource records in the additional section. The number of signatures for those resource records is indicated with a number after a slash "/".
- tc : Whether or not the TC flag is set, indicated by a check mark.
- # sigs : The number of signatures in the response over TCP.

The name schemes are indicated before the (bordered) rows. If more than one name scheme produces the same value, they are combined. With all current root-server software, naming schemes 5.1 and 5.2 produce equal priming responses.

When the properties of the responses differ depending on whether IPv4 or IPv6 was used, the content of the table cell is split in two rows, the top one showing the values for IPv6 and the bottom one showing them for IPv4.

The responses in [Table 5](#) have the following properties.

- Addresses for the values of the NS records are added in the additional section.
 - First the addresses for the IP version over which the query was sent,
 - Then the addresses of the other IP version as space permits.
- No RRSIGs are ever included in the additional section.
- When the RRSIG for the . NS RRset does not fit, the TC flag is set.
- All responses with the TC flag set have an empty answer and additional section.
- 5.6 and 5.6.1 are the only schemes that do not have the TC flag set for any of the query parameters. This is the case for all groups except for [Group VII](#) and [Group VIII](#).
- All UDP responses stay below 1232 bytes. Only for scheme 5.5.1 the TCP DO responses exceed the 1232 limit, and hence this is the only scheme for which the additional address records are incomplete with “4096 DO” query parameters.
- ⚠ No additional address records are added with naming schemes in which they are authoritative in the root zone itself (5.3, 5.3.1, 5.6 and 5.6.1). This behavior is only seen in this group and in [Group II](#) which consists of only BIND.
- The software in this group responds to DNS Cookies.

		noedns				512 DO				4096 DO				TCP		#
		size	NS	A	AA tc	size	NS	A	AA tc	size	NS	A	AA tc	size	size	
5.1&	6	492	13	0	9	56	0	0	0 ✓	1137	13/1	13	13	851	1137	1
5.2	4	504	13	13	2											
5.3		236	13	0	0	56	0	0	0 ✓	561	13/1	0	0	275	561	1
5.3.1		211	13	0	0	56	0	0	0 ✓	536	13/1	0	0	250	536	1
5.4	6	488	13	0	9	56	0	0	0 ✓	1133	13/1	13	13	847	1133	1
	4	500	13	13	2											
5.5	6	500	13	0	8	56	0	0	0 ✓	1173	13/1	13	13	887	1173	1
	4	512	13	13	1											
5.5.1	6	503	13	0	3	56	0	0	0 ✓	1220	13/1	7	13	1030	1316	1
	4	499	13	5	0					1232	13/1	13	10			
5.6		42	1	0	0	367	1/1	0	0	367	1/1	0	0	81	367	1
5.6.1		46	1	0	0	371	1/1	0	0	371	1/1	0	0	85	371	1

Table 5: Response sizes and properties for C Alma BIND 9.16.39, C CentOS BIND 9.16.39, F FreeBSD BIND 9.16.39, G Linux BIND 9.16.29, X4 (conf), X6 (conf) and X9 (conf)

Group II

This Group consists of only BIND software. This group has similar properties as [Group I](#), except that DNS Cookies are missing in the responses, which results in 28 byte smaller responses. The smaller responses have the additional effect that also scheme 5.3.1 does not cause a TC flag set for any of the query parameters. A property it has in common with [Group III](#), [IV](#), [V](#) and [VI](#).

		noedns				512 DO				4096 DO				TCP #		
		size	NS	A	AA tc	size	NS	A	AA tc	size	NS	A	AA tc	size	DO si	
5.1&	6	492	13	0	9	28	0	0	0 ✓	1109	13/1	13	13	823	1109	1
5.2	4	504	13	13	2					1109	13/1	13	13	823	1109	1
5.3		236	13	0	0	28	0	0	0 ✓	533	13/1	0	0	247	533	1
5.3.1		211	13	0	0	508	13/1	0	0	508	13/1	0	0	222	508	1
5.4	6	488	13	0	9	28	0	0	0 ✓	1105	13/1	13	13	819	1105	1
	4	500	13	13	2											
5.5	6	500	13	0	8	28	0	0	0 ✓	1145	13/1	13	13	859	1145	1
	4	512	13	13	1											
5.5.1	6	503	13	0	3	28	0	0	0 ✓	1224	13/1	9	13	1002	1288	1
	4	499	13	5	0					1232	13/1	13	11			
5.6		42	1	0	0	339	1/1	0	0	339	1/1	0	0	53	339	1
5.6.1		46	1	0	0	343	1/1	0	0	343	1/1	0	0	57	343	1

Table 6: Response sizes and properties for E FreeBSD BIND 9.16.39, K CentOS BIND 9.16.39, K CentOS BIND 9.18.13 and X1 (conf)

Group III

This group consists primarily of NSD software without explicit EDNS message size settings. The responses in this group as shown in [Table 7](#) have the following properties:

- Addresses for the values of the NS records are added in the additional section with a preference for the IP version over which the query was sent, similar as in [Group I](#) and [Group II](#).
- Signatures for the addresses in the additional section are included if those addresses are authoritatively present in the root zone itself, but in those cases only complete address RRsets with signatures are included.
 - ⚠ This results in large responses (more than 8k) with “TCP DO” for schemes 5.3 and 5.3.1.
- When the RRSIG for the . NS RRset does not fit, the TC flag is set.
- All responses with the TC flag set have an empty answer and additional section.

		noedns				512 DO				4096 DO				TCP		#
		size	NS	A	AA AA tc	size	NS	A	AA AA tc	size	NS	A	AA AA tc	size	size	gs
5.1&	6	508	13	0	10	28	0	0	0 ✓	1097	13/1	13	13	811	1097	1
5.2	4	492	13	13	2											
5.3	6	504	13	0	10	28	0	0	0 ✓	1151	13/1	0	2/2	807	8555	27
	4	488	13	13	2					1127	13/1	2/2	0			
5.3.1	6	507	13	0	11	496	13/1	0	0	1126	13/1	0	2/2	782	8530	27
	4	491	13	13	3					1102	13/1	2/2	0			
5.4	6	504	13	0	10	28	0	0	0 ✓	1093	13/1	13	13	807	1093	1
	4	488	13	13	2											
5.5	6	488	13	0	8	28	0	0	0 ✓	1133	13/1	13	13	847	1133	1
	4	500	13	13	1											
5.5.1	6	487	13	0	8	28	0	0	0 ✓	1132	13/1	13	13	846	1132	1
	4	499	13	13	1											
5.6	6	406	1	0	13	339	1/1	0	0	990	1/1	0	13/1	625	1485	3
	4	250	1	13	0					834	1/1	13/1	0			
5.6.1	6	410	1	0	13	343	1/1	0	0	994	1/1	0	13/1	629	1489	3
	4	254	1	13	0					838	1/1	13/1	0			

Table 7: Response sizes and properties for D Linux NSD 4.6.1, H Linux NSD 4.5.0, K NSD 4.6.1, L Ubuntu NSD 4.6.0, X2 (conf) and X5 (conf)

Group IV

This group has a single software NSD 4.1.20 with EDNS UDP message size set explicitly to 1450. This results in larger responses for naming scheme 5.3 and 5.3.1. All other responses are equal to those in [Group III](#).

		noedns				512 DO				4096 DO				TCP		#
		size	NS	A	AA AA tc	size	NS	A	AA AA tc	size	NS	A	AA AA tc	size	size	gs
5.3	6	504	13	0	10	28	0	0	0 ✓	1151	13/1	0	2/2	807	8555	27
	4	488	13	13	2					1430	13/1	3/3	0			
5.3.1	6	507	13	0	11	496	13/1	0	0	1126	13/1	0	2/2	782	8530	27
	4	491	13	13	3					1405	13/1	3/3	0			

Table 8: Response sizes and properties for D Linux NSD 4.1.20

L Ubuntu NSD 4.6.0 (not in this group) has the EDNS UDP message size explicitly set to 4096. However that version of NSD adds additional addresses (after the first IPv4 and IPv6 address) only if the resulting response size stays below the size of the minimal response setting.

Group V

This group contains Knot-DNS and the confidential configuration X3.

- The TC flag is set when the signature for the NS RRset does not fit, but unlike [Group I](#), [II](#), [III](#) and [IV](#), the NS RRset is still included. Unlike [Group VIII](#), no additional addresses are included in truncated responses.
- This group, together with [Group VI](#), [VII](#) and [VIII](#), tries to add equal amounts of IPv4 and IPv6 additional addresses regardless of the IP version of the used transport.
- Additional addresses, which are authoritatively present in the root zone, are included, but signatures are left out when they do not fit the size restrictions for the response.

	noedns				512 DO				4096 DO				TCP		#
	size	NS	A	AA tc	size	NS	A	AA tc	size	NS	A	AA tc	size	size	
5.1 & 5.2	508	13	2	2	431	13	0	0 ✓	1217	13/1	12	11	1003	1289	1
5.3	500	13	3	3	379	13	0	0 ✓	1209	13/1	13	12	951	8699	27
5.3.1	507	13	7	7	512	13/1	1	0	1068	13/1	13	13	782	8530	27
5.4	500	13	3	3	379	13	0	0 ✓	1209	13/1	13	12	951	1237	1
5.5	500	13	6	5	275	13	0	0 ✓	1133	13/1	13	13	847	1133	1
5.5.1	511	13	3	2	418	13	0	0 ✓	1232	13/1	12	12	990	1276	1
5.6	250	1	13	0	339	1/1	0	0	1198	1/1	13/1	13	625	1485	3
5.6.1	254	1	13	0	343	1/1	0	0	1202	1/1	13/1	13	629	1489	3

Table 9: Response sizes and properties for K Knot 3.1.9, K Knot 3.2.6 and X3 (conf)

Group VI

This single software group has EDNS UDP message size configured to be 4096. This results in larger UDP responses when queried with EDNS UDP message size larger than 1232. Other than that, the properties and sizes of the responses in this group are equal to those of [Group V](#).

	noedns				512 DO				4096 DO				TCP		#
	size	NS	A	AA tc	size	NS	A	AA tc	size	NS	A	AA tc	size	size	
5.1 & 5.2	508	13	2	2	431	13	0	0 ✓	1289	13/1	13	13	1003	1289	1
5.3	500	13	3	3	379	13	0	0 ✓	3820	13/1	13/5	13/4	951	8699	27
5.3.1	507	13	7	7	512	13/1	1	0	3938	13/1	13/5	13/5	782	8530	27
5.4	500	13	3	3	379	13	0	0 ✓	1237	13/1	13	13	951	1237	1
5.5	500	13	6	5	275	13	0	0 ✓	1133	13/1	13	13	847	1133	1
5.5.1	511	13	3	2	418	13	0	0 ✓	1276	13/1	13	13	990	1276	1
5.6	250	1	13	0	339	1/1	0	0	1485	1/1	13/1	13/1	625	1485	3
5.6.1	254	1	13	0	343	1/1	0	0	1489	1/1	13/1	13/1	629	1489	3

Table 10: Response sizes and properties for L Ubuntu Knot 3.1.8

Group VII

This group consists of a single software: X7 (conf).

- This group is the only group in which the responses have the TC flag set when not all additional addresses fit in the additional section.

	noedns				512 DO				4096 DO				TCP		#
	size	NS	A	AA tc	size	NS	A	AA tc	size	NS	A	AA tc	size	size	
5.1& 5.2	508	13	7	6 ✓	239	13	0	0 ✓	1097	13/1	13	13	811	1097	1
5.3	504	13	7	6 ✓	521	13/1	0	0 ✓	1183	13/1	2/1	2/1 ✓	807	8555	27
5.3.1	507	13	7	7 ✓	496	13/1	0	0 ✓	1158	13/1	2/1	2/1 ✓	782	8530	27
5.4	504	13	7	6 ✓	235	13	0	0 ✓	1093	13/1	13	13	807	1093	1
5.5	500	13	6	5 ✓	275	13	0	0 ✓	1133	13/1	13	13	847	1133	1
5.5.1	499	13	6	5 ✓	274	13	0	0 ✓	1132	13/1	13	13	846	1132	1
5.6	250	1	13	0 ✓	339	1/1	0	0 ✓	1198	1/1	13/1	13 ✓	625	1485	3
5.6.1	254	1	13	0 ✓	343	1/1	0	0 ✓	1202	1/1	13/1	13 ✓	629	1489	3

Table 11: Response sizes and properties for X7 (conf)

Group VIII

This group consists of a single software: X8 (conf).

- The TC flag is set when the signature for the NS RRset does not fit, but like with [Group V](#), [VI](#) and [VII](#), the NS RRset itself remains. Unlike any other group, additional addresses are still included with truncated responses without NS signature.
- Unlike [Group II](#), [III](#), [IV](#) and [V](#), naming scheme 5.3.1 for 512 DO is truncated, even though without additional addresses there would have been space for the NS signature.

	noedns				512 DO				4096 DO				TCP		#
	size	NS	A	AA tc	size	NS	A	AA tc	size	NS	A	AA tc	size	size	
5.1& 5.2	508	13	7	6	503	13	6	6 ✓	1097	13/1	13	13	811	1097	1
5.3& 5.4	504	13	7	6	499	13	6	6 ✓	1093	13/1	13	13	807	1093	1
5.3.1	507	13	7	7	490	13	7	6 ✓	1068	13/1	13	13	782	1068	1
5.5	500	13	6	5	511	13	6	5 ✓	1133	13/1	13	13	847	1133	1
5.5.1	499	13	6	5	510	13	6	5 ✓	1132	13/1	13	13	846	1132	1
5.6	502	1	13	9	485	1	13	8 ✓	911	1/1	13	13	625	911	1
5.6.1	506	1	13	9	489	1	13	8 ✓	915	1/1	13	13	629	915	1

Table 12: Response sizes and properties for X8 (conf)

Priming responses properties

Property	Group							
	I	II	III	IV	V	VI	VII	VIII
Answers DNS Cookies	✓							
Truncated messages contain no RRsets	✓	✓	✓	✓				
Set the truncate flag when not all additional addresses fit							✓	
Prefer additional address records over NS signature								✓
Does <i>not</i> include <i>in-zone</i> additional addresses (<i>naming schemes 5.3, 5.3.1, 5.6 and 5.6.1</i>)	✓	✓						
Prefer additional address records from same IP	✓	✓	✓	✓				
Include signatures for additional (in-zone) address records			✓	✓	✓	✓	✓	
Include signatures for out-of-zone additional addresses ¹⁴								
Randomize RRs in NS RRset	✓	✓					✓	✓
Additional address types are interlaced (IPv6 with IPv4)					✓	✓	✓	✓

Table 13: Overview of properties of priming responses for all groups of authoritative name server software in use by the root servers.

Table 13 provides an overview of the properties of priming responses for all groups of authoritative name server software in use by the root servers. They differ with respect to:

- DNS Cookies : Only software in [Group I](#) returns a DNS Cookie on request.
- Truncated responses : All software sets the TC flag when the signature for the NS record does not fit.

Truncated responses from software in [Group I](#), [II](#), [III](#) and [IV](#) are empty and contain no RRsets in the answer, authority and additional section. The other groups ([V](#), [VI](#), [VII](#) and [VIII](#)) do at least have the NS RRset in the answer section.

Only responses from [Group VII](#) have the TC flag set when the additional section does not contain all additional addresses.

¹⁴ None of the current software includes signatures for out-of-zone additional addresses, but we did observe this property with BIND 9.10.3-P4 when [reproducing the RSSAC028 Appendix A results](#).

Only responses from [Group VIII](#) have additional addresses when a response is truncated due to a missing NS signature. With a choice, this group prefers adding additional addresses over the NS signature.

- **Additional addresses** : All groups add additional addresses for the names of the root servers in responses to a priming query, when the names for those addresses are out-of-zone; i.e. the root zone is not authoritative for the names. This is the case with naming schemes 5.1, 5.2, 5.4, 5.5 and 5.5.1.

Only responses from [Group I](#) and [II](#) do not in-zone include additional addresses when those addresses are authoritatively present within the root zone.

When not all additional addresses fit, responses from [Group I](#), [II](#), [III](#), [IV](#) prefer additional addresses for the same IP version as over which the priming query and response are conveyed.

Responses from Group [III](#), [IV](#), [V](#), [VI](#) and [VII](#) also have the signatures of the in-zone additional addresses (from naming schemes 5.3, 5.3.1, 5.6 and 5.6.1) included. This results in large responses (larger than 8,000 bytes) over TCP, and large and fragmented responses over UDP with [Group VI](#).

None of the current and future name server software includes signatures for out-of-zone addresses. During this study this *has* been perceived while reproducing RSSAC028 Appendix A results with BIND version 9.10.3.

- **The order of RRs and RRsets** : Group I, II, VII and VIII have the RRs in the NS RRset in randomized. Group V, VI, VII and VIII provide additional address types interlaced, the other provide first all the addresses for one type and then all the addresses for the other type.

Testbed results

In the last two sections, we grouped the root server { Letter, OS, Software } combinations by their response to priming queries and studied their properties and sizes. In this section, we show the impact of the naming schemes, root server behavior and resolver behavior on priming.

The first test case shows how resolvers would react if we would introduce the new naming schemes today. The root zone is served by their current software and the recursive resolvers are not configured with a hint file but use their built-in hint file based on the current naming scheme. The root servers in the root zones have alternative (new) IP addresses different from those in the built-in hint file, to be able to test whether or not priming succeeded. Priming is considered successful when a resolver learns and queries any of the alternative (new) IP addresses for the root servers. For each naming scheme and resolver, we carry out the tests as described in [Running the testbed](#).

[Figure 9](#) shows the results of the measurements. The color of each oval indicates if a resolver:

- failed resolving the check domains and failed priming (red),
- resolved the check domains successfully but failed priming (orange),
- failed resolving at least one of the check domains but succeeded priming (yellow),
- succeeded resolving the check domains and succeeded priming (green).

Within each oval, we indicate when the resolver initiated the priming query:

- '<', the resolver sends the priming query during start-up.
- '=', the resolver sends the priming query before it starts working on the test query.
- '>', the resolver sends the priming query just after it starts working on the test query

The numbers within the oval show the number of priming queries before the slash, and the number of direct address queries for the names in the priming response after the slash

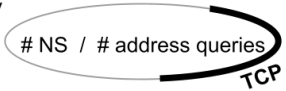
Finally, a bold border of a circle indicates that queries were sent via TCP.

The color of the oval and the border indication TCP are used in all the tables presented in [Testbed results](#). The metric given by the text within the oval can be different, but is always listed in the caption with the Figure showing the table.

[Figure 9](#) shows that priming and test queries succeed in the majority of cases. However, a few resolver-naming-schema combinations stand out. Also, some resolvers prime and resolve successfully, but not with all root server authoritative software. In the following sections, we analyze these cases further.

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
knot-resolver-5.5.3	< 1	< 2	< 1/41	< 1/34	< 1	< 1/3	< 1/40	< 1/3	< 1/5
knot-resolver-5.6.0	< 1	< 1	< 1/42	< 1/34	< 1	< 2	< 1/38	< 1/3	< 1/2
pdns-recursor-4.0.9	< 3	< 3	< 7	< 3	< 3	< 3	< 3	< 3	< 3
pdns-recursor-4.1.15	< 4	< 4	< 8	< 10	< 4	< 4	< 4	< 9	< 7
pdns-recursor-4.2.1	< 4	< 4	< 11	< 8	< 4	< 4	< 4	< 6	< 8
pdns-recursor-4.7.5	< 1	< 1	< 3	< 1	< 1	< 1	< 1	< 1	< 1
pdns-recursor-4.8.4	< 1	< 1	< 2	< 1	< 1	< 1	< 1	< 2	< 2
unbound-1.5.10	= 1	= 1	= 1	= 1	= 1	= 1	= 1	= 1/3	= 1/3
unbound-1.6.8	= 1	= 1	= 2/6	= 1	= 1	= 1	= 1	= 1/2	= 1/2
unbound-1.7.3	= 1	= 1	= 1/13	= 1/7	= 1	= 1	= 1	= 1/3	= 1
unbound-1.8.3	= 1	= 1	= 1	= 1	= 1	= 1	= 1	= 1/1	= 1
unbound-1.9.6	= 1	= 1	= 1/11	= 1/7	= 1	= 1	= 1	= 1/3	= 1/3
unbound-1.13.0	= 1	= 1	= 1/14	= 1/8	= 1	= 1	= 1	= 1	= 1/3
unbound-1.14.0	= 1	= 1	= 1	= 1	= 1	= 1	= 1	= 1	= 1
unbound-1.17.1	= 1	= 1	= 1/9	= 1	= 1	= 1	= 1	= 1	= 1/3
bind-9.9.11	> 2	> 2	> 1/26	> 1/26	> 1/26	> 1/26	> 1/26	> 1/2	> 2/2
bind-9.10.8	> 2	> 2	> 2/26	> 2	> 2	> 2	> 2	> 2/4	> 1/4
bind-9.11.6	> 2	> 2	> 2/26	> 1	> 2	> 2	> 2	> 1/4	> 1/4
bind-9.12.4	> 2	> 2	> 2/26	> 2/26	> 2	> 2	> 2	> 1/4	> 1/4
bind-9.13.7	> 2	> 2	> 2/26	> 1/26	> 2	> 2	> 2	> 1/4	> 2
bind-9.14.10	> 2	> 2	> 2	> 1/26	> 2	> 2	> 2	> 1/4	> 1/4
bind-9.15.8	> 2	> 2	> 2/26	> 1/26	> 2	> 2	> 2	> 1/2	> 1/2
bind-9.16.41	> 2	> 2	> 2	> 2/10	> 2	> 2	> 2	> 1/4	> 1/2
bind-9.18.15	> 2	> 1	> 1/10	> 1	> 1	> 1	> 1	> 1	> 1/2
bind-9.19.13	> 1	> 1	> 1/10	> 1	> 2	> 2	> 1	> 1/2	> 1

< Priming starts immediately after resolver starts
 = Priming starts just before work on test query
 > Priming starts just after work on test query



● Priming succeeded
 ● Test queries succeeded
 ● Priming succeeded
 ● A test query failed

● Priming failed
 ● Test queries succeeded
 ● Priming failed
 ● Both test queries failed

Figure 9: Naming scheme by Resolver table for the current root setup and current hints
 Metric: priming timing, the number of priming queries / the number of address queries

Naming Scheme 5.4 and Knot Resolver

Figure 9 shows Knot Resolvers unable to prime and resolve the check domain names in naming schema 5.4. Figure 10 shows that Knot Resolver fails this naming scheme with all root server authoritative software. The number shown in the oval is the total number of NS queries to the root-servers domain. This is only applicable to naming schemes in which the root servers have their own domain (5.1, 5.2, 5.4, 5.5 and 5.5.1). Knot Resolver is the only resolver in the testbed that sends NS queries to the root servers domain. No other resolvers do this.

When Knot Resolver fails, it is stuck repeating sending NS queries for 'root-servers.'. The message 'cannot resolve address [a-m].root-servers.' is logged despite the fact that we cannot identify such a query in our traffic traces.

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
C Alma Bind 9.16.39	3	3			104	22	4		
C CentOS Bind 9.16.39	3	3			129	19	3		
D Linux NSD 4.1.20	3	3			104	24	3		
D Linux NSD 4.6.1	3	3			128	22	3		
E FreeBSD Bind 9.16.39	3	3			129	21	4		
F FreeBSD Bind 9.16.39	2	3			128	23	3		
F FreeBSD' Bind 9.16.39	3	3			129	20	4		
G Linux Bind 9.16.29	3	3			130	25	4		
H Linux NSD 4.5.0	3	3			130	23	4		
K CentOS Bind 9.16.39	3	3			129	22	3		
K CentOS Bind 9.18.13	3	3			114	21	3		
K CentOS Knot 3.1.9	3	2			130	21	4		
K CentOS Knot 3.2.6	3	3			128	24	3		
K CentOS NSD 4.6.1	3	3			129	22	3		
L Ubuntu Knot 3.1.8	3	3			129	21	3		
L Ubuntu NSD 4.6.0	3	3			130	22	3		
X1 (conf)	3	3			128	23	3		
X2 (conf)	3	3			129	17	4		
X3 (conf)	3	3			129	20	3		
X4 (conf)	3	3			130	22	3		
X5 (conf)	3	3			129	21	5		
X6 (conf)	3	3			127	20	3		
X7 (conf)	3	2			129	76	36		
X8 (conf)	3	2			129	3	19		
X9 (conf)	3	4			129	20	4		

Figure 10: Naming scheme by { Letter, OS, Software } table for Knot Resolver.
Metric: the number of NS queries for the root servers domain.

Missing additional addresses of Group I and II name servers

	I & II	III, IV V & VI	VII	VIII
bind-9.10.8 ... 9.15.8	= 26			
bind-9.16.41	< 16			
bind-9.18.15 ... 9.19.13	< 16			
unbound-1.5.10 ... 1.17.1				
bind-9.9.11	= 26	= 26	= 26	= 26
knot-resolver-5.5.3 ... 5.6.0	> 26	> 26	> 26	> 26
pdns-recursor-4.0.9 ... 4.8.4				

Figure 11: { Letter, OS, Software } by Resolver table for Naming scheme 5.3.

Metric: The number of address queries for root servers, where:

"< 16" is less than 16, "= 26" is exactly 26 and "> 26" is more than 26

Figure 9 indicates larger numbers of address queries for the root servers for naming schemes 5.3, 5.3.1, 5.6 and 5.6.1: The naming schemes where the root server addresses are present authoritatively in the root zone. As discussed in [Priming responses from all root server software, Group I and II](#) (consisting primarily of BIND name servers) do not contain additional addresses in priming responses for those naming schemes.

Figure 11 shows the number of address queries for root servers in a { Letter, OS, Software } by Resolver table. Both rows and columns are collated. By grouping the number of address queries into classes ("< 16" is less than 16, "= 26" is exactly 26 and "> 26" is more than 26), the columns collate into the "Groups" identified in [Priming responses from all root server software](#).

Figure 11 confirms that the larger number of address queries seen in Figure 9 are indeed caused by name server software from [Group I](#) and [II](#), and all resolvers except for PowerDNS Recursor, follow up with queries for root server addresses when served by [Group I](#) or [II](#). PowerDNS Recursor, however, never sends follow-up address queries and as a result fails to prime and resolve the test queries for naming schemes 5.3, 5.3.1, 5.6 and 5.6.1 with authoritative name server software from [Group I](#) and [II](#).

It is furthermore noteworthy that BIND 9.9.11 and Knot Resolver perform follow-up address queries for the root servers. In our traces we noticed this happens with all naming schemes for BIND 9.9.1 and for all naming schemes except for 5.1 and 5.2 with Knot Resolver.

Failed priming

Figure 9 shows that priming failed for naming schemes 5.1 and 5.2 for PowerDNS Recursor 4.1.15 and 4.2.1, and BIND 9.9.11. Figure 12 and Figure 13 show the Naming scheme by { Letter, OS, Software } tables for these resolvers. From these we can read this is the case with all root server software. Figure 12 also shows that PowerDNS Recursor fails resolving all together with naming schemes 5.3, 5.3.1, 5.6 and 5.6.1 as was discussed in the previous Section [Missing additional addresses of Group I and II name servers](#).

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
I, II	4	4	16	16	4	4	4	16	16
III, IV, V, VI	4	4	4	4	4	4	4	4	4
VII	4	4	8	8	4	4	4	8	8
VIII	4	4	4	4	4	4	4	4	4

Figure 12: Naming scheme by { Letter, OS, Software } table for PowerDNS Recursor 4.2.1
 Metric: The number of priming queries
 (The same table for PowerDNS Recursor 4.1.15 has similar results)

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
I, II, III, IV, V, VI	2	2	1/26	1/26	1/26	1/26	1/26	1/2	1/2
VII	2	2	2/26	2/26	1/26	1/26	1	2/2	2/2
VIII	2	2	1/26	1/26	1/26	1	1	1/2	1/2

Figure 13: Naming scheme by { Letter, OS, Software } table for BIND 9.9.11
 Metric: The number of priming queries / number of root server address queries

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
bind-9.10.8 ... 9.16.41	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
bind-9.18.15 ... 9.19.13	1	1	1	1	1	1	1	1	1
unbound-1.5.10 ... 1.17.1									
bind-9.9.11	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
pdns-recursor-4.1.15 ... 4.2.1	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
knot-resolver-5.5.3	1	1	1/42	1/34	1	1	1/40	1/3	1/2
knot-resolver-5.6.0	1	< 5	1/40	1/34	1	1	1/40	1/3	1/3
pdns-recursor-4.0.9	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
pdns-recursor-4.7.5 ... 4.8.4	1	1	1	1	1	1	1	1	1

Figure 14: Naming scheme by Resolver table for X8 (conf)
 Where: Resolvers are configured with the root hints for the tested naming scheme
 Metric: The number of priming queries classes / number of root server address queries
 Classes: "1" is 1, "< 5" is more than 1 but less than 5

To further investigate these cases, we scheduled measurements in which the resolvers are configured with root hint files for the tested naming scheme. We created “Naming scheme by Resolver”-tables for all { Letter, OS, Software } combinations. [Figure 14](#) shows this table for the X8 (conf) root server software. We can see that, when explicitly configured with a hints file, the BIND 9.9.11 and PowerDNS Recursor 4.1.15 and 4.2.1 fail priming. Furthermore, newer versions of PowerDNS Recursor (4.7.5 and 4.8.4) now also fail priming when *explicitly* configured with root hints for the naming scheme.

[Figure 15](#) shows the same table for C Centos bind-9.16.39. In this table we can see that all BIND resolvers, as well as PowerDNS Recursor 4.0.9 now also fail priming for the naming schemes in which the root server addresses are authoritatively present in the root zone (5.3, 5.3.1, 5.6 and 5.6.1). We see similar results from other { Letter, OS, Software } combinations from [Group I](#) and [II](#) that do not include the additional addresses for these naming schemes in priming responses.

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
bind-9.10.8	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
bind-9.11.6 ... 9.12.4	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
bind-9.13.7 ... 9.16.41	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
bind-9.18.15 ... 9.19.13	1	1	< 5	< 5	1	1	1	< 5	< 5
bind-9.9.11	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
pdns-recursor-4.1.15 ... 4.2.1	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
knot-resolver-5.5.3	1	1	1/43	< 5/34	1	1/2	1/40	< 5/3	1/3
knot-resolver-5.6.0	1	1	1/38	1/35	1	1/2	1/40	1/2	1/3
pdns-recursor-4.0.9	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
pdns-recursor-4.7.5 ... 4.8.4	1	1	1	1	1	1	1	1	1
unbound-1.13.0	1	1	1/11	1/9	1	1	1	1/3	1/3
unbound-1.14.0	1	1	1/11	1/8	1	1	1	1/3	1/3
unbound-1.17.1	1	1	1/14	1/8	1	1	1	1/3	1/3
unbound-1.5.10 ... 1.6.8	1	1	1/6	1/6	1	1	1	1/2	1/2
unbound-1.7.3	1	1	1/10	1/7	1	1	1	1/3	1/3
unbound-1.8.3	1	1	1/12	1/8	1	1	1	1/3	1/3
unbound-1.9.6	1	1	1/13	1/9	1	1	1	1/3	1/3

Figure 15: Naming scheme by Resolver table for C Centos bind-9.16.39 (conf)

*Where: Resolvers are configured with the root hints for the tested naming scheme
Metric: The number of priming queries classes / number of root server address queries
Classes: “1” is 1, “< 5” is more than 1 but less than 5*

Influence on UDP/TCP traffic

As shown in [Figure 9](#) and as expected, different naming schemes and different resolver behavior can lead to truncated responses and priming queries via TCP.

Avoiding TCP with naming scheme 5.3.1

As shown in [Table 4](#) in [Priming responses from all root server software](#), BIND versions 9.10.8 ... 9.16.41 send a priming query with an EDNS(0) buffer size set to 512 bytes. In general, this leads to truncated responses and results in a TCP retry. Here, naming scheme 5.3.1 is the exception; the priming query can fit into 512 byte UDP messages when sent to { Letter, OS, Software } combinations that do not return DNS cookies. [Figure 16](#) shows priming results for BIND 9.10.8 up to 9.16.41 with naming scheme 5.3.1 per { Letter, OS, Software } combinations.

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
I	2	2	2/26	2/26	2	2	2	1/4	1/4
II	2	2	2/26	1/26	2	2	2	1/4	1/4
III, IV	2	2	2	1/26	2	2	2	1/4	1/4
V, VI	2	2	2	1	2	2	2	1/4	1/4
VII	2	2	2	2	2	1	1	2	2
VIII	2	2	2	2	2	1	1	2	2

Figure 16: Naming scheme by { Letter, OS, Software } table for BIND 9.10.8

Metric: The number of priming queries / number of root server address queries

An oval with a black outline indicates priming queries over TCP

BIND 9.11.6 .. 9.16.41 have similar tables.

Increase in TCP traffic

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
bind-9.10.8	1097	1097	1448	8530	1093	1133	835	1485	1489
bind-9.11.6 ... 9.16.41	1097	1097	8555	8516	1093	1133	1132	1485	1489
bind-9.18.15 ... 9.19.13			8555	8516				1485	1489
bind-9.9.11			8555	8530			1132	1485	1489
knot-resolver-5.5.3	130		8555	8530			918	1485	1489
knot-resolver-5.6.0	219		8555	8530			918	1485	1489
pdns-recursor-4.0.9 ... 4.8.4			8555	8530				1485	1489
unbound-1.5.10 ... 1.17.1			8555	8530				1485	1489
pdns-recursor-4.1.15 ... 4.2.1			8555	8530				1485	1489

Figure 17: Naming scheme by Resolver table for [Group VII](#) - X7 (conf)

Metric: The total number of bytes transferred over TCP.

The largest increase in TCP traffic, both in number of TCP sessions and in size, can be observed between resolvers and root server configuration [X7 \(conf\)](#), since X7 always sets the TC flag when not all additional addresses fit (See [Figure 17](#)).

Still high in the number of TCP sessions, but causing the smallest amount of TCP traffic from all the root server software that still work with PowerDNS Recursor (i.e. all software except from [Group I](#) and [II](#) as discussed in [Missing additional addresses of Group I and II name servers](#) and Section [Failed priming](#)), is root server configuration [X8 \(conf\)](#). This can be explained by the fact that this server includes additional addresses, but not the signatures for them with the naming schemes where those addresses are authoritatively present in the root zone (5.3, 5.3.1, 5.6 and 5.6.1).

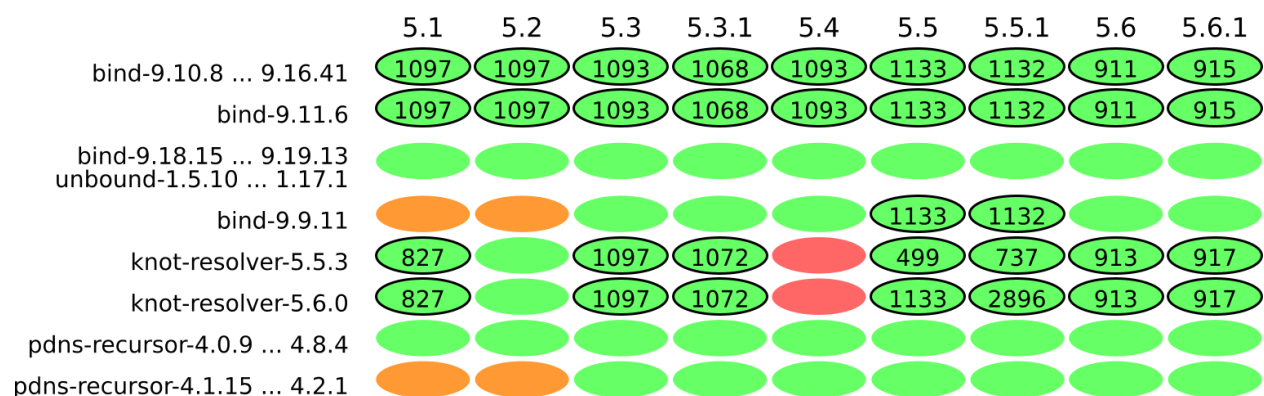


Figure 18: Naming scheme by Resolver table for [Group VIII](#) - X8 (conf)
Metric: The total number of bytes transferred over TCP.

Name server software from Group III ... VI, which consist of primarily NSD and Knot DNS, cause the least TCP sessions, and are sizewise in between X7 (Largest) and X8 (Smallest)

Failure scenarios

Aside from test cases in which a resolver *should* be able to prime successfully, we also tested cases in which resolver operations were impaired in some way.

Unable to validate

Resolvers that would be unable to validate DNSSEC, for example because of a clock which is out of sync, behave differently when priming. [Figure 19](#) shows priming results for root server setup [X8 \(conf\)](#) and resolvers with a clock set to the year 2000.

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
bind-9.10.8 ... 9.12.4	1	1	1	1	1	1	1	1	1
bind-9.13.7 ... 9.16.41	1	1	1	1	1	1	1	1	1
bind-9.18.15 ... 9.19.13	1	1	1	1	1	1	1	1	1
unbound-1.5.10 ... 1.17.1	1	1	1	1	1	1	1	1	1
bind-9.9.11	1	1	1/26	1/26	1/26	1/26	1/26	1/2	1/2
knot-resolver-5.5.3 ... 5.6.0	1	1	1	1	1	1	1	1	1
pdns-recursor-4.0.9	3	3	3	3	3	3	3	3	3
pdns-recursor-4.1.15 ... 4.2.1	4	4	4	4	4	4	4	4	4
pdns-recursor-4.7.5 ... 4.8.4	1	1	1	1	1	1	1	1	1

Figure 19: Naming scheme by Resolver table for [Group VIII](#) - X8 (conf)

Where: The resolvers clock is set to outside the DNSSEC validity period

Metric: The number of priming queries / number of root server address queries

Most resolvers prime successfully, but some fail to resolve the test domain name. Priming of Knot Resolver always fails and also PowerDNS Recursor versions 4.7.5 ... 4.8.4 fail priming with all naming schemes except naming schemes 5.1 and 5.2. This may indicate that those versions of PowerDNS Recursor and Knot Resolver validate priming responses. We assume that we would see similar results with other causes for DNSSEC failure.

Unable to send queries via TCP

Resolvers that are unable to send queries over TCP fail to prime in some cases. [Figure 20](#) shows that for root server setup [X8 \(conf\)](#), priming fails with BIND versions 9.10.8 ... 9.16.41. These are the versions that start priming with EDNS UDP Message size 512 (See [Table 4](#)).

In case resolvers without TCP support query the root servers of setup [X7 \(conf\)](#), resolvers start failing to resolve with some or all naming schemes (see [Figure 21](#)). This is due to the fact that this setup returns the TC flag in case of missing additional addresses, and as a result returns responses with the TC flag set more frequently than the other root server authoritative software.

If Unbound is unable to get priming responses, it will also not resolve the following queries and the same applies to some versions of BIND (at least 9.13.7 ... 9.14.10). Since X7 (conf) does include additional addresses in truncated responses, this suggests that Unbound and those versions of BIND discard the content of truncated responses all together and rely in those cases on the response returned over TCP.

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
bind-9.10.8	< 100	< 100	< 20	< 20	< 20	< 20	< 100	< 20	< 100
bind-9.11.6 ... 9.16.41	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20
bind-9.13.7	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20
bind-9.18.15 ... 9.19.13	1	1	1	1	1	1	1	1	1
pdns-recursor-4.7.5 ... 4.8.4	1	1	1	1	1	1	1	1	1
unbound-1.5.10 ... 1.17.1	1	1	1	1	1	1	1	1	1
bind-9.9.11	< 5	< 5	1/26	1/26	1/26	1/26	1/26	1/2	1/2
knot-resolver-5.5.3	1	1	1/78	1/57	1	< 5	< 5/77	1/6	1/8
knot-resolver-5.6.0	1	1	1/56	1/52	1	1	1/58	1/5	1/10
pdns-recursor-4.0.9	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5
pdns-recursor-4.1.15 ... 4.2.1	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5	< 5

Figure 20: Naming scheme by Resolver table for [Group VIII](#) - X8 (conf)

Where: The resolvers cannot reach the authoritative over TCP

Metric: The number of priming queries classes / number of root server address queries

Classes: "1" is 1, "< 5" is more than 1 but less than 5,

"< 20" is 5 or more than 5 but less than 20,

"< 100" is 20 or more than 20 but less than 100

	5.1	5.2	5.3	5.3.1	5.4	5.5	5.5.1	5.6	5.6.1
bind-9.10.8	< 100	< 100	< 100	< 100	< 100	< 100	< 100	< 100	< 100
bind-9.11.6 ... 9.12.4	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20
bind-9.13.7 ... 9.14.10	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20
bind-9.15.8	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20
bind-9.16.41	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20	< 20
bind-9.18.15 ... 9.19.13	1	1	< 20	< 20	1	1	1	< 20	< 20
bind-9.9.11	< 5	< 5	< 20	< 20	1/26	1/26	1/26	< 20	< 20
knot-resolver-5.5.3	1	1	< 5	< 5	1	1	1/42	< 5	< 20
knot-resolver-5.6.0	1	1	< 5	< 5	1	1	1/37	< 5	< 20
pdns-recursor-4.0.9	< 5	< 5	< 100	< 100	< 5	< 5	< 5	< 100	< 100
pdns-recursor-4.1.15 ... 4.2.1	< 5	< 5	> 100	> 100	< 5	< 5	< 5	> 100	> 100
pdns-recursor-4.7.5	1	1	< 20	< 20	1	1	1	< 20	< 20
pdns-recursor-4.8.4	1	1	< 20	< 5	1	1	1	< 5	< 20
unbound-1.7.3 ... 1.17.1	1	1	< 100	< 100	1	1	1	< 100	< 100
unbound-1.5.10 ... 1.6.8	1	1	< 5	< 5	1	1	1	< 5	< 20

Figure 21: Naming scheme by Resolver table for [Group VII](#) - X7 (conf)

Where: The resolvers cannot reach the authoritative over TCP

Metric: The number of priming queries classes / number of root server address queries

Classes: "1" is 1, "< 5" is more than 1 but less than 5,

"< 20" is 5 or more than 5 but less than 20,

"< 100" is 20 or more than 20 but less than 100

Properties of resolvers

This table summarizes the general behavior of resolvers derived from our tests.

Property	Resolvers	Observed behavior
Moment of priming	knot-resolver-*	After startup
	pdns-recursor-*	
	unbound-*	Before it starts working on a query
	bind-*	Just after it starts working on a query
Address queries	pdns-recursor-*	Never query for root server addresses
	unbound-*	Query root server addresses when there were none in the priming response
	bind-9.10.8 ... 9.19.13	
	bind-9.9.11 & knot-resolver-*	Always query for root server addresses
DNSSEC validation	pdns-recursor-4.7.5 ... 4.8.4 knot-resolver-5.5.3 ... 5.6.0	Does not resolve with a clock skewed outside of DNSSEC validity period
Truncated responses ¹⁵	unbound-*	The content of truncated responses is ignored. The content is expected to be acquired in the follow-up query over TCP
	bind-9.13.7 ... 9.14.10	

¹⁵ This property is deduced from the results seen in [Unable to send queries via TCP](#)

Impact of search lists on different naming schemes

According to RSSAC028, all naming schemes that introduce a new TLD or a new name in the root zone increase the potential of name collisions with existing resolver search lists. This concerns naming schemes 5.3, 5.4, 5.5 and 5.6.

SAC53¹⁶ and SAC64¹⁷ discuss the impact of search lists and the introduction of new TLDs. SAC64 Recommendation 1.b states *“When a user enters a single label name, that name may be subject to search list processing if a search list is specified, but must never be queried in the DNS in its original single-label form.”* and 1.c states *“When a user queries a hostname that contain two or more labels separated by dots, such as www.server, applications and resolvers must query the DNS directly. Search lists must not be applied even if such names do not resolve to an address (A/AAAA). Therefore www.server is always a FQDN.”*

Applying these recommendations to the proposed naming schemes, this would mean that single label names like “a” and “b” could be subject to search list processing. Additionally, recommendation 1.b even goes further and recommends not to query single labels at all.

For this report we tested two commonly used stub resolvers: systemd-resolved¹⁸ and dnsmasq¹⁹. Both support DNSSEC validation. Dnsmasq does not support the configuration of search lists for when resolving but can only instruct DHCP clients to use search lists. Systemd-resolved does support search lists and we tested its effect on scenario 5.3.1. Here, we added a domain name under our control (e.g. example.) to the search lists with existing subdomain-labels [a-m] (e.g. a.example.) and an A.

When querying for the A record of root server letter (e.g. “a”), resolved responded with the record of our test domain. When querying for the corresponding DNSKEY record, resolving failed with return code SERVFAIL. However, since stub resolvers do not perform priming queries, we do not expect this to impact regular operations.

¹⁶ ICANN Security and Stability Advisory Committee (SSAC), “SSAC Report on Dotless Domains”, 2012. <https://www.icann.org/en/system/files/files/sac-053-en.pdf>

¹⁷ ICANN Security and Stability Advisory Committee (SSAC), “SAC064 SSAC Advisory on DNS ‘Search List’ Processing”, 2014. <https://www.icann.org/en/system/files/files/sac-064-en.pdf>

¹⁸ <https://github.com/systemd/systemd>

¹⁹ <https://dnsmasq.org/>

Discussion and Conclusion

In the RSSAC028 report, alternative naming schemes for the root servers were considered, and evaluated with respect to the effect and risks they would induce on the root server system. The primary recommendation was to make no changes to the current naming scheme until more studies have been conducted. One of the recommended follow-up studies was into understanding the current behavior of DNS resolvers and how each naming scheme would affect these behaviors. This document is a report on such a follow-up resolver behavior study.

RSSAC028 further broke down the follow-up study recommendation into a list of topics relevant for further research. Those topics are enumerated below referencing the parts of the report in which the topic is addressed.

- **The acceptable response size for priming queries**

We carried out a literature study to identify the maximum supported packet size on the Internet in [Acceptable response sizes](#).

Since the RSSAC028 report, the DNS community discussed packet size support extensively which eventually led to DNS Flag Day 2020 with the recommendations for maximum safe message sizes. The implementation of those recommendations can be seen in the current software landscape, both in the size of priming responses (see [Reproducing RSSAC028 Appendix A results](#)) as well as EDNS UDP Message size parameters of resolver software (see [Most prevalent query parameters](#)).

The limitation on the response size impacts other properties of priming responses, such as the number of address records for the root servers in the additional section, whether or not DNSSEC signatures for additional addresses are included and whether or not the TC (truncate) flag is set in the response. These properties impact resolver behavior and thereby the performance of the root server system. These properties vary per naming scheme and per authoritative name server software.

To inventory the authoritative name server software in use on the root server system, we conducted a survey of the root server operators (see [Survey of Root Server Operators](#)). We received information on all current and future root server software (see [Survey results](#)). Two proprietary authoritative name server software were reported for which we arranged priming responses for the naming schemes. With these we created simulations of the authoritative name server software based on Idns-testns datafiles (see [Simulating the proprietary authoritative name server software](#)).

In [Priming responses from all root server software](#) we provide a detailed overview of the priming responses for all the current and future versions of the root server software with all the naming schemes. We identify eight groups of root server software that have equal priming responses for those naming schemes.

- **How resolver software responds to a reduced set of glue records**

We created a [Resolver testbed](#), consisting of a simulation of the root server system, where each root server is running on the in the survey reported open source software, on the reported operating system, with the reported network parameters, and the proprietary software is simulated with `ldns-testns`. The testbed contains various versions of open source resolver software and provides the means to evaluate priming behavior and resolver behavior in general, for simulations of the current and future root server system - as well as for other root server software configurations - equipped with the naming schemes to be evaluated.

Many of the priming responses identified in the eight groups of priming responses in [Priming responses from all root server software](#), already provided an incomplete, or even completely absent set of addresses for the root servers in the additional section. The impact of these properties is evaluated by studying resolver behavior with simulations of the root server system consisting entirely from software from the group with those properties.

In [Missing additional addresses of Group I and II name servers](#) we see that absent address records for the root servers in the additional section will cause follow-up queries for those addresses from all resolvers except for PowerDNS Recursor, which is then no longer able to resolve.

Absent additional addresses furthermore causes BIND and PowerDNS Recursor resolvers to fail learning new IP addresses for the root servers when the NS RRset for the root is equal to the one in the built-in or provided by a root hints file (see [Figure 15](#) in [Failed priming](#)). Note that some versions of PowerDNS Recursor and BIND 9.9.11 also fail to learn new IP addresses for the root servers even with additional addresses which is discussed in the same section.

Related to the question of absent additional addresses, is how the *presence* of additional addresses and the NS RR in the answer section in truncated responses affects resolvers. This is studied in [Unable to send queries via TCP](#). From [Figure 20](#) and [Figure 21](#) in that section we can deduce that Unbound disregards the content of truncated responses altogether.

- **How resolver implementations handle a bogus response to priming queries**

We address the impact on resolvers of a clock skewed to outside of the DNSSEC validity period of the root and root servers zones in [Unable to validate](#). PowerDNS Recursor 4.7.5 and 4.8.4 and Knot Resolver 5.5.3 and 5.6.0 stop working altogether in those cases. This may indicate that those versions of PowerDNS Recursor and Knot Resolver validate priming responses. We assume that we would see similar results with other causes for DNSSEC failure.

- **How search lists might be relevant**

This is addressed in [Impact of search lists on different naming schemes](#). Systemd-resolved was evaluated with naming scheme 5.3.1 and single letter label name in a test domain which was in the search list. The search list did not have an impact on resolving and DNSSEC in general and for the single letter label name in the test domain.

Observation per naming scheme

In the below sections we put forward some noteworthy observations and potential and actual issues we have seen with resolver behavior per naming scheme. We note that problematic resolver behavior in relation to specific properties of priming responses can be addressed by “fixing” the authoritative name server software to be deployed on the root system, but that problematic resolver behavior for naming schemes regardless of the properties of the priming responses would be impossible to address with “fixes” to resolver software on the short-term, because they are out of our or anyone’s reach.

- **5.1 and 5.2, Current and current + DNSSEC**

BIND 9.9.1 and PowerDNS Recursor version 4.1.15 and 4.2.1 fail to learn new (alternative) IP addresses for root servers in the additional section of priming responses, when the root NS RRset matches that of the built-in or loaded hints file (see [Failed priming](#)).

With a loaded hints file (not relying on the built-in root hints), also PowerDNS versions 4.7.5 and 4.8.4 do not learn new IP addresses from the additional section in priming responses.

- **5.3 and 5.3.1, In-zone NS names**

[Group I](#) and [II](#) authoritative name servers (primarily BIND) do not include the addresses for the root servers in the additional section.

All tested versions of PowerDNS Recursor fail to prime and resolve, when this naming scheme is served by software from [group I](#) or [II](#) (see [Missing additional addresses of Group I and II name servers](#)). This is also the case for the other naming schemes that have the addresses for the root server authoritatively present in the root zone (5.3.1, 5.6 and 5.6.1).

Other resolvers generate more queries for priming because of the follow-up address queries for the root servers.

Software from [group III](#), [IV](#), [V](#), [VI](#) and [VII](#) include DNSSEC signatures for the address records which leads to a limited number of addresses for root servers in the additional section as well as very large responses (more than 8500 bytes) over TCP.

Software from [group VII](#) furthermore causes many TCP sessions from resolvers, by setting the TC flag on responses that do not have all IP addresses for the root servers in the additional section (see [Increase in TCP traffic](#)).

- **5.4, Shared delegated TLD**

The tested versions of Knot Resolver fail to prime and resolve with this naming scheme regardless of how this naming scheme is served.

- **5.5 and 5.5.1, Names delegated to each operator**

We have not observed any notable deviating or problematic behavior with this naming scheme with any of the tested resolvers.

- **5.6 and 5.6.1, Single shared label for all Operators**

As with 5.3 and 5.3.1, authoritative software from [group I](#) and [II](#) do not include addresses for the root servers in the additional section, which causes PowerDNS Recursor to fail to prime and to resolve.

As with 5.3 and 5.3.1, authoritative software from [group III](#), [IV](#), [V](#), [VI](#) and [VII](#) include DNSSEC signatures for the address records, however because naming schemes 5.6 and 5.6.1 only need to include a single DNSSEC signature for all IPv4 and a single DNSSEC signature for all IPv6 addresses instead of 13 for each, the resulting responses are much smaller. As a consequence more additional addresses are included in the responses over UDP and the TCP responses are also much smaller, resulting in fewer TCP sessions and less TCP traffic.

The complete priming response with all additional addresses and signatures is still larger than 1232 bytes (DNS Flag-day recommended UDP size) though, resulting in responses with the TC bit set from authoritative server software from [group VII](#) (X7 (conf)).

Acknowledgement

Many thanks to Jennifer Bryce, Matt Larson, Paul Hoffman for their patience, support and positive feedback, to the root operators for helping us out with all the information we needed to do this research, to the providers of the proprietary software for performing priming responses on their software on our behalf, and another thank you to Paul Hoffman for the design and work on the original resolver testbed from which we developed the testbed used in this research.

Appendix A: RSO Survey

Email, sent to RSO's.

Survey of RSO Authoritative Server Software

Introduction

Our consortium of NLnet Labs and SIDN is carrying out a study on behalf of ICANN on the naming scheme used for the root servers (the RSSAC028 Implementation study). This is a follow up study of the Advisory from the ICANN Root Server System Advisory Committee (RSSAC) RSSAC028.

The goal of our study is to do an impact analysis of the different alternative naming schemes proposed in section 5 of RSSAC028. Part of this, is an extensive survey of software and configurations used by Root Server Operators (RSOs). The different naming schemes proposed in RSSAC028 are tested in combination with the used configurations in a testbed.

An announcement of this work has been presented at the RSO meeting that was held the 24th of July alongside the IETF114 meeting in Philadelphia .

Survey

In order to test the naming schemes in a realistic testbed, we kindly request the RSOs to provide us with some information, like which software and version is used to serve the root zone and how it is configured. This information is only used for performing the study and we plan to include the result and the configuration in the final report, which will be shared with ICANN, the RSSAC Caucus, and the general community. For each answer, please indicate under which conditions you are willing to provide this information. Note that we will not share anything without your explicit consent.

Could you please answer the questions in the questions section below by replying to this email and providing the answers inline in the reply. Any additional files can be attached to the reply email. The reply may be PGP encrypted with the PGP key of any of this study's team members:

Redacted

For convenience, I (Willem) have attached my PGP key to the email.

We would appreciate it if you could respond within two weeks (before Monday the 26th of September) so that we can move to the analysis part of this work in a timely manner.

Questions

Question 1: What authoritative name server software are you using to serve the root zone or are you planning to use in the near future? Which version of the software do you use and on which platform? If different software is used in parallel or as a backup, please include them as well.

Question 2: Is the name server software compiled with custom compile-time options and/or configuration for compilation? If so, are you able and willing to provide those options and/or configuration?

Question 3: If the software used is open source, would you be willing to provide us with the configuration files? If yes, please attach the configuration files to this response.

Question 4: If the software is proprietary, would you willing to share it with us or provide us with an installation we can use to perform our tests?

Question 5: Do you have public facing load balancers or other infrastructure you consider important for this study?

Question 6: Are there any other parameters of your name server setup relevant to this study? For example, do your name servers support DNS cookies? Which path MTU(s) do you have configured? Have you enabled minimal responses or similar functions?

Many thanks for your feedback and cooperation,

Willem Toorop on behalf of the consortium

Appendix B: Reproducing RSSAC028 Appendix A

The `rssac028-appendix-a.yml` ansible playbook in the resolver testbed performs all the necessary tasks to reproduce the results in the table; including installation of older versions of BIND, NSD and Knot-DNS. The playbook needs only the `servers-vm` to deploy the zones for the schemes to be tested and to perform the testing queries. The playbook will create a file `rssac028-appendix-a.txt` in the `ansible/results` directory containing the measured sizes.

Reproduced RSSAC028 Appendix A results

BIND 9.10.3-P4

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	508	811	1097	1097
5.2	508	811	3833	3833
5.3	507	782	3938	3938
5.4	504	807	4093	4093
5.5	264	275	561	561
5.6	250	625	1485	1485

NSD 4.1.13

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	492	811	1097	1097
5.2	492	811	1097	1097
5.3	491	782	1405	1126
5.4	488	807	1093	1093
5.5	500	847	1133	1133
5.6	250	625	1485	990

Knot DNS 2.2.1

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	508	811	1097	1097
5.2	508	811	1097	1097
5.3	507	782	3938	3938
5.4	504	807	1093	1093
5.5	500	847	1133	1133
5.6	250	625	1485	1485

Knot DNS 2.3.0

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	228	239	525	525
5.2	228	239	525	525
5.3	507	782	3938	3938
5.4	224	235	521	521
5.5	264	275	561	561
5.6	250	625	1485	1485

Results for recent versions of BIND, NSD and Knot DNS

BIND 9.18.13

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	504	823	1109	1109
5.2	504	823	1109	1109
5.3	211	222	508	508
5.4	500	819	1105	1105
5.5	512	859	1145	1145
5.6	42	53	339	339

NSD 4.6.1

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	492	811	1097	1097
5.2	492	811	1097	1097
5.3	491	782	1102	1126
5.4	488	807	1093	1093
5.5	500	847	1133	1133
5.6	250	625	834	990

Knot DNS 3.2.5

Scenario	No EDNS	No DNSSEC	IPv4 DNSSEC	IPv6 DNSSEC
5.1	508	1003	1217	1217
5.2	508	1003	1217	1217
5.3	507	782	1068	1068
5.4	500	951	1209	1209
5.5	500	847	1133	1133
5.6	250	625	1198	1198

Appendix C: Naming schemes cheat sheet

5.1. Current : [a-m] → **root-servers** → **net** → ○

5.2. Current + DNSSEC

The authoritative servers for the root zone have the names '[a-m].root-servers.net'. The 'root-servers.net' zone is served by name servers that also serve the root zone. In scheme 5.1, the 'root-servers.net' zone continues to be unsigned. In scheme 5.2 the 'root-servers.net' zone will be signed.

5.3. In-zone NS Names : [a-m].root-servers → ○

5.3.1. or [a-m] → ○

The root zone will have an NS RRset consisting of in-zone names with the A and AAAA records of the root servers. Scheme 5.3 uses '[a-m].root-servers' as the name for the root servers. Scheme 5.3.1, that uses a single letter '[a-m]' for the root servers.

5.4. Shared Delegated TLD : [a-m] → **root-servers** → ○

The root zone will have an NS RRset that consists of 13 domain names that share a new common delegated TLD (for example, the names 'a.root-servers', 'b.root-servers', and so on). There will be 13 records in the root zone's NS RRset pointing to the root server name server instances. The new shared TLD will be delegated to the same set of nameservers.

5.5. Names Delegated to Each Operator



A new domain will be delegated to each root server operator. The root zone will have an NS RRset consisting of server names that are managed by the corresponding root server operators. The names for this proposal can either be short labels in the root zone ('a', 'b', and so on in scheme 5.5) or can have all records under a common label ('a.root-servers', 'b.root-servers' in scheme 5.5.1). No other delegations are involved.

5.6. Single Shared Label for All Operators

all-root-servers → ○

5.6.1. root-servers → ○

Instead of having individual names for each root server, the set of root servers could be given one name at the top level (such as 'all-root-servers' in scheme 5.6 or 'root-servers' in scheme 5.6.1) and that one name has the 13 IPv4 addresses and 13 IPv6 addresses of the root servers as two RRsets.