Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 23, 2016

L. Song
S. Kerr
D. Liu
Beijing Internet Institute

November 20, 2015

Experiences from Root Testbed in the Yeti DNS Project
draft-song-yeti-testbed-experience-00

Abstract

This document reports and discusses issues in DNS root services, based on experiences from the experiments in the Yeti DNS project. These issues include IPv6-only operation, the root DNS server naming scheme, DNSSEC KSK rollover, root server renumbering, multiple root zone signer, and so on. This project was founded in May 2015 and has since built a live root DNS server system testbed with volunteer root server and resolver operations.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 23, 2016.

Table of Contents

1. Introduction

RFC 1034[RFC1034] says the domain name space is a tree structure. The top level of the tree for the unique identifier system is the DNS root system. It has been operational for 25+ years. It is pivotal to making the current Internet useful. So it is considered somewhat ossified for stability reasons. It is hard to test and implement new ideas evolving to a more advanced level to counter challenges like IPv6-only operation, DNSSEC key/algorithm rollover [RFC4986], scaling issues, and so on. In order to make the test more practical, it is also necessary to involve users' environments which are highly diversified, in order to study the effect of the changes in question.

To benefit Internet development as a whole, the Yeti Project [Yeti-DNS-Project] was proposed to build a parallel, experimental, live IPv6 DNS root system to discover the limits of DNS root name service and deliver useful technical output. Possible research agenda will be explored on this testbed, covering several aspects (but not limited to):

- o IPv6-only operation
- o DNSSEC key rollover
- o Renumbering issues
- o Scalability issues
- o Multiple zone file signers

Starting from May 2015, three coordinators began to build this live experimental environment and called for participants. At the time of writing, there are 14 Yeti root servers with 13 operators, and experimental traffic from volunteers, universities, DNS vendors, mirrored traffic non-Yeti traffic, and RIPE Atlas probes. Some experiments have been proposed and have been verified in lab tests.

Note that the Yeti DNS project has complete fealty to IANA as the DNS name space manager. All IANA top-level domain names will be precisely expressed in the Yeti DNS system, including all TLD data and meta-data[Root-Zone-Database]. So, the Yeti DNS project is not an "alternative root" in the usual sense of that term. It is expected to inform the IANA community by peer-reviewed science as to future possibilities to consider for the IANA root DNS system.

In order to let people know the technical activities in Yeti DNS project, this document reports and discusses issues on root DNS services, based on experiences so far from the experiments in the Yeti DNS project.

2. Problem Statement

Some problems and policy concerns over the DNS Root Server system stem from centralization from the point of view of DNS content consumers. These include external dependencies and surveillance threats.

- o External Dependency. Currently, there are 12 DNS Root Server operators for the 13 Root Server letters, with more than 500 instances deployed globally. Yet compared to the number of connected devices, AS networks, and recursive DNS servers, the number of root instances is far from sufficient. Connectivity loss between one autonomous network and the IANA root name servers usually results in loss of local service within the local network, even when internal connectivity is perfect
- o Surveillance risk. Even when one or more root name server anycast instances are deployed locally or in a nearby network, the queries

sent to the root servers carry DNS lookup information which enables root operators or other parties to analyze the DNS query traffic. This is a kind of information leakage [RFC7626] which is to some extent not acceptable to some policy makers

People are often told that the current root system with 13 root servers is not able to be extended to alleviate the above concerns, because it is limited to 13 by the current DNS protocol [ROOT-FAQ]. To the best of author's knowledge, there is no scientific evidence to support this assertion. It remains an open question.

There are some technical issues in the areas of IPv6 and DNSSEC, which were introduced to the DNS root server system after it was created. Renumbering DNS root servers also creates some technical issues.

- IPv6-only capability. Currently some DNS servers including root which support both A and AAAA (IPv4 and IPv6) records still do not respond to IPv6 queries. IPv6 introduces larger IP packet MTU (1280 bytes) and a different fragmentation model [RFC2460]. It is not clear whether DNS can survive without IPv4 (in an IPv6-only environment), or what the impact of IPv6-only environment introduces to current DNS operations especially in the DNS root server system.

- KSK rollover. Currently, IANA rolls the ZSK every six weeks but the KSK has never been rolled as of writing. Is the 512 bytes DNS packet size limitation still observed? Is [RFC5011] widely supported by resolvers? How about longer key with different encryption algorithm? There are many issues still unknown.

- Renumbering issue. It is likely that root operators may change their IP addresses for root servers as well. There is no dynamic update mechanism to inform resolvers and other Internet infrastructure relying on root service of such changes.

3. Yeti Testbed and Experiment Setup

To use the Yeti testbed operationally, the information that is required for correct root name service is a matching set of the following:

- a root "hints file"

- the root zone apex NS record set

- the root zone's signing key

o root zone trust anchor

Although Yeti DNS project publishes strictly IANA information for TLD data and meta-data, it is necessary to use a special hint file and replace the apex NS RRset with Yeti authority name servers, which will enable the resolves to find and stick to the Yeti root system. In addition, unless IANA was to help Yeti sign its root zone with a different root set, it is necessary to use a different ZSK and KSK (the DNSSEC trust anchor) in Yeti system.

Below is a figure to demonstrate the topology of Yeti and the basic data flow, which consists of the Yeti distribution master, Yeti root server, and Yeti resolver:

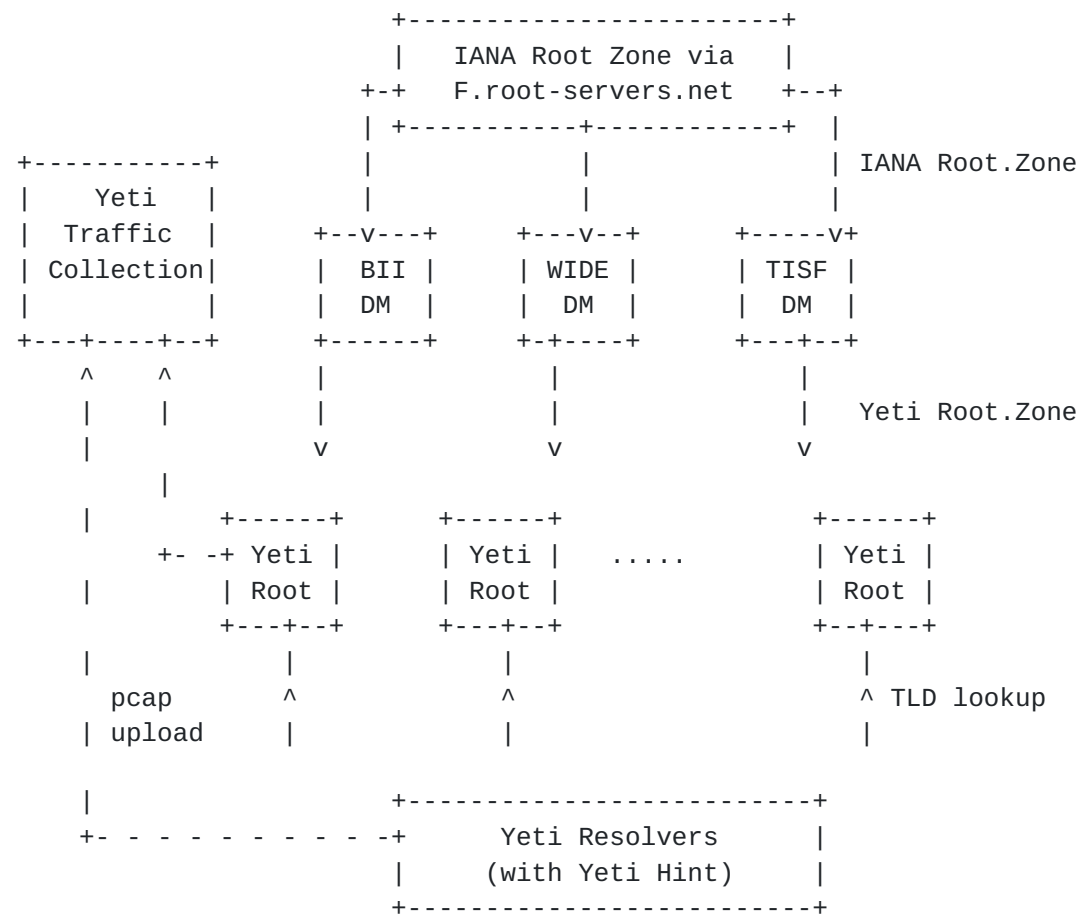

Figure 1. The topology of Yeti testbed

3.1. Distribution Master

As shown in figure 1, the Yeti Root system takes the IANA root zone and performs minimal changes needed to serve the zone from the Yeti root servers instead of the IANA root servers. In Yeti, this modified root zone is generated by the Yeti Distribution Masters (DM), which provide it to the Yeti root servers.

So the generation process is:

- DM downloads the latest IANA root zone at a certain time
- DM makes modifications to change from the IANA to Yeti root servers
- DM signs the new Yeti root zone
- DM publishes the new Yeti root zone to Yeti root servers

While in principle this could be done by a single DM, Yeti uses a set of three DMs to avoid any sense that the Yeti project is run by a single organization. The three Distribution Masters (DMs) can independently fetch the root zone from IANA, sign it and publish the latest zone data to Yeti root servers.

In the same while, these DMs coordinate their work so that the resulting Yeti root zone is always consistent. There are two aspects of coordination between three DMs: timing and information synchronization.

3.1.1. Yeti root zone SOA SERIAL

Consistency with IANA root zone except the top level apex record is one of most important point for the project. As part of Yeti DM design, the Yeti SOA SERIAL which reflect the changes of yeti root zone is one factor to be considered.

Currently IANA SOA SERIAL number for root zone is in the form of YYYYMMDDNN, like 2015111801. In Yeti root system, IANA SOA SERIAL is directly copied in to Yeti SOA SERIAL. So once the IANA root zone has changed with a new SOA SERIAL, a new version of the Yeti root zone is generated with the same SOA SERIAL.

There is a case of Yeti DM operation that when a new Yeti root server added, DM operator change the Yeti root zone without change the SOA SERIAL which introduces inconsistency of Yeti root system. To avoid inconsistency, the DMs hold on every changes to Yeti apex record and

only new IANA SOA SERIAL will trigger the operation of adding these changes to Yeti root zone.

A analysis of IANA convention shows IANA SOA SERIAL change 2 times every day (NN=00, 01). And that since October 2007 the maximum of NN was 03 while 13 times it is observed that the versions with NN=02. So in the worst case, the changes of Yeti apex record is updated into Yeti root zone in less than 12 hours.

3.1.2. Timing of Root Zone Fetch

Yeti root system operators do not receive notify message from IANA when IANA root zone updates with a new SOA serial number. So Yeti DMs check the root zone periodically. At the time of writing, each Yeti DM checks to see if the IANA root zone has changed hourly, on the following schedule:

DM Operator	Time
BII	hour+00
WIDE	hour+20
TISF	hour+40

Note that Yeti DMs can check IANA root zone more frequently (every minute for example). A test done by Yeti participant shows that the delay of IANA root zone update from the first IANA root server to last one is around 20 minute. Once a Yeti DM fetch the new root zone, it will notify all the Yeti root server with a new SOA serial number. So normally yeti root server will be notified in less than 20 minute after new IANA root zone generated. Ideally, if IANA DM notifies the Yeti DMs, Yeti root zone will be updated more timely.

3.1.3. Information Synchronization

Given three DMs operational in Yeti root system, it is necessary to prevent any inconsistency caused by human mistakes in operation. The straight method is to share the same parameters to produce the Yeti root zone. There parameters includes following set of files:

- the list of Yeti root servers, including:
  - public IPv6 address and host name
  - IPv6 addresses originating zone transfer
  - IPv6 addresses to send DNS notify to

- the ZSKs used to sign the root
- the KSK used to sign the root
- the SERIAL when this information is active

The theory of operation is straight that each DM operator runs a Git repository, containing files with the information needed to produce the Yeti root zone. When a change is desired (such as adding a new server or rolling the ZSK), a DM operator updates the local Git repository. A SOA SERIAL in the future is chosen for when the changes become active. The DM operator then pushes the changes to the Git repositories of the other two DM operators. When the SOA SERIAL of the root zone passes the number chosen, then the new version of the information is used.

3.2. Yeti Root Servers

In Yeti Root system, authoritative servers donated and operated by Yeti volunteers are configured as a slave to the Yeti DM. As the time of writing, there are 14 Yeti root servers distributed around the world. As one of operational research goal, all authoritative servers are required to work in an IPv6-only environment. In addition, different from IANA root, Yeti root server only serve the Yeti root zone, other than root-servers.org zone and .arpa zone.

```
.                               3600000    IN   NS       bii.dns-lab.net
bii.dns-lab.net                 3600000    IN   AAAA     240c:f:1:22::6
.                               3600000    IN   NS       yeti-ns.tisf.net
yeti-ns.tisf.net                3600000    IN   AAAA     2001:559:8000::6
.                               3600000    IN   NS       yeti-ns.wide.ad.jp
yeti-ns.wide.ad.jp              3600000    IN   AAAA     2001:200:1d9::35
.                               3600000    IN   NS       yeti-ns.as59715.net
yeti-ns.as59715.net             3600000    IN   AAAA
2a02:cdc5:9715:0:185:5:203:53
.                               3600000    IN   NS       dahu1.yeti.eu.org
dahu1.yeti.eu.org               3600000    IN   AAAA
2001:4b98:dc2:45:216:3eff:fe4b:8c5b
.                               3600000    IN   NS       ns-yeti.bondis.org
ns-yeti.bondis.org              3600000    IN   AAAA     2a02:2810:0:405::250
.                               3600000    IN   NS       yeti-ns.ix.ru
yeti-ns.ix.ru                   3600000    IN   AAAA     2001:6d0:6d06::53
.                               3600000    IN   NS       yeti.bofh.priv.at
yeti.bofh.priv.at               3600000    IN   AAAA     2a01:4f8:161:6106:1::10
.                               3600000    IN   NS       yeti.ipv6.ernet.in
yeti.ipv6.ernet.in              3600000    IN   AAAA     2001:e30:1c1e:1::333
.                               3600000    IN   NS       yeti-
dns01.dnsworkshop.org
yeti-dns01.dnsworkshop.org      3600000    IN   AAAA     2001:1608:10:167:32e::
53
.                               3600000    IN   NS       yeti-ns.conit.co
yeti-ns.conit.co                3600000    IN   AAAA     2607:ff28:2:10::47:a010
.                               3600000    IN   NS       dahu2.yeti.eu.org
dahu2.yeti.eu.org               3600000    IN   AAAA     2001:67c:217c:6::2
.                               3600000    IN   NS       yeti.aquaray.com
yeti.aquaray.com                3600000    IN   AAAA     2a02:ec0:200::1
.                               3600000    IN   NS       yeti-ns.switch.ch
yeti-ns.switch.ch               3600000    IN   AAAA     2001:620:0:ff::29
```

Figure 2. the Yeti root server in hint file

Since Yeti is a a live root DNS server system testbed, it needs to capture DNS traffic sent for later analysis. Today some servers use dnscap, which is a DNS-specific tool to produce pcap files. There are several versions of dnscap floating around; some people use the VeriSign one. Since dnscap loses packets in some cases (tested on a Linux kernel), some people use pcapdump. It requires the patch attached to this bug report [dnscap-bug-report]

System diversity is also a requirement and observed for current 14 Yeti root server. Here are the results of a survey regarding the machine, operation system and DNS software:

- Machine: 11 out of 14 root server operator are using a VPS to provide service.

- OS: 6 operators use Linux (including Ubuntu, Debian, CentOS). 5 operators use FreeBSD and 1 NetBSD. 2 other servers are unknown.

- DNS software: 8 our of 14 root server use BIND (varying from 9.9.7 to 9.10.3). 4 of them use NSD (4.10 and 4.15). The other 2 servers use Knot (2.0.1 and 2.1.0).

3.3. Yeti Resolvers and Experimental Traffic

In client side of Yeti project, we expect participants and volunteers from individual researchers, labs of universities, companies and institutes, and vendors (for example, the DNS software implementers), developers of CPE devices &IoT devices, and middle box developers who can test their product and connect their own testbed into Yeti testbed. Resolvers donated by Yeti volunteers are required to be configured with Yeti hint file and Yeti DNSSEC KSK. It is required that Yeti resolver can speak both IPv4 and IPv6, given that not all the stub resolver and authoritative servers on the Internet are IPv6 capable.

At the time of writing several universities and labs have joined us and contributed certain amount of traffic to Yeti testbed. But it is far from the desired volume of the experiment traffic. So Yeti adopts two alternative ways to increase the experimental traffic in Yeti testbed to check the functionality of Yeti root system.

One approach is to mirror the real traffic by off-path method and reply it into Yeti testbed; this is implemented by one of the Yeti root server operators. Another approach is to use some traffic generating tool such as RIPE Atlas probes to generate specific queries against Yeti servers.

4. Experiments in Yeti Testbedd

The main goal of Yeti DNS Project is to act as an experimental network. Experiments will be conducted on this network. In order to make the findings that result from these experiments more rigorous, an experiment protocol is proposed.

A Yeti experiment goes through four phases:

- Proposal. The first step is to make a proposal. It is discussed and if accepted by the Yeti participants then it can proceed to the next phase.

- Lab Test. The next phase is to run a version of the experiment in a controlled environment. The goal is to check for problems such

as software crashes or protocol errors that may cause failures on the Yeti network, before putting onto the experimental network.

- Yeti Test. The next phase actually running the experiment on the Yeti network. Details of this will depend on the experiment. It must be coordinated with the Yeti participants.

- Report of Findings. When completed, a report of the findings of the experiment should be made. It need not be an extensive document.

In this section, we are going to introduce some experiments implemented and planned in Yeti project.

4.1. Naming Scheme and Glue Issue

In root server history, the naming scheme for individual root servers was not fixed. Current IANA Root server adopt [a-m].root-servers.net to represent 13 servers which are labeled with letter from A to M. For example, L root operated by ICANN uses l.root-servers.net to represent their server as NS. One reason behind this naming scheme is that the common suffix 'root-servers.net' can be compressed in DNS message to produce a smaller DNS response.

Different from the IANA root naming scheme, the Yeti root system uses separate and normal domains for root servers (shown in figure 2). The motivation of this naming scheme in Yeti is that it intentionally produces larger packets for priming responses. Note that currently, the Yeti root has a priming response which is almost the same size as the IANA root. Yeti has no compression, and has one more name server, but it also has no IPv4 addresses.

the change of name scheme not only affects the size of priming response, but also changes the content in additional section of the response.When a resolver bootstraps, it sends a 'NS-for-dot' query to one of the root servers that it knows about, which is called a priming query. It looks like this with the "dig" command:

```
$ dig @a.root-servers.net -t ns +norecurse +edns +nodnssec
```

Normally in IANA root system the priming response contains the _names_ of the root servers in the answer section and the _addresses_ of the root servers in the additional section. The additional section data is what the resolvers need to actually perform recursion. Shown as below:

In priming response :

Answer Section:

```
---------------
.        518400  IN  NS  a.root-servers.net.
.        518400  IN  NS  b.root-servers.net.
  ...
.        518400  IN  NS  m.root-servers.net.
```

Addtional section:

```
------------------
a.root-servers.net.     3600000  IN  A 198.41.0.4
b.root-servers.net.     3600000  IN  A 192.228.79.201
  ...
m.root-servers.net.     3600000  IN  AAAA  2001:dc3::35
```

In IANA root system, all the root server returns the "a.root-servers.net" addresses in the additional section, because root servers not only answer for root zone, but also answer for "root-servers.net" zone. Note that BIND will not behave like this if it is not configured for the "root-servers.net" zone. NSD and Knot happily return such "glue" in the additional section, whether configured for the "root-servers.net" zone or not.

The Yeti root naming scheme uses separate and independent domain for individual root servers. It this case, the priming response from Yeti root servers will only contain the A&AAAA records of that responding server in BIND 9. However it is desired that the Yeti root servers to respond to priming queries with the addresses of all Yeti root servers in the additional section. This will make them operate as similar to the IANA root servers as possible.

In Yeti root system, there are two approaches adopted in different root servers. One is to patch BIND 9 so that it includes the glue addresses in the additional section. The other one is to add a zone file for each root server and answer for all of them at each Yeti server. That means each Yeti root server would have a small zone file for "bii.dns-lab.net", "yeti-ns.wide.ad.jp", "yeti-ns.tisf.net", and so on.

4.2. Multiple-Signers with Multi-ZSK

According to the Problem statement of Yeti DNS project, more independent participants and operators of root system is desirable. As the name implies, multi-ZSK mode will introduce different ZSKs sharing a single unique KSK, as opposed to the IANA root system

(which uses a single ZSK to sign the root zone). On the condition of good availability and consistency on root system, the Multi-ZSK proposal is designed to give each DM operator enough room to manage their own ZSK, by choosing different ZSK, length, duration, and so on; even the encryption algorithm may vary.

According to the Yeti experiment protocol, a lab test was done to verify the concept and validity of Multi-ZSK. The purpose of the test is two-fold: 1) To test whether this proposal can be implemented by current DNS protocol&software (to see if there should be some extra modification to protocol or software), and 2) To demonstrate the impact of Multi-ZSK proposal to the current root system.

The experiment is run like this: build a test topology like figure 3, with 2 Root servers and a resolver (BIND 9). The hint file of this test only contains the two DM servers. In the first time slot, Root A is up and Root B is turned off. Let resolver bootstrap from Root A and query a certain signed TLD (or junk query). For the second time slot, turn off Root A and turn on Root B. Let resolver shift to Root B to look up another TLD (or a junk query). the test result of different time slot is compared to see whether the resolver can validate the DNSSEC signature.

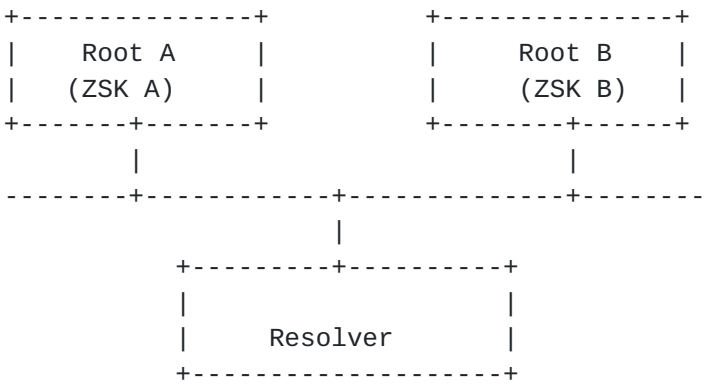

Figure 3. Multi-ZSK lab test topology

There are two cases in this test:

- Case 1: Assign one ZSK to the smart sign process on each DM, which means the root zone only contain one single ZSK.

- Case 2: Assign both ZSK A and ZSK B to the smart sign process on each DM, which means the root zone contains two ZSK. In this case, it is required that Root A and Root B to share their public ZSK to each other before root zone is signed.

In case 1 SERVFAIL is received during switching because the resolver can not validate the signature signed by Root B after switching. In case 2 NOERROR is received. It is the actual demonstration of how Multi-ZSK works by adding multiple ZSK to the root zone. As a result, the resolver will cache the key sets instead of single ZSK to validate the data no matter it is signed by Root A or Root B. As follow-up test, Unbound also passed the test with more than 10 DMs and 10 ZSKs.

Although more DM and ZSK can be added into the test, adding more ZSKs to root zone enlarges the DNS response size for DNSKEY queries which may be a concern given the limitation of DNS packet size. Current IANA root server operators are inclined to keep the packets size as small as possible. So the number of DM and ZSK will be parameter which is decided based on operation experience. In the current Yeti root testbed, there will be 3 DMs, each with a separate ZSK.

4.3. Root Renumbering Issue and Hint File Update

With the nearing renumbering of H root Server's IP address, there is a discussion of ways that resolvers can update their hint file. Traditional ways include using FTP protocol by doing a wget and using dig to get the servers' addresses manually. Each way would depend on operators manual operation. As a result, there are many old machines that have not updated their hint files. As a proof, after done renumbering for thirteen years, there is an observation that the "Old J-Root" can still receive DNS query traffic [Renumbering-J-Root].

This experiment proposal aims to find an automatic way for hint-file updating. The already-completed work is a shell script tool which provides the function that update a hint-file in file system automatically with DNSSEC and trust anchor validation.

The methodology is straightforward. The tool first queries the NS list for "." domain and queries A and AAAA record for every domain on the NS list. It requires DNSSEC validation for both signature and trust anchor for all the answers. After getting all the answers, the tool compares the new hint file to the old one. If there is a difference, it renames the old one with a time-stamp and replaces the old one with the new one. Otherwise the tool deletes the new hint file and nothing will be changed.

Note that in current IANA root system the servers named in the root NS record are not signed due to lack of incentive. So the tool can not fully work in the production network. In Yeti root system some of the names listed in the NS record are signed, which provides a test environment for such a proposal.

4.4. DNS Fragments

In consideration of new DNS protocol and operation, there is always a hard limit on the DNS packet size. Take Yeti for example: adding more root servers, using the Yeti naming scheme, rolling the KSK and Multi-ZSK increase the packet size. The fear of large DNS packets mainly stem from two aspects: one is IP-fragments and the other is frequently falling back to TCP.

In Yeti testbed, a mechanism is implemented which supports larger DNS packet working around the IP-layer fragment caused by middle box misbehavior (in IPv4) and IPv6 MTU limitation by splitting a single DNS message across multiple UDP datagrams. This DNS fragments mechanism is documented in [I-D.muks-dns-message-fragments] as an experimental IETF draft.

4.5. The KSK Rollover Experiment in Yeti

The Yeti project provides a good basis to conduct a real-world experiment of a root KSK roll. It is not a perfect analogy to the IANA root because all of the resolvers to the Yeti experiment are "opt-in", and are presumably run by administrators who are interested in the DNS and knowledgeable about it. Still, it can inform the IANA root KSK roll.

The IANA root KSK has not been rolled. ICANN put together a design team to analyze the problem and make recommendations. The design team put together a proposal [ICANN-ROOT-ROLL]. Whether this proposal or a different one is adopted, the Yeti project can use it as a basis for an experimental KSK roll. The experiment may not be identical, since the time-lines laid out in the current IANA plan are very long, and the Yeti project would like to conduct the experiment in a shorter time.

The Yeti project would also like to conduct an experiment to try rolling the root KSK using a straightforward method, such as a double-DS approach outlined in [RFC6781]. If this ends up being adopted for the IANA root, then only a single Yeti experiment will need to be conducted.

< <It's worthwhile to mention that in Yeti testbed there is a lessen when the KSK rollover was implemented at the first time. It lasted for one month and has been held off afterwards. In the first trial, it made old KSK inactive in one week after new key was created, and delete it in another week, which is totally unaware of [RFC5011]. Because the hold-down timer is not correctly set in the server side, some clients(UNBOUND) SERVFAILs (like dig without +cd) because the new key is still in AddPend state when old key is inactive. The

lesson from the first KSK trial is that both server and client should compliant to [RFC5011] to set proper timer. The question is how can authority server know the resolver is ready for RFC5011？ [I-D.wkumari-dnsop-trust-management] tries to address the problem. >>

5. Other Technical findings and bugs

Besides the experiments with specific goal and procedures, some unexpected bugs have been reported. It is worthwhile to record them as technical findings from Yeti DNS Project. Hopefully, these experiences can share and help.

5.1. IPv6 fragments issue

There are two cases in Yeti testbed reported that some Yeti root servers on VPS failed to pull the zone from Distribution Master via AXFR/IXFR. Two facts have been revealed in both client side and server side after trouble shooting.

One fact in client side is that some operation system on VPS can not handle IPv6 fragments correctly which causes failure when they are doing AXRF/IXFR in TCP. The bug covers several OS and one VM platform (listed below).

OS	VM
NetBSD 6.1 and 7.0RC1	VMware ESXI 5.5
FreeBSD10.0	
Debian 3.2	

Another fact is from server side in which one TCP segment of AXRF/IXFR is fragmented in IP layer resulting in two fragmented packets. This weird behavior has been documented IETF draft [I-D.andrews-tcp-and-ipv6-use-minmtu]. It reports a situation that man implementations of TCP running over IPv6 neglect to check the IPV6_USE_MIN_MTU value when performing MSS negotiation and when constructing a TCP segment. It will cause TCP MSS option set to 1440 bytes, but IP layer will limit the packet less than 1280 bytes and fragment the packets which finally result two fragmented packets.

The latter is not a technical error though, but it will cause the error in the former fact which deserves much attention in IPv6 operation when VPS is already widely used.

5.2. Root compression issue

[RFC1035]specifies DNS massage compression scheme which allows a domain name in a message to be represented as either: 1) a sequence of labels ending in a zero octet, 2) a pointer, 3) or a sequence of labels ending with a pointer. It is designed to save more room of DNS packet.

However in Yeti testbed, it is found that Knot 2.0 server compresses even the root. It means in a DNS message the name of root (a zero octet) is replaced by a pointer of 2 octets. As well, it is legal but breaks some tools (Go DNS lib in this bug report) which does not expect such name compression for root. Now both Knot and Go DNS lib have fixed that bug.

6. IANA Considerations

This document requires no action from the IANA.

7. Acknowledgements

The editors fully acknowledge that this memo is based on works and discussions with Paul Vixie and Akira Kato in Yeti DNS project[Yeti-DNS-Project]. Thanks to Antonio Prado and Stephane Bortzmeyer who helped to review the document and give many editing suggestions

Acknowledgment to all the Yeti participant and volunteers who make the experimental testbed become functional and workable.

8. References

[dnscap-bug-report]
Bortzmeyer, S., "pcaputils: IWBN to have an option to run a program after file rotation in pcapdump", 2009, <https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=545985>.

[I-D.andrews-tcp-and-ipv6-use-minmtu]
Andrews, M., "TCP Fails To Respect IPV6_USE_MIN_MTU", draft-andrews-tcp-and-ipv6-use-minmtu-04 (work in progress), October 2015.

[I-D.muks-dns-message-fragments]
Sivaraman, M., Kerr, S., and D. Song, "DNS message fragments", draft-muks-dns-message-fragments-00 (work in progress), July 2015.

[I-D.wkumari-dnsop-trust-management]
Kumari, W., Huston, G., Hunt, E., and R. Arends, "Signalling of DNS Security (DNSSEC) Trust Anchors", draft-wkumari-dnsop-trust-management-01 (work in progress), October 2015.

[ICANN-ROOT-ROLL]
"Root Zone KSK Rollover Plan", 2015, <https://www.icann.org/en/system/files/files/root-zone-ksk-rollover-plan-draft-04aug15-en.pdf>.

[Renumbering-J-Root]
Wessels, D., "Thirteen Years of "Old J-Root"", 2015, <https://indico.dns-oarc.net/event/24/session/10/contribution/10/material/slides/0.pdf>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4986] Eland, H., Mundy, R., Crocker, S., and S. Krishnaswamy, "Requirements Related to DNS Security (DNSSEC) Trust Anchor Rollover", RFC 4986, DOI 10.17487/RFC4986, August 2007, <http://www.rfc-editor.org/info/rfc4986>.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <http://www.rfc-editor.org/info/rfc5011>.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <http://www.rfc-editor.org/info/rfc6781>.

[RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <http://www.rfc-editor.org/info/rfc7626>.

[ROOT-FAQ]
Karrenberg, D., "DNS Root Name Server FAQ", 2007,
<https://www.isoc.org/briefings/020/>.

[Root-Zone-Database]
"Root Zone Database",
<http://www.iana.org/domains/root/db>.

[Yeti-DNS-Project]
"Website of Yeti DNS Project", <http://www.yeti-dns.org>.

Authors' Addresses

Linjian Song
Beijing Internet Institute
2/F, Building 5, No.58 Jinghai Road, BDA
Beijing 100176
P. R. China

Email: songlinjian@gmail.com
URI: http://www.biigroup.com/

Shane Kerr
Beijing Internet Institute
2/F, Building 5, No.58 Jinghai Road, BDA
Beijing 100176
CN

Email: shane@biigroup.cn
URI: http://www.biigroup.com/

Dong Liu
Beijing Internet Institute
2508 Room, 25th Floor, Tower A, Time Fortune
Beijing 100028
P. R. China

Email: dliu@biigroup.com
URI: http://www.biigroup.com/