NSEC Caching Revisited

Geoff Huston, Joao Damas
APNIC Labs
October '19

What is NSEC Caching?

RFC 8198, July 2017:

- If a DNS zone is pre-signed then each zone entry is “connected” to the next with a NSEC (or NSEC3) record
- Proof of non-existence of a name is provided by a signed NSEC record that “spans” the query name
- If a DNSSEC-validating recursive resolver receives an NSEC record it can cache this record and use it to answer all subsequent queries for names in the NSEC span for the TTL of the record

NSEC Caching Benefits (according to RFC8198)

- Reduced Latency
- Decreased Recursive Resolver Load
- Decreased Authoritative Server Load
- Random name attack mitigation

Does it Deliver?

- Petr Špaček's report to DNS OARC 28 (2018)

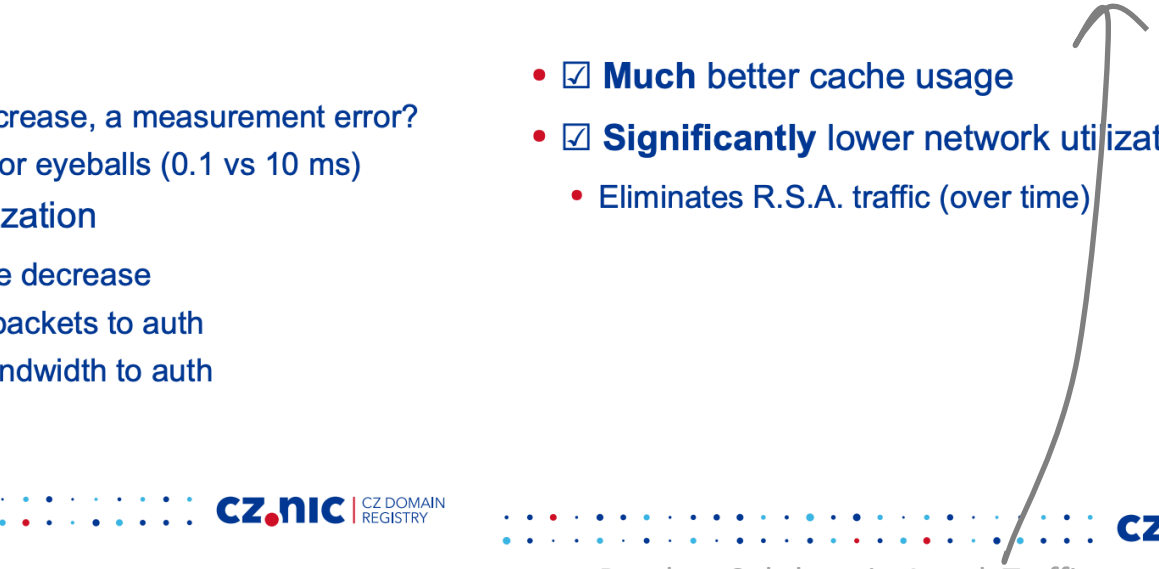
https://indico.dns-oarc.net/event/28/contributions/509/attachments/479/786/DNS-OARC-28-presentation-RFC8198.pdf

RFC 8198's promises & normal traffic

- ☐ Lower latency
  - Some unexplained increase, a measurement error?
  - Likely not significant for eyeballs (0.1 vs 10 ms)
- ☑ Lower network utilization
  - Small but reproducible decrease
  - 1-2 % decrease of # packets to auth
  - 3-4 % decrease of bandwidth to auth

RFC 8198's promises & R.S.A. traffic

- ☑ **Much** better cache usage
- ☑ **Significantly** lower network utilization
  - Eliminates R.S.A. traffic (over time)



Random Subdomain Attack Traffic

Does it Deliver?

- Petr Špaček's report to DNS OARC 28 (2018)

https://indico.dns-oarc.net/event/28/contributions/509/attachments/479/786/DNS-OARC-28-presentation-RFC8198.pdf

Normal traffic: Experimental setup

- Replay query PCAP to BIND 9.12.0
  - synth-from-dnssec yes / no;
- Record to PCAP
  - traffic to auth
  - answers
- Analyze
  - # packets to auth
  - bandwidth to auth
  - latency for answers

R.S.A. traffic: Experimental setup

- Auth server with a test zone
- Replay random query names to Knot Resolver
- Record **traffic to auth** into PCAP
- Analyze
  - # packets to auth
  - bandwidth to auth

cz.nic | CZ DOMAIN REGISTRY

Benchtop vs Live Measurement

- Petr's results were obtained by feeding a log of query data into a resolver and measuring the query traffic from the resolver to the authoritative server(s)
- We thought it would be useful to see if these results are reproducible in the Internet:

- **To what extent is NSEC caching visible in the DNS today?**
- **How well does NSEC caching work?**

Measurement Setup

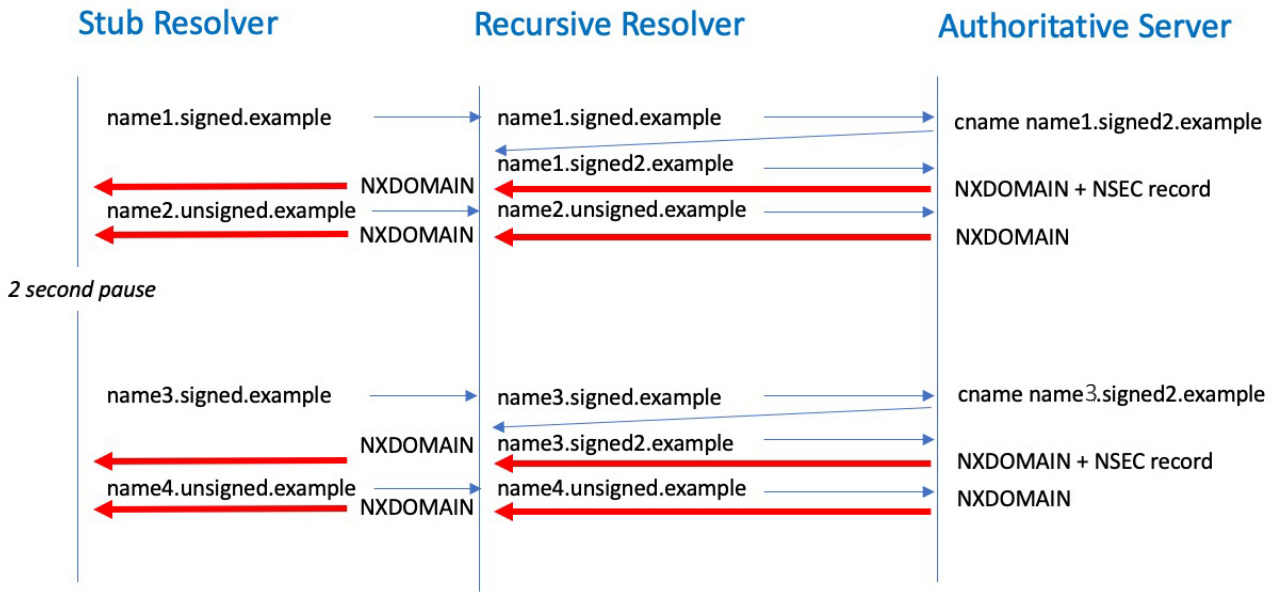
- We use APNIC Lab's Ad-based measurement platform in conjunction with a set of servers
- The Ad campaign delivers some 8M – 12M ads per day to eyeball networks across the entire Internet
- Each ad contains an HTML5 script that runs the NSEC experiment:
  - Generate unique label name within a DNSSEC-signed domain name and query
  - The DNSSEC response is an NXDOMAIN code with an NSEC record that includes a span across the label name space in the domain
  - Wait 2 seconds
  - Generate a new unique label name sits within the span of the NSEC record

Measurement Setup

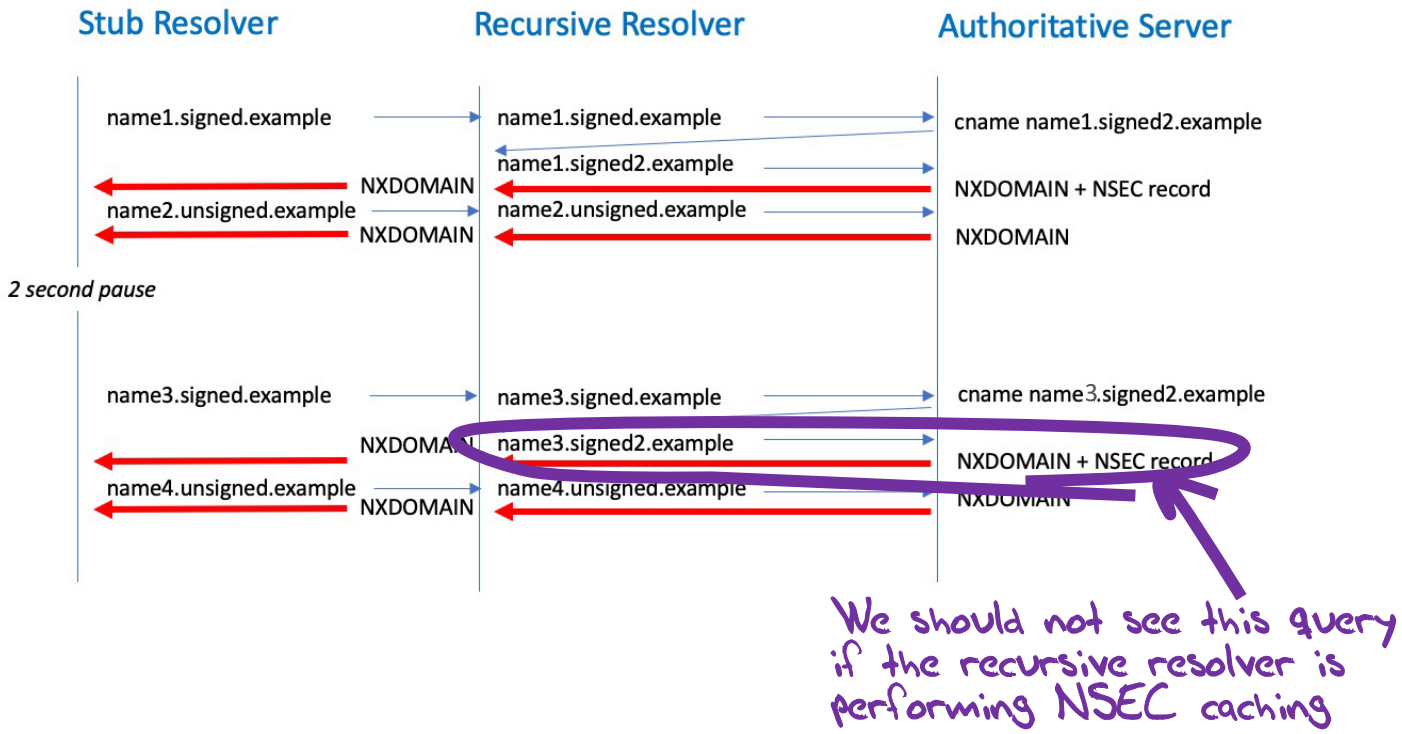
Controls:

- We use a pair of unique labels that generate a CNAME response to a unique non-existent name
  - That way the authoritative server will see the query that generates the CNAME response and depending on whether the recursive resolver is performing NSEC caching the authoritative server may (or may not) see the second CNAME target name query
- We use a second pair of labels that are generated within an unsigned zone where the server always returns NXDOMAIN

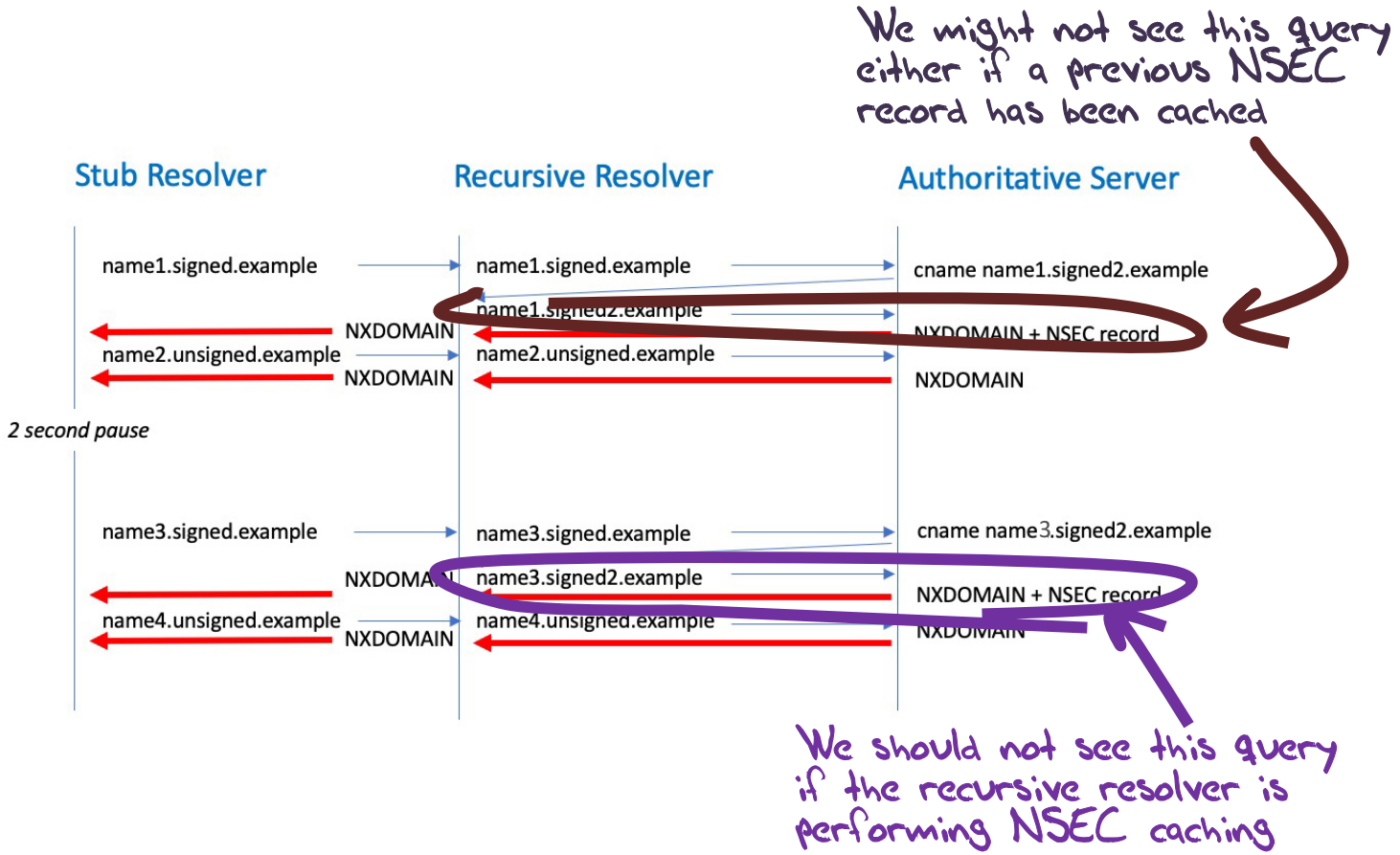
Measurement Details



Measurement Details



Measurement Details

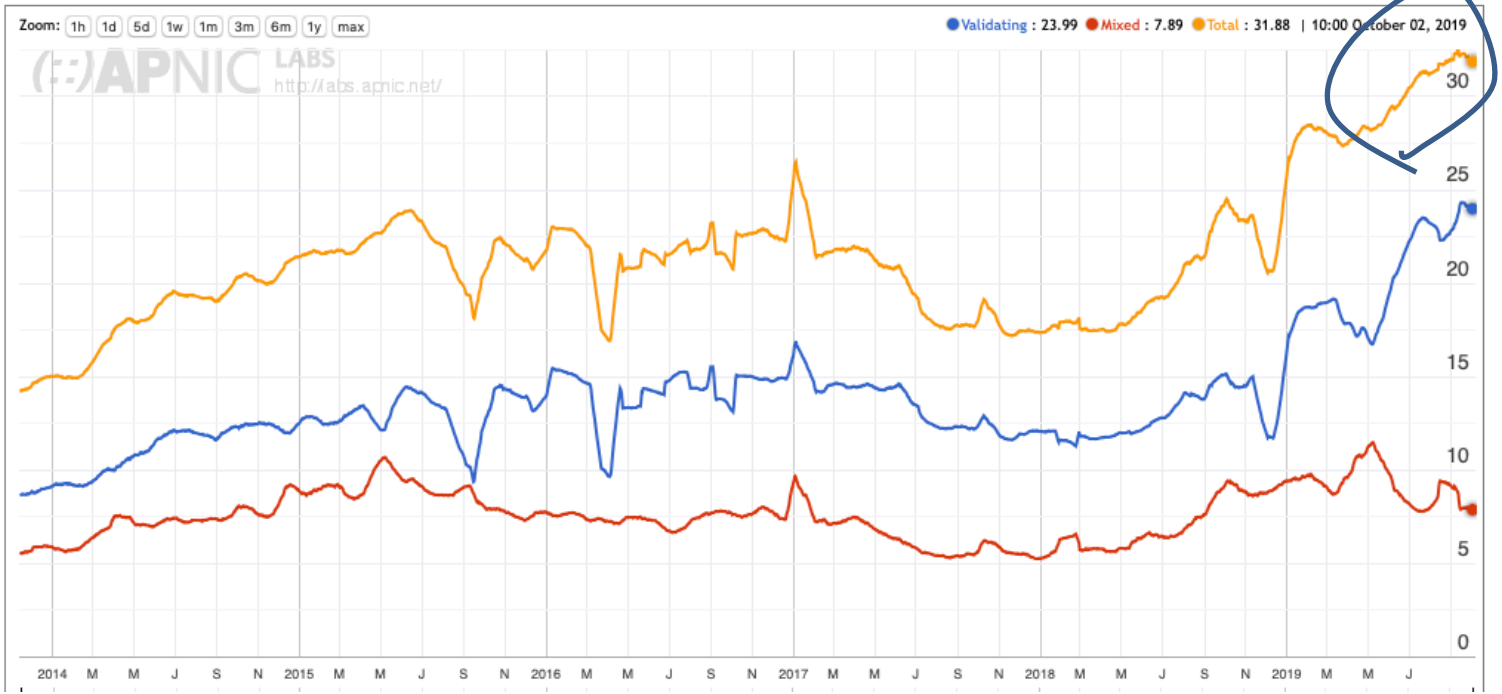


Predictions

- How many users sit behind DNSSEC validating resolvers?
- In this case the NSEC record is validly signed, so we are interested only in the user's "first choice" resolver

DNSSEC Validation Rates

Use of DNSSEC Validation for World (XA)



30% of users send queries to DNS resolvers that perform DNSSEC validation

https://stats.labs.apnic.net/dnssec/XA

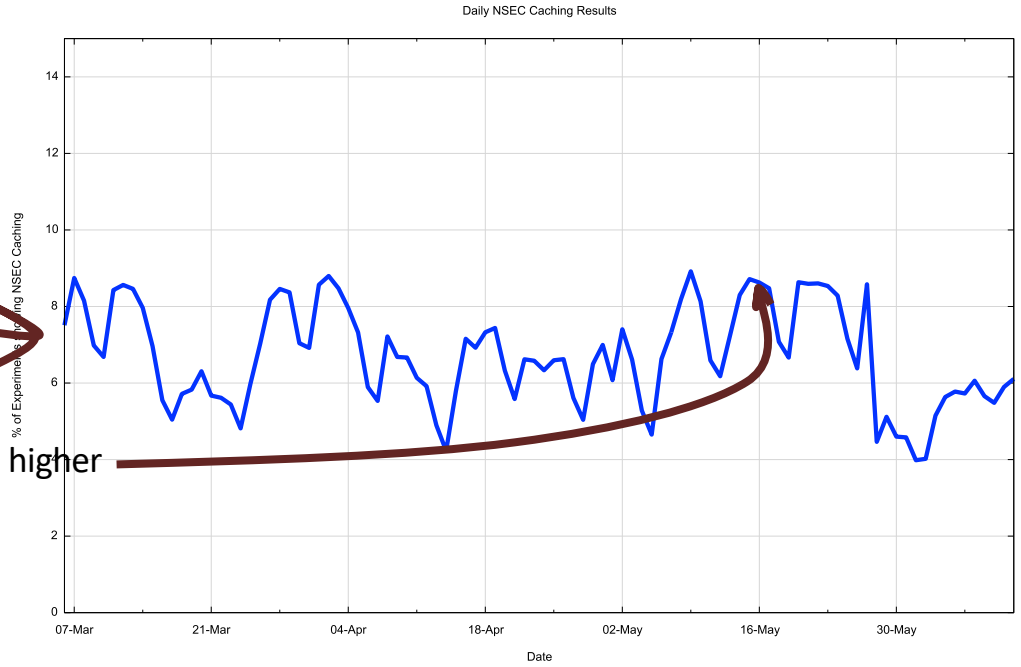
Predictions

- In up to 30% of these measurements we expect to see a query pattern consistent with DNSSEC validation being performed by the recursive resolver
- The maximum NSEC caching rate would therefore be visible for some 30% of users
  - While some resolver code has NSEC caching enabled by default other code sets have it as a configurable option
  - Somewhere between 0% and 30% of all measurements is the range of possible outcomes from this experiment

Results

7% average outcome

Weekdays tend to be higher



98 days, 266M experiments

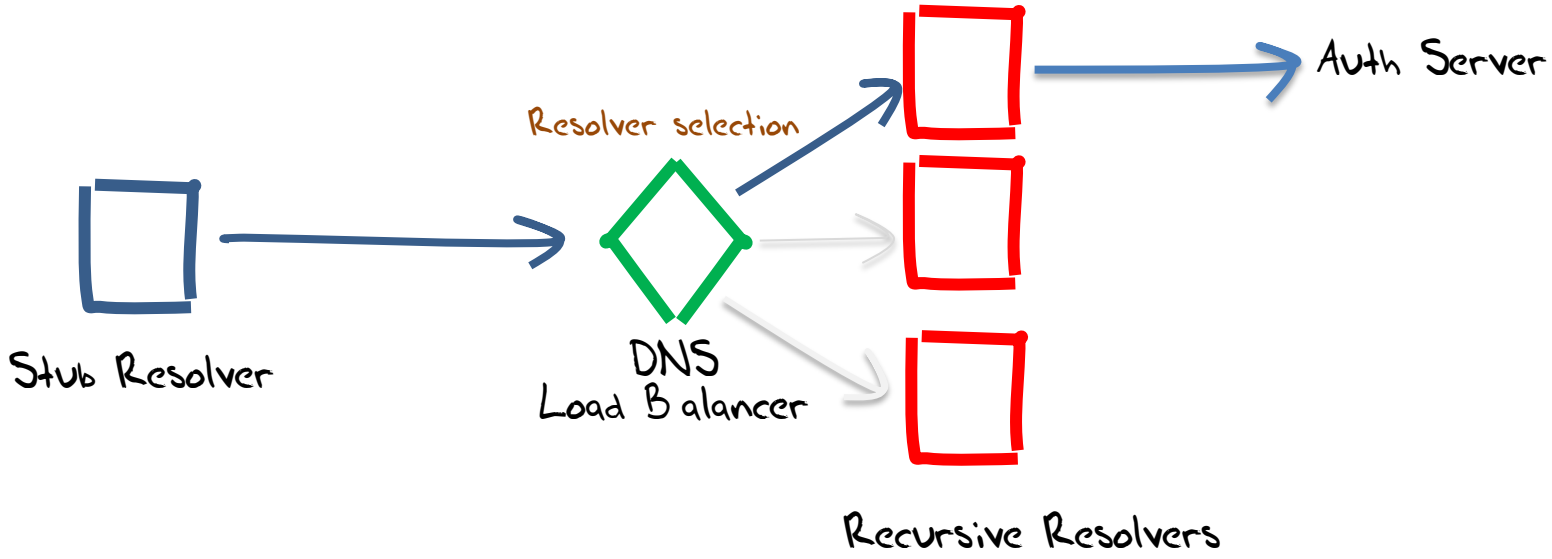
This is lower than we had anticipated

- And some open resolvers that we had thought were NSEC caching were showing variable behaviour
  - Sometimes they queried as if they were NSEC caching and other times not

What's going on?

DNS Load Balancing

The scenario of a stub resolver sending queries to a single instance of a DNS recursive resolver is being overshadowed by DNS load balancing scenarios



Measuring Load Balancing by IP addresses

- Over the 98 day period we observed **559,357** distinct IP addresses that queried the experiment's authoritative name servers with query names generated by the experiment
- If we group these addresses using a /24 (IPv4) and /48 (IPv6) subnet we observed **295,546** distinct subnets

- **More than one half of the visible recursive resolver set share a common subnet with other recursive resolvers**

Query Distribution in Resolver Farms

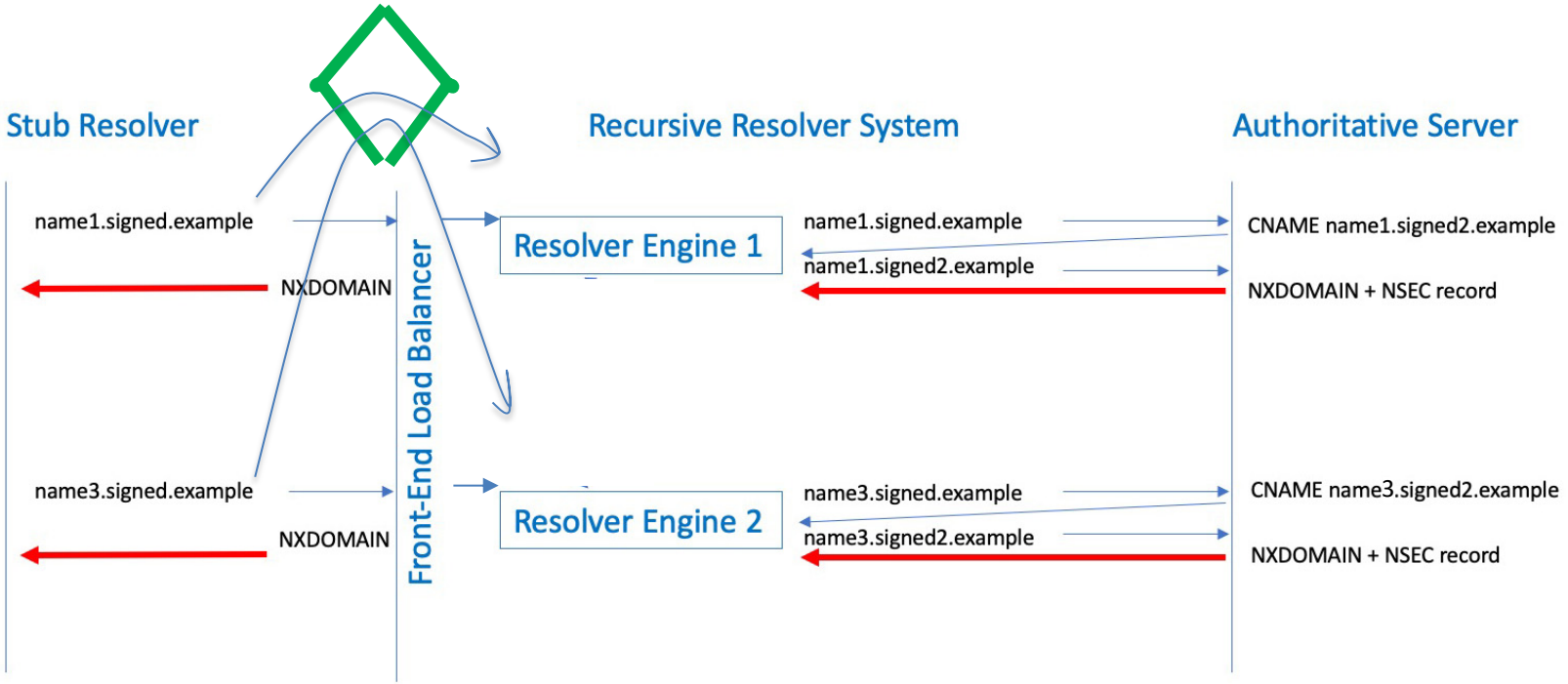
- How do you distribute DNS queries in a resolver farm and maximise cache performance?

  – Run some form of cache memory bus across the resolvers to share cached data across all members of the resolver farm

  or

  – Hash the query name so that the same resolver handles all queries for a given name

Query Name Hashing and NSEC caching



Does NSEC Caching Deliver?

- The results are not all that promising for conventional query traffic
- Despite a large number of recursive resolvers performing DNSSEC validation (and possibly even performing NSEC caching) the apparent widespread use of qname-hashing DNS load balancers works against NSEC caching

What about NSEC caching as a response to Random Subdomain Attack?

- It depends…
- If the attack query pattern is widely distributed then each recursive resolver may not experience sufficient query intensity to load all resolver farm members with the cached NSEC records

But

  - We didn't test this scenario using the ad platform using a very high query intensity as we don't have the capacity in this platform to operate at very high query loads over sustained periods

NSEC Caching?

- ☹ The recursive resolver needs to perform range checks against its help cache state - this may make lookups into the cache database slightly slower
- 😊 The cache does not contain individual NXDOMAIN entries for signed zones, so the cache efficiency increases with NSEC caching
- ☹ Commonly used DNS load balancers appear to spread query names across individual resolver engines and this appears to reduce the effectiveness of NSEC caching for the resolver farm as a whole

NSEC Caching?

Mostly Harmless!

- NSEC caching does not appear to be harmful to the DNS
- But in today's resolver environment the interplay between commonly used load balancers and queries for non-existent names tends to negate much of the potential benefit of NSEC caching

One more thing…

- The observed model of DNS operational deployment at scale is diverging from the classical stub-resolver-authoritative model that many still use as a reference
- Caching has long been a fundamental property of DNS but the current deployment model with extensive use of DNS load balancers alters aggregate cache behaviour
- Is it time to consider how DNS load balancers fit within the larger DNS architecture?

Standards?

- This is similar to the situation with NATs for many years
- The absence of standard specifications that describe how such units should behave mean that individual implementors are able to make their own choices
- And this can result in unexpected variance in behaviours for aspects of DNS resolution

Thanks!