Operational Experience with DNSSEC Signed Zones

Shumon Huque
February 16th 2023
DNS-OARC 40 Workshop
Atlanta, GA, USA

This presentation mentions the names of software implementations and vendors that have had some specific DNSSEC related bugs and/or operational inefficiencies. All of the issues presented have previously been described in conferences, public presentations, bug reports, and public/semi-public mailing lists and other online fora.

One of the goals of this talk is to make these issues known more widely and help other enterprises planning a large scale DNSSEC deployment to avoid many of the pitfalls we encountered.

Background

- OARC-28, March 2018: “DNSSEC for a Large Enterprise” (Pallavi Aras & Shumon Huque)
  - Described planning & testing for deploying DNSSEC
  - Main business Drivers: Fedramp, DOD IL4/5/6 Certifications for cloud provider services to the US Government; but more generally some large commercial customers also request it.
- Complex environment
  - Mix of inhouse DNS services, commercial on-premise DNS appliances, many different commercial DNS providers, etc.
  - Differing DNS requirements for many different types of application services.
  - Different units and subsidiaries involved in different parts of the DNS service.

Scope of Deployment for DNSSEC

- Salesforce Core CRM and its closely related services
  - Phased incremental deployment over a number of years (leaf zones on load balancers were signed last in some areas).
- More recently: other business units (Heroku, Slack, ..)
  - DNS operated by other teams, not the focus of this presentation.

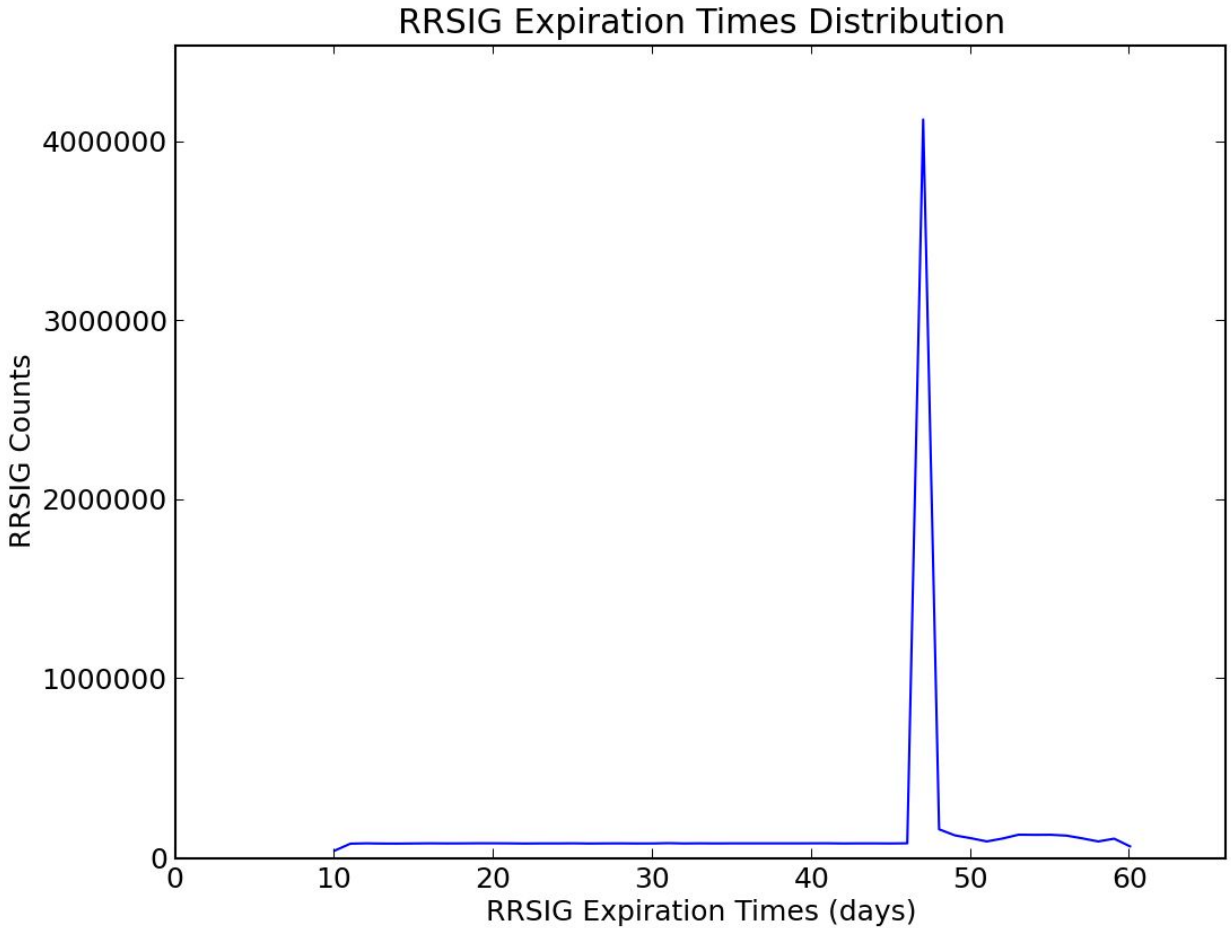
- Out of Scope: Corp IT managed services outsourced to 3rd Parties like corp website, mail services, etc.
  - DNS managed separately by Corporate IT teams.
  - No business imperative to deploy DNSSEC for these services.

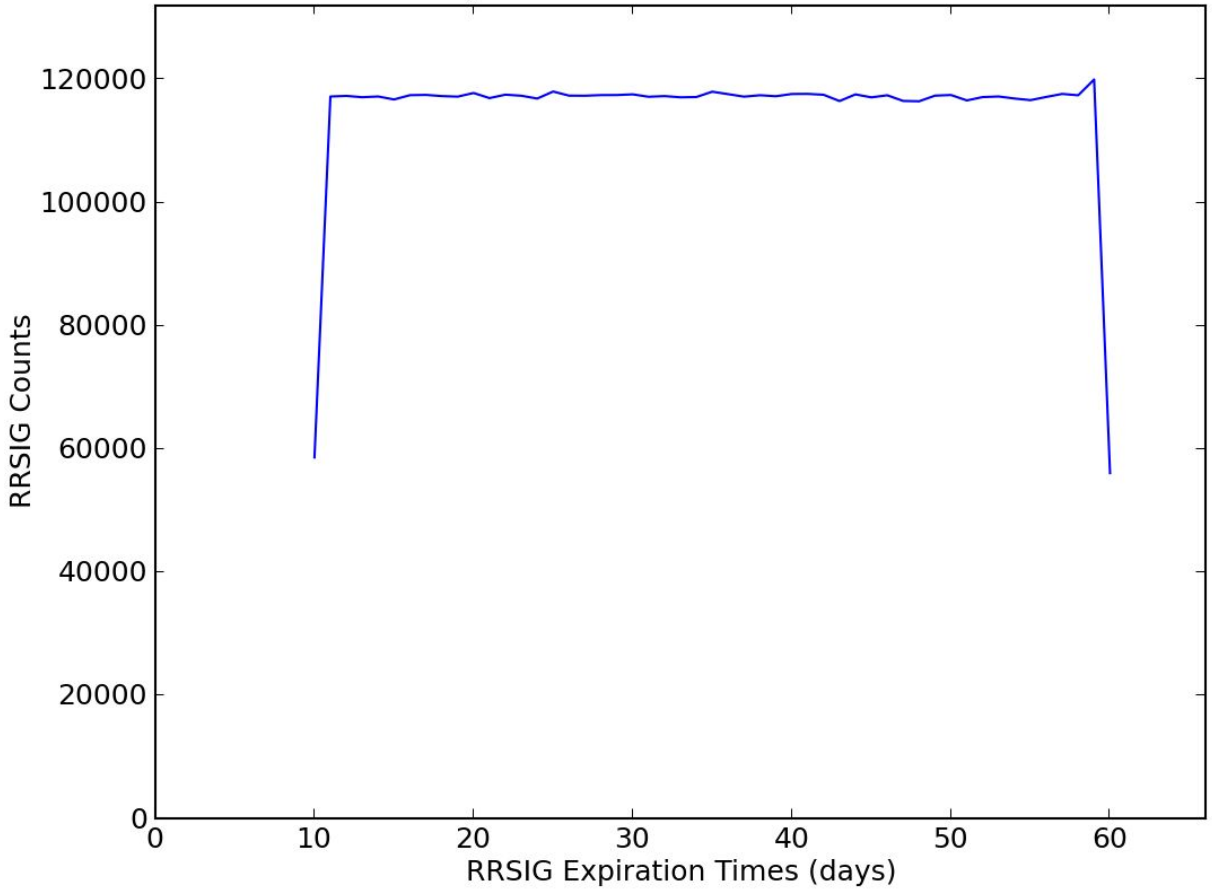
Zone Size Scaling Challenges

- Many of our zones are very large ~ 10 to 20 million records
  - Tenant/customer specific names; customer feature specific names, etc.
- Record count increases substantially with pre-computed signatures:
  - 1 NSEC/NSEC3 & corresponding RRSIG per name, and 1 RRSIG per data RRset.
  - Zone dominated by CNAMES: ~ 4x RR count, ~ 15-20x size increase.
- Limits the candidate list of viable DNS operators
  - A relatively small number of commercial DNS providers can support zones of our scale.
  - Even those took several rounds of performance tuning, infrastructure improvements, over many months, before we could deploy.

Managing Zone Re-signing Load

- With pre-computed signatures and large zones, we want **incremental signing, re-signing, & incremental zone transfer** (IXFR).
- Still end up with a large amount of records being resigned and corresponding updates being sent out.
- I**ncrease the sig validity window** (not ideal - larger window for replay attacks)
- To avoid large spikes, we also want to make sure initial signing of the zone **“jitters” the signature expirations** across the signature validity period.
  - BIND originally supported jitter only for the offline signer (dnssec-signzone’s “-j” option).
  - https://gitlab.isc.org/isc-projects/bind9/-/issues/418


RRSIG Expiration Times Distribution
RRSIG Counts
4000000
3000000
2000000
1000000
0
0
10
20
30
40
50
60
RRSIG Expiration Times (days)


RRSIG Counts
120000
100000
80000
60000
40000
20000
0
0
10
20
30
40
50
60
RRSIG Expiration Times (days)

Issues with data propagation at scale

- Some DNS **vendors have struggled with the update rates** generated by our large zones.
  - Steady state fine, but some specific types of activities (site switches, org/tenant cleanup related processes) spike up the update rates, and some vendors struggled to get those updates out to the edges of their network in a timely fashion.
  - Necessitating more rounds of gradual performance improvement.
- DNS vendor network visibility: **Looking Glasses**
  - But we need a more precise picture.
  - Many of our vendors have developed Looking Glass services for us. Allows us to obtain full list of active nodes, and direct queries to each individually.

Transition Large Zones to Wildcard + CDN model

- Long term goal: collapse all tenant specific records with **wildcards that point to a Content Delivery Network (CDN)**
- The CDN database then maintains the endpoint to backend origin server mapping - better scaling characteristics.
- This is partly in progress already. We have an **in-house CDN (Salesforce Edge network)** where subsets of customers have already migrated, and we are contemplating using 3rd party commercial CDNs to handle full scale.

Response Size Considerations

- DNSSEC increases response packet sizes.
- To decrease the potential of any issues, wherever we could, we used **Elliptic Curve signature algorithms** (ECDSAP256SHA256), which produce relatively small signatures.
- On some load balancer appliances that only supported **RSA**, we needed to **drop the ZSK key length to 1024-bits** (nothing in between 2048 and 1024 was supported) to keep negative responses from exceeding the Internet path MTU
    - Not ideal, trying to get vendor to implement Elliptic Curve signatures.
- **Cap UDP message size** to 14xx octets, and **truncate** & force TCP beyond.

Response Size - issues observed

- Very few
- One customer case: “reports of unresolvability of our domains”
  - A few calls with the customer identified the problem, which they promptly fixed.
  - Resolvers were asking for DNSSEC signatures (setting DNSSEC_OK bit)
  - However, their **firewall positioned between their resolver and the outside world, restricted DNS packets to 512 octets** (sigh) and blocked responses from us.
  - (It is unclear if their resolvers were actually doing DNSSEC validation, or whether their default behavior was to set DO=1)
- Solution: fix firewall!

DNSSEC Implementation Bugs

- **Most bugs encountered were related to negative responses** (“Authenticated Denial of Existence”)
- Authoritative Server side (many cases):
  - Incorrect NSEC/NSEC3 proofs (incorrect records, missing records, incorrect type bitmaps)
- Resolver side (1 case):
  - Inability to authenticate certain complex types of valid negative responses, leading to SERVFAIL

F5 Type Bitmap Bug - episode 1 (July 2019)

- Load Balancer returning **incorrect NSEC3 type bitmap for NODATA** responses (only a static TXT RR type)
- Implementation does on-the-fly signing, and is unable to efficiently obtain the exact list of types located at the name.
- Worked fine until the emergence of resolvers that implemented aggressive negative caching (RFC 8198) with type inference.
- **Fix**: vendor introduced **2 configurable parameters to allow operators to explicitly configure the static set of types for the zone apex and names under the zone apex**.
- See also
  - Neda Kianpour's talk at RIPE79
  - F5 Knowledge Base doc

F5 Type Bitmap Bug - episode 2 (October 2021)

- Intermittent incorrect NSEC3 type bitmap in responses; only for NXDOMAIN responses (P. Van Dijk brought this to my attention)
  - **NSEC3 record that matches the closest encloser had an empty type bitmap!**
  - Subsequent queries for the closest encloser (which had valid address records) would be answered negatively from the aggressive cache.
  - Note that direct NODATA queries for closest encloser yielded the correct type bitmap!
- **Workaround**: ISPs affected deployed **NTA (Negative Trust Anchors)** for Salesforce :(
- Fix by vendor in ~3 months, but deployed by us in May 2022 (7 months later!)
  - Budget plenty of time for vendor fix; and then for your network ops team to roll out the fix.

eu38-fra.fra.r.force.com has an A record
foo.eu32-fra.fra.r.force.com does not exist
but NXDOMAIN response included proof that former A record didn't exist!!

```
$ dig +dnssec @ns1-fra.salesforce.com. foo.eu38-fra.fra.r.force.com
;; AUTHORITY SECTION (RRSIGs omitted for brevity)
[...]
j53bhf4tahtb817l901tv0cprees1t45.fra.r.force.com. 30 IN    NSEC3 1 0 1
E7E5875A85368D3F J53BHF4TAHTB817L901TV0CPREES1T46
gi238gloo9o29idbi2ufp7r54c52stba.fra.r.force.com. 30 IN    NSEC3 1 0 1
E7E5875A85368D3F GI238GLOO9O29IDBI2UFP7R54C52STBC
4o95decpsr3gcl6prpdanvkbunuhi2gm.fra.r.force.com. 30 IN    NSEC3 1 0 1
E7E5875A85368D3F 4O95DECPSR3GCL6PRPDANVKBUNUHI2GO
```

empty type bitmap at closest encloser, denying existence of A record.

NSEC3 math for reference

```
$ nsec3hash E7E5875A85368D3F 1 1 eu38-fra.fra.r.force.com (Closest Encloser)
J53BHF4TAHTB817L901TV0CPREES1T45
$ nsec3hash E7E5875A85368D3F 1 1 foo.eu38-fra.fra.r.force.com (Next Closer Name)
GI238GLOO9O29IDBI2UFP7R54C52STBB
$ nsec3hash E7E5875A85368D3F 1 1 *.eu38-fra.fra.r.force.com (Wildcard @Closest Enc)
4O95DECPSR3GCL6PRPDANVKBUNUHI2GN
```

Knot DNS missing NSEC3 record (May 2021)

- PCH introduced Knot into their footprint (in addition to NSD)
  - In order to improve propagation performance, but within a day, reports of problems from some high profile customers.
- Failing customer names matched a wildcard CNAME, pointing to another CNAME, which pointed to a name in an unsigned zone below it.
  - The signed response for this is a bit complex:
    - RRSIGs for the wildcard and explicit CNAMEs
    - **NSEC3 record that covers the hash of the next closer name (wildcard no closer match proof)**
    - NSEC3 record that matches the delegated zone, demonstrating that no DS record exists
- Knot failed to include the first NSEC3 record (and its RRSIG) in the response, although it was actually present in the zone.

(Notes: actual customer name redacted to “cust123”; RRSIGS excluded for brevity)

```
;; ANSWER SECTION:
cust123--c.na113.visual.force.com. 300 IN CNAME na113.force.com.    → wildcard match
na113.force.com.                   300 IN CNAME na113-ia2.force.com.
na113-ia2.force.com.               300 IN CNAME na113-ia2.ia2.r.force.com.

;; AUTHORITY SECTION:
85bjl8pbq0jo98rbcpluocjd2vkrnqeu.force.com. 10 IN NSEC3 1 0 0 C7EB9E0E
85BN5GQGVIKHI7AS5GVSQGVBPEEUDNF1 CNAME RRSIG
ia2.r.force.com.          86400    IN      NS    ns1-ia2.salesforce.com.
ht1dh0gh3d875rdaofl3ch29jqtf8ohg.force.com. 10 IN NSEC3 1 0 0 C7EB9E0E
HT1H5C9J2NLJMHCOUOH65AARSVPOG875 NS
```

Missing!

Ok. (insecure delegation)

---

nsec3 math for next closer name than wildcard:

```
  $ nsec3hash <salt> 1 0 cust123--c.na113.visual.force.com
  85BLBGTAEICVE7PFDQNMGSA3CBCPOOL2 (salt=<salt>, hash=1, iterations=0)
```

(Note: the leaf zone has since been signed, so the 2nd NSEC3 is now replaced by a DS record)

Resolver behavior mystery?

- But so what? The provider had another implementation, and we had a 2nd provider (Neustar XTLD) running a 3rd implementation (BIND)
- Shouldn’t resolvers be robust to this kind of unvalidatable answer, and re-query other available authoritative servers for the zone, until they get a good answer?
  - They should, and all the resolvers we tested had this behavior.
  - **And yet, some of our customers are using resolvers that are seemingly not robust to this situation and permanently fail when encountering such errors.**

Mitigation

- This zone had 2 providers.
- FIX 1: Temporarily remove the problematic provider from the NS set.
- (worked, modulo TTLs etc)

- But a few days later, the single remaining provider was under DDoS attack.
- FIX 2: Asked PCH to remove the buggy software implementation from their footprint; then re-instated PCH into the NS set.

(Nameserver change agility needed. See later slide)

NSD omits required NSEC3 record for wildcard NODATA

- After the Knot incident, we learned that PCH had already seen a similar issue the previous year with NSD (~October 2020) that was promptly fixed at that time (but they didn’t notify us).
- Missing closest encloser NSEC3 record for wildcard NODATA type.

Google Public DNS SERVFAIL; October 2019

- Google unable to authenticate some Salesforce DNS responses.
- A customer facing record involving **a CNAME chain that traversed multiple zones, involving a wildcard, and an insecure delegation to another zone**.
  - Resulting response contained 2 NSEC3 records from different zones proving 2 different facts (wildcard no closer match & insecure delegation).
  - Our response was entirely correct; but Google had a validation bug.
- Interim solution:
  - Found an alternative URL path for some affected customers to use that didn't traverse wildcards, and thus wouldn't trigger the google bug.
  - Instructed other customers to use a different resolver.
- Bug fixed by Google in late November 2019, rolled out in early 2020.
  - https://partnerissuetracker.corp.google.com/issues/143165170?pli=1

Slack DNSSEC outage (mostly skip)

- September 30th 2021
- https://slack.engineering/what-happened-during-slacks-dnssec-rollout/
- Also presented at OARC36 (November 2021)

- Cloud DNS provider (Route53) had a critical bug in their wildcard NODATA responses - the NSEC type bitmap was incorrect.

Slack DNSSEC outage

*.slack.com has a valid wildcard A record, but no AAAA record

"A" record type missing!

```
$ dig app.slack.com. AAAA

;; AUTHORITY SECTION
app.slack.com.      300     IN   NSEC\000.app.slack.com. NSEC RRSIG
app.slack.com.      300     IN   RRSIG NSEC [ … omitted … ]
```

Resolvers that do aggressive negative caching will conclude that app.slack.com has no A record and return NODATA responses for subsequent queries for A records for the cached period of the NSEC record. (Note: the query sequence of AAAA followed by A is common for many dual stack systems)

Is it really a wildcard though? *(minutiae for DNS protocol geeks)*

- Previous response doesn’t look like a wildcard NODATA response.
- For many online signers, the DNS wildcard is an internal provisioning instruction in the zone database, and not a protocol element exposed in the wire response. They pretend the name that matched the wildcard explicitly existed in the zone and provide DNSSEC proofs for an explicit match.
    - This behavior actually had a beneficial effect in this case, because it likely reduced the scope of the damage caused by the bug for aggressive cachers.
    - The effect of the incorrect bitmap was on the basis of each specific matched record, not on _any_ name that matched the wildcard.

Why are there so many bugs in negative responses?

- These were fairly fundamental bugs (in my view).
- Yet have plagued implementations almost across the board.
- Insufficiently developed test suites for all possible permutations of negative response types?
- Composition of different types of negative assertions in the same response confusing validators?
- New behavior in resolvers (ANC) that triggered existing bugs in type bitmaps.
  - [See also OARC36 talk on NSEC type bitmap bugs - P. Spacek]
  - This feature has caused a lot of damage.
  - Should this have needed a “Flag Day”, and a prerequisite study of ecosystem wide impacts, before rollout in the field?

What could help?

- Most important: complete DNSSEC test suites by implementers.
- Enhance Nameserver checking systems to include DNSSEC
  - Automatically Find RFC Compliance Bugs in Nameservers
- Note: static zone file analysis won't catch many of these errors, because:
  - Online Signers!
  - NSEC/3 record may be present in zone, but not returned correctly responses - so only query/response interrogation will identify such issues.
- DNSViz can catch many of these errors (if they exist at the apex or well known locations), but only *if* you turn on Denial of Existence checking
  - That checking should be the default (I've communicated this suggestion to Casey Deccio, and he's working on it)

Nameserver/Provider change agility

- Several issues were worked around by having multiple DNS providers and temporarily withdrawing the services of malfunctioning provider.
- Challenge at 2LD level: Long, unchangeable NS TTLs at the TLDs.
- Resolvers should lock on to child zone NS instead (ns-revalidation)
- Longer term: TLDs should offer ways for registrants to deploy much shorter delegation TTLs
  - May need EPP TTL extension
- Similarly, DS records also need low TTL agility.
- Get resolvers to cap all TTLs to a much lower value?

Parting Thoughts & Advice

- Even if your DNS operations are outsourced to 3rd parties, you need to have substantial DNSSEC expertise on your staff.
  - because you will often be exposing, and debugging problems and bugs in their implementations.
- Knowing key DNS folks at other companies and open source projects has helped tremendously.
- Pay acute attention to testing of denial of existence.
- Don’t use DNS zones as massive application tenant configuration databases.
- Use multiple DNS providers where you can.
  - Ask online signers to implement Multi-Signer DNSSEC (RFC 8901)

Appendix: Extra Slides

DNSSEC Signing Methods

Two General Methods - we employ both, depending on technical feature set needed by the zone. They have pros and cons (see Appendix for more detail)

1. Pre-computed Signatures
    a. Sign zone data with hidden master (ISC BIND) and Zone Transfer out to DNS provider(s).
2. Online Signing
    a. DNS Provider signs responses on-the-fly.

Pre-computed Signatures

- Pros:
  - More secure, because only we hold the signing keys, and they can be kept either offline, or on systems not exposed to the general Internet. DNS provider cannot send out bogus responses without detection by validating resolvers.
  - Can support multiple distinct DNS providers easily.
- Cons:
  - Zone size scaling issues much more acute
  - Cannot support non-standard features (GSLB, and other types of dynamic responses)
- Where used:
  - Zones that don’t use non-standard features & where we must have multiple providers.

Online Signing

- Pros:
  - Better scalability with respect to zone sizes.
  - Can support non-standard features, e.g. GSLB, by dynamically signing their response data.
- Cons:
  - Less secure (for us), since we don't hold the signing keys. A compromise of any of the providers edge servers (which hold the ZSK private keys) can lead to false responses.
  - Needs "Multi-Signer DNSSEC (RFC 8901)" to support multiple providers - very few providers have implemented that to date, and there isn't a lot of interop testing yet to be confident that it will work flawlessly in the field.
- Where used:
  - Zones with non-standard features or other requirements to be on an online signing provider.

Ballpark hypothetical estimate of “re-signing” activity

Large zones and pre-computed signatures can result in an extremely large amount of near constant zone maintenance activity.

```
#records = 20,000,000
sig_validity_interval = 15 (days) w/ re-signing at 3/4th of window (11.25)

If signature expirations are spread out uniformly ..
    resignings_per_day = records / resigning_interval = 1,777,777.8
If say 15 records are procured for each IXFR message (exact number depends on zone
structure, quantum, & sig_signing_nodes …
    ixfr_messages per day = resignings_per_day/ 15 = 118,518.5
    ixfr / second = _ / 86400 = 1.37
Then multiply by number of outbound XFR clients.

Notify throttles and record coalescing can reduce these some, but the general point is,
there is a lot of zone maintenance activity, apart from actual data updates to the zone.
```

Unsigned leaf zone tricks

- Company is slowly shifting much of its infra from 1st party datacenters to large IaaS cloud providers
- Leaf zone architecture changing from on-premise commercial load balancers that do support DNSSEC, to cloud provider offered Network Load Balancers residing in DNS zones that aren’t signed.
- Trick: use signed ‘ALIAS’ records to hide the unsigned (target) path from view of the resolver.
- Security gap in name lookup process:
  - Cloud providers should sign these load balancer endpoint zones
  - Auth servers should validate ALIAS target lookups to secure the hidden path if it is signed (they by and large don’t today)

Other tidbits (ask me later)

- Unexpected signed wildcard TTL behavior in BIND resolver.
- NS1 custom filters in large zones: to collapse sets of records pointing to the same endpoint that share a specific pattern.
- Getting online signers to use separate signing keys for us.