

Beyond Loop Limits

CNAME and Delegation Loop Handling in OpenDNSCache

Dejan Donin
Cisco DNS Security

Dipa Thakkar
Cisco DNS Security

Sean Mooney
Cisco DNS Security



Stopping a loop isn't the hard part

The question is how quickly, at what upstream cost, and whether the next client pays again

- CNAME and delegation loops are old, well-understood failure modes
- Recursive resolvers terminate them – none loops forever
- What actually matters:
 - how quickly the loop is recognised
 - how much upstream work is spent before giving up
 - whether equivalent work is suppressed for the next query
 - whether operators can see why the SERVFAIL happened

Bounded failure $\neq$ early, explainable, efficiently suppressed failure

Four controls that bound a loop

Each catches a different shape; none of them covers everything on its own

1

Visited-name detection

Stop when a CNAME target
repeats

Early, exact – but only for what it
tracks

2

Alias limit

Cap CNAMEs followed per
query

Catches long chains, loops or not

3

Work / depth budget

Cap recursion levels and
total steps

Always terminates – after
spending the budget

4

Failure cache + telemetry

RFC 9520: remember the
failure briefly, count why

Stops the next query paying again

Loop shapes we studied

Loop shapes covered by our regression tests and code review



Which loop shapes our counters see

response.loop.* sees CNAME loops; only the failure cache sees delegation loops

Loop shape (slide 4)	response.loop.*	Failure cache: first query	Consecutive Queries – Use of failure cache (within 5 s)
CNAME self-loop	✓ cname-direct	— not cached answered NOERROR	pays again
CNAME ring across 3 zones	✓ cname-direct	— not cached answered NOERROR	pays again
Mutual glueless delegation	— none	✓ set	hit, 0 upstream
Delegation ring (3 zones)	— none	✓ set client's zone only	hit, 0 upstream
Fan-out (4 glueless NS)	work-exhausted only if budget runs out	✓ set	hit, 0 upstream
NS names that don't exist	— none	✓ set	hit
Lame: only server REFUSES	— none	— not cached	pays again
CNAME into a loop	— none no name repeats	✓ set (for zx)	hit

Each loop shape shows up in exactly one place: the loop counters or the failure cache, never both. Some shapes show up in neither. Our alias limit counter only fires inside nameserver lookups, so client-side chains past 16 CNAMEs are stopped but not counted.

Experiment Methodology

Real traffic for frequency and cache effect; a controlled lab for loop shapes

Mirrored traffic

- Lab resolver instance receiving mirrored traffic
- Resolvers with the failure cache and loop counters
- Counters only – responses unchanged with cache off
- Failure cache ON, then OFF
- All rates normalized per million responses

Top-1M replay (lab)

- Lab resolver replaying the public Cisco Umbrella Top 1M list
- Queries go to the real authoritative servers on the internet
- 2 × 14 min: failure cache OFF, then ON
- Upstream measured at the resolver: authoritative queries per client query

RFC 9520 failure cache

Remember "this delegation has no usable nameserver" for 5 seconds

Design

- Keyed by zone name – siblings share the failure
- Cached: no NS name resolves, or every server lame
- Fixed 5 s TTL (RFC 9520: ≥ 1 s, ≤ 5 min recommended)
- Hit → SERVFAIL with no upstream work – same answer as without the cache
- Off by default; option `failure_cache_enabled`

Deliberately not cached

Outcomes that depend on this query, not the zone:

- client timeout
- work budget already exhausted
- forwarded or RD=0 queries
- QNAME-minimized pass that will be retried

REFUSED / SERVFAIL from every server: not failure-cached;
Failing servers are tracked per address in the existing SERVFAIL cache

Failure cache in two experiments

Cache ON vs OFF in matched windows · per million responses

Traffic	Mirrored	Mirrored	Top-1M	Top-1M
Failure cache	ON	OFF	ON	OFF
Failure-cache hits	6,249	0	138	0
Zones cached (sets)	570	0	84	0
Hits per cached zone	≈ 11	—	≈ 1.7	—
SERVFAILs served from cache	35 %	—	2.9 %	—
SERVFAIL responses	17,870	17,230	4,705	4,803
Upstream per client query	0.233	0.234	0.43 *	0.51 *

Mirrored traffic: 2 × 24 h, ON then OFF. Top-1M: replay of the public Cisco Umbrella Top 1M list, 2 × 14 min, OFF then ON.

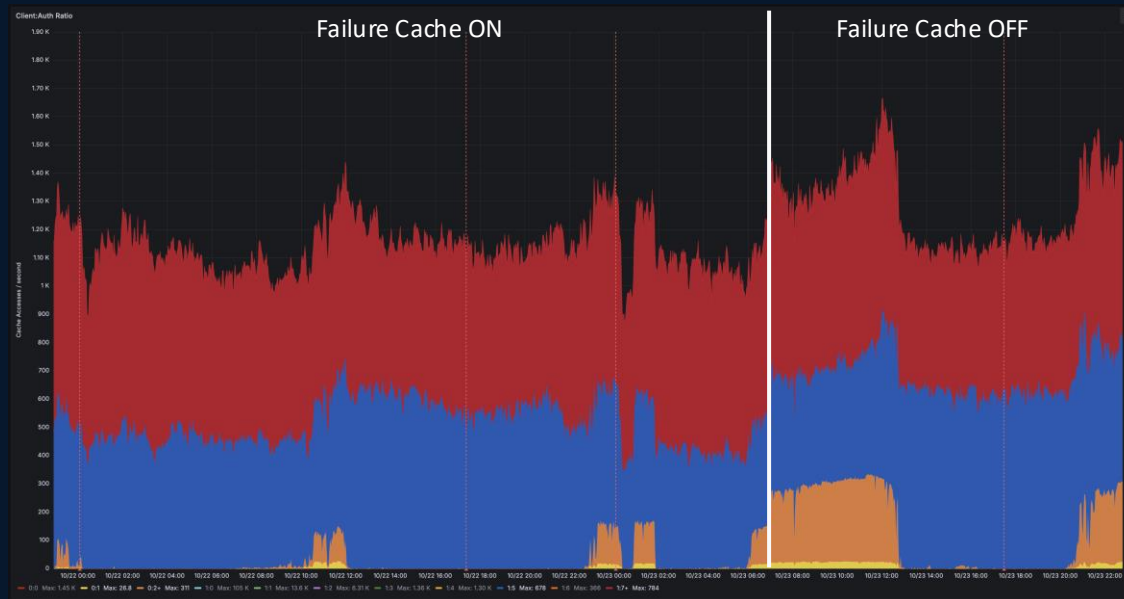
* Top-1M upstream falls through the run as the cache warms, so ON vs OFF is not comparable.

What it says

- Benefit scales with repetition: real traffic re-asks a failed zone ~11× within 5 s; a once-through list barely does
- Same SERVFAIL rate – the cache changes costs, not answers

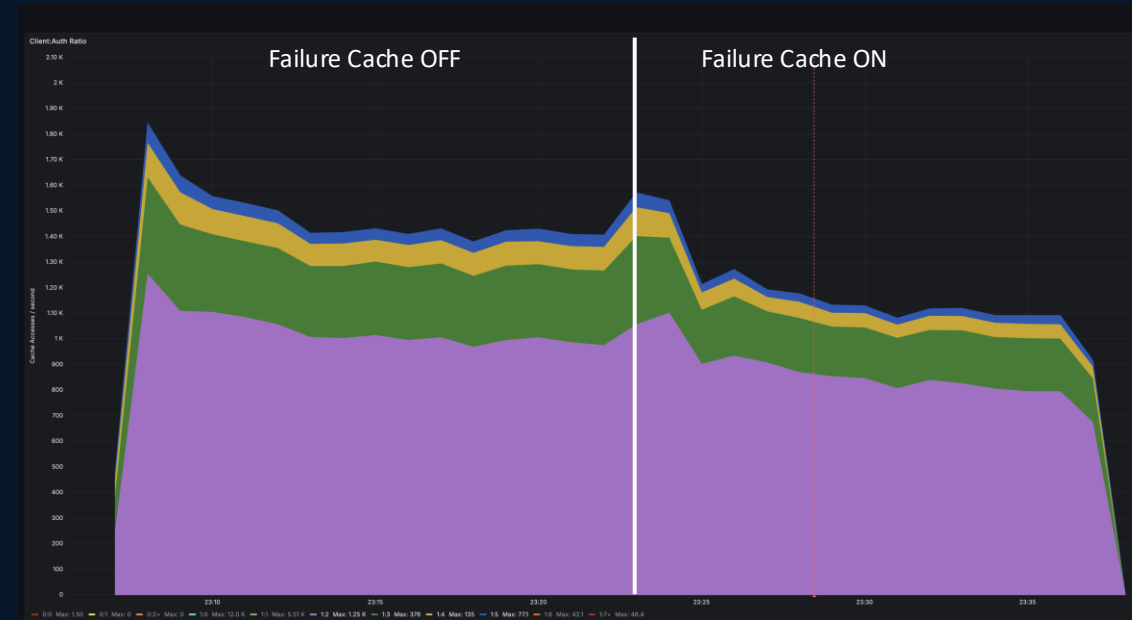
Client:Auth ratio in both experiments

- Left: Mirrored traffic
- Right: top-1M replay



Mirrored traffic · 10-22 00:00 – 10-23 23:00 UTC; cache off from 10-23 06:57

Shown: 0:1, 0:2+, 1:5, 1:7+. 1:5 and 1:7+ follow the traffic level. 0:2+ bursts appear with the cache on (10-22 10:30–12:30, 23:00–01:30) and off (10-23 06:00–12:30, from 20:30), so they don't track the switch.



Top-1M replay · 23:05–23:38 UTC; cache on from 23:22

Shown: 1:2–1:5. The total falls by about a quarter through the run as the cache warms; the drop at 23:24–23:25 is too close to the 23:22 switch to separate the two.

Loop counters don't move with the cache enablement

Events per million responses, both experiments

Counter	Traffic Mirrored	Traffic Mirrored	Traffic Top-1M	Traffic Top-1M
	Failure cache ON	Failure cache OFF	Failure cache OFF	Failure cache ON
cname-direct	14.1	13.8	15.3	19.4
work-exhausted	0.60	0.73	0	10.7
alias-exhausted	0.003	0.002	0	0

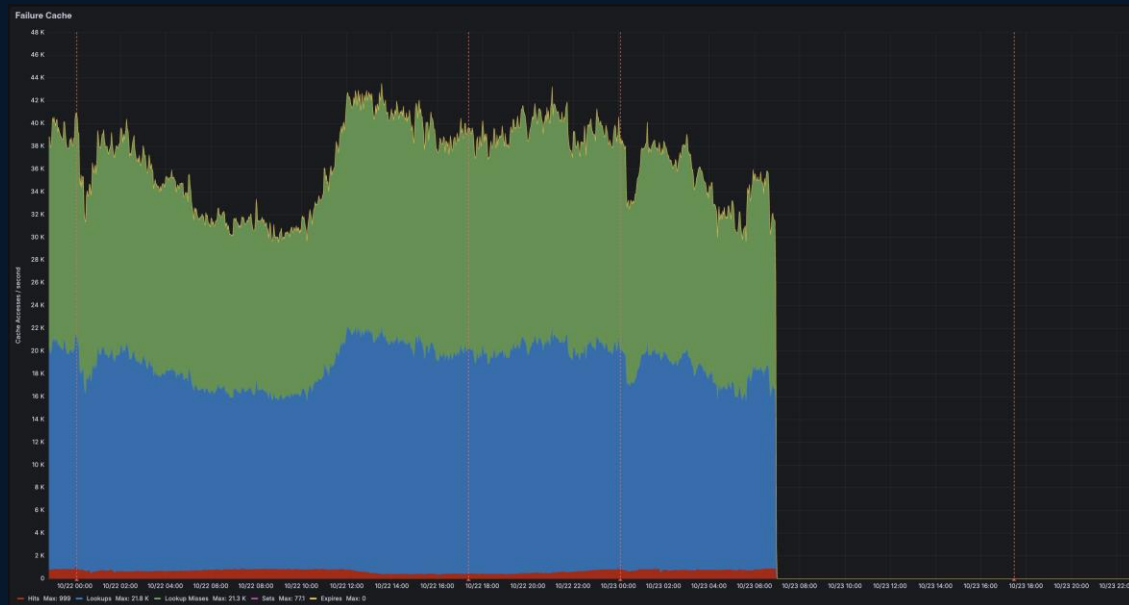
Mirrored traffic: 2 × 24 h. Top-1M: 2 × 14 min – small counts, so treat the top-1M columns as indicative. Counters count events, not responses: one client query can raise work-exhausted once per level that gives up.

What it says

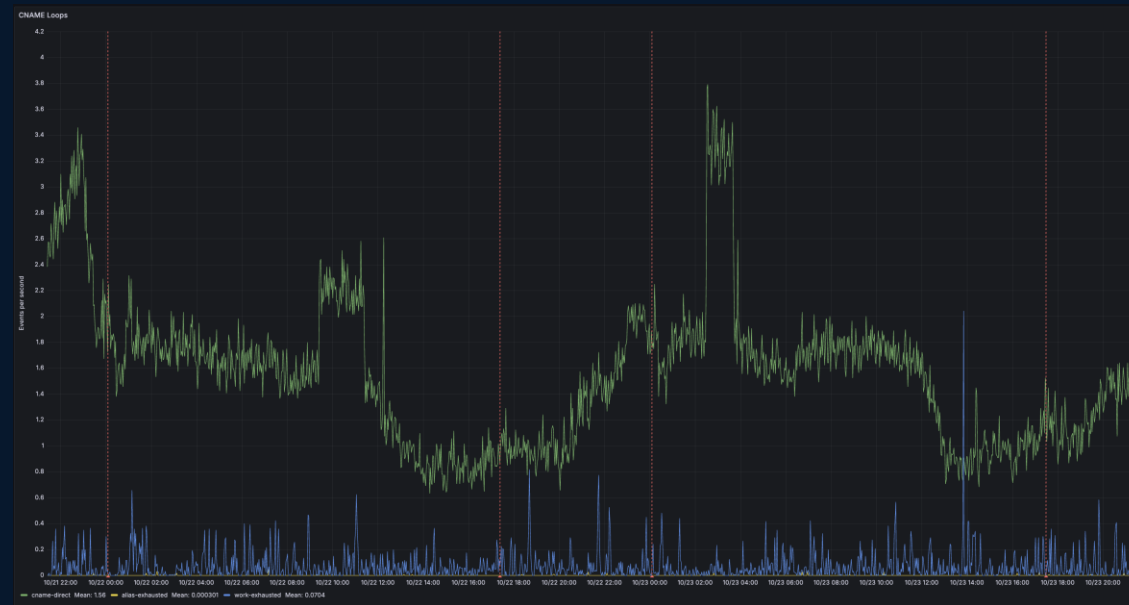
- CNAME loops: a steady ~14–19 per million in both traffic mixes, cache on or off – by design they are answered, never cached
- Budget exhaustion is rare (< 1 per million on mirrored traffic); the top-1M burst at 23:34–23:36 follows list position, not the cache
- alias-exhausted ≈ never: it only counts chains met while resolving an NS address
- On mirrored traffic, CNAME loops are ~20× more frequent than budget exhaustion

Mirrored traffic: failure cache and loop counters

Mirrored traffic, 10-22 – 10-23; cache switched off 10-23 06:56:50 UTC



Failure cache: lookups and hits drop to zero when the cache is switched off at 06:57.



Loop counters: cname-direct level shifts follow traffic, not the switch. Red dashed lines are dashboard annotations.

Plan

- Keep all four controls – they are complementary, not alternatives
- Implement RFC 9520 failure caching keyed by zone, short TTL, excluding query-specific outcomes
- Count loop and limit events separately from SERVFAIL
 - e.g. cname-direct, alias-exhausted, work-exhausted, failure-cache hit
- Proposal: shared counter names and Extended DNS Error mapping for loop / unusable-delegation failures
- Measure before acting: ship counters with feature disabled first, then enable behavior

Takeaways

- 1 Stopping a loop is not the same as stopping its cost
- 2 On real traffic, repeated work across queries matters more than detecting the loop
- 3 A cheap zone-keyed failure cache closes most of the gap

Questions? • ddonin@cisco.com • dthakkar@cisco.com • seamoone@cisco.com

Backup

OpenDNSCache limits

B2 · The four controls in OpenDNSCache

Values in the tested build

Control	OpenDNSCache	Outcome when it fires
Visited-name CNAME detection	Stops when a CNAME target repeats in the chain	Answer returned as-is (NOERROR)
Alias limit	16 CNAMEs per query	Lookup abandoned
Recursion depth	6 levels of dependent lookups	Lookup abandoned
Total work budget	356 steps, minus 3 per level	SERVFAIL
Failure cache (RFC 9520)	Zone-keyed, 5 s, opt-in	SERVFAIL, no upstream work
Failure-cause telemetry	response.loop.*, cache.failure.*	Counted; response unchanged

What didn't work: signature-based loop detection

Our first design tried to catch every cycle as it happened

The idea

Track (level, state, name) signatures across CNAME, delegation and NS-address lookups

SERVFAIL as soon as one repeats

What went wrong

Legitimate revisits look like loops:

- forwarder fallback
- skipped-zone recovery

→ **SERVFAIL for good traffic**

The lesson

Existing limits already bound the work inside one query

The waste is the same doomed work repeated by every query

That's a caching problem, not a detection problem