Evaluation of anti-DDoS features in full-service resolvers

Yoshitaka Aharen

JPRS

Introduction

- JPRS operates a TLD ".jprs" as an R&D platform
  - https://tldlabs.jprs/en/
  - "TLD Anycast DNS Serves to ISPs" (APRICOT 2017)
    - https://2017.apricot.net/program/schedule/#/day/9/network-operations-2
  - "Deployment of TLD DNS Anycast node to ISPs for stability and resiliency" (APRICOT 2021)
    - https://2021.apricot.net/program/schedule-conference/#/day/10/network-operations
- We conducted an experiment of anti-DDoS functionalities implemented in full-service resolver implementations with Japanese domestic ISPs
- I am going to show the results and some findings

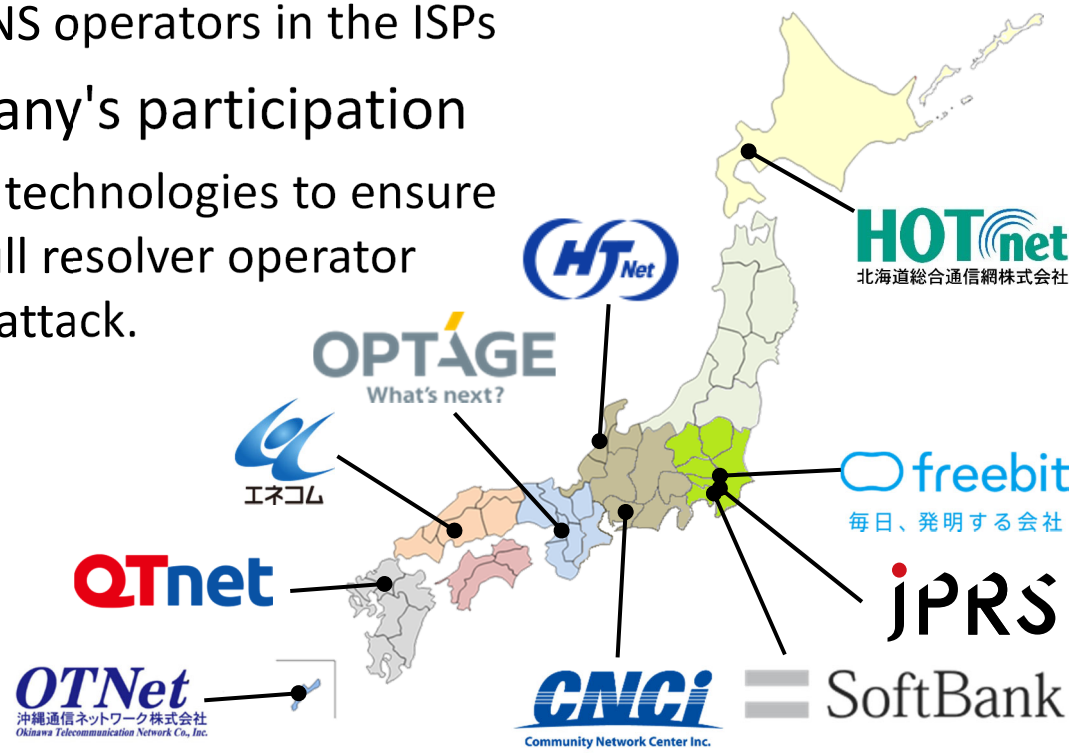
Purpose of the evaluation

- Evaluate anti-DDoS functionalities with DNS operators of the ISPs
  - Try the functionalities in an evaluation environment
  - Give some feedbacks to software implementers (if possible)

- What we tried:
  - BIND 9 (BIND 9.14.5) and Unbound (Unbound 1.9.3)
  - fetch-limit (BIND 9) / ratelimit (Unbound)
  - serve-stale (BIND 9) / serve-expired (Unbound)
- What we didn't try:
  - NSEC aggressive use: .jprs is signed with NSEC3 opt-out
  - DNS Cookies: update to RFC 7873 was ongoing in dnsop

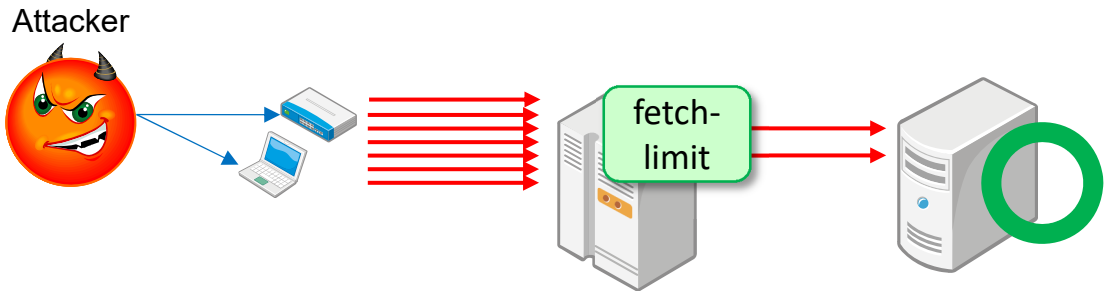
Participants

- Joint research effort with 9 domestic ISPs
  - Regional and nation-wide ISPs: CNCI, ENECOM, Freebit, HOTnet, HTNet, OPTAGE, OTNet, Qtnet and Softbank
  - They operate full-service resolvers for their customers
  - We cooperated with DNS operators in the ISPs
- Purpose of each company's participation
  - Verify countermeasure technologies to ensure stable operation as a full resolver operator in the event of a cyber attack.



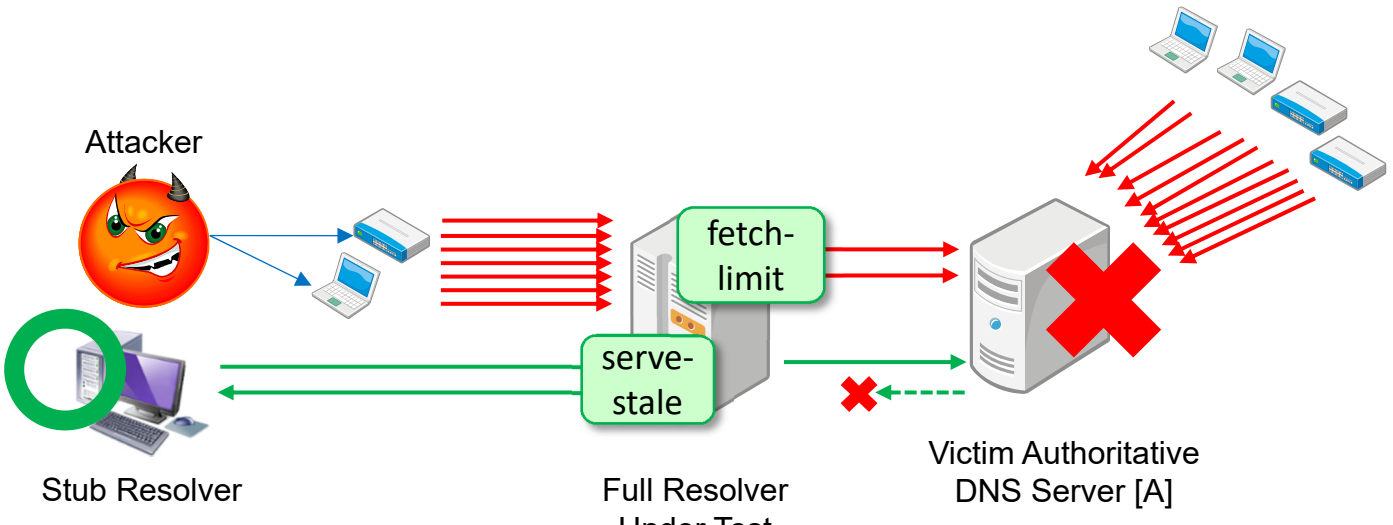
Main evaluation scenario (1)

- Set up a full-service resolver with both fetch-limit and serve-stale enabled
- Suppose a situation that an authoritative server is under DDoS attack: rd2020-theme2.jprs
  - We set up an authoritative server on the Internet
  - Generate random sub-domain queries to the full-service resolver
- Fetch-limit will suppress outbound iterative queries



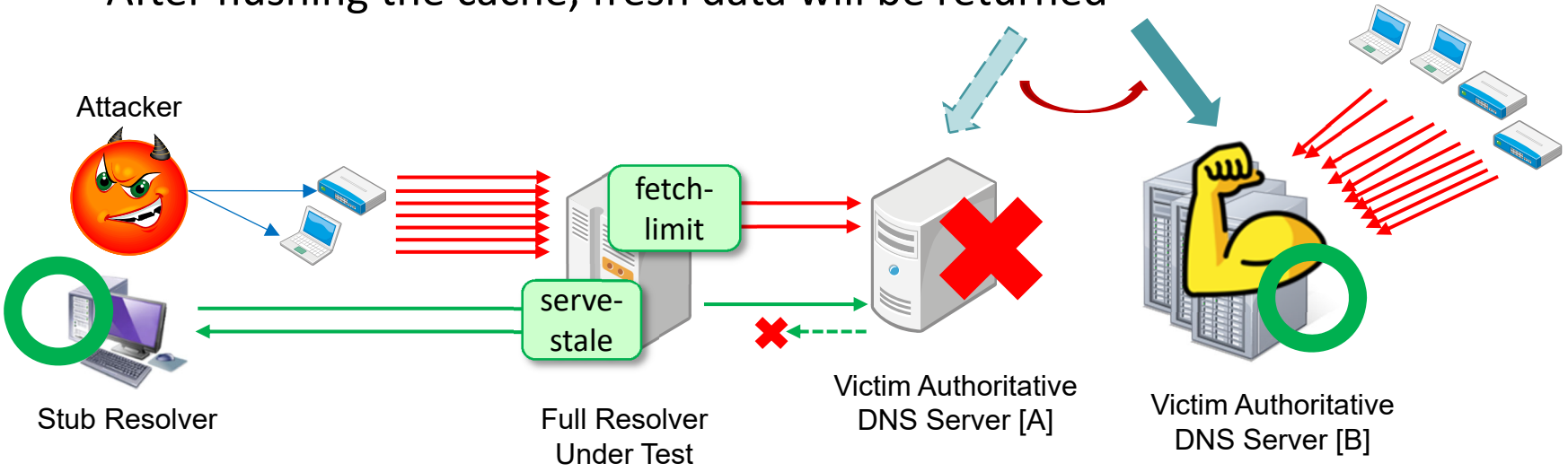
Main evaluation scenario (2)

- The authoritative server [A] becomes not responding
  - DDoS traffic from others increases and server [A] is not massive
  - Actually it is simulated by dropping inbound queries
- Serve-stale will use stale cache to answer queries
  - Legitimate clients still can resolve the domain name



Main evaluation scenario (3)

- Then, the victim switches to another DNS provider [B]
  - The servers [B] are massive enough to absorb DDoS traffic
  - Actually it is simulated with changing delegation in the parent zone to another NS RRset
- However the stale cache entry is still used
  - The full-service resolver answers with the stale cache
  - After flushing the cache, fresh data will be returned



Results

- The combination worked as expected with Unbound
  - ratelimit restricted number of iterations to the authoritative server
  - Full-service resolver continues to respond with cached answers even after the authoritative server becomes not responding and the TTL of the RRset expired
- BIND 9 worked differently than expected
  - fetch-limit restricted number of iterations to the authoritative server as expected
  - However, it responds with SERVFAIL after the authoritative server becomes not responding
  - `` `rndc dumpdb` `` shows there is a stale cache entry
- Stale cache may be used unexpectedly
  - While evaluation, maybe we should wait some time after changing delegation to see what happens

Comments from the operators

- serve-stale in BIND 9 did not work as expected (as explained in the previous slide)
- There is a concern on the stale cache
  - It may cause name resolution failure or getting old contents even after delegation or zone content change
  - After the experiment we realized it can be controlled with `infra-host-ttl` option
- If the stale cache remains after the authoritative server is back, it would cause name resolution outage for the full-service resolvers
  - However it would be difficult to flush the stale cache in real operation

Feedback to the implementers

- We sent some thoughts to ISC and NLnet Labs
  - They generously gave us a response
- ISC
  - We told that `fetches-per-{server,zone}` interferes `stale-answer-enable`; it cannot be used in combination
  - There already was an open issue: https://gitlab.isc.org/isc-projects/bind9/-/issues/1712
  - They confirm the behavior seems to be wrong and it is fixed in 9.16.13 (ChangeLog 5573)
- NLnet Labs
  - We told a concern on the default value of `serve-expired-ttl` is set to 0: different from suggested in RFC 8767 (1 day to 3 days)
  - They confirm it is a good idea to change but changing it may complicates operators; we agree with it

Conclusion & Acknowledgements

- We evaluated anti-DDoS functionalities with DNS operators of some Japanese domestic ISPs
- We found some implementations did not work as expected
- We gave some feedback to the implementers and got positive response

- I would like to express thanks to:
  ISC and NLnet Labs for handling feedback from us
  and
  Participating ISPs: CNCI, ENECOM, Freebit, HOTnet, HTNet, OPTAGE, OTNet, Qtnet and Softbank