Poster: Measuring Open Resolvers Negative Caching

Yehuda Afek
Tel-Aviv University, Israel

Anat Bremler-Barr
Reichman University , Israel

Neta Peleg
Reichman University, Israel

ACM Reference Format:
Yehuda Afek, Anat Bremler-Barr, and Neta Peleg. 2021. Poster: Measuring Open Resolvers Negative Caching. In *Proceedings of ACM IMC 21'*. ACM, New York, NY, USA, 2 pages. https://doi.org/TBA

1 INTRODUCTION

We investigate the negative caching (caching of NXdomain responses) behavior on nine large open DNS resolvers. We measure the amount of time an NXDomain response is kept in the cache in various TTL configurations and compare it to the time an existent domain is kept in the cache. First we show that while most open-resolvers cache NXDomain responses, they however cap the NXDomain record TTL to 1 hour, regardless of what the corresponding authoritative tries to dictate. Secondly, there is almost no difference in the time a response is cached, if the corresponding query is re-queried multiple times or is a "one time" query. Thirdly, the caching time of existent queries is usually much longer than that of non existent domain queries.

These measurements shed some light on actual performance of negative caching in open resolvers, which are known to respond to 50% of all DNS traffic[1]. Negative caching can have a significant impact on the DNS system, Chen et al. [2] analyze real-life DNS traces from a large ISP operating a recursive resolver cluster, and observed that almost 40% of the responses received by the resolver are NXDomain. Recently, floods of NXDomain requests played a key role different DDoS attacks on the DNS system, e.g., the notorious Mirai botnet [3], and *water torture* [4]. Since the sub-domains are random, their records are not present in the recursive resolvers caches, thus always reaching the target authoritative server. During such an attack the negative cache is filled with NXDomain records until it overflows, which might critically degrade the system performances. Understanding negative caching behaviour is thus key first step in improving the system efficiency and in the development of DDoS attacks mitigation methods.

ACM IMC 21', November 2–4, 2021, Virtual Event

ACM ISBN TBA…$TBA
https://doi.org/TBA

2 MEASUREMENT OVERVIEW

Experiment Setup. The setup has two machines under our control, a client that issues "A" record queries as specified below to the different open resolvers, and an authoritative DNS server, running BIND9, with our domain (nxdomainidc.xyz), containing an authoritative zone file with a SOA record.
Unlike the TTL value of valid "A" records that is inherited from the corresponding response of the authoritative server, the TTL value of a new negative response (NXDomain response) is set by the TTL value found in the SOA (start of authority) record of the corresponding zone file. We ran our experiments with three different values set in the SOA record - 30 min, 1 hour and 6 hours, and measure how long records are actually cached, by observing the TTL on repeated responses in the client. The TTL on the response to a first time NX request, called initial TTL, is the value set by the resolver, which suppose to be the value in the SOA record. Subsequent responses from the cache come with TTL values that are smaller than the initial one. Since caching time of a record is unpredictable and is effected by other queries and the load on the cache, we run an experiment that measures the maximum caching time seen during a time window. We run it multiple times and measure the median, average and variance of the caching time for negative responses in a resolver. We repeated this measurement on a list of open resolvers which serve a large portion of the DNS traffic. First observation we noticed that 7 out of 9 resolvers truncate the SOA TTL and set it to 1 hour (when 6 hours is specified in the SOA). We thus calculate the caching time of a cached response as the difference between the initial TTL and the TTL returned in a subsequent query. We provide the results of the 6 hour TTL as they emphasis the negative caching behaviour of all the open resolvers. We also alternate between "hot" and "one time" queries in order to observe how this effects caching duration, since in some caching evictions policies, e.g., Least Recently Used and Most Frequently Used, "hot" entries are likely to be cached longer. Finally we compare these results with the corresponding results for existent domain by adding "A" records in the Authoritative server for our queries and setting each record with the chosen TTL. **Experiment, Query Set** - We create a set of (217) query names (called the "one time" queries). Each query name is prefixed with a number from {100, 200, ..., 21600 (TTL-TIME)} where

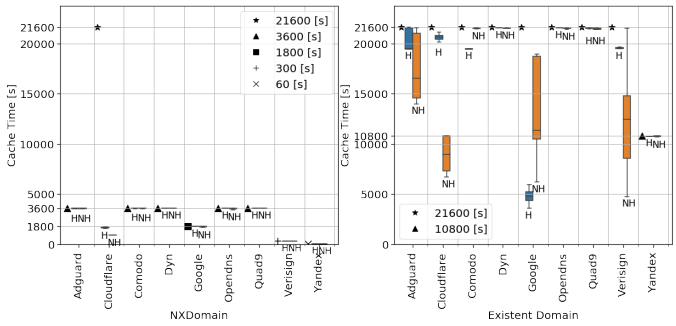

Figure 1: Cache time observed in seconds per resolver for TTL value of 6 hours. H - Hot Query, NH - Non Hot Query. Each Resolver has a marker to the left of the H/NH plots indicating actual TTL returned in initial request, values are presented in the marker legend.

TTL-TIME is the negative caching TTL time set in the SOA record. e.g., 100.nxdomainidc.xyz, 200.nxdomainidc.xyz etc. We also add one query, abc.nxdomainidc.xyz, to the query set (called the "hot query").

Initial Cache Filling - At time zero a burst of ten "A" queries per each query name in the query set, is sent to the open resolver in order to fill the different caches. As shown by Randall et. al. [5] each open resolver has a different server topology and different cache architecture. Repeating each query 10 times increases the probability that the query gets to all the different caches of the resolver.

Re-Query - In time interval jumps of 100 seconds we send five "A" queries of the query name that starts with the current time, and save the TTL returned in the response. We also send five queries of the "hot query", abc.nxdomainidc.xyz, at each time interval. That is after 100 seconds from the begging of the measurement we send five "A" queries for 100.nxdomainidc.xyz and five "A" queries for the "hot query", after 200 seconds we send five "A" queries for 200.nxdomainidc.xyz and five "A" queries for the "hot query", and so on. We query multiple times in this stage too, to increase the probability of hitting a cache holding the response.

Get Max Cache Time - We retrieve the response with the minimal TTL value, corresponding to the longest caching duration. Two values are retrieved - longest caching duration for hot and non hot query.

Repeating the experiment above gives us caching duration data points. In figure 1 We show box plots per resolver for "Hot" and "Non Hot" queries maximum cached time. The box of each resolver shows us a five number summary, “min”, 1st quartile, median, 3rd quartile, and “max”.

3 KEY OBSERVATIONS

TTL Truncation - 8 out of 9 open resolvers ignore the TTL returned from the authoritative server, and set the initial TTL of negative responses to one hour. Google Public DNS (GPDNS), holding 35.94% of the DNS traffic [1], set it to 30 minutes. Outstanding is cloudflare resolver, which takes the value from the SOA record, but as seen in the next observation, does not perform well in terms of caching time.

Caching Time - We see that although most resolvers truncate the TTL returned by the authoritative servers, the record is cached for the entire duration of the TTL. Notably poor performance is showed by cloudflare which on one hand does not truncate the TTL, but effectively caches for less time than other resolvers.

"Hot" vs "Non Hot" queries - We observe little differences in the caching behaviour, since most resolvers respect the TTL to its full duration. Note that cloudflare, is able to cache the record of the "Hot queries" for a longer time than the "one time" queries, which might imply that the cache is filling up and throwing away cold records.

Existent vs NXDomain - The resolvers respect the TTL returned from the authoritative servers for existent domain responses, and cache the records for the TTL period of time. We do see differences between the Hot vs Non-Hot queries caching time as should be expected from a caching system with an eviction policy. A point to notice is GPDNS which has counter intuitive results, where non hot queries are cached longer than hot queries, this behaviour is seen through all the different TTL measurements.

The key finding here is that negative caching on open resolvers is managed differently than the regular existent cache. The resolvers aggressively truncate the TTLs and cache the responses for a short amount of time. We assume this is done to handle NXDomain attacks from flooding the caches with garbage data. On the other hand this limits the ability of domain administrators to control the negative caching for their domain.

Future Work - We are looking to understand the logic behind the TTL caping in all the open resolvers, and understand if negative caching is efficient and improving the performance in the DNS system

REFERENCES

[1] Roxana Radu and Michael Hausding. Consolidation in the dns resolver market–how much, how fast, how dangerous? *Journal of Cyber Policy*, 2020.

[2] Yizheng Chen, Manos Antonakakis, Roberto Perdisci, Yacin Nadji, David Dagon, and Wenke Lee. Dns noise: Measuring the pervasiveness of disposable domains in modern dns traffic. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014.

[3] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX Security Symposium*, 2017.

[4] Yuya Takeuchi, Takuro Yoshida, Ryotaro Kobayashi, Masahiko Kato, and Hiroyuki Kishimoto. Detection of the dns water torture attack by analyzing features of the subdomain name. *Journal of Information Processing*, 2016.

[5] Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M Voelker, Stefan Savage, and Aaron Schulman. Trufflehunter: Cache snooping rare domains at large public dns resolvers. In *Proceedings of the ACM IMC*, pages 50–64, 2020.