NXDOMAIN Answer for A Record Queries

Yves Dantas Neves

BRIEF INTRO TO DNS RECORDS AND ERROR CODES

DNS records are instructions that live in authoritative DNS servers and provide information about a domain including what IP address is associated with that domain and how to handle requests for that domain. The following are common DNS Records which are part of this technical case study:

- **A Record** - The record that holds the IP address of a domain.
- **AAAA Record** - The record that contains the IPv6 address for a domain (as opposed to A records, which list the IPv4 address).

A DNS error code is a standardized response code used by the Domain Name System (DNS) to indicate the outcome of a query.The **NXDOMAIN** Error (Response Code 3) stands for a non-existent domain and represents an error DNS message received by the Recursive DNS server when the requested domain cannot be resolved to an IP address.

LAB ENVIRONMENT

Requirements

To replicate the scenario, a linux machine with docker installed is required. The environment is composed by a linux container which runs tcpdump and save the results to a .pcap file and a containerized DNS (Knot Resolver) which will resolve our testing queries. The deployment is defined and streamlined with **docker-compose**.

docker-compose.yml

Traffic Monitor

Container which sniffs traffic on lo interface and port 53

Dns Resolver

Container which runs the Knot Resolver

KNOT RESOLVER CONFIG

config.yaml

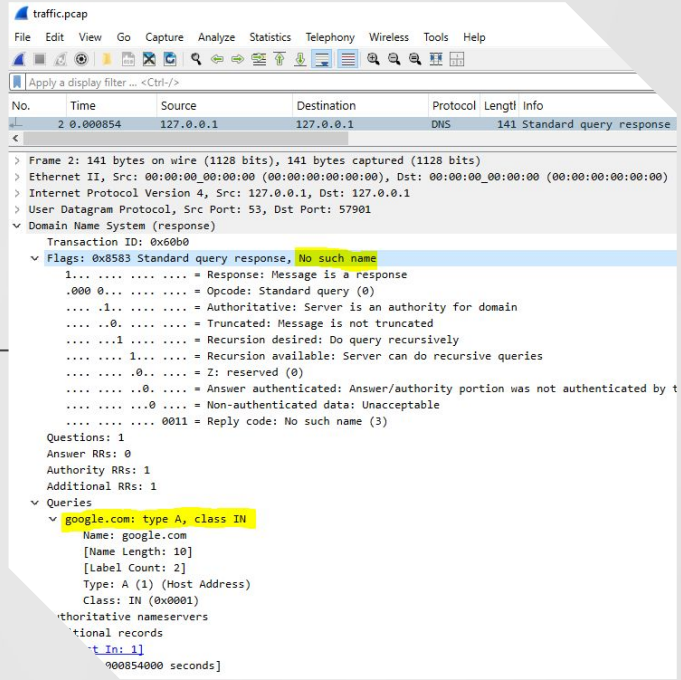
The default DNS Resolver configuration was kept and a customized filter to match and block *A Record Queries and domains saved in the blocklist.rpz file* was created and added to the configuration.

```yaml
network:
  listen:
    - interface: lo@53
    - interface: lo@853
      kind: dot
    - interface: lo@443
      kind: doh2
management:
  interface: 127.0.0.1@5000
lua:
  script: |

    function endswith(str, ending)
        -- Check if the ending is longer than the string
        if #ending > #str then
            return false
        end
        -- Compare the end of the string with the ending
        return string.sub(str, -#ending) == ending
    end

    function filter_blocklist(action, target_qtype, block_qname)
        return function (state, query)
            query_check = endswith(kres.dname2str(query.sname), block_qname)
            if query.stype == target_qtype and query_check then
                return action
            else
                return nil
            end
        end
    end

    file = io.open("/etc/knot-resolver/blocklist.rpz", "r")
    for line in file:lines() do
        dname = string.gsub(line, "[\r\n]+$", "")
        policy.add(filter_blocklist(policy.DENY, kres.type.A, dname))
```

EXPERIMENT RESULTS

dig google.com @127.0.0.1 A

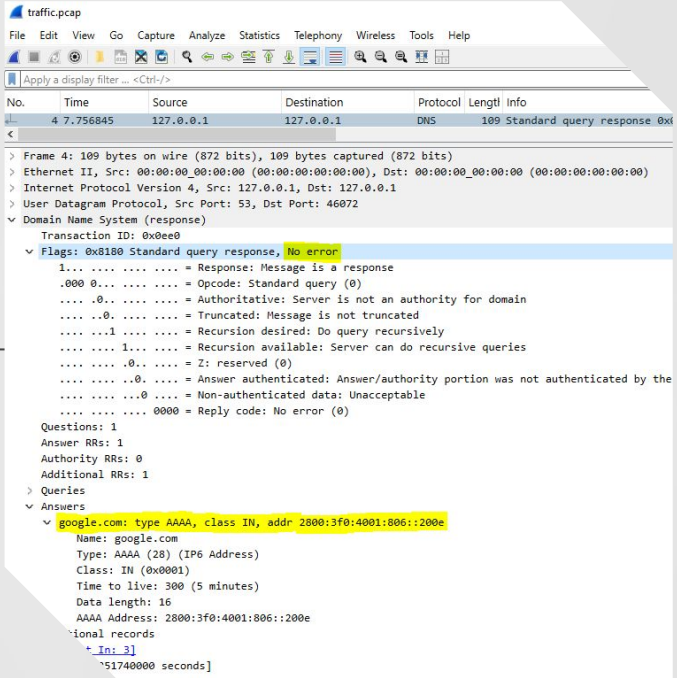
The DNS Resolver inspects the query and the policy for query blocking is matched since google.com is added to the *blocklist.rpz* file. The resolver returns a *NXDOMAIN Error* to this query.



EXPERIMENT RESULTS

dig google.com @127.0.0.1 AAAA

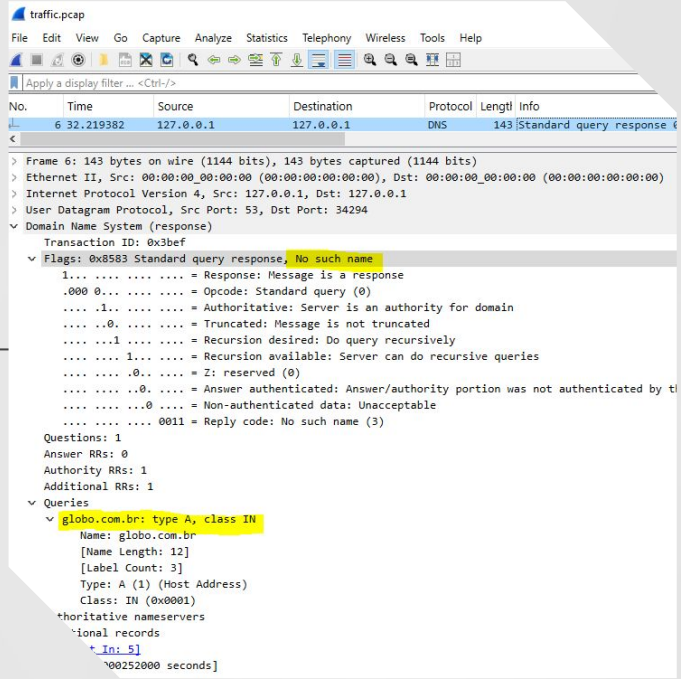
As the blocking policy is only applicable to A queries, the DNS Resolver is able to answer this query.



EXPERIMENT RESULTS

dig globo.com.br @127.0.0.1 A

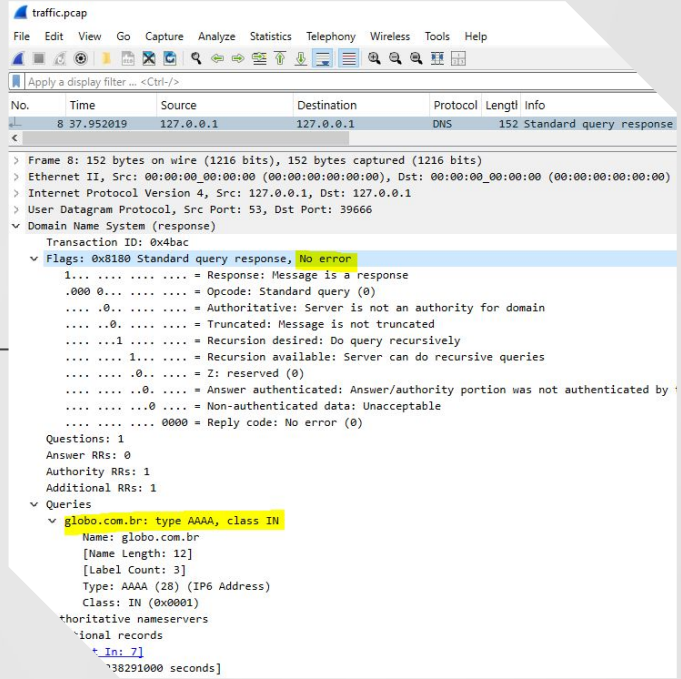
This is another example of a query which was added to the *blocklist.rpz* file and consequently matches the blocking policy.



EXPERIMENT RESULTS

dig globo.com.br @127.0.0.1 AAAA

For this example, as expected the policy is not matched but the query does not bring any answers since the domain globo.com.br does not offer IPv6 services at this moment.



REFERENCES

- https://datatracker.ietf.org/doc/html/rfc1035
- https://www.knot-resolver.cz/documentation/latest/config-lua.html
- https://knot-resolver.readthedocs.io/en/stable/modules-policy.html
- https://docs.whalebone.io/en/immunity/knot_tips_tricks.html#deny-list-of-domains
- https://github.com/yvesdantas/dns-resolver-nxdomain

Thank you!