IETF 115 Hackathon, DNS error reporting

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 8 November 2022

https://www.bortzmeyer.org/hackathon-ietf-115.html

At the IETF 115 <https://www.ietf.org/how/meetings/115/> hackathon in London, I worked on DNS error reporting.

Remember that IETF develops the standards for the Internet, roughly from layer 3 to some parts of layer 7. One of the most important technologies at IETF is of course the DNS. The role of the IETF hackathons, that take place the weekend just before the IETF meeting, is to try to implement new ideas (not yet standardized) to see if it works (IETF prides itself on relying on "running code"). This time, we were four persons working on DNS error reporting. This technique is currently specified in an Internet Draft `draft-ietf-dnsop-dns-error-reporting`. The problem it tries to solve is the one of informing the managers of a zone that there is some technical problem detected by a DNS resolver. Today, if you don't test your zone thoroughly with tools like DNSviz <https://dnsviz.net/> or Zonemaster <https://zonemaster.fr/>, you'll know that there is a problem only when users will complain on Twitter. It would be better to be told by your clients, the resolvers, before that.

The way it works is a follows :
— The authoritative server for the zone returns in its replies an EDNS option indicating a **report receiving domain**.
— The resolver, when it detects a problem, such as a wrong DNSSEC key, sees this indication and creates a DNS query encoding the problem and appending the report receiving domain.
— The servers for the report receiving domain receives the query, processes it and stores it.
— The zone administrator reads about the problem and acts accordingly.

You can see there are three actors and therefore three programs to modify (the last ones, the servers for the report receiving domain, could be unmodified authoritative name servers). I worked on the authoritative side, using the **experimental** name server Drink <https://www.bortzmeyer.org/drink.html>. Willem Toorop worked on the Unbound resolver and Shane Kerr on another (proprietary) authoritative server. Roy Arends helped, replied and modified the draft with our feedback.

The draft is quite simple and straightforward. Drink was modified to emit a new EDNS option :

```
report_to = Binary.from_list(encode(config()["reporting-agent"]))
length = byte_size(report_to)
Binary.append(<<Drink.EdnsCodes.reporting::unsigned-integer-size(16),
length::unsigned-integer-size(16)>>, report_to)
```

And then to process the reports (the query type for the reports is TXT) :

```
"report" ->
   if config()["services"]["report"] do
              case qtype do
                :txt ->
                  result = Drink.Reports.post(Enum.join(qname, "."))
                  %{:rcode => Drink.RCodes.noerror,
                    :data => [result],
                    :ttl => @report_ttl}
                :any -> @any_hinfo
                _ -> %{:rcode => Drink.RCodes.noerror}
              end
```

Reports are stored in a separate agent. Note that we check they are properly formed, with the label `_er` at the beginning and at the end :

```
  def post(value) do
    labels = String.split(value, ".")
    if List.last(labels) == "report" do
      labels = List.delete_at(labels, length(labels)-1) # Remove report
      if List.first(labels) == "_er" and List.last(labels) == "_er" do
        labels = List.delete_at(List.delete_at(labels, length(labels)-1), 0) # Remove _er
        error_code = List.last(labels)
        qtype = List.first(labels)
        qname = Enum.join(List.delete_at(List.delete_at(labels, length(labels)-1), 0), ".")
        Agent.update(__MODULE__,
          fn state ->
            [{error_code, qtype, qname} | state]
          end)
        "Thanks for the report of error #{error_code} on #{qname}"
      else
        # Else do nothing, probably QNAME minimization (or may be broken report)
        "Invalid report"
      end
    end
  end
```

From the outside, this looks like this :

```
% dig _er.1.foobar.example.9._er.report.dyn.courbu.re TXT
...
;; ANSWER SECTION:
_er.1.foobar.example.9._er.report.dyn.courbu.re. 24 IN TXT "Thanks for the report of error 9 on foobar.examp

;; AUTHORITY SECTION:
dyn.courbu.re. 15 IN NS dhcp-80f2.meeting.ietf.org.
...
```

---

https://www.bortzmeyer.org/hackathon-ietf-115.html

Since Drink is fully dynamic, it allows us to send a response depending on the analysis of the report (but the draft says you can return anything, you can use a static TXT record). If the report is invalid :

```
% dig _er.1.foobar.example.9.missing-end.report.dyn.courbu.re TXT
...
;; ANSWER SECTION:
_er.1.foobar.example.9.missing-end.report.dyn.courbu.re. 30 IN TXT "Invalid report"
```

One can then retrieve the report, by asking the name server :

```
%   echo report | socat - UNIX-CONNECT:/home/stephane/.drink.sock
REPORT state:
foobar.example, 9
funny.example, 9;2
```

Lessons from the hackathon ? Drink is well suited for rapid exprimentation (it was one of its goals), the draft is clear and simple and interoperability is fine. By querying the modified Unbound resolver, we see report queries arriving and being accepted. Here, a DNSSEC error :

```
% dig @2a04:b900:0:100::28 www.bogus.bortzmeyer.fr
...
;; OPT PSEUDOSECTION:
...
; EDE: 9 (DNSKEY Missing): (validation failure <www.bogus.bortzmeyer.fr. A IN>: key for validation bogus.bortzme
...
```

(Errors are the errors specified in RFC 8914[1].) And Drink saw :

```
nov. 08 18:17:14 smoking Drink[365437]: [info]  185.49.141.28 queried for _er.1.www.bogus.bortzmeyer.fr.9._er.re
```

And we can get it from the server :

```
% echo report | socat - UNIX-CONNECT:/home/stephane/.drink.sock
REPORT state:
foobar.example, 9
funny.example, 9;2
www.bogus.bortzmeyer.fr, 9
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8914.txt`

So, it is a success. You can see all the reports made at the end of the hackathon `<https://github.com/IETF-Hackathon/ietf115-project-presentations>` (ours is "IETF115-DNS-Hackathon-Results.pdf "). Outside of that, there is a Wireshark implementation of the EDNS option `<https://gitlab.com/wireshark/wireshark/-/merge_requests/8815>`:

Because it is a work in progress, note that there were changes before, during and after the hackathon. The query type for the reports changed from NULL to TXT (the code for NULL was tested and works), the EDNS option is now returned only when the resolver asks for it (this is not yet done in Drink), the format of the query report changed, etc. The goal is to test, not to make stable code.

The code, written in Elixir, is available online `<https://framagit.org/bortzmeyer/drink/>`. It is not yet merged in the master, you have to use the git branch `error-reporting`.