DNS Extended Error reporting at the IETF hackathon

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 25 March 2019

https://www.bortzmeyer.org/hackathon-ietf-104.html

On 23 and 24 March 2019 took place the now traditional hackathon of the IETF. I worked on a mechanism to report more detailed errors for DNS requests.

This IETF meeting was located in Prague (a very common city for IETF meetings). As we now do at each meeting, there is a hackathon the weekend before. This one was the biggest ever <https://twitter.com/ietf/status/1109376786138361857>, with more than 350 persons coding in the room.

One of the projects (involving four people, Shane Kerr, Ralph Dolmans, Vladim[Caractère Unicode non montré [1]]r [Caractère Unicode non montré]un[Caractère Unicode non montré]t and myself) was to implement the "Extended DNS Errors <https://datatracker.ietf.org/doc/draft-ietf-dnsop-extended-error/>" Internet-Draft. (It was later published as RFC 8914 [2].) A long-time problem of the DNS is the fact there is little error reporting to the client. Most standardized return codes <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-6> are rare or useless and one, SERVFAIL (Server Failure), is used to report almost every possible error. Therefore this draft, which adds a EDNS option to be returned by the server, adding to the normal return code :

— An "info-code", an integer giving details,
— Optional "extra-text", a human-readable character string,
— The "retry" bit, indicating to the clients if it makes sense to retry to another server or not.

My choice for the hackathon was to add this extended reporting system to the Knot resolver <https://www.knot-resolver.cz/>, with Vladim[Caractère Unicode non montré]r [Caractère Unicode non montré]un[Caractère Unicode non montré]t (Shane Kerr worked on dnsdist <https://dnsdist.org/> and Ralph Dolmans on Unbound). Knot is written in C but a part of the extra modules is in Lua.

Adding an extra EDNS option in the reply is quite simple (the real difficulty is after that, don't stop reading !). I choose a code for the option, 65500, from the range dedicated to experimentations <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-11>:

1. Car trop difficile à faire afficher par LaTeX
2. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8914.txt

```
#define KNOT_EDNS_OPTION_EXTENDED_ERROR 65500
```

Of course, in the future, this definition will be in libknot, the general DNS library of the Knot authoritative server, library which is also used by the Knot **resolver**.

I then created a new module for Knot (many functions are not done by the core but by modules), `extended_error`, with the usual data structures (code is available, you'll see it later). Extended errors are defined as :

```
struct extended_error_t {
bool valid; /* Do we have something to report? */
bool retry;
uint16_t response_code;
uint16_t info_code;
const char *extra_text; /* Can be NULL.  Allocated on the kr_request::pool or static. */
};
```

One small issue : serializing such a data structure in the layout decided by the draft (one bit for `retry`, four for `response_code`, matching RFC 1035, section 4.1.1, twelve for the `info_code`), required me to re-learn bit manipulations in C :

```
uint32_t serialize(struct extended_error_t err) {
uint32_t result = 0;
result = (u_int32_t)err.retry << 31;
result = result + ((err.response_code & 0x0000000F) << 12);
result = result + (err.info_code & 0x00000FFF);
return(htonl(result));
}
```

(Je sais, ce n'est pas beau d'utiliser les opérateurs arithmétiques comme le plus, pour des opérations sur des bits. Cela a été corrigé par la suite.)

Now, once the module is done (`modules/extended_error/`), compiled (Knot uses Meson) and ran, we indeed get the new EDNS option in dig output. (We did not develop a client using extended error codes, just used dig or Wireshark to see what the server returned.) But the real difficulty of the project was not here : it is to extract the correct information from the depths of the DNS resolver. On some resolvers (at least on Knot), the place where the error is noticed can be quite far from the place where the answer is built, with its EDNS options. In practice, we had to add data to the request object `kr_request`, for the extended error information to be carried to the module that emits the extended error code EDNS option. So, the real difficulty is not in the draft, but in knowing and understanding your resolver.

First example, the case where all the authoritative name servers for a zone are down (or reply wrongly). This is in `lib/resolve.c` after some search in the code to see where the resolver decided it is time to give in :

```
if (qry->ns.score > KR_NS_MAX_SCORE) {
        request->extended_error.valid = true;
        if (kr_zonecut_is_empty(&qry->zone_cut)) {
             request->extended_error.retry = true;
             request->extended_error.response_code = KNOT_RCODE_SERVFAIL;
             request->extended_error.info_code = KNOT_EXTENDED_ERROR_SERVFAIL_NO_AUTHORITY;
             request->extended_error.extra_text = "no NS with an address"; /* Also used when all authoritative na
```

Younger C programmers would use the syntax :

```
request->extended_error = (struct extended_error_t){
             .valid = true,
             .retry = true,
             .response_code = KNOT_RCODE_SERVFAIL,
             .info_code = KNOT_EXTENDED_ERROR_SERVFAIL_NO_AUTHORITY,
             .extra_text = "no NS with an address",
};
```

Another example is with DNSSEC, because DNSSEC is a very important use case for extended errors since it is important to know if the error is local to the resolver, or due to a broken DNS zone. This is in `lib/layer/validate.c` and, again, the difficulty was not to write the code but to find where to put it :

```
if (ret != 0) {
        if (ret != kr_error(DNSSEC_INVALID_DS_ALGORITHM) &&
            ret != kr_error(EAGAIN)) {
            req->extended_error.valid = true;
            req->extended_error.retry = false;
            req->extended_error.response_code = KNOT_RCODE_SERVFAIL;
            if (vctx.rrs_counters.expired > 0) {
                   req->extended_error.info_code = KNOT_EXTENDED_ERROR_SERVFAIL_DNSSEC_EXPIRED;
                   req->extended_error.extra_text = "DNSSEC expired signatures";
```

One last example, using Knot's policy module. "Policy" means the ability to lie, for instance to block domains used in advertising and tracking (or to implement state censorship <`https://labs.ripe.net/Members/stephane_bortzmeyer/dns-censorship-dns-lies-seen-by-atlas-probes`>). Most Knot modules are in Lua. This one is in `modules/policy/policy.lua`, and it shows that extended error codes are not only useful for SERVFAIL but here for NXDOMAIN, to allow the lying resolver to explain why it lied :

```
function policy.DENY_MSG(msg) -- TODO: customizable extended error info code
    return function (_, req)
...
          req.extended_error.valid = true
          req.extended_error.retry = true
          req.extended_error.response_code = kres.rcode.NXDOMAIN
          req.extended_error.info_code = 1 -- "Blocked" TODO KNOT_EXTENDED_ERROR_NXDOMAIN_BLOCKED
```

With this code, we can configure the resolver to load the "extended error" module, and we block a domain, to check the policy module. Here is the configuration (itself a Lua file) :

```
modules = {
   'hints', 'nsid', 'extended_error'
   }
...
policy.add(policy.suffix(policy.DENY_MSG("No tracking"), {todname('googleanalytics.com.')}))
```

Let's try it with dig :

```
% dig  @::1 A brk.internautique.fr
...
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 15368
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; OPT=65500: 80 00 20 07 6e 6f 20 4e 53 20 77 69 74 68 20 61 6e 20 61 64 64 72 65 73 73 (".. .no NS with an
;; QUESTION SECTION:
;brk.internautique.fr. IN A
...
```

We get a SERVFAIL, which is expected for this broken zone, plus an extended error code, displayed in hexadecimal by dig. We get the retry bit (such error might be temporary, or it may be server-dependent, for instance because of a local routing problem), and the "info code" 7 ("SERVFAIL Extended DNS Error Code 7 - No Reachable Authority").

Another example, with a zone which has a DNSSEC problem (in that case, deliberately introduced, for testing) :

```
% dig  @::1 A expired.caatestsuite-dnssec.com
...
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 38360
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; OPT=65500: 00 00 20 02 44 4e 53 53 45 43 20 65 78 70 69 72 65 64 20 73 69 67 6e 61 74 75 72 65 73 (".. .DN
;; QUESTION SECTION:
;expired.caatestsuite-dnssec.com. IN A
...
```

This time, no "retry" bit (using another resolver won't help) and info code 2, `KNOT_EXTENDED_ERROR_SERVFAIL_DNSSEC_EXPIRED`.

One last example, showing the effect of the policy declared in the configuration file :

```
% dig  @::1 -p 9053 A googleanalytics.com
...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 46242
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; OPT=65500: 80 00 30 01 4e 6f 20 74 72 61 63 6b 69 6e 67 ("..0.No tracking")
;; QUESTION SECTION:
;googleanalytics.com. IN A

;; AUTHORITY SECTION:
googleanalytics.com. 10800 IN SOA googleanalytics.com. nobody.invalid. (
1          ; serial
3600       ; refresh (1 hour)
1200       ; retry (20 minutes)
604800     ; expire (1 week)
10800      ; minimum (3 hours)
)

;; ADDITIONAL SECTION:
explanation.invalid. 10800 IN TXT "No tracking"
```

Note the retry bit (if you don't approve the lie, you may go to another resolver), the forged SOA record, and the extra text which contains the message indicated in the configuration file.

If you want to see the full code, it is on the public site <https://gitlab.labs.nic.cz/knot/knot-resolver/tree/extended_error>, in Git branch `extended_error`. A merge has been requested <https://gitlab.labs.nic.cz/knot/knot-resolver/merge_requests/794>.

Thanks to Vladim[Caractère Unicode non montré]r [Caractère Unicode non montré]un[Caractère Unicode non montré]t for a nice collaboration and for answering my (not always clever) questions, to Sara Dickinson for presenting the results <https://twitter.com/NLnetLabs/status/1109815581283356672>, and to the organisers of the hackathon : a useful event !