# DNS Myths and Misconceptions

An FAQ, sort of

# The DNS can take anything so let's just shove everything there

- The DNS is a success story
  - and so it also carries a burden
- Lots of stuff end up in the DNS
  - And the fast path to dump new stuff in the DNS is the one available free-form Resource Record
  - aka the TXT RR
    - Yes, I realise there is an 1993 RFC suggesting how to do this but…
- Like many things, it is a good idea if you are the first and only squatter
  - But envy is a very human emotion
  - And soon you won't be alone (and, unfortunately, not in the Liverpool sense)

# Why throwing everything in TXT RRs is not a good idea

DNS Rrsets are indivisible

You get all  the records in the set even if you want only yours

```
google.com. 3569 IN TXT "v=spf1 include:_spf.google.com ~all"
google.com. 269 IN TXT "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com. 3569 IN TXT "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
google.com. 3569 IN TXT "facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
google.com. 269 IN TXT "docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
```

# Just get a new RR type

- But that is so hard!!!!
- No, it really isn’t. You just need to write a clear spec and request an expert review, for which the IETF has a standing panel eagerly waiting for your requests

- You might be getting this confused with getting your own top level name (TLD)
- That is hard…
- … and expensive…
- ... and pointless

# How do you get a new RR type if you need one?

- Step 1: Define what you want (this is the hardest part)*
- Step 2: Ask the standing set DNS review experts to do their job, wait a few weeks, be happy
- Step 3: there is no step 3

* If you don’t define special processing, where the DNS server would have to apply rules beyond a key/value lookup, a new RR Type is opaque to the server software everywhere so please, go ahead. You don’t need to wait for any customized DNS server or resolver support. Just do it!

# While we are here…

- Apart from the (mis|ab)use of the TXT RR:

- DON’T GET A TLD!
  - It’s expensive
  - It’s hard (it will involve lawyers)
  - It takes a long time
  - But above all
  - It is most probably NOT WHAT YOU WANT

- DON'T MAKE UP YOUR OWN TLD
  - Someone else might get the same idea – collisions
  - Clashes in private space
    - Others may have the same idea you did
    - Two (or more) networks are joined and have different uses for the same name
  - Clashes in public space
    - When public identifiers stop being unique, the working network stops

# If names are for humans…

- How do you write those names in your language/script?
- IDN (Internationalised Domain Names)
  - Encodes Unicode characters for many scripts in DNS-compatible ASCII
  - Works a treat but…
  - Turns out distinct characters, in different scripts, have equal or very similar visual representations
    - Homoglyphs (a in latin and cyrillic scripts), look the same but are not, so a name containing different “a” is a different name even if it looks the same, or A in cyrillic and greek
    - even within ASCII we have had long lived problems with 0/0 and 1/l
    - ı /i (good luck spotting the difference if you are not used/not expecting the other variation)

# Nah, I will not get caught by this…

Right, so which one is the good one? (If any)

http://google.com

http://google.com

http://google.com

Have fun at, for instance, https://www.irongeek.com/homoglyph-attack-generator.php

# Valid characters in the DNS

- Domain names and host names
  - The DNS only places a length restriction on each label (63 bytes) and a total length restriction on the name (255 bytes) and does not restrict what values a label can have.
  - However, restrictions may be placed on labels used by applications
  - A host name, which identifies a host, inherited restrictions from before the DNS. RFC 952
  - Most accepted norm is that a hostname can contain 0-9, a-z and hyphen, but not start with a hyphen and should contain at least one letter.
  - What is allowed or not has evolved since the before the existence of the DNS specificaction
  - RFC 1033, 1034, 1035, then the 3Com RFC (1123), later relaxations and clarifications (e.g. RFC 2181) and a bit of whatever people could get away with (e.g. 411.org), as well as vendor interpretation of the RFCs

# But that's not all

- There is a difference between what is STORED in the DNS and what we think of as DNS names
- There are implied Unicode rules of valid Unicode character and character sequences as well
    - Think of Unicode encoding to support non-ascii character sets...

      As an extreme example, characters such as U+2024, U+2025 and U+2026

      xn--dots-tc7a.example.com ⟺ dots….example.com

- There is an ongoing issue between adding new glyph sets into Unicode and stripping out obvious (and not-so-obvious) ambiguities in the IDNA standards!

# A parenthesis

Repeat after me: The DNS is NOT a search engine!

# An asterisk character is not a valid label?

- It is, but what it means depends on where it appears (hint, read RFC 4592)
  - If used in a query it is a wildcard, not a label per-se
    - It can only appear once, by itself and it has to be at leftmost label
    - When it appears as defined above it is called the “wildcard label”
    - Wildcard != regular expression
      - E.g., “a*” probably doesn’t mean what you expect
  - If present in a zone it is just that, an asterisk in a name

- Wildcards in the DNS are not the same as wildcards in a certificate DNS-like name
  - Certs: only one level deep
  - DNS: can match any numbers of labels to the left

# Question: So how should this work?

- Given a hypothetical example.com zone with these records:

```
*    IN    TXT    "*.example.com."
a.*  IN    TXT    "a.*.example.com."
a    IN    TXT    "a.example.com.”
a.b  IN    TXT    "a.b.example.com."
```

- Match these queries to the DNS response:

```
dig a.example.com.    txt → ?
dig c.example.com.    txt → ?
dig *.example.com.    txt → ?
dig a.*.example.com.  txt → ?
dig b.*.example.com.  txt → ?
```

# Answer: This is how it should work!

- Given a hypothetical example.com zone with these records:

```
*    IN    TXT    "*.example.com."
a.*  IN    TXT    "a.*.example.com."
a    IN    TXT    "a.example.com.”
a.b  IN    TXT    "a.b.example.com."
```

- Match these queries to the DNS response:

```
dig a.example.com.    txt → "a.example.com."
dig c.example.com.    txt → "*.example.com."
dig *.example.com.    txt → "*.example.com."
dig a.*.example.com.  txt → "a.*.example.com."
dig b.*.example.com.  txt → NXDOMAIN
```



Why doesn’t it match the wildcard?
If it is not the letftmost label in the query, it is not a wildcard query nor name

# When you don’t get data: NXDOMAIN vs NODATA vs SRVFAIL

- NXDOMAIN is  the common name for RCODE 3 (originally Name Error)
  - It means the name does not exist and neither does any node below it in the DNS tree (see RFC 8020)
- NODATA is not an RCODE. It is short-hand for a DNS response with RCODE==0 (NOERROR) and no data in the Answer Section
- How can this be?
  - 1) The query was for a QTYPE for which there is no data associated with that owner name. Asked for AAAA but only A data is present
  - 2) The name is an “empty non-terminal” in Ed Lewis’s words
- SRVFAIL
- Cache behaviour

# ENT

- We are not talking about middle earth beings here
- An Empty Non-Terminal is a Domain name for which there is no data but does not represent a terminal node in the DNS tree
  - a.b.example.com may exist but:
    - b.example.net does not and it is not a delegation point
- Answering NXDOMAIN would be wrong as there is data below that name

# CNAMEs can be used anywhere?

- Let’s remember:
  - A CNAME RR identifies its owner name as an alias, and specifies the corresponding canonical name in the RDATA section of the RR
- So you can’t say you are an alias for someone else and at the same time provide data for yourself as that would be putting words on the “real name’s” mouth.
- So, a CNAME MUST be the only record at a name that has one
- THEREFORE, by definition, since a DNS Zone’s apex must have an SOA record and NS records, it CAN NOT have a CNAME.

# All servers have the same data?

- As initially designed, this is true most of the time
  - DNS is a *loosely coherent* protocol.
  - Servers can be slightly off-sync but a well run set of servers will be sync'ed in near real time
- However, there is more it
  - People play tricks, OMG!
    - E.g. Geo localised data, meaning different responses for the same name depending on who is asking
    - ECS
    - Split DNS
  - Migrations not always go according to plan
  - Humans make mistakes, processes fail (e.g. sync may fail). This is what loose means above

# Flag days nowadays

- The DNS forms the substrate, together with routing, that makes the Internet work
- It’s deployment is universal, with differing levels of maintenance in the deployed base
- Any new feature must expect to see a multi-year deployment adoption
- Vendors don’t control operation (typically)

# CPEs work well as DNS resolvers?

- Not really:
  - Most CPEs have forwarders, not full resolvers.
  - These rely on a full resolver, typically operated by the ISP, and simply serve as a small local cache
- It might be good to have a local recursive resolver in the CPE
  - None of the resolver operators would see/receive all your queries, giving yo
- It might be bad to have a local recursive resolver
  - You may degrade performance by having a less efficient cache
    - DNS cache efficiency has been shown to depend on how “hot” it is which depends mostly on how many users target it.
  - Your resolver will issue small amounts of queries so it will be easier, if desired, to identify people behind that resolver, if traffic can be monitored.

# DNSSEC is too slow?

- Too slow for what or for whom?
- OK, so any additional operation will be slower than a straight ASCII fetch from a database, in DNS, in the Web, anywhere else
  - Or maybe not…
- DNSSEC data is cacheable (and so are the results of validation)
  - This property can be used to speed up queries for names that have already been queried BUT ALSO for names that haven't been, particularly if they don't exist by avoiding queries. This makes those DNS faster with DNSSEC than without

# If DNSSEC is slow may be the problem is elsewhere

- SIZE!!!!
  - Does your network carry big packets without issue?
  - Does it handle fragments properly?
  - Do your servers handle fragments properly?
  - This too can be tuned out (be smart with your EDNS settings)
- Are the DNSSEC records being filtered out by ancient middleboxes or sysadmins?

# TCP and the DNS

- The DNS does TCP?
  - If it has to, it can, and it will, mostly
    - TCP is always used for zone transfers
    - TCP is a standard fallback when UDP has issues (which it does sometimes)
      - If DNS responses generate fragmented UDP streams, you may have problems
    - Clients can choose to use TCP straight from the get go but that is not recommended
- A few things to keep in mind
    - The servers need tuning as you want to keep the minimum amount of state possible
    - And state requires memory
- It’s totally doable, just need to know what you are doing (aka be a pro)

# What is type ANY?

- QTYPE 255, a real query type
- The standard actually calls it '*'
  - But we already have enough trouble with the other '*' so we usually call it ANY

- It means “give me all you have got”
- Which **might not be** all there is.
- For instance, a caching server may only have some of the record types
  - And in practice if all you have got is not all there is then an implementation could reasonably say "this is all I want to give you!"