Public Suffix List DNS Query Service

CA/B Forum
February 22, 2022

https://publicsuffix.zone/

Peter Thomassen
peter.thomassen@securesystems.de

SSE

peter:~$ whoami

- Historically, a particle physicist (Ph.D. 2016)
  - Big data analysis at the CERN Large Hadron Collider (LHC)
- Running Internet services for ~20 yrs; long-term interest in Internet Security
  - Founded deSEC, a managed DNS platform to fill in the “DNSSEC gap”
    - Let’s Encrypt for DNSSEC
- Working for **Secure Systems Engineering** (Berlin)
  - **IT Security is not binary.**
    - **Not 0 or 1:** Security is gradual, 100% security does not exist.
    - **A process, not a state:** The security posture is subject to continual adaptation.
    - **It’s tech, but also culture:** We teach our clients to live the security mindset.
  - Both **defensive** (plan, implement, audit/review) and **offensive** (penetration testing)
  - **Various industries** (media & tech, financial, health, public/gov)

The Public Suffix List (PSL)

> *A "public suffix" is one under which Internet users can (or historically could) directly register names. Some examples of public suffixes are `.com`, `.co.uk` and `pvt.k12.ma.us`. The Public Suffix List is a list of all known public suffixes.*
> — https://publicsuffix.org/

What does that mean?

- Informs about **organization and policy boundaries** in the domain space
  - one level below (+ 1 label): **“registrable domain”**
- Supports wildcards, and exceptions from wildcards
- Maintained by the community (on GitHub) and provided as a text file

CA/B Forum Baseline Requirements (current version 1.8.1)

3.2.2.6 Wildcard Domain Validation

Before issuing a Wildcard Certificate, the CA MUST establish and follow a documented procedure that determines if the FQDN portion of any Wildcard Domain Name in the Certificate is “registry-controlled” or is a “public suffix” (e.g. “*.com”, “*.co.uk”, see RFC 6454 Section 8.2 for further explanation).

If the FQDN portion of any Wildcard Domain Name is “registry-controlled” or is a “public suffix”, CAs MUST refuse issuance unless the Applicant proves its rightful control of the entire Domain Namespace. (e.g. CAs MUST NOT issue “*.co.uk” or “*.local”, but MAY issue “*.example.com” to Example Co.).

Determination of what is “registry-controlled” versus the registerable portion of a Country Code Top-Level Domain Namespace is not standardized at the time of writing and is not a property of the DNS itself. Current best practice is to consult a “public suffix list” such as the Public Suffix List (PSL), and to retrieve a fresh copy regularly.

If using the PSL, a CA SHOULD consult the “ICANN DOMAINS” section only, not the “PRIVATE DOMAINS” section. The PSL is updated regularly to contain new gTLDs delegated by ICANN, which are listed in the “ICANN DOMAINS” section. A CA is not prohibited from issuing a Wildcard Certificate to the Registrant of an entire gTLD, provided that control of the entire namespace is demonstrated in an appropriate way.

PSL: Use Cases

- **Certificate Authorities**
  - should not issue certificates for wildcards directly under public suffix (such as `*.co.uk`)
- **Browsers**
  - may reject certificates of the above type
  - anti-phishing (domain highlighting), ordering of lists (e.g. downloads, cookies)
  - cookie/script scoping; computing `document.domain`
  - more examples: https://wiki.mozilla.org/Public_Suffix_List/Uses
- Multi-tenant **DNS hosting**
  - **our motivation** (DNS platform desec.io)
  - customer creating `co.uk`, blocking others from creating `example.co.uk`
- DMARC

Why a PSL Query Service?

Status Quo:

- Applications have to bring a **copy of the list**, and need to **keep it up to date**
- Applications have to **parse the list**
- Extracting information from the PSL requires a multi-staged algorithm

With a DNS-based Query Service:

- **No need to parse or refresh the PSL** altogether
- Public suffix can be **retrieved ad-hoc with a simple lookup**, cacheable
- No need for specialized tooling

… unless *privacy* or *latency* are an issue (browsers)

How it works

- In a special zone, **public suffixes are stored as subdomains with PTR values**
  - Zone: `query.publicsuffix.zone`
  - `co.uk PTR co.uk.`
- All **other names have a CNAME record** (or are covered by a CNAME wildcard)
- A domain's **public suffix is retrieved as the PTR record at the domain's name**
  - CNAMEs take care of "routing"
- **Auxiliary rules** that influenced the PTR outcome are **given as a TXT record**
  - e.g. in case of wildcard exceptions: parent rule is given in PTR, wildcard + exception in TXT
- **Authenticity** is provided by DNSSEC

Implementation Challenges

- The PSL **parsing algorithm is not trivial**
  - for example, it's important to get rule precedence right
- Also, PSL rules on a deeper level cause what's called "empty non-terminals"
  - intermediate levels need CNAME but can't be covered with a DNS wildcard

  → Things need to be glued together with a CNAME chain
- **~75k records total**
  - ~10k PSL rules → ~20k records for the mapping
  - ~55k for DNSSEC signatures
  - incremental updates require **calculating large diff**

Implem

- The P
  - f
- Also,
  - ir

→ Th

- ~75k
  - ~
  - ~
  - ir

https://github.com/sse-secure-systems/psl-dns/blob/main/psl_dns/parser.py

```
139     def _process(self):
140         # This algorithm transforms Public Suffix List input into RRsets so that the
141         # suffix of a domain is given by the PTR record of <domain>.<SERVICE>. The pe
142         # matching algorithm is described here: https://publicsuffix.org/list/
143
144         # Add regular rules
145         self._process_regular_rules()
146
147         # May be overwriting wildcard CNAME from regular rules, so has to go after re
148         self._process_regular_wildcard_rules()
149
150         # Find the next wildcard in the hierarchy and point to the rule covering its
151         self._process_wildcard_exception_rules()
152
153         # Remove rules that do not apply any longer
154         self._prioritize_wildcard_exception_rules()
155
156         # Needs to run before the wildcard shadowing step because it relies on this o
157         self._add_root_rule()
158
159         # Once the general structure is clear, fix up some stuff
160         self._fix_wildcard_shadowing()
```

Examples

Standard cases:

```
$ dig +noall +answer PTR cabforum.org.query.publicsuffix.zone
cabforum.org.query.publicsuffix.zone.    21530 IN CNAME  org.query.publicsuffix.zone.
        org.query.publicsuffix.zone.     7199 IN PTR    org.

$ dig +noall +answer PTR s3.dualstack.eu-west-1.amazonaws.com.query.publicsuffix.zone
s3.dualstack.eu-west-1.amazonaws.com.query.pu… 21600 IN PTR    s3.dualstack.eu-west-1.amazonaws.com.

$ dig +noall +answer PTR s4.dualstack.eu-west-1.amazonaws.com.query.publicsuffix.zone
s4.dualstack.eu-west-1.amazonaws.com.query.pu…  7198 IN CNAME  dualstack.eu-west-1.amazonaws.com.query.pu…
   dualstack.eu-west-1.amazonaws.com.query.pu…  7198 IN CNAME            eu-west-1.amazonaws.com.query.pu…
            eu-west-1.amazonaws.com.query.pu…  7198 IN CNAME                     amazonaws.com.query.pu…
                     amazonaws.com.query.pu…  7198 IN CNAME                              com.query.pu…
                              com.query.pu…  7198 IN PTR    com.
```

Wildcard with exception:

```
$ dig +noall +answer ANY www.ck.query.publicsuffix.zone | grep -v RRSIG
www.ck.query.publicsuffix.zone.                21600 IN PTR     *.
www.ck.query.publicsuffix.zone.                21600 IN TXT     "!www.ck"
www.ck.query.publicsuffix.zone.                21600 IN TXT     "*.ck"
```

Implementations / Demo

- Lookup zone **implemented under `query.publicsuffix.zone`**
  - hosted by **deSEC Managed DNS**
- https://publicsuffix.zone/ has a **live demo**
  - uses JavaScript requests to Google's DoH resolver
- Python implementation: https://pypi.org/project/psl-dns/
  - library + CLI
  - implements both querying and parsing (for preparing zone updates)
  - currently supports deSEC implementation, but interface is provider-agnostic

Limitations

Updates

- currently **every once in a while** (not automated)
- can be **automated easily** based on GitHub action or webhook

Inline wildcards no longer an issue.

- `foo.*.example.com` not possible in DNS
- PSL ~~allows them~~ **disallowed them** two weeks ago
  - https://github.com/publicsuffix/list/issues/145

→ DNS implementation provides **full coverage**

Addressing Privacy Concerns

DNS **resolvers learn about domains** that get queried.

- **Not a concern for CAs** (names leaked in CT Logs anyways)

Solution ideas:

- **Resolver-local copy** (e.g. via zone transfer/AXFR)
  - deSEC use case: we resolve directly against our own auth → no leakage
- ***k*-anonymity**: replace all labels by truncated hashes
  - **answers are fuzzy**, only client can sort it out
  - devil is in the detail

Next Steps?

The PSL Query Service works perfectly well for internal use case at deSEC.

- Do we need extra features to cover all use cases?
  - Distinguish between ICANN and PRIVATE section?
  - Find better solutions to address privacy concerns?
- It has been suggested to make this a community service
  - Does that make sense / is there support?
  - Who runs it (e.g. ICANN, CA/B Forum, somebody else)?
  - Does it need oversight (e.g. ICANN, CA/B Forum, somebody else)?
- …

Thank you!

Questions?

Addressing Privacy Concerns

DNS **resolvers learn about domains** that get queried.

- **Not a concern for CAs** (names leaked in CT Logs anyways)

Solution ideas:

- **Resolver-local copy** (e.g. via zone transfer/AXFR)
  - deSEC use case: we resolve directly against our own auth → no leakage
- ***k*-anonymity**: replace all labels by truncated hashes → collisions intended
  - **queries are fuzzy** and return list of potential matches (client selects item of interest)
  - inference from hierarchy patterns still possible
  - required API changes not very DNS-like → perhaps **not the best idea**