Improving DNS Privacy
and: the Battle for the Namespace

Roland van Rijswijk-Deij
Today

• **Who am I:**
  • *Associate professor* at University of Twente (EEMCS-DACS)
  • *Principal Scientist at NLnet Labs* -- not for profit developing open source software for core Internet protocols and real-world research on Internet protocols

• **Today:**
  • I will talk about privacy in the Domain Name System (**DNS**); my **goal** is to **show you how complex privacy can be in the context of real-world Internet protocols**
Introduction

- That the DNS has privacy issues is a public secret

- Protocol from 1980s with clear-text communication over UDP and TCP

- Snowden revelations just made this public secret very painful, as it turned out this was one of the Internet vulnerabilities being exploited en masse by intelligence services of the "Five Eyes"
IETF to the rescue!

• The IETF took action for many protocols post-Snowden

• October 2014: establishment of the DNS PRIVate Exchange (DPRIVE) working group

• Goal: analyse privacy issues in the DNS and propose protocol changes to alleviate these
First step: identifying problems

- **RFC 7626** gives a comprehensive overview of privacy risks in the whole DNS ecosystem

- Identifies all the points in the DNS ecosystem where privacy sensitive information can leak

- Today we're going to focus on client to resolver traffic

*a -bis of this RFC is in the final phase of standardisation: https://tools.ietf.org/html/draft-ietf-dprive-rfc7626-bis-04*
Recap: the DNS

You

Your ISP

You

DNS resolver

Focus for today

Internet

authoritative name servers

UNIVERSITY OF TWENTE.
Well actually...

Also going to talk about these folks

UNIVERSITY OF TWENTE.
Behavioural measures

- There are two behaviour changes for DNS resolvers that help privacy.

- **QNAME minimisation**, where resolvers limit what parts of a query string are sent to authoritative name servers.

- **Caching measures**, where resolvers can run parts of the name space locally, to limit sending, e.g., queries to the root onto the Internet (not going to talk about these in detail).
QNAME minimisation

- In **"classic" DNS**, resolver sends full query name to every server in hierarchy → to **enhance privacy**, only send necessary labels

<table>
<thead>
<tr>
<th>Standard DNS resolution</th>
<th>qmin Reference (RFC 7816)</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.</td>
<td>com. NS ← .</td>
</tr>
<tr>
<td>a.b.example.com A → com.</td>
<td>example.com NS → com.</td>
</tr>
<tr>
<td>example.com NS ← com.</td>
<td>example.com NS ← com.</td>
</tr>
<tr>
<td>a.b.example.com A → example.com.</td>
<td>b.example.com NS → example.com</td>
</tr>
<tr>
<td>a.b.example.com. A ← example.com.</td>
<td>b.example.com NS ← example.com</td>
</tr>
<tr>
<td>a.b.example.com. A ← example.com.</td>
<td>a.b.example.com NS ← example.com</td>
</tr>
<tr>
<td>a.b.example.com. A ← example.com.</td>
<td>a.b.example.com A ← example.com</td>
</tr>
</tbody>
</table>

This reference algorithm, however, faces two challenges on the real Internet:

- First, it does not handle configuration errors in the DNS well [26]. E.g., in case b.domain.example does not have any RRs but a.b.domain.example does, a name server should respond with **NOERROR** for a query to b.domain.example [8], but in fact often responds with **NXDOMAIN** or another invalid RCODE. This would force resolvers that conform to the standard to stop querying and thereby not successfully resolve the query. Also, operators report other issues, such as name servers that do not respond to **NS** queries, which would break qmin as well [25].

- Second, qmin can lead to a large number of queries. For example, a name with 20 labels would make the resolver issue 21 queries to authoritative name servers, causing excessive load at the resolver and authoritative. Attackers can abuse this for DoS attacks by querying excessively long names for victim domains.

Both of these issues led resolver implementors to modify their qmin implementations, as well as adding so called “strict” and “relaxed” modes, which we investigate in Subsection 3.2 and Section 5.

As of October 2018, three major DNS resolvers support qmin. Unbound supports qmin since late 2015 and turned relaxed qmin on by default in May 2018 [25]. Knot resolver uses relaxed qmin since its initial release in May 2016 [13], and the recursive resolver of BIND supports qmin and turned the relaxed mode on by default in July 2018 [23]. Another frequently used resolver, PowerDNS Recursor, does not support qmin yet [9].

**Related Work:**

Hardaker et al. [19] showed that root servers receive a considerable amount of privacy-sensitive query names, and propose using local instances of root servers to alleviate this issue. Imana et al. [22] study this aspect from a broader perspective, covering all name servers above the recursive resolver, and report similar privacy issues.

Schmitt et al. [32] propose Oblivious DNS, an obfuscation method introducing an additional intermediate resolver between recursive resolver and authoritative name servers. Oblivious DNS prevents the additional resolver from learning the user’s IP address and the recursive resolver from learning the query name.
QNAME minimisation

- **QNAME minimisation** is seeing quite a bit of deployment already.

- Supported by e.g. **1.1.1.1 and 9.9.9.9** (among others), but also e.g. SURFnet (ISP for Dutch universities).
DNS over TLS

- RFC 7858: simple idea, let the stub talk to the recursive over a TLS connection

- Raises some issues:
  - TCP + TLS handshake overhead
    (partially alleviated by TCP Fast Open and TLS Session Resumption)
  - Resource consumption on the recursor is a potential issue
    (TCP buffers, TLS state, ...)

- Generally speaking, though, works quite well
Padding

- An interesting aspect of encrypting DNS traffic is that **padding** may be required.

- Otherwise, the **size of queries and responses** can still be **used to profile users**!

- **EDNS0 padding** allows **stub resolvers** to **pad requests** and **recursors** that support it must also **pad responses** if the query was padded.

- There are multiple approaches to padding; block-length padding seems the most sensible.

(plot courtesy of Daniel Kahn Gillmor, based on data from SURFnet)
Issues in DNS over TLS

• Encrypting DNS traffic means some on-path **security monitoring** will no longer work; **requires a shift from on-path (A) to on-resolver (B)**

• **Little experience** in production **with resource requirements** of DoT

• **Dedicated TCP port 853 may be blocked**, making DoT unavailable
DoT implementation status

- **DNS over TLS** is already **well-supported** in recursors; **all** the **popular resolver implementations** support it (Unbound, BIND, Knot Resolver, PowerDNS Recursor)

- **Client support** jumped with the advent of **Android P** (DoT support, enabled by default)

- Other end users can use, e.g. **getDNS Stubby**

- **Service providers** also **widely support it** (e.g. all public cloud resolvers)
Next steps in DoT

- **Improve performance** by supporting, e.g., out-of-order processing

- **More support** in built-in system **stub resolvers** (slowly arriving, e.g., systemd-resolved now has support)

- Also use **TLS on recursor to authoritative path**; but how do we make this work? How to build the trust relationship (is it even possible/necessary?)
Privacy conscious monitoring

• **Remember** that encrypting traffic makes monitoring harder

• In 2018/2019, **we developed a potential solution** to this: use of so-called **Bloom Filters**

• Tested in production at SURFnet (national research network in NL)
Bloom Filters

- Developed in the 1970s to speed up database lookups
- Highly efficient, insertion and lookup are $\sim O(1)$
- Bloom Filters are like a set with a probabilistic membership test
- For a given Bloom Filter $B$ and an element $n$, we can test the following:
  
  $n \in B$?
  
  - no $\rightarrow$ $n$ is guaranteed not to be in $B$
  - yes $\rightarrow$ $n$ is highly likely in $B$, with a small probability $p_\varepsilon$ of this being a false positive
Bloom Filters

www.example.com

(set of) hash function(s)

a029e8a9  c3faa9f8  cb745caa  8136503e  3a6dccaa  c9f4c130  574c0e58  7235970e

index #1  index #2  index #3  index #4  index #5  index #6  index #7  index #8

set bits to 1 in bit array using indices
Bloom Filters

- www.example.com
- www.example.org
- true-negative.net
- false-positive.org

**set elements**

- www.example.com
- www.example.org

**look up elements**

- true-negative.net
- false-positive.org
The idea

• Insert all queries from clients of a resolver into a Bloom Filter

• Then, we can check if a name was queried for, but not by whom and also not exactly when; this is sufficient for network-level threat monitoring

• Privacy properties of Bloom Filters:
  • Non-enumerable
  • By mixing queries from many users in a single filter, tracking becomes harder
  • Due to mathematical properties of Bloom Filters, we can combine different filters, so we can further aggregate data over time (making it even harder to track user)
Real-world tests

• We tested this for three weeks on busy DNS resolvers at SURFnet

• We studied three use cases:
  • Detection of so-called "Booters"
  • Hits on e-mail blacklists
  • Hits of high-value indicators-of-compromise for the so-called National Detection Network
National Detection Network

- NDN is **managed by** the Dutch National Cyber Security Centre (NCSC) and is supposed to have "high value" indicators-of-compromise (from *e.g.* intelligence services)

- SURFnet could previously **not monitor** for threats reported in NDN because monitoring **DNS traffic** was considered too privacy sensitive

- **With Bloom Filter approach** it was **now possible**, and we **found actual compromises** (*e.g.* WannaCry infected machine)
Future of Bloom Filter solution

- **First version** of code already released as open source
  https://github.com/SURFnet/honas

- **SURFnet** is planning to take this into production

- **Future integration** in NLnet Labs **open source software**
  to make this approach more widely available and easy to deploy

- **Proof** that security and privacy can go hand in hand!

---

UNIVERSITY OF TWENTE.
DNS over HTTPS

- Google had experimental "DNS over HTTPS" for ages; using their own REST protocol, seemed abandoned (nobody used it)

- Then an IETF draft was published, and things started moving... FAST!

- DoH working group formed in September 2017, draft adopted October 2017, RFC 8484 officially published October 2018

- Incredibly fast for the IETF; lot of momentum behind this idea
DoH basic outline

- DoH simply sends **Base64-encoded wire format DNS datagrams** over either **HTTP GET** or **HTTP PUSH**

- **Two modes** of operation:
  - **Dedicated**: the service end point only functions as a **DoH DNS resolver**
  - **Mixed**: DNS traffic is **mixed into other HTTP traffic**
  - **DoH server configured as** a **URI** end point in the client
DoH, where did it come from?

- Browser community wanted a web-style API to access DNS

- Argumentation browser community uses to push for it:
  - Enhance privacy of browser users (encrypted transport, mixing with HTTP traffic), arguing that adoption of e.g. DoT is too slow
  - Port 443 does not get blocked, so can circumvent traffic filtering
  - Improve user experience by reducing latency (really?!)
  - Longer term: new features (JSON, Server Push, "resolverless")
Issues with DoH

- The **rest of this talk will focus on issues with DoH** in several dimensions

- **Why?** Because DoH may have far-reaching consequences for the DNS and the Internet

- Dimensions we will look at:
  - Issues with privacy
  - Issues for network operators
  - Impact on the DNS name space
DoH and privacy

- **Proponents push** DoH arguing privacy; there are **issues with** that claim.

- **DoH imports** all of the **privacy issues of** the HTTP ecosystem into the DNS resolution process (**e.g. user agent profiling**), which has sparked a new Internet draft to address this.

- **DoH proponents** appear to **advocate** that a "**public trusted recursive resolver**" (TRR) is always better. This is **simply not true** in many cases, consider **e.g. EU citizens** who are **protected by** the GDPR in relation to their ISP.
NXDOMAIN hijacking

• Cited by DoH proponents as one of the "bad things" operators do

• Fun fact 1: Deutsche Telekom has a bit of a bad reputation in this regard but: GDPR + German law forbids monetising surfing behaviour of customers

• Fun fact 2: This is how OpenDNS (now Cisco Umbrella) initially made some of their money
DoH and privacy

- Mozilla is forcing DoH on users

- Mozilla has DoH support in Firefox since version 61, and enabled by default since version 69 and their default TRR is currently Cloudflare

- Other browsers still taking a different approach; Chrome supports DoH since version 78, but default is to only use DoH if the system-configured resolver is whitelisted, Safari does not support DoH and Apple does not have plans yet

- Users are highly unlikely to turn this off if it's the default, experience with users switching to 8.8.8.8 illustrates user inertia on this
Side step: user inertia viz. DNS

Graphs show Google Public DNS use in the Ziggo network (big ISP in NL) after a DoS attack on their resolvers.

Takeaway: once users change their config, they never go back

(graph from [1])

"Disable Protection"

- Mozilla's **approach for getting users to enable** DoH is **pretty drastic**

- Seriously, **who is going to click** "Disable Protection"?

- Sure, Cloudflare may have a good privacy policy now, but will it stay that way?
DoH and performance

• Remember DoH proponents cite "performance" as reason to deploy?

• Firefox put "classic DNS" and DoH side-by-side (blog here)

• Here are the weasel words from the blog:
  "The slowest 20% of DNS exchanges are radically improved [...], while the majority of exchanges exhibit a small tolerable amount of overhead when using a cloud service. This is a good result."

• A "small tolerable amount of overhead" is an average of 6ms per query!
DoH and network operators

• Where **DNS over TLS may require** operators to **re-think security monitoring**, DoH makes it impossible

• Use of **DoH circumvents any local security policy** for the DNS

• Use of **DoH is (almost) impossible to track**, especially in mixed mode

• **Security officers** can look forward to **having to wrangle browser configs for managed desktops** to disable DoH

• **Prediction:** Firefox will be banned on enterprise desktops
DoH and the DNS name space

- The **biggest** expected **impact may not be** the most **obvious**

- **Remember** that word "**resolverless**" a few slides back?

- Deployment of **DoH may radically change the DNS name space** as we know it

- **Why?**
DoH and the name space

- Browsers vendors and others have floated the idea of a "repository of TRRs" for looking up specific parts of the name space.

- Imagine a cabal very much like the CAB Forum for the X.509 Web PKI deciding on a common TRRs in browsers (and in the future OSes too).

- Suddenly, they decide how names are resolved.

- Who ever gave these folks the right to make this decision? What about the multi-stakeholder model for Internet governance?
DoH and the name space

- Imagine what this might mean!

- Parts of the name space are directly resolved through browser-embedded TRRs, circumventing the current DNS hierarchy

- Next step: ICANN and the current DNS hierarchy become obsolete

- What about the "level playing field"? How do I claim my name?

- Facilitates further centralisation of the Internet, and even stronger monopolies for certain big players
DoH and the name space

- **Current DNS operators** are **heavily invested in** an infrastructure that does **UDP** really well, **and** also handles a **bit of TCP**

- For **resolver operators**, it is relatively **simple to** also **support DoT**

- **DoH is a game changer**, it has a relatively **low bar of entry for players** that are already **heavily invested in** the **HTTP** ecosystem, but requires **major re-engineering for "traditional" DNS players**
Application Behavior Considering DNS (abcd)

No documents match your query.

First draft with potential consequences for the name space
What will the future look like?

- No reason to attribute malice to the browser folks, they are probably just trying to do what they think is "the right thing for privacy"

- That "right thing" may have unintended and irreversible side effects

- Because it is tilting thinking about how we view the name space

- This has not happened in earnest for over 30 years

- So we should be paying close attention!
What can/should you do?

• If you do not support **DNS over TLS** on your resolver: **turn it on!**

• **Consider running** a **DNS over HTTPS server**, to at least offer some diversity

  • This is **not simple**; there is **insufficient open source code available** to do this -- at NLnet Labs we are working on this (next slide)

• **GET INVOLVED IN THE DEBATE!** If you agree DoH has issues, speak up!
DoH in open source

- NLnet Labs will support **DoH in an upcoming Unbound release** funded by Mozilla Open Source Support foundation.

- We will also develop **web server plugins for Apache and NGINX** for **mixing DoH endpoints with regular web traffic** funded by Comcast Innovation Fund.

- Other DoH support: BIND, PowerDNS "dnsdist", Knot Resolver.
Thank you!

Questions?

NLnet Labs is hiring!

We are looking for a C developer for our open source DNS projects

nl.linkedin.com/in/rolandvanrijswijk
@reseauxsansfil
r.m.vanrijswijk@utwente.nl
roland@nlnetlabs.nl