DNS Privacy Tutorial

Shumon Huque
January 11th 2020
bdNOG11 Conference
Cox’s Bazar, Bangladesh
Timeline (continued)

• 2013: Edward Snowden Revelations
• 2013: IETF: RFC 6973: “Privacy Considerations for Internet Protocols”
• 2014: IETF: RFC 7258: “Pervasive Monitoring is an attack”
• 2014: DPRIVE (DNS Privacy) working group formed in IETF
• 2016: RFC7858: DNS over TLS (DoT)
• 2018: DNS over HTTPS (DoH)
• 2019: Web browsers implement DoH with Cloud DNS providers
RFC 7258

• “Pervasive Monitoring is an attack on the privacy of Internet users and organizations.”

• “… that needs to be mitigated where possible, via the design of protocols that make Pervasive Monitoring significantly more expensive of infeasible.”
DNS Privacy around 2013?

- Original DNS protocol largely unchanged from RFC 1034/1035 (1987) is still dominant
- UDP, with TCP used for fallback (when MTU is exceeded), and for specialized functions like zone transfer.
- No encryption or privacy protections
  - All DNS packets are sent in cleartext
- Prevailing expectation: DNS data are public, so no specific need for confidentiality.
- Note: DNSSEC does not provide confidentiality – its goal was data origin authentication (i.e. integrity of DNS data)
DNS Privacy - confidentiality

• Gradually, we realized that privacy was also very important.
  • Edward Snowden revelations (2013) had a big impact.

• DNS queries and responses are metadata, and this metadata can reveal important information about your communications.
  • e.g. the fact that you did a DNS lookup of a drug rehab site, may give away some clues about you and your intentions.
  • So, the DNS records for that site may be public info, but the fact that a specific user looked up those DNS entries should not be.

• Even without user meta data, traffic analysis is possible via many methods, e.g. timing & size measurements, cache snooping, etc.
DNS Privacy – data minimization (skip?)

- DNS Query Name Minimization (RFC 7816)
  - Limit the amount of information about domain names visible to authoritative servers.

- NXDOMAIN Cut (RFC 8020) & Aggressive Negative Caching (RFC 8198)
  - Increase the scope of local negative caching.
  - Minimize leakage of queries for non-existing domain names from the recursive server to other authoritative servers.
  - (won’t discuss any further today – refer to RFCs for additional details)
The *full* query name is being sent to both the Root and the COM authoritative DNS servers, which is more than they need to know.
Query-name Minimization!

Full Query name visible only at the final target zone.
DNS Privacy – anti data minimization trends

- CDNs and the EDNS Client Subnet Option
Content Delivery Networks want to know the IP address or subnet of the client, so that they can produce an answer best suited for the location of the client system. Use of public resolvers that may be far away from the actual client thus pose a problem.
Solution: the recursive server includes a “Client Subnet” option (ECS) in outbound queries to Auth servers. Many consider this a privacy leak.
DNS Privacy – confidentiality

• Protocols:
  • DNS over TLS (DoT)
  • DNS over HTTPS (DoH)
  • DNS over QUIC (DoQ?, future looking, not ready yet)

• Ecosystem scope of operation
  • Stub resolver to Recursive server
  • Recursive server to Authoritative server
DNS Privacy – confidentiality

• Some non-standardized options have existed for some time:
  • **DNSCurve** (D.J. Bernstein)
    • Encrypts recursive to authoritative path. No adoption really
  • **DNSCrypt** (OpenDNS and others)
    • Encrypts stub to recursive server. Some implementations

• No efforts to standardize these have been made in bodies like the IETF, so they are unlikely to see wide adoption.
DNS Privacy – side channel protection

- Padding queries and responses to avoid size based side channels:
  - RFC 7830: EDNS0 Padding Option
DNS over TLS (DoT) and HTTPS (DoH)

• The DNS protocol remains mostly unchanged, apart from the transport of DNS messages (and authentication of DNS servers).
• DNS over TLS: send DNS messages over TLS
• DNS over HTTPS: send DNS messages over HTTPS (HTTP over TLS)
• DNS over TLS came first, and has seen some marginal deployment
• DNS over HTTPS, came only recently, but is already causing dramatic and fast moving changes across the ecosystem!
  • Pushed by browsers, where a small number of companies dominate the market and can cause quick, far-reaching changes!
DNS over TLS (DoT)

- RFC 7858: Specification for DNS over Transport Layer Security (TLS)
- New dedicated port: 853/tcp
DNS over HTTPS (DoH)

- RFC 8484: DNS Queries over HTTPS (DoH)
- Send DNS over HTTPS (i.e. HTTP over TLS)

- TLS already provides transport encryption and server authentication.
- HTTPS is clearly additional overhead
- So what is the actual benefit/advantage over DoT?
DNS over HTTPS (DoH) Motivations

• Primary proponents are some of the big web browser vendors (Mozilla Firefox & Google Chrome)

• 2 Primary motivations:
  • Allowing web apps to access DNS info via existing browser APIs
  • Preventing on-path devices from interfering with DNS operations
DNS over HTTPS (DoH) Motivations

• Primary proponents are some of the big web browser vendors (Mozilla Firefox & Google Chrome)

• 2 Primary motivations:
  • Allowing web apps to access DNS info via existing browser APIs
  • Preventing on-path devices from interfering with DNS operations

• Runs on the same port as Web/HTTPS, so makes it possible to **comingle Web and DNS at the same server addresses**, in a way that makes it difficult to identify, inspect, block DNS traffic, without collateral damage! (will come back to this later)
DoH vs Network Operators tussle

- Preventing on-path devices from interfering with DNS operations
  - Actually, not just “interfering”, but also “inspecting”
- But network operators (e.g. for corporate networks, campus networks etc) often want to inspect their DNS (and other traffic) as part of their network management/monitoring/security strategy.
- DNS filtering is common for malware/abuse protection, parental controls, etc.
  - DoH may make this difficult or impossible, depending on how it is deployed.
- Others: Split view/Internal DNS
Current Scope: Stub to Recursive

• The current scope of both DoT and DoH, is encrypting the path between the “Stub Resolver”, i.e. client system and the “Recursive Server” only.

• Rationale:
  • This is where DNS queries for individual users can be most easily identified, so it is deemed the most important.
  • Baby steps; don’t boil the ocean from the start.
Current Scope of DoT and DoH
Future Scope: Stub to Recursive

• The next task is to encrypt the path(s) between the Recursive Server and the Authoritative Servers

• Work on this leg is already in progress in the IETF, but will take some time to complete

• In the interim, some early movers have already started some experimental deployments in this space
  • Cloudflare and Facebook experiment
Phase 2 of DoT and DoH

stub (client)

www.amazon.com

recursive

www.amazon.com

referral to .com

www.amazon.com

referral to amazon.com

www.amazon.com

answer: 1.2.3.4

auth: .com

auth: . (root)
Cloudflare & Facebook experiment (there are others)
Note: this is DNS over TLS (not HTTPS)
DoH announcements

• May 2019: Mozilla Firefox DoH experiment
  • https://bugzilla.mozilla.org/show_bug.cgi?id=1552783

• Sep 2019: Mozilla Firefox: Making DoH the default (later: in US only):
Hot topic: Many other DoH announcements

• Oct 2019
  • US ISP Comcast announces DoH and DoT trials

• Nov 2019: Microsoft announces DoH plans in Windows:

• Dec 2019: Google Chrome DoH auto-upgrade experiment:
  • https://groups.google.com/a/chromium.org/forum/#!msg/net-dev/lIm9esAFjQ0/vJ93oMbAAgAJ

• Dec 2019: UK ISP BT announces DoH Trial:
Mozilla plans and experiments

• Select Trusted Recursive Resolvers (TRR):
  • Cloudflare (1.1.1.1)
  • There is 1 more now, but Cloudflare is the default

• TRR Policies (some key ones)
  • Strictly limit data collection; do not sell/monetize/transfer to other parties
  • Must not filter DNS queries/responses
  • Must use Query Name Minimization
  • Must not use EDNS Client Subnet, unless resolver-auth path encrypted

• Firefox itself, will send DNS queries using DoH to this TRR, bypassing the system’s local DNS resolver completely.
Mozilla plans and experiments

• Concession to managed network operators
  • “use-application-dns.net” canary domain
  • Allows network operators to disable DoH for users on their network
• Initial plan was to default *all* Firefox users to using DoH to Cloudflare
• Backlash from various quarters.
• Now will default only US users.
Mozilla implementation shortcomings

• Doesn’t require that TRR implement DNSSEC validation.

• In fact, Firefox falls back to local resolver on DNSSEC validation failure by Cloudflare!

• Falls back to local resolver if name doesn’t resolve via DoH
  • To allow split-horizon to partially work. Doesn’t work if same name exists in both inside and outside views.

• use-application-dns.net easily spoofed by adversarial network. Users expecting DNS privacy may be in for a rude surprise.
DoH and Deployment Models

• Distinguish between DoH, the protocol, and the manner (model) in which DoH is being deployed.

• Local vs Cloud DoH server deployment

• System wide DoH resolver function vs Application specific DoH function.

• Mozilla/Cloudflare TRR model is: Cloud DoH server and Application-specific (i.e. browser) DoH function.
Back to this diagram ...

Current Scope of DoT and DoH

stub (client)

DoH Recursive

1. www.amazon.com
2. www.amazon.com referral to .com
3. www.amazon.com referral to .com
4. www.amazon.com
5. www.amazon.com referral to amazon.com
6. www.amazon.com
7. www.amazon.com answer: 1.2.3.4
8. www.amazon.com answer

auth: .(root)

auth: .com

auth: amazon.com
Critical question:

Who runs this DoH Recursive Server?
Model: DoH Recursive operated by the local network

Campus/Corporate/ISP Network

stub (client)  DoH Recursive

Authoritative DNS Servers on the Internet

Local network operator cannot monitor DNS traffic on the wire, but can inspect, filter, block it at the DoH server.
Model: DoH Recursive operated by an external service such as Cloudflare

Local network operator can only block DNS traffic to the cloud provider. That may be difficult if it is co-located with a popular web service.
Is DNS a good control point for this?

• My view: not really. But managed network operators have used it in this way forever, and some are really upset.

• Ideally, monitoring and control is best achieved at the endpoints themselves:
  • “Moving control to the endpoints: Motivations, challenges, and the path forward” – M. Nottingham
  • We may be heading in this direction, but it won’t be easy or smooth (capital costs, rearchitecture, IoT and BYOD challenges, etc.)
Not a new issue

• Long TLS 1.3 debate about preventing on-path transparent TLS interception proxies.
Who do you want to trust?

• Local network operator?
• ISP?
• Internet application service provider? (esp. US tech giants)

• There are wildly different views about who is more trustworthy.
Regulation?

• Many non-US folks see Mozilla’s plans as distinctly US centric, where (some) ISPs are known to spy on/monetize DNS traffic.

• In Europe, ISPs are strongly regulated and aren’t permitted to do this. Furthermore, there are strong privacy regulations, like GDPR that apply, not only to ISPs, but across the board.

• In their view (and actually many others), it is US internet companies that can’t really be trusted.

• Should cloud based DoH providers be regulated also?
Centralization Concerns

• If all (US) Firefox users centralize their DNS on Cloudflare, is that a good thing?

• A huge part of the strength and resilience of the Internet has come from its decentralized nature. That has been slowly changing, with the rise of CDNs and Big Tech companies.
Google Chrome plans?

- The largest browser by market share
- Taking a more conciliatory approach to network operators
- Won’t default users to using DoH, and won’t default to directing DoH traffic to their own Public Resolver (politically wise for them)
- Will try to detect DoH support in configured local resolver and then automatically upgrade to using DoH with it.
- Dec 2019: Google Chrome DoH auto-upgrade experiment:
  - https://groups.google.com/a/chromium.org/forum/#!msg/net-dev/lIm9esAFjQ0/vJ93oMbAAgAJ
New negative tracking potential

• Something to think about..

• TLS and HTTPS introduce new protocol elements into the DNS that may make it easier to track people/devices across locations:
  • TLS session resumption tickets
  • HTTP headers
Protecting dissidents and whistleblowers

• A section of the DoH advocacy camp insists that DoH is needed to offer real protection to folks like political dissidents and corporate or government whistleblowers.

• If so, it must be bulletproof against compromise, with no fallbacks to insecure modes. Not true for Firefox DoH.

• And critically, we need to comprehensively plug *all* privacy leaks of domain names, not just the ones seen in DNS query and responses!
Comprehensively plugging DNS privacy leaks

• Encrypt TLS SNI extension, which currently carries the server hostname in cleartext (ESNI spec is coming).

• Disabling OCSP checking by client, and having servers staple OCSP responses in their encrypted handshake message.

• Hiding in the crowd, in large co-located services, or access service through a fronting server.
  • Otherwise IP address alone is often enough to identify the service name.

• Maybe we really need true Anonymity networks: Tor, and mix networks? DoH is decidedly a partial solution.
Technology alone is seldom enough

From: https://xkcd.com/538/
Performance & Scaling Considerations
Performance & Scaling Considerations

• DNS over TLS/HTTPS is much more computationally expensive than plain DNS

• Recommendations:
  • Persistent Connections
  • Pipelining
  • Out of Order processing
  • TCP Fast Open
  • TLS Session Resumption (without server side state)
  • 0-RTT in TLS 1.3?
Setting up Certificates for TLS
Setting up Certificates

• Self-signed is the quickest, but not secure, and many clients doing strict server authentication will fail.
• LetsEncrypt offers free certificates that are (relatively) easy to acquire
  • But some things need to be setup correctly.
Getting a certificate from LetsEncrypt

- Certbot
  - The normal way, if you have a real webserver
  - A different way if you are actually deploying DNS over TLS/HTTPS
    - Use Certbot’s standalone mode to bootstrap, answering HTTP challenges on port 80
Creating a self-signed SSL certificate, putting the private key and certificate portions in separate files: Note for self-signed cert, you need to specify "-x509" and "-days NNN" with "openssl req".

```bash
openssl req -nodes -new -x509 \
    -newkey rsa:2048 -sha256 \
    -out server.crt -keyout server.key -days 3650
```

Will create server.key and server.crt

Example Subject Name (for rosetta):
Country        US
State/Province Pennsylvania
Locality      Philadelphia
Organization  University of Pennsylvania
Organizational Unit  ISC Networking
Common Name   rosetta.upenn.edu
Better tool: mkcert (written in Go – need to build with Go compiler)

https://github.com/FiloSottile/mkcert

$ mkcert host.example.com

To install root certs too (to avoid certificate errors)

$ mkcert -install
Ubuntu Packages for labs
Ubuntu packages you may need to install for some of the labs.

unbound
getdns-utils
gcc
g++
pkg-config
libedit-dev
lua5.3
lua5.3-dev
libssl-dev

>> golang for mkcert
Go 1.13.x
libnss3-tools
Deploying DoT
DNS over TLS implementations

• Client side:
  • Android
  • systemd-resolved (TLS disabled by default)
  • Getdns project and Stubby
  • Misc diagnostic tools: kdig, getdns, ...

• Server:
  • Public DoT services from Cloudflare, Google, Quad9, etc
  • Unbound
  • Knot Resolver
  • dnsdist (DNS proxy from PowerDNS)
  • Nginx (TLS proxying of DNS)
  • [ISC BIND: expected in late 2020; implementation funded by Moz/Cloudflare]
Unbound DoT server

- Has supported TLS natively for quite a while

Main configuration directives involved:

```plaintext
interface: 0.0.0.0@853  # listen on IPv4
interface: ::@853       # listen on IPv6

tls-service-key: "/path/to/privatekey.pem"  # private key
 tls-service-pem: "/path/to/certificate.pem"  # certificate
tls-port: 853           # do TLS on this port
```
Stubby/getdns DoT client

• From the ”getdns” project

• A local stub resolver application, that sends client DNS requests over TLS to an upstream DNS over TLS server.

• Details:
  • https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Daemon+-+Stubby
  • https://github.com/getdnsapi/stubby
  • https://dnsprivacy.org/wiki/display/DP/Configuring+Stubby
systemd-resolved

- DNS resolver service integrated into system
- Need a pretty recent version for DoT support
- Configuration file: /etc/system/resolved.conf
- Add: “DNSOverTLS=opportunistic”
- Then: sudo systemctl restart system-resolved

See:
  - https://thottingal.in/blog/2019/02/08/how-to-setup-dns-over-tls-using-systemd-resolved/
kdig

• Command line DNS query tool from the Knot DNS project, by CZ.NIC
Kdig command line options:

+tls // TLS with opportunistic security
+tls-ca[=FILE] // TLS with certificate validation
+tls-pin=BASE64 // TLS with server key pin
+tls-hostname=STR // TLS with specified hostname to check
Query google.com with DoT at 1.1.1.1 (Cloudflare DoT service):

$ kdig @1.1.1.1 +tls google.com

;; TLS session (TLS1.2)-(ECDHE-X25519)-(ECDSA-SHA256)-(AES-256-GCM)
;; -->HEADER<<- opcode: QUERY; status: NOERROR; id: 7040
;; Flags: qr rd ra; QUERY: 1; ANSWER: 1; AUTHORITY: 0; ADDITIONAL: 1

;; EDNS PSEUDOSECTION:
;; Version: 0; flags: ; UDP size: 1452 B; ext-rcode: NOERROR

;; QUESTION SECTION:
;; google.com. IN A

;; ANSWER SECTION:
google.com. 7 IN A 172.217.10.78

;; Received 55 B
;; Time 2020-01-05 15:02:50 EST
;; From 1.1.1.1@853(TCP) in 1.6 ms
getdns

• [TBD: advanced DNS library project]
• getdns_query – standalone query & debugging tool.

• Ubuntu:
  • sudo apt-get install getdns-utils
Getdns_query

Relevant options:

@<ip>[@<scope_id>][@<port>][#<tls port>][~<tls name>]

- s Set stub resolution (default: full recursion)
- L TLS transport only
- E TLS with fallback to TCP
- l Set Transport list (UTL)

- m TLS authentication required
- n TLS authentication not required (default)
Getdns_query example

getdns_query -s @1.1.1.1 -L facebook.com. A

getdns_query -s @1.1.1.1~one.one.one.one -m -L www.huque.com. A
dnsdist configuration

- An example configuration that runs a normal DNS server bound only to the loopback interface, and then runs a dnsdist proxy server on the normal interface that listens for DoT and DoH..
Installing dnsdist 1.4 package from powerdns repo:

See https://repo.powerdns.com/ for Ubuntu 18.04:

dnsdist - version 1.4.X
Create the file '/etc/apt/sources.list.d/pdns.list' with this content:

deb [arch=amd64] http://repo.powerdns.com/ubuntu bionic-dnsdist-14 main
And this to '/etc/apt/preferences.d/dnsdist':

Package: dnsdist*
Pin: origin repo.powerdns.com
Pin-Priority: 600
and execute the following commands:

curl https://repo.powerdns.com/FD380FBB-pub.asc | sudo apt-key add - &&
sudo apt-get update &&
sudo apt-get install dnsdist
$ cat /usr/local/etc/dnsdist.conf

newServer("127.0.0.1")
setACL({'0.0.0.0/0', '::/0'})
addTLSLocal('0.0.0.0',
    '/usr/local/etc/certs/doth.huque.com/fullchain.pem',
    '/usr/local/etc/certs/doth.huque.com/privkey.pem')
addTLSLocal('::',
    '/usr/local/etc/certs/doth.huque.com/fullchain.pem',
    '/usr/local/etc/certs/doth.huque.com/privkey.pem')
Nginx TLS/DNS proxy

- Using Nginx as a DoT/DoH gateway:
Web based debugging

• https://getdnsapi.net/query/
Deploying DoH
DNS over HTTPS implementations

• Client side:
  • Mainly browsers: Mozilla Firefox, Google Chrome
  • Misc diagnostic tools: curl, ...

• Server:
  • Public DoH services from Cloudflare, Google, Quad9, etc
  • Knot Resolver
  • dnsdist (DNS proxy from PowerDNS)
  • Nginx (HTTPS proxying of DNS)
  • [ISC BIND: expected in late 2020; implementation funded by Moz/Cloudflare]
Public DoH Services

• A list:
  • [https://github.com/curl/curl/wiki/DNS-over-HTTPS#publicly-available-servers](https://github.com/curl/curl/wiki/DNS-over-HTTPS#publicly-available-servers)

• Many, including Google, Cloudflare, Quad9, Comcast, OpenDNS, Cox, NextDNS, PowerDNS
Firefox DoH

• Details:
• “about:studies” → “DNS over HTTPS US Rollout”
• Manually enable:
  • Preferences -> General -> Network Settings: Enable DNS over HTTPS”
    • Select “On”
    • Select a provider or setup a custom provider (default Cloudflare) – sets network.trr.mode to 2
• ”about:config”
  • “network.trr.mode” 0 (off), 5 (off by user choice)
Firefox DoH

• Excluding specific domains:
  • “about.config”
  • ”network.trr.excluded-domains” -> Edit button
  • Add domains separated by commas to the list
  • Includes sub domains

• Firefox DoH uses a Mozilla specific Cloudflare DNS server name:
  • mozilla.cloudflare-dns.com
  • 2606:4700::6810:f9f9
  • 2606:4700::6810:f8f9
  • 104.16.248.249
  • 104.16.249.249
Configure networks to disable Firefox/DoH

• Details:
Microsoft has announced DoH

• “Windows will improve user privacy with DNS over HTTPS”:
curl-doh

• Command line DNS over HTTPS query tool from the author of curl
  • https://github.com/curl/doh
Curl-doh

$ doh www.example.com
www.example.com from https://dns.cloudflare.com/dns-query
TTL: 2612 seconds
A: 93.184.216.34

$ doh www.yahoo.com https://dns.google/dns-query
www.yahoo.com from https://dns.google/dns-query
TTL: 36 seconds
A: 87.248.98.8
A: 87.248.98.7
AAAA: 2a00:1288:0110:001c:0000:0000:0000:0004
AAAA: 2a00:1288:0110:001c:0000:0000:0000:0003
CNAME: atsv2-fp.wg1.b.yahoo.com
CNAME: atsv2-fp.wg1.b.yahoo.com
dnsdist configuration

• An example configuration that runs a normal DNS server bound only to the loopback interface, and then runs a dnsdist proxy server on the normal interface that listens for DoT and DoH ..
$ cat /usr/local/etc/dnsdist.conf

newServer("127.0.0.1")
setACL({'0.0.0.0/0', '::/0'})
addTLSLocal('0.0.0.0',
    '/usr/local/etc/certs/doth.huque.com/fullchain.pem',
    '/usr/local/etc/certs/doth.huque.com/privkey.pem')
addTLSLocal('::',
    '/usr/local/etc/certs/doth.huque.com/fullchain.pem',
    '/usr/local/etc/certs/doth.huque.com/privkey.pem')
addDOHLocal('0.0.0.0',
    '/usr/local/etc/certs/doth.huque.com/fullchain.pem',
    '/usr/local/etc/certs/doth.huque.com/privkey.pem',
    '/dns-query')
addDOHLocal('::',
    '/usr/local/etc/certs/doth.huque.com/fullchain.pem',
    '/usr/local/etc/certs/doth.huque.com/privkey.pem',
    '/dns-query')

Will serve DoH at: https://doth.huque.com/dns-query
Nginx HTTPS/DNS proxy

- Using Nginx as a DoT/DoH gateway:
Questions and/or comments?
Extra Slides (Time Permitting)
DoT/DoH vs DNSSEC?

• Mistaken assumption I’ve heard from some:
  • With DoT/DoH there is no longer any need for DNSSEC
  • This is **wrong**.

• DoT/DoH provide channel security (secure transport) of DNS messages to/from the DoT/DoH Recursive Server.

• Unless the DoT/DoH Recursive Server performs DNSSEC validation of responses that it receives, it’s cache can be poisoned with bogus responses, and it will happily relay those bogus responses downstream to its clients.
DoT/DoH vs DNSSEC?

• Adding DoT/DoH between the Recursive and Authoritative strengthens the picture a bit, but still does not obviate DNSSEC.

• You need to be sure that you are connecting to the right authoritative server for the zone, and it needs to be done at *all* layers of the DNS authoritative hierarchy.

• DNSSEC employs an object security model, that doesn’t require you to make sure you securely connected to the right authority servers at all levels of the hierarchy. If using pre-computed signatures, with the signing server offline or inaccessible to the Internet, it even protects you against authoritative servers that have been compromised.
DNS over QUIC?

• Possibly in our future.
• QUIC is a new transport protocol:
  • Over UDP (faster protocol evolution)
  • Multi-streaming with no head-of-line blocking, etc.
• HTTP3 is already designed to run over QUIC transport, so if DoH evolves to DNS over HTTP(3)S, then it will have moved to QUIC anyway.
• But there are some other compelling reasons to design DNS to run directly over QUIC.
• Early work: