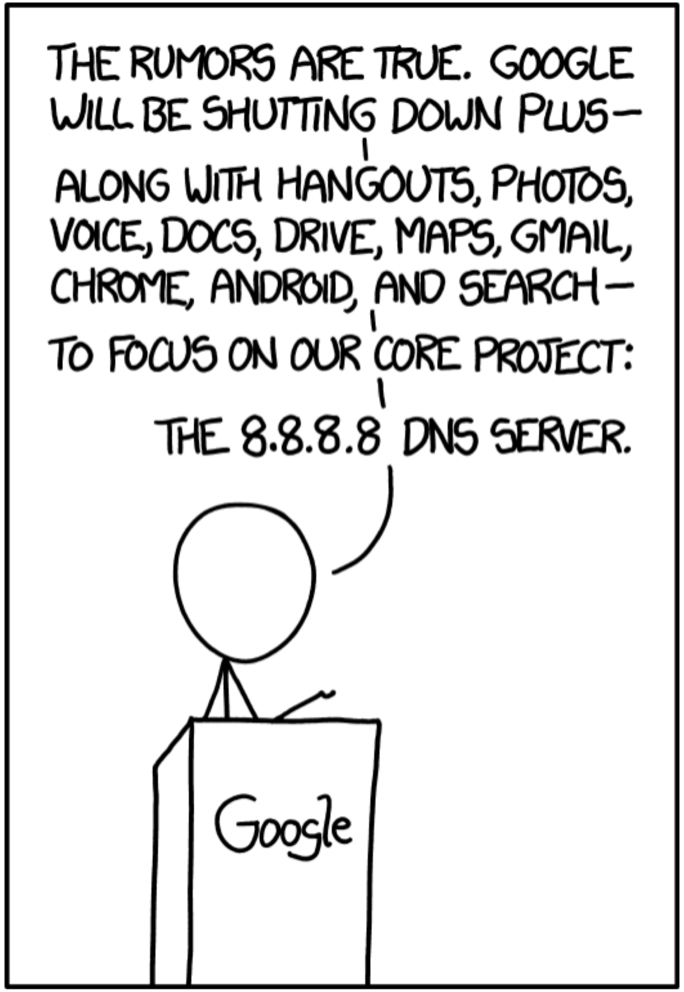
DNS Privacy

Geoff Huston
APNIC


THE RUMORS ARE TRUE. GOOGLE WILL BE SHUTTING DOWN PLUS—
ALONG WITH HANGOUTS, PHOTOS, VOICE, DOCS, DRIVE, MAPS, GMAIL, CHROME, ANDROID, AND SEARCH—
TO FOCUS ON OUR CORE PROJECT:
THE 8.8.8.8 DNS SERVER.
Google

https://xkcd.com/1361/

Why?

- Because everything you do on the net starts with a call to the DNS
- If we could see your stream of queries in real time we could assemble a detailed profile of you and interests and activities
- Do we have any evidence of DNS data mining?
  - Data miners don't disclose their sources as a rule

Why?

- Because everything you do on the net starts with a call to the DNS
- If we could see your stream of queries in real time we could assemble a detailed profile of you and interests and activities
- Do we have any evidence of DNS data mining?
  - Data miners don't disclose their sources as a rule
- How about something related:
  - Do we have any evidence of DNS stalking?

What if…

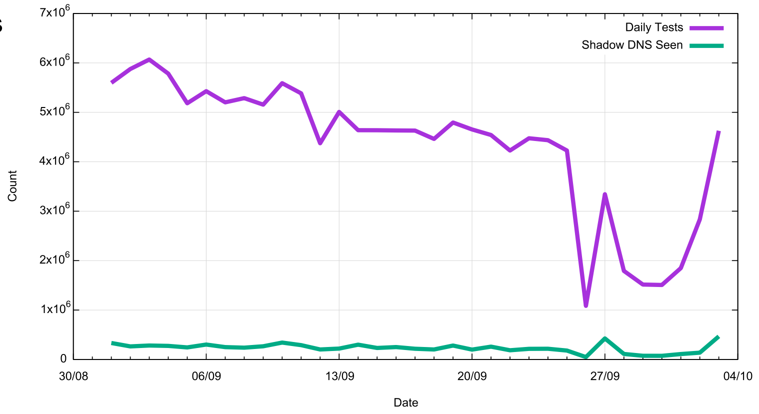
- I gave you an absolutely unique name to resolve:
  - The name never existed before now
  - The name will never be used again
  - The name includes the time when the name was created
- If I am the authoritative server for the name's zone then I should see your efforts to resolve the name
- Then I should never see the name as a resolution query ever again

  Unless you have attracted a digital stalker who performs re-queries of your DNS names!

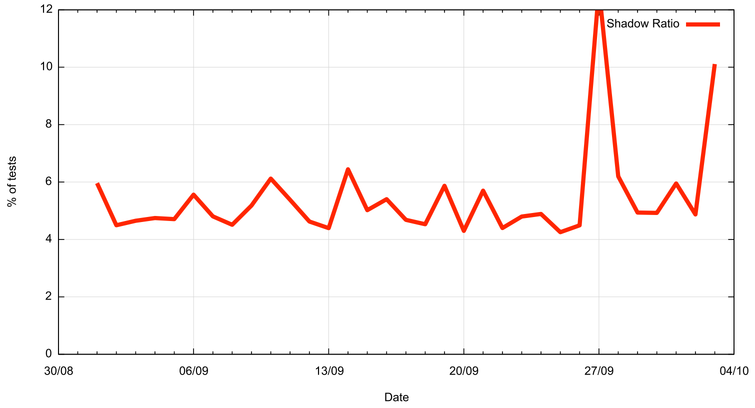
DNS Re-query Rate

- Over 30 days, some 5.3% of users had some kind of DNS stalker that is asking the same query more than 30 seconds after the initial query
- Only some of these ‘ghost’ queries could be explained by incredibly slow DNS resolver systems

Daily Counts

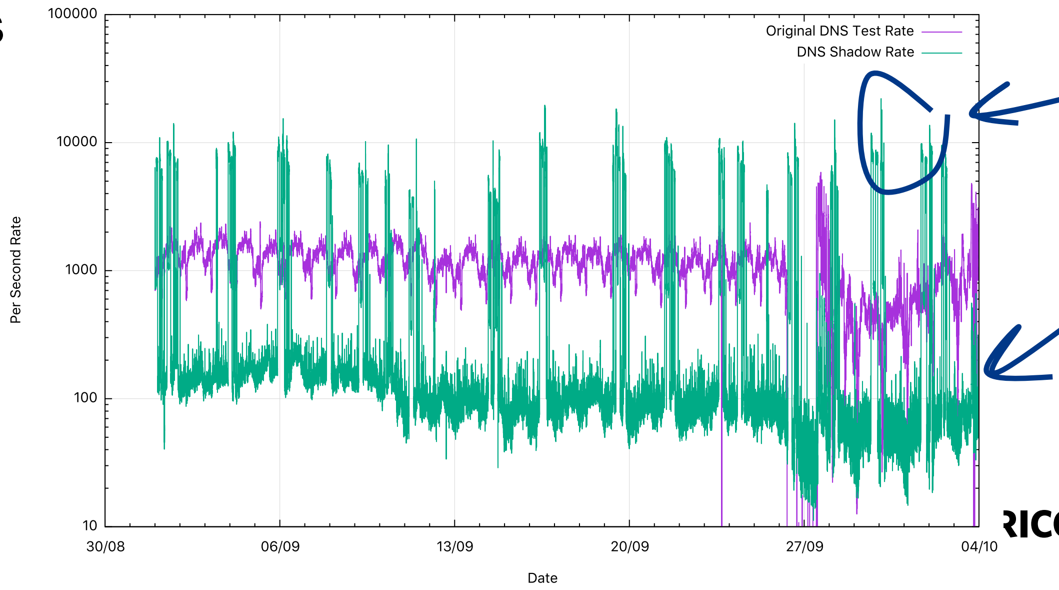


Daily Ratios



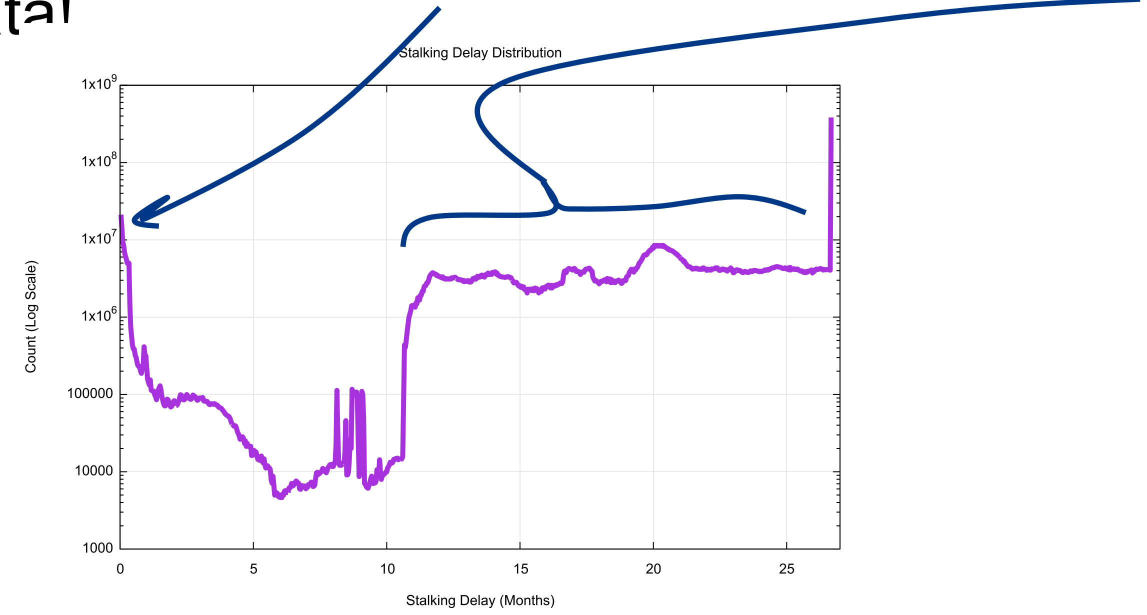
DNS Stalking

- There are two kinds of DNS stalkers
  - Rapid tracking stalkers that appear to track users in real time
  - Bulk replay stalkers that replay DNS queries at a very high rate in bursts



DNS Stalking

DNS Stalking uses both really recent data and more than year-old data!



DNS Surveillance

- The DNS appears to be used by many actors as a means of looking at what we do online and censoring what services we can access online

DNS Surveillance

- The DNS appears to be used by many actors as a means of looking at what we do online and censoring what services we can access online

- Can we stop DNS surveillance completely?
  - Probably not!

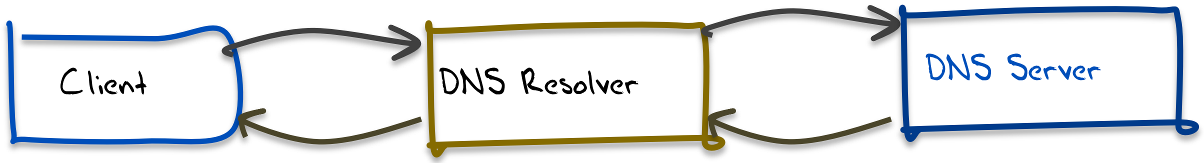
- Can we make it harder to collect individual profiles of activity?
  - Well, yes
  - And that's what I want to talk about today

The DNS Privacy Issue

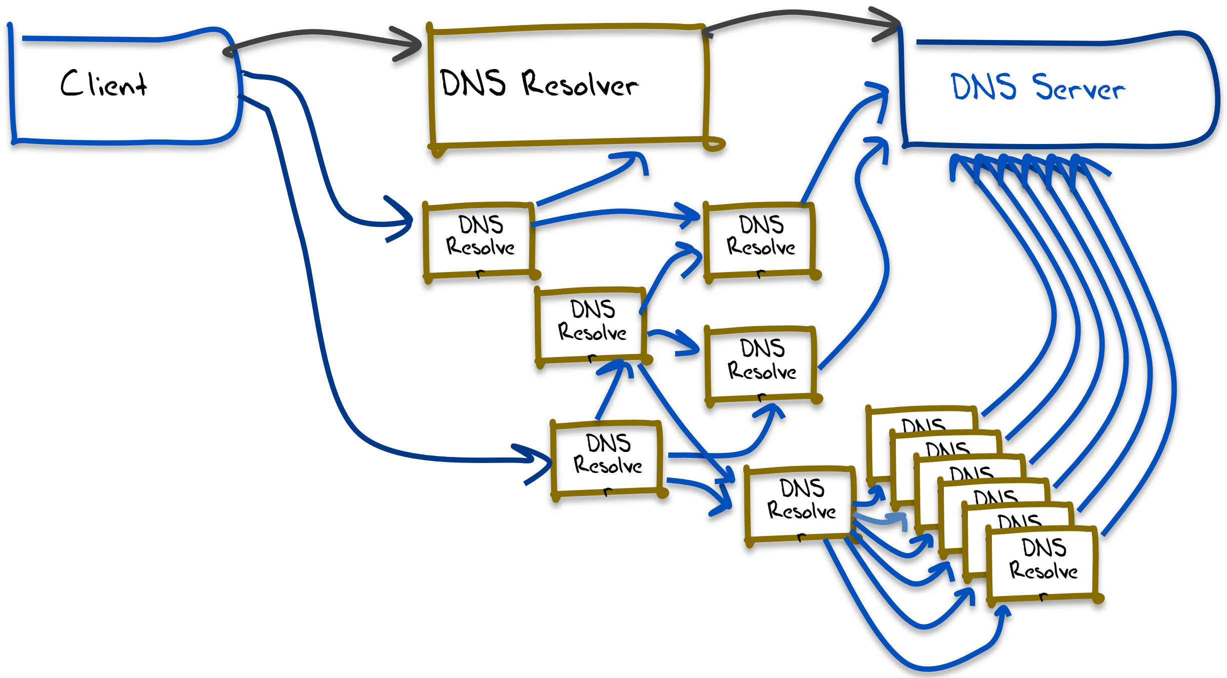
- Lots of actors get to see what I do in the DNS
  - My OS platform provider
  - My ISP's recursive resolver
  - Their forwarding resolver, if they have one
  - Authoritative Name servers
  - Snoopers on the wire

- Can we make it harder for these "others" to snoop on me?

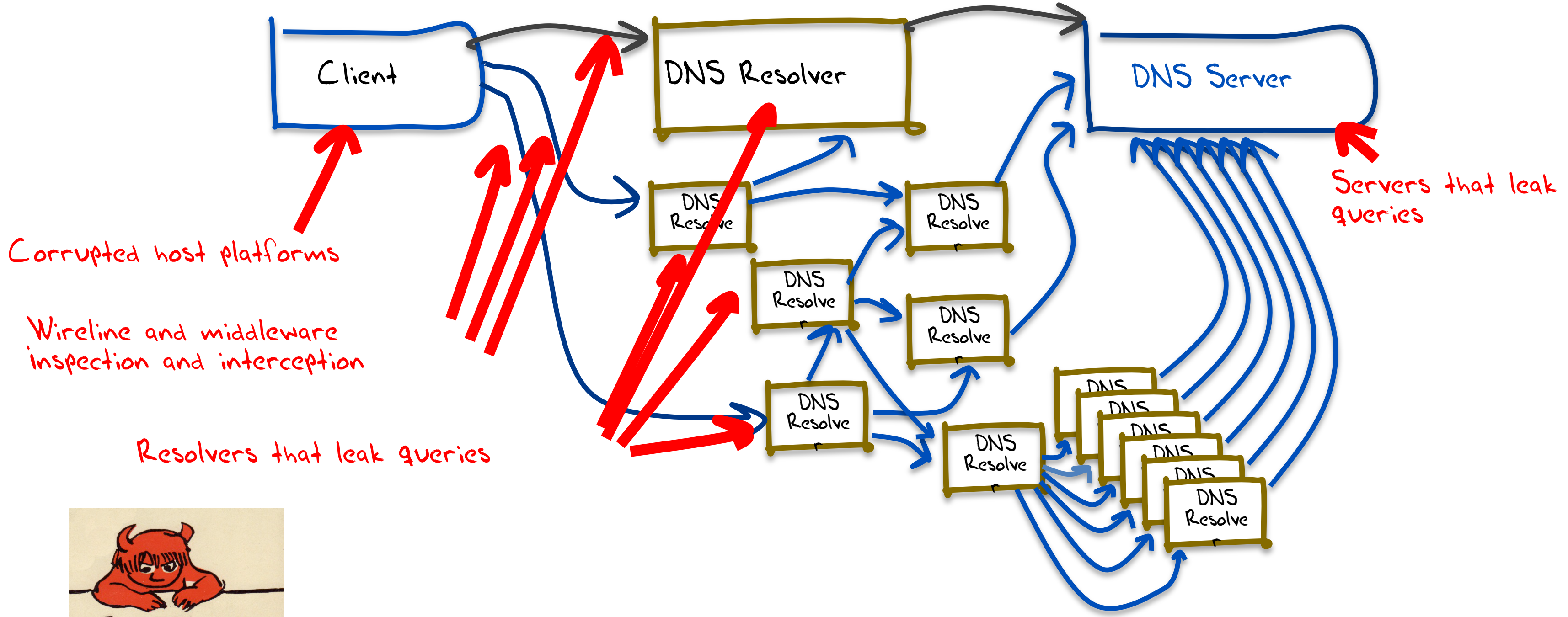
How we might think the DNS works



What we suspect the DNS is like



What we suspect the DNS is like



Why pick on the DNS?

- The DNS is very **easy to tap**
  - Its open and unencrypted
- DNS traffic is **easy to tamper with**
  - Its payload is not secured and tampering cannot be detected
  - Its predictable and false answers can be readily inserted
- The DNS is **hard to trace**
  - Noone knows exactly where their queries go
  - Noone can know precisely where their answers come from

Second-hand DNS queries are a business opportunity these days

FARSIGHT SECURITY

Solutions Resources Blog Partners Community Company

IDC

Farsight Security: Providing Real-Time DNS Data to Threat Intelligence

IDC Report:

Farsight Security - Providing Real-Time DNS Data to Threat Intelligence

EVERYTHING STARTS WITH DNS

LATEST NEWS

How ThreatConnect®

New Research on Domain

IPs, Address Ranges, a

How can we improve DNS Privacy?

- Lets look at a few behaviours of the DNS and see what we are doing to try and improve its privacy properties

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

Hi .com server, I want to resolve www.example.com

Not me – try asking the servers for example.com

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

Hi .com server, I want to resolve www.example.com

Not me – try asking the servers for example.com

Hi example.com server, I want to resolve www.example.com

Sure – its 93.184.216.34

The DNS is overly chatty

The DNS uses the full query name to discover the identity of the name servers for the query name

Why are we telling root servers all our DNS secrets?

In our example case, both a root server and a .com server now know that I am attempting to resolve the name www.example.com

Maybe i don't want them to know this!

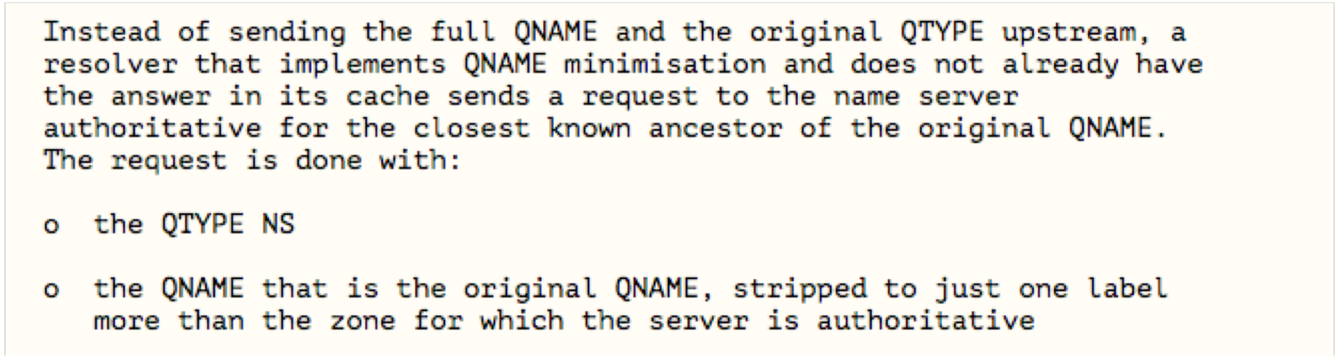
The DNS is overly chatty

Is there an alternative approach to name server discovery that strips the query name in iterative search for a zone's servers?

- Yes – the extra information was inserted into the query to make the protocol simpler and slightly more efficient in some cases
- But we can alter query behaviour to only expose as much as is necessary to the folk who need to know in order to answer the query

QNAME Minimisation

- A resolver technique intended to improve DNS privacy where a DNS resolver no longer sends the entire original query name to the upstream name server
- Described in RFC 7816



```
Instead of sending the full QNAME and the original QTYPE upstream, a
resolver that implements QNAME minimisation and does not already have
the answer in its cache sends a request to the name server
authoritative for the closest known ancestor of the original QNAME.
The request is done with:

o  the QTYPE NS

o  the QNAME that is the original QNAME, stripped to just one label
   more than the zone for which the server is authoritative
```

Example of QNAME Minimisation

Ask the authoritative server for a zone for the NS records of the next zone:

Hi Root server, I want to know the nameservers for com

  Sure, here are the servers for .com

Hi .com server, I want to know the nameservers for example.com

  Sure, here are the servers for example.com

Hi example.com server, I want to resolve www.example.com

  Sure – its 93.184.216.34

Interception and Rewriting

- The DNS is an easy target for the imposition of control over access
  - Try asking for www.thepiratebay.org in Australia
  - Try asking for www.facebook.com in China
  - Etc etc
- These days interception systems typically offer an incorrect response
- How can you tell is the answer that the DNS gives you is the genuine answer or not?

DNSSEC

- DNSSEC is defined in RFCs 4033, 4034 & 4035
  - Adds a number of new RRtypes that allow a digital signature to be attached to RRsets in a zone and to define how keys that are used to generate signatures are also stored in the zone
- DNSSEC validation of the DNS response can tell you if the response is genuine or if it is out of date or has been altered
- DNSSEC can't tell you what the "good" answer is, just that the answer you got was not it!
- DNSSEC will also tell if is an NXDOMAIN response is authentic

DNSSEC and Recursive Resolvers

- A DNS response that has been modified will fail to validate.

  When:
  - a client asks a security-aware resolver to resolve a name, and
  - sets the EDNS(0) DNSSEC OK bit, and
  - the zone is DNSSEC-signed

  then the recursive resolver will only return a RRset for the query if it can validate the response using the attached digital signature

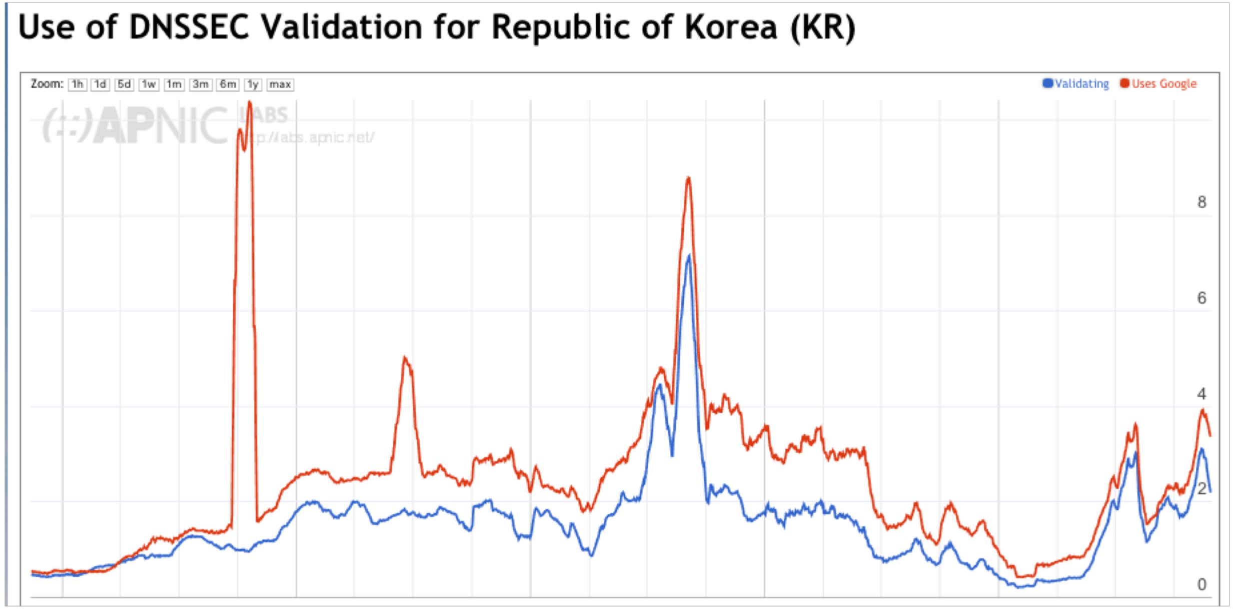
  It will set the AD bit in the resolver response to indicate validation success

  Otherwise it will return SERVFAIL
- But SERVFAIL is not the same as "I smell tampering"
- Its "nope, I failed. Try another resolver"

DNSSEC and Recursive Resolvers

- If you are going to use a DNSSEC-validating recursive resolver
  - Such as 1.1.1.1, 8.8.8.8, 9.9.9.9 or any other validating open resolver
- Then make sure that all your resolvers perform DNSSEC validation if you don't want to be mislead
  - Because SERVFAIL from a validating resolver means "try the next resolver in your resolver list"

DNSSEC in Korea



Negative Chattiness

- Names that do not exist in the DNS are also passed to the authoritative servers in order to return the NXDOMAIN response
- Sometimes the questions we ask are informative, whether or not the DNS gives an answer

Aggressive NSEC Caching

- Described in RFC 8198
- When a zone is DNSSEC-signed an authoritative server will answer a query for a non-existent name with an authenticated denial of existence response (NSEC or NSEC3)
- These records describe two adjacent labels in the zone that span the non-existent record
- If recursive resolvers cache the NSEC span record then they can answer subsequent queries for any label within the span without passing the specific query onward for the cache lifetime of the NSEC record

Aggressive NSEC caching

- It’s a tool for recursive resolvers
- Stops queries for non-existent names always heading to the zone authoritative servers
- Improves the efficiency of the resolver’s cache

Middleware and WireTapping

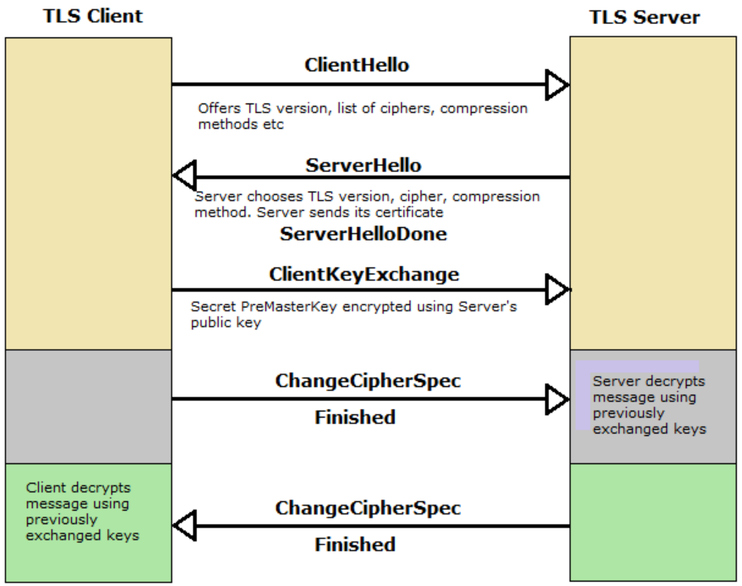
- Protecting the content of DNS responses is part of what we need to make the DNS more robust
- If we want to prevent DNS inspection we also should look at encrypting the transport used by DNS queries and responses
- Today the standard tool is TLS, which uses dynamically generated session keys to encrypt all traffic between two parties
- We could use TLS between the end client and the client’s recursive resolver
  – It’s more challenging to use encryption between recursive resolvers and authoritative servers

DNS over TLS

- Similar to DNS over TCP:
  - Open a TLS session with a recursive resolver
  - Pass the DNS query using DNS wireline format
  - Wait for the response
- Can use held DNS sessions to allow the TLS session to be used for multiple DNS queries
- The queries and the responses are hidden from intermediaries
- The client validates the recursive resolver’s identity

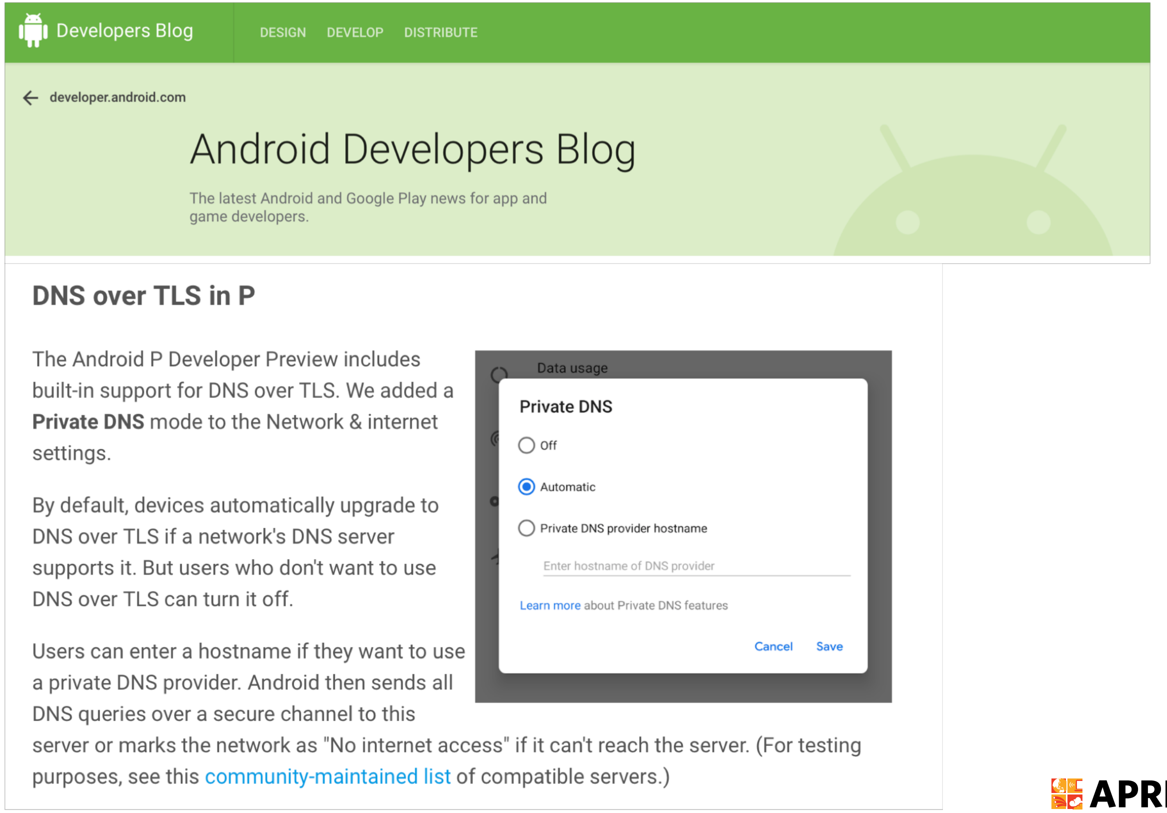
DNS over TLS

- TLS is a TCP 'overlay' that adds server authentication and session encryption to TCP
- TLS uses an initial handshake to allow a client to:
  - Validate the identity of the server
  - Negotiate a session key to be used in all subsequent packets in the TCP session
- RFC 7858, RFC 8310


TLS Client
TLS Server
ClientHello
Offers TLS version, list of ciphers, compression methods etc
ServerHello
Server chooses TLS version, cipher, compression method. Server sends its certificate
ServerHelloDone
ClientKeyExchange
Secret PreMasterKey encrypted using Server's public key
ChangeCipherSpec
Finished
Server decrypts message using previously exchanged keys
Client decrypts message using previously exchanged keys
ChangeCipherSpec
Finished

TLS 1.2 handshake

DNS over TLS and Android



Android Developers Blog

The latest Android and Google Play news for app and game developers.

DNS over TLS in P

The Android P Developer Preview includes built-in support for DNS over TLS. We added a **Private DNS** mode to the Network & internet settings.

By default, devices automatically upgrade to DNS over TLS if a network's DNS server supports it. But users who don't want to use DNS over TLS can turn it off.

Users can enter a hostname if they want to use a private DNS provider. Android then sends all DNS queries over a secure channel to this server or marks the network as "No internet access" if it can't reach the server. (For testing purposes, see this community-maintained list of compatible servers.)

Data usage

Private DNS

Off [✓ Automatic] Private DNS provider hostname

Enter hostname of DNS provider

Learn more about Private DNS features

https://android-developers.googleblog.com/2018/04/dns-over-tls-support-in-android-p.html

DNS over TLS

- Will generate a higher recursive resolver memory load as each client may have a held state with one or more recursive resolvers
- The TCP session state is on port 853
  - DNS over TLS can be readily blocked by middleware
- The privacy is relative, as the recursive resolver still knows all your DNS queries
- Supported by Bind (stunnel), Unbound, DNSDist

DNS over TLS

- It seems odd to me that the IETF standardised DNS over TLS on TCP port 853
- If you want to hide then it makes far more sense to reuse port 443
- And if you really want to hide the DNS place both a web service and a DNS recursive resolver behind the same port 443 on the server

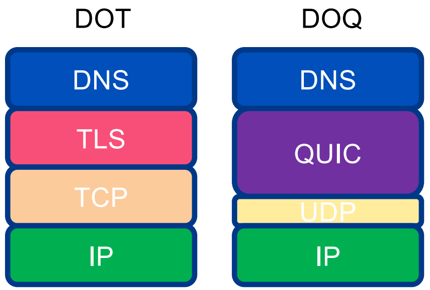
DNS over DTLS

- DTLS is a UDP variant of TLS that is intended to work over UDP rather than TCP (RFC 8094)
- However:
  - DTLS is intolerant of fragmentation
  - It appears to have similar overheads to TLS
  - I'm not sure if there are any implementations of DNS over DTLS

DNS over QUIC

- QUIC is a transport protocol originally developed by Google and passed over to the IETF for standardised profile development
- QUIC uses a thin UDP shim and an encrypted payload
  - The payload is divided into a TCP-like transport header and a payload
- The essential difference between DOT and DOQ is the deliberate hiding of the transport protocol from network middleware with the use of QUIC
- No known production implementations of DNS over QUIC exist, though IETF work continues

draft-huitema-quic-dnsoquic-05



DNS over HTTPS

- DNS over HTTPS
- Uses an HTTPS session with a resolver
- Similar to DNS over TLS, but with HTTP object semantics
- Uses TCP port 443, so can be masked within other HTTPS traffic
- Can use DNS wireformat or JSON format DNS payload

https://dns.google.com/query?name=www.potaroo.net

DNS over HTTPS within the Browser

- Firefox's "Trusted Recursive Resolver"
- Avoids using the local DNS resolver library and local DNS infrastructure
- Has the browser sending its DNS queries directly to a trusted resolver over HTTPS
- Servers available from Cloudflare, Google, Cleanbrowsing

https://blog.usejournal.com/getting-started-with-dns-over-https-on-firefox-e9b5fc865a43

Push DNS

HTTP/2 allows for server push

For example, the HTTP header:

```
Link: </css/styles.css>; rel=preload
```

What if…

```
Link: </dns-query/dns=<query>; rel=preload
```

Push DNS

- It can make web loads a whole lot faster
  - This can allow a web page to push a DNS result to the client without the client performing a query
  - There is no need for these DNS names to be separately defined by conventional DNS queries
- But can you trust the answer?
- Raises some disturbing possibilities for segmented namespaces

Push DNS

- What if the only way to resolve a particular name was via pushed DNS over HTTPS ?
- What if we push an entire dictionary of resolved names?
- Pushed DNS over HTTPS would permit a server to associate an entire private namespace with a web resource that is loaded into the user's browser context

DIY DNS

- Run your own resolver
  - The issue here is limited opportunity for caching, which means that performance could be painful!
- Run your own validating stub resolver
  - Getdns API interface
  - Stubby stub resolver
  - Still need to select a recursive resolver, and take advantage of caching, but allows the local resolver to perform its own DNSSEC validation

If you care about DNS privacy

- You might want to think about how want to interact with the DNS
- The careful choice of an open recursive resolver and an encrypted DNS session might go a long way along the path to DNS privacy
- But a poor choice of recursive resolver will probably compromise your privacy more than just doing nothing!

My current DNS Config

I use DNS over HTTPS

- I have configured Cloudflare's Cloudflared * to listen of localhost:53
- I have set up my local /etc/resolv.conf to contain 127.0.0.1
- All my DNS queries leave my laptop in an HTTPS port 443 stream towards 1.1.1.1

* https://developers.cloudflare.com/1.1.1.1/dns-over-https/cloudflared-proxy/

Thanks!