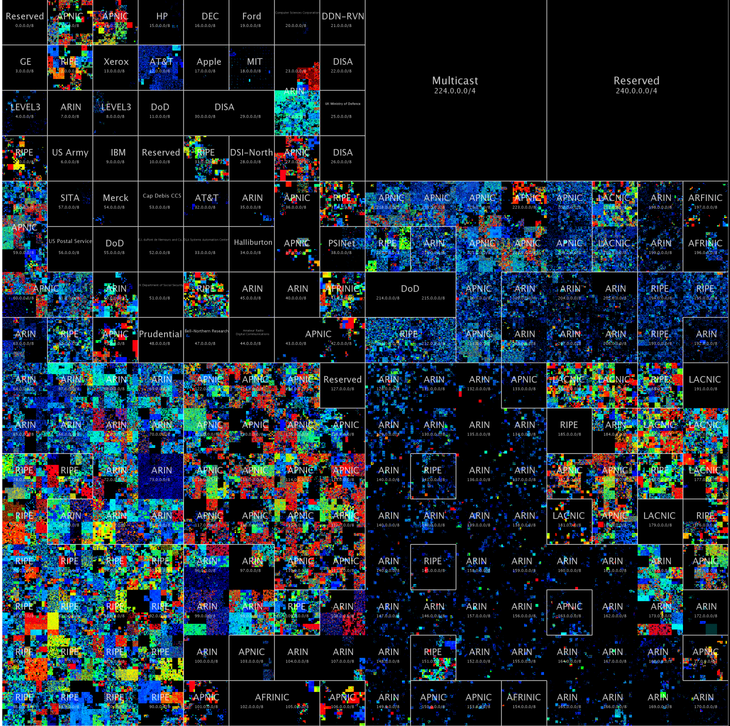
Domain Name System (DNS) (rfc1035)

Prime Purpose of DNS:
translate hostnames / domains to IP address

Prime Purpose of DNS:
translate hostnames / domains to IP address
“phonebook of internet”

Hostnames vs. IP addresses

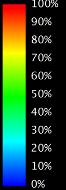
- hostname
  - e.g., www.cbc.ca and www.ucalgary.ca
  - easy to remember
  - useful to humans
  - variable length, full alphabet
  - little to no location information
- IP addresses
  - e.g., 194.40.217.50 and 194.150.245.142
  - fixed length, binary numbers
  - useful to routers
  - hierarchical, related to host location



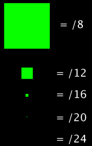
IPv4 Census Map

June – October 2012

Utilization



Prefix Sizes



420 Million hosts that responded to ICMP Ping at least 2 times between June and October 2012
Source: Carna Botnet

Mapping Names to Addresses

- domain name system divides namespace by sub-trees (zones)
  - dots separate it as domain becomes more refined right to left
  - .ca, ucalgary.ca, cs.ucalgary.ca, etc.
  - consider www.bank.com and mail.bank.com
  - and www.bank.com and www.bank.ru
- root is '.' and hardwired into DNS resolver
- top-level domain (TLD) is next: .com, .org, .ca, etc.
- turning domain to IP goes piece by piece
  - different computers are responsible for each node of subtree

```
; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> www.cpsc.ucalgary.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60461
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.cpsc.ucalgary.ca.          IN      A

;; ANSWER SECTION:
www.cpsc.ucalgary.ca.   3474    IN      CNAME   cpsc.ucalgary.ca.
cpsc.ucalgary.ca.       3474    IN      A       136.159.2.21

;; Query time: 10 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Thu Mar 04 18:42:36 MST 2021
;; MSG SIZE  rcvd: 79
```

```
; <<>> DiG 9.11.3-1ubuntu1.13-Ubuntu <<>> -t TXT www.cpsc.ucalgary.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17110
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.cpsc.ucalgary.ca.          IN      TXT

;; ANSWER SECTION:
www.cpsc.ucalgary.ca.   3439    IN      CNAME   cpsc.ucalgary.ca.

;; AUTHORITY SECTION:
cpsc.ucalgary.ca.       3466    IN      SOA     ns1.cpsc.ucalgary.ca. jenn.cpsc.ucalgary.ca.
2021030301 3600 7200 2419200 7200

;; Query time: 11 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Thu Mar 04 18:43:10 MST 2021
;; MSG SIZE  rcvd: 108
```

DNS Zone

- set of hostnames / IPs all managed by one server
- e.g., www.ucalgary.ca, www.cs.ucalgary.ca, mail.ucalgary.ca all part of ucalgary.ca zone

Nameserver

- server software that answers DNS questions
  - e.g., what is the IP address for www.ucalgary.ca
- if it is **responsible** for knowing the answer it answers directly
  - i.e., it is the **authoritative nameserver** for the zone
- if it is a **recursive nameserver** it goes out on the internet and asks around
  - not all nameservers are configured to be recursive
  - clients may be subject to access control
    - e.g., an ISP provides nameservice only to customers
  - it will use cached values when possible
- different software does this service differently
  - BIND, PowerDNS, djbdns

Authoritative Nameserver

- for every zone somebody has to keep a file of the hostnames and the IP addresses
  - x.y.z is 10.3.5.23, etc.
- this is generally an administrative function done by a human
- most cases, one machine has the file
  - is called **primary DNS server** or **zone master**
- if there are multiple public nameservers then the zone data is replicated to additional secondary nameservers
  - all secondary servers are considered authoritative to the outside world
  - why would you want to use secondary nameservers?

Resolver

- this is the client part of the DNS system
  - it asks the questions about hostnames
- sends questions to a nearby nameserver
  - on UNIX systems they are specified in `/etc/resolv.conf`
- resolver is lightweight and relies on the server to do all the work

Resource Record

- DNS does more than give hostname-to-IP mappings
  - "A" record is IP address
  - "NS" record is nameserver for hostname
  - "MX" record is mail exchanger
  - "TXT" record is arbitrary text
    - can be descriptive or be used by computers
  - "CNAME" record is canonical name
    - multiple domains sharing a server
    - only need to update IP for the CNAME
  - "SOA" record is start of authorities
    - data about zone administrator

DNS is a Distributed Database

DNS is a Distributed Database
Queries for a key (hostname+record type)
Replies is the key again and the value (the record)

Delegation

- when nameserver doesn't have the contents of a zone but know how to find the owner
- "I know the zone you want, go ask (hostname) for it"

Following a simple DNS Query

Alice wants www.ucalgary.ca

Alice wants www.ucalgary.ca
DNS client goes to ISP's nameserver

Alice wants www.ucalgary.ca
DNS client goes to ISP's nameserver
DNS client requests "A" record

Alice wants www.ucalgary.ca
DNS client goes to ISP's nameserver
DNS client requests "A" record
Nameserver knows it is not authoritative

Alice wants www.ucalgary.ca
DNS client goes to ISP's nameserver
DNS client requests "A" record
Nameserver knows it is not authoritative
Nameserver doesn't have it in cache

Alice wants www.ucalgary.ca
DNS client goes to ISP's nameserver
DNS client requests "A" record
Nameserver knows it is not authoritative
Nameserver doesn't have it in cache
Nameserver goes out on the Internet.

List of Root Servers

HOSTNAME	IP ADDRESSES	MANAGER
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

192.112.36.4

Star

Columbus, Ohio, US

anycast

Summary

Geolocation

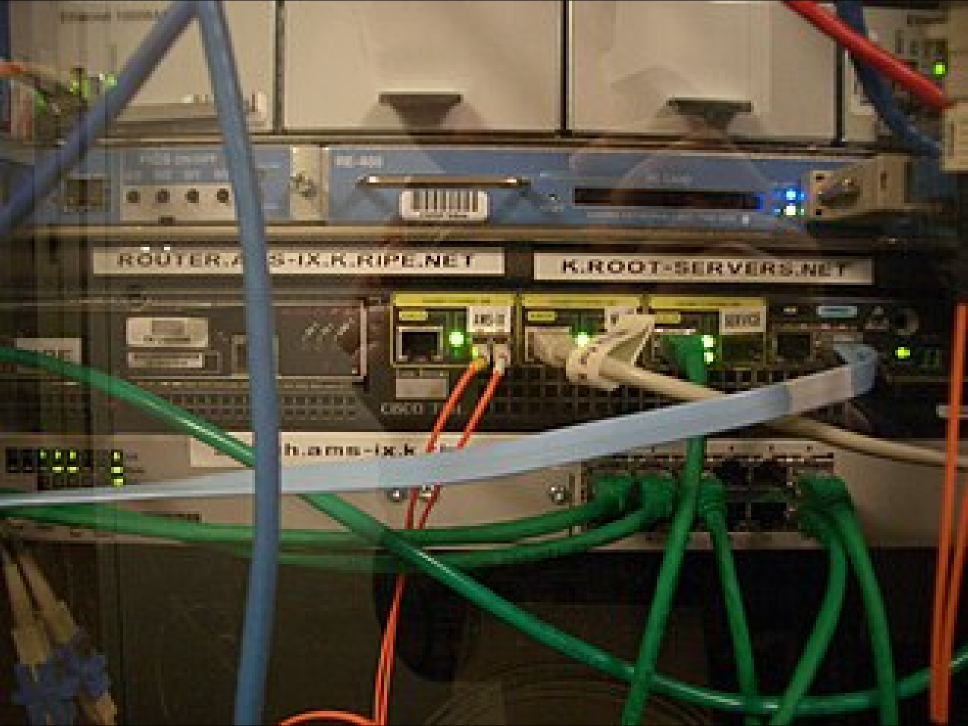
Privacy

ASN

Company

Summary

ASN	AS5927 - United States Department of Defense (DoD)
Hostname	g.root-servers.net
Range	192.112.36.0/24
Company	DoD Network Information Center


ROUTER.A S-IX.K.RIPE.NET
K.ROOT-SERVERS.NET

Recursive nameserver are preconfigured with root servers.

Recursive nameserver are preconfigured with root servers.
It picks one at random and asks for the “A” record

Recursive nameserver are preconfigured
with root servers.
It picks one at random and asks for the “A” record
Root server doesn’t know anything about “ucalgary.ca”
But it sends a list of “.ca” nameservers who may know

Recursive nameserver are preconfigured
with root servers.
It picks one at random and asks for the "A" record
Root server doesn't know anything about "ucalgary.ca"
But it sends a list of ".ca" nameservers who may know
It sends a list of "NS" records of more qualified servers

Recursive nameserver are preconfigured
with root servers.
It picks one at random and asks for the “A” record
Root server doesn’t know anything about “ucalgary.ca”
But it sends a list of “.ca” nameservers who may know
It sends a list of “NS” records of more qualified servers
These are called Global Top Level Domain (GTLD) servers

Recursive nameserver are preconfigured with root servers.

It picks one at random and asks for the “A” record

Root server doesn’t know anything about “ucalgary.ca”

But it sends a list of “.ca” nameservers who may know

It sends a list of “NS” records of more qualified servers

These are called Global Top Level Domain (GTLD) servers

It also provides “A” records (called **glue** records)

Recursive nameserver are preconfigured
with root servers.
It picks one at random and asks for the “A” record
Root server doesn’t know anything about “ucalgary.ca”
But it sends a list of “.ca” nameservers who may know
It sends a list of “NS” records of more qualified servers
These are called Global Top Level Domain (GTLD) servers
It also provides “A” records (called **glue** records)
This is to save time in looking it up.

Recursive nameserver goes to a random .ca NS

Recursive nameserver goes to a random .ca NS

“What’s the A record for www.ucalgary.ca?”

Recursive nameserver goes to a random .ca NS
"What's the A record for www.ucalgary.ca?"
Also not authoritative

Recursive nameserver goes to a random .ca NS
"What's the A record for www.ucalgary.ca?"
Also not authoritative
It gives referrals to NS records as before

Recursive nameserver goes to a random .ca NS
"What's the A record for www.ucalgary.ca?"
Also not authoritative
It gives referrals to NS records as before
These are likely ucalgary's authoritative NSes

Recursive nameserver goes to a random ucalgary.ca NS

Recursive nameserver goes to a random ucalgary.ca NS
This time it responds with the A record

Recursive nameserver goes to a random ucalgary.ca NS
This time it responds with the A record
It also includes a flag saying “this is authoritative”

Recursive nameserver goes to a random ucalgary.ca NS
This time it responds with the A record
It also includes a flag saying “this is authoritative”
The recursive nameserver returns the answer to the client

Recursive nameserver goes to a random ucalgary.ca NS
This time it responds with the A record
It also includes a flag saying “this is authoritative”
The recursive nameserver returns the answer to the client
The recursive nameserver keeps answer in its cache

Recursive nameserver goes to a random ucalgary.ca NS
This time it responds with the A record
It also includes a flag saying "this is authoritative"
The recursive nameserver returns the answer to the client
The recursive nameserver keeps answer in its cache
The recursive nameserver does not keep the "authoritative" flag
It is considered a non-authoritative answer.

This is happening at the trillions scale daily.

This is happening at the trillions scale daily.
Google alone does billions daily.

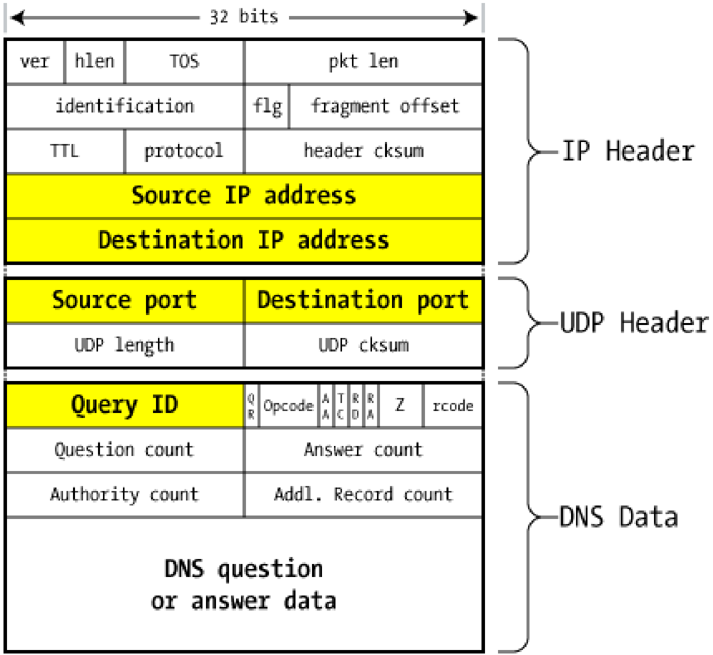
This is happening at the trillions scale daily.
Google alone does billions daily.
DNS is fast so this eight packet dance is not noticed

This is happening at the trillions scale daily.
Google alone does billions daily.
DNS is fast so this eight packet dance is not noticed
Caching also helps

DNS is distributed: no one machine knows everything
They find what is needed through delegation

Fake Nameserver

- nothing stops Eve running a nameserver for anything she wants
- she can claim to be authoritative for anything she wants
- but no higher-level NS will delegate to her



DNS packet on the wire

DNS Packet

- source / dest IP address
  - sender and receiver of the packet
  - these are possible to forge, you can write what you want
- source / dest port
  - DNS servers listen of port 53/udp for queries
  - source port varies
    - sometimes is also 53
    - sometimes it's a fixed random port chosen by the OS
    - sometimes it's a random port everytime
  - source port doesn't matter as long as replies reach it

DNS Packet

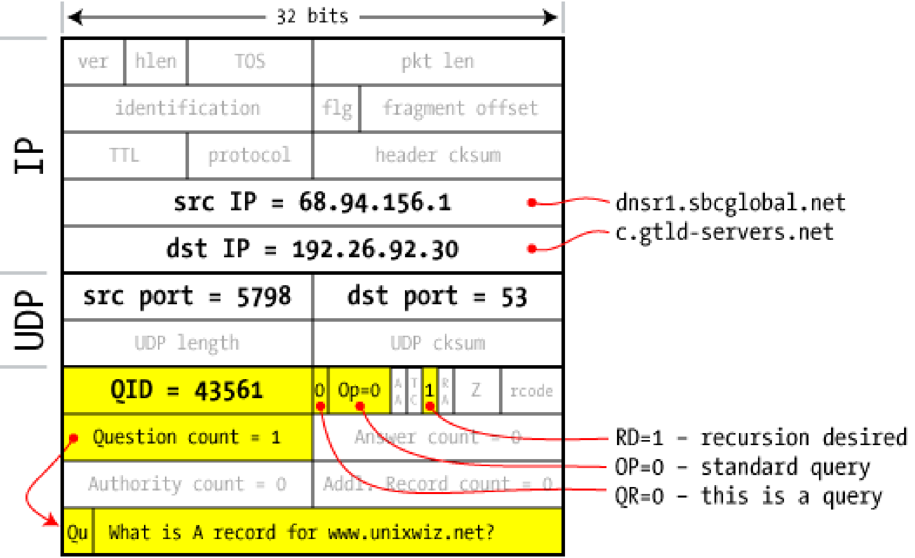
- query ID
  - unique identifier created in the query packet
  - left intact by the server
    - allows client to associate the answer
  - nameserver may have many queries at one time
    - include to the same server
    - query ID matches answers to awaiting questions
- QR (query / response)
  - 0 for query by client
  - 1 for response from a server

DNS Packet

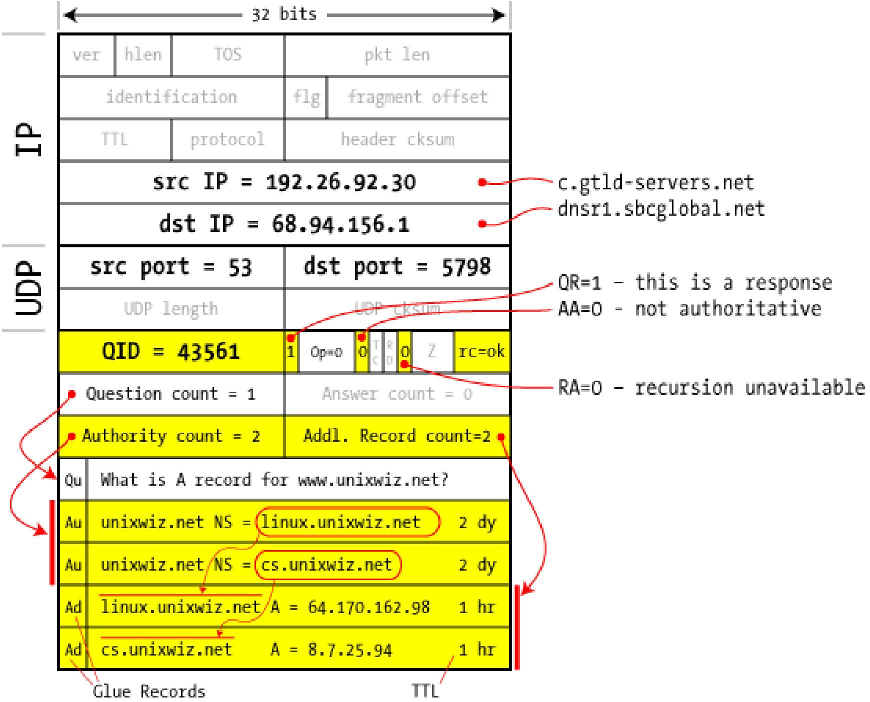
- AA: authoritative answer bit
- TC: truncated bit
  - answer can't fit in 512-byte UDP packet
  - try again with TCP
- RD: recursion desired
  - client wants the server to perform the lookup recursively
  - otherwise client will do the work itself
  - does not need to be honoured
- RA: recursion available bit
  - 0 means nameserver won't honour RD

DNS Packet

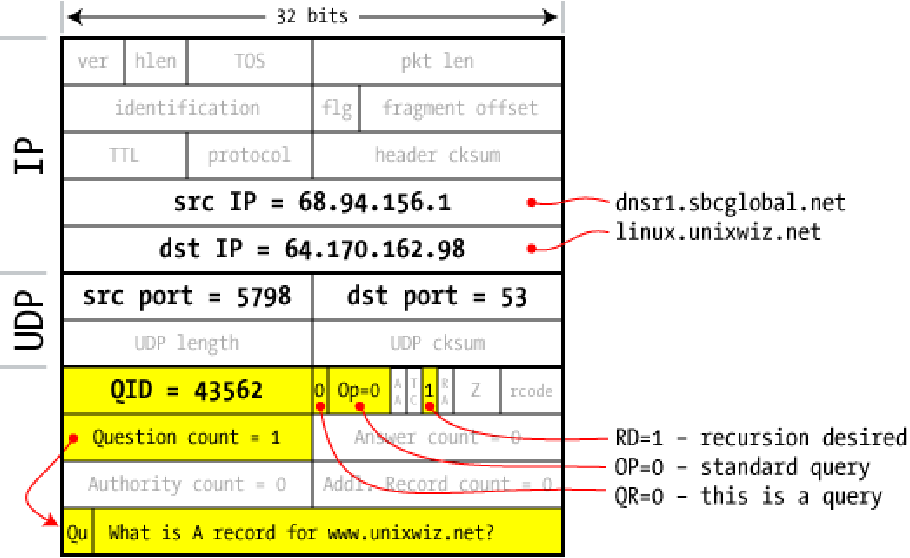
- Z: reserved, must be zero
  - why do we do things like this?
- rcode: response code success or failure
- Question record count: number of questions
  - server reply repeats the question
- Answer / authority / additional record count
  - number of replies by the server
- DNS question / answer data
  - area storing data referenced by the counts



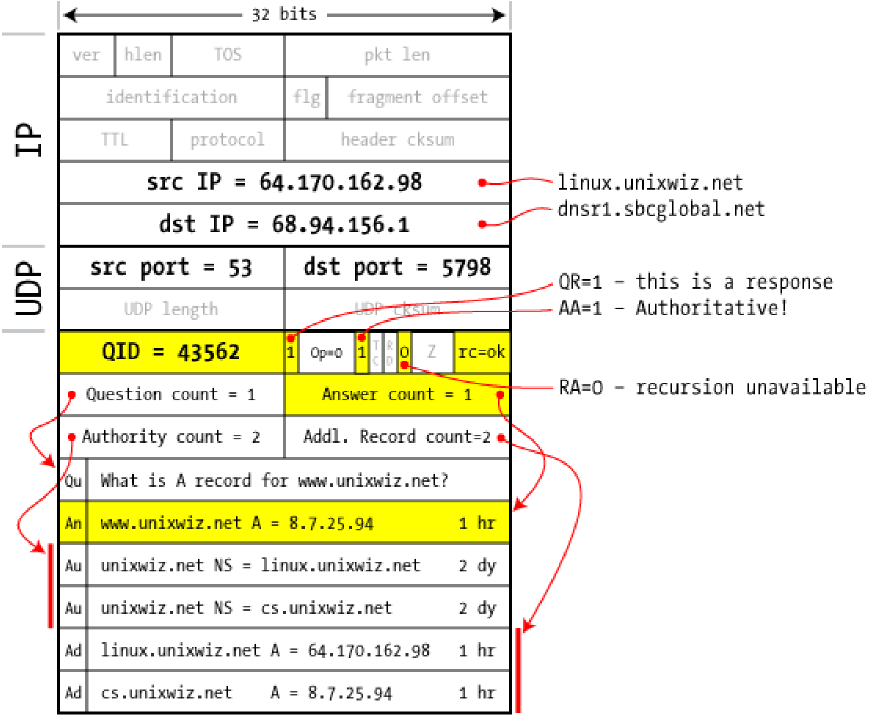
32 bits
IP
UDP
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
src IP = 68.94.156.1
dst IP = 192.26.92.30
dnsr1.sbcglobal.net
c.gtld-servers.net
src port = 5798
dst port = 53
UDP length
UDP cksum
QID = 43561
0
Op=0
AA
TC
1
RA
Z
rcode
Question count = 1
Answer count = 0
Authority count = 0
Addl. Record count = 0
Qu
What is A record for www.unixwiz.net?
RD=1 - recursion desired
OP=0 - standard query
QR=0 - this is a query



32 bits
IP
UDP
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
src IP = 192.26.92.30
dst IP = 68.94.156.1
c.gtld-servers.net
dnsr1.sbcglobal.net
src port = 53
dst port = 5798
UDP length
UDP cksum
QID = 43561
1
Op=0
0
TC
RD
0
Z
rc=ok
QR=1 - this is a response
AA=0 - not authoritative
RA=0 - recursion unavailable
Question count = 1
Answer count = 0
Authority count = 2
Addl. Record count=2
Qu What is A record for www.unixwiz.net?
Au unixwiz.net NS = linux.unixwiz.net 2 dy
Au unixwiz.net NS = cs.unixwiz.net 2 dy
Ad linux.unixwiz.net A = 64.170.162.98 1 hr
Ad cs.unixwiz.net A = 8.7.25.94 1 hr
Glue Records
TTL



32 bits
IP
UDP
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
src IP = 68.94.156.1
dst IP = 64.170.162.98
dnsr1.sbcglobal.net
linux.unixwiz.net
src port = 5798
dst port = 53
UDP length
UDP cksum
QID = 43562
0
Op=0
AA
TC
1
RA
Z
rcode
Question count = 1
Answer count = 0
Authority count = 0
Addl. Record count = 0
RD=1 - recursion desired
OP=0 - standard query
QR=0 - this is a query
Qu
What is A record for www.unixwiz.net?



32 bits
IP
UDP
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
src IP = 64.170.162.98
linux.unixwiz.net
dst IP = 68.94.156.1
dnsr1.sbcglobal.net
src port = 53
dst port = 5798
UDP length
UDP cksum
QR=1 - this is a response
AA=1 - Authoritative!
QID = 43562
1
Op=0
1
TC
RD
0
Z
rc=ok
RA=0 - recursion unavailable
Question count = 1
Answer count = 1
Authority count = 2
Addl. Record count=2
Qu What is A record for www.unixwiz.net?
An www.unixwiz.net A = 8.7.25.94 1 hr
Au unixwiz.net NS = linux.unixwiz.net 2 dy
Au unixwiz.net NS = cs.unixwiz.net 2 dy
Ad linux.unixwiz.net A = 64.170.162.98 1 hr
Ad cs.unixwiz.net A = 8.7.25.94 1 hr

DNS Caching

- in practice the full effort is not needed
  - caching is used by recursive nameservers to speed up the process
  - authoritative answers can be cached to use again later
  - glue records send expected data needed afterwards
- DNS can't be cached forever
  - stale data breaks the system
  - IPs do change, etc.

DNS Time-to-Live (TTL)

- TTL is how long the data can be considered valid
- TTL set by **administrator of the zone**
  - can be minutes or weeks, up to administrator
- recursive nameserver does not need to guess how long to keep cache
  - once the record expires, it is removed from the cache
- not just A records are cached
- all other authority data is cached
  - NS plus glued A are cached with their own TTL

DNS Poisoning

- attacker manages to put bad data into a recursive nameserver's cache
  - bad information is given to unsuspecting clients
- DNS does not accept unsolicited responses
  - it must be a response to a pending query

DNS Query Checks

- response is on the same UDP port it was sent from
- question section matches the question in the pending query
- query ID matches pending query
- authority and additional section represent names that were being queried
  - **bailiwick checking**
  - prevents replies from piggybacking on other bad records

DNS Poisoning

- if the attacker can send bad data that passes all the checks it is accepted
- this affects not only the recursive nameserver but all its clients
- typically, it involves changing the IP for some website to the attacker's server
    - this allows an **off-path** attacker to act as a MITM or impersonator
- attacker doesn't need to be right everytime
    - bad packets are dropped but one good one will be accepted

Guessing the Query ID

- old nameservers just incremented it by one
  - not-cryptographically suitable randomness
  - need **unguessable** query IDs for this to be hard
- SECURITY BY DESIGN
  - Query ID was never **meant** to stop DNS poisoning

But how can the attacker learn the current query ID?

But how can the attacker learn the current query ID?
Ask resolver for an IP in its own domain.

But how can the attacker learn the current query ID?
Ask resolver for an IP in its own domain.
Then it gets the query ID from the DNS request.

DNS Query Checks

- UDP port: often not random
- Query ID: learnable
- question matches query
  - spurious responses ignored
- bailiwick
  - cannot glue other domains

DNS Query Forcing

- attacker can force the DNS server to retrieve the query
  - simply ask for it
  - if it is cached eventually it will expire
- attacker sends a flurry of replies
  - candidate query IDs
- simple rule in response
  - first good answer wins
    - only attacker knows about it
  - there is no authentication on the response

Poisoning Antidote: why not just ignore DNS responses coming from another IP?

Poisoning Antidote: why not just ignore DNS responses coming from another IP?
UDP packets can come from anywhere and can spoof the sender's IP.

Poisoning Antidote: randomize the query ID

- if attacker knows QID=9999, then it floods with QIDs 10000–10030
- but if the QID can be any 16-bit value it makes it much harder
  - making and sending that many packets in DNS-resolution time is harder
- again: security by design
  - QID is only 16 bits because it wasn't meant for this purpose
  - if it were, it would be at least 128 bits

Dan Kaminsky DNS Attack (2008)

Dan Kaminsky DNS Attack (2008)
He found a more effective approach.

Dan Kaminsky DNS Attack (2008)
He found a more effective approach.
DNS servers worldwide patched for this.

Dan Kaminsky DNS Attack (2008)
He found a more effective approach.
DNS servers worldwide patched for this.
Approach: highjack the authority records.

Kaminsky Attack

- nothing stops anyone from setting up a nameserver for anyone else
  - it can offer A records, MX records, claim to be authoritative
  - no one ever asks it any questions
- higher-level servers are configured with administration
  - .ca NS knows the .ucalgary.ca NS
  - it doesn't need to request it, it **knows it**
  - this is the distributed database

Kaminsky Attack

- but the resolver **doesn't** know the .ucalgary.ca or the .ca domain
  - that's why it's asking the distributed database for it
- if we poison responses for the .ca NS, we can poison *.ca trivially
  - we will be the ones **asked** to answer future queries
  - this means we don't need to invest any effort

Kaminsky Attack

- step 1: attacker requests a random name from the victim domain
  - e.g., vnrqpjvnlj.victimbank.com
  - this is unlikely to be in cache, so will require DNS query
- step 2: attacker sends a flurry of forged packets
  - not A record like before
  - instead delegate to another nameserver via Authority records
    - "I don't know, but you can ask over there"
    - attacker can give honest NS for victimbank.com but fake glue records
    - glue points to attacker's IP addresses
    - attacker now is treated as IP for victimbank's authoritative NS

That's the attack. Once the adversary controls the NS the rest of the attack is DNS-working-as-intended.

That's the attack. Once the adversary controls the NS the rest of the attack is DNS-working-as-intended. The attacker needs to control only one point in the chain.

That's the attack. Once the adversary controls the NS the rest of the attack is DNS-working-as-intended.
The attacker needs to control only one point in the chain.
Controls everything after it.

Attacker can redirect all web traffic
or all mail traffic with MX records.

Attacker can redirect all web traffic
or all mail traffic with MX records.
Set TXT records to satisfy CA challenges

Attacker can redirect all web traffic
or all mail traffic with MX records.
Set TXT records to satisfy CA challenges
Attacker can set a maximum TTL.

Even private DNS recursive nameservers are vulnerable.
(Why?)

Even with all the TLS and certificates infrastructure in place this attack still works. (Why?)

Even with all the TLS and certificates infrastructure in place this attack still works. (Why?)

site could simply not use HTTPS

Even with all the TLS and certificates infrastructure in place this attack still works. (Why?)

site could simply not use HTTPS

site could spoof DNS for a certificate authority first to get a cert

What is the fix?

What is the fix?

The small space of the query ID makes it possible

What is the fix?
The small space of the query ID makes it possible
Even with randomization, repeated tries make it likely.

What is the fix?

The small space of the query ID makes it possible

Even with randomization, repeated tries make it likely.

“Solution”: also randomize the UDP port

What is the fix?

The small space of the query ID makes it possible

Even with randomization, repeated tries make it likely.

“Solution”: also randomize the UDP port

with 2048 random ports we have $2^{16} \cdot 2^{11} = 2^{27} = 134M$.

Further Fix 0x20 encoding

Further Fix 0x20 encoding
in ASCII a–z XOR 0x20 yields A–Z

Further Fix 0x20 encoding
in ASCII a–z XOR 0x20 yields A–Z
in DNS www.POtatoCRUNchCeReal.com is
the same as www.potatocrunchcereal.com

Further Fix 0x20 encoding

in ASCII a–z XOR 0x20 yields A–Z

in DNS www.POtatoCRUNchCeReal.com is the same as www.potatocrunchcereal.com

DNS servers repeat query back with case

Further Fix: RFC 7873

- client computes a "cookie" based on
  - its IP
  - server's IP
  - some secret
- client includes cookie in DNS request
- server repeats cookie back
  - how does this stop poisoning?
- server also includes its own cookie back
  - this is used to defend DoS amplification (how?)
- requires both client and server support

What can we do to fix this?

A better solution: DNSSec

A better solution: DNSSec

Let's use cryptography to trust DNS more.

Idea 1: do DNS over TLS

Idea 1: do DNS over TLS

DNS is UDP so we need Datagram TLS (DTLS)

TLS DNS

- secure all connections
  - from computer to local DNS server (resolver)
  - from resolver to root DNS server
  - from resolver to TLD DNS server
  - from resolver to authoritative DNS server

Issues with TLS DNS

- performance
  - DNS is lightweight
  - TLS is not
- caching
  - crucial for DNS scalability
  - we need **object** security, not **channel** security

Consider the DNS record to be an atomic piece of data.
We need it to be authentic, not defeat eavesdroppers.

Consider the DNS record to be an atomic piece of data.
We need it to be authentic, not defeat eavesdroppers.
Consider GPS timestamps

Idea 2: make DNS results like **certs**

Idea 2: make DNS results like **certs**

e.g., a **verifiable signature** that guarantees

who generated the data, signing is offline

DNSSEC

- standardized DNS security extention
  - "currently" being deployed
- resolver works its way from DNS root to final name server
  - at each step it gets a signed statement for the next level's keys
- resolver knows root DNS keys (offline, can't bootstrap trust)
  - this is O(1) data
- root DNS knows all TLD servers **and** their signing keys
  - this is only O(1) more information per thing it was already storing

DNSSec con't

- the final answer is signed by the authoratitive DNS server
- resolver can trust its signature because of a chain of support from higher levels
- all keys **and** signed results are **cacheable**!
  - critical for scalability
  - like tickets (object security)

Idea 3: DNS-over-HTTPS (DoH) (rfc8484)

- this is effectively DNS-over-TLS
  - DoH benefits by "looking like" HTTPS traffic
  - network operators cannot force their DNS resolver
- difference
  - HTTPS is used between you and a highly-shared resolver
  - that resolver then does insecure DNS on the open Internet
  - does not require DNS servers at every site to use DNSSec
  - resolver still performs all caching functions
- act as a DNS proxy with a secure connection
  - need to poison DoH resolver
  - e.g., peers on hotel WiFi cannot poison you

Enable DNS over HTTPS using:

Default Protection

Firefox decides when to use secure DNS to protect your privacy.

- Use secure DNS in regions where it's available
- Use your default DNS resolver if there is a problem with the secure DNS provider
- Use a local provider, if possible Learn more
- Turn off when VPN, parental control, or enterprise policies are active
- Turn off when a network tells Firefox it shouldn't use secure DNS Learn more

Increased Protection

You control when to use secure DNS and choose your provider.

- Use the provider you select
- Only use your default DNS resolver if there is a problem with secure DNS

Choose provider:

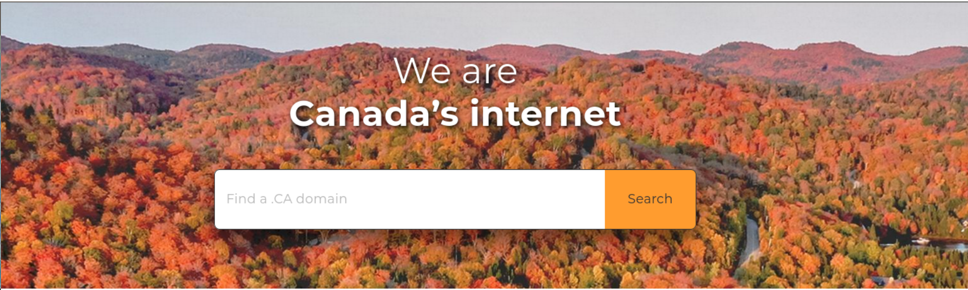
CIRA Canadian Shield (Default)

No.	Time	Source	Destination	Protocol	Length	Info
170	5.379957	192.168.0.116	149.112.121.10	TCP	74	52872 → 443 [SYN] Seq=4092458011 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TS
171	5.404472	149.112.121.10	192.168.0.116	TCP	66	443 → 52872 [SYN, ACK] Seq=1674013340 Ack=4092458012 Win=30660 Len=0 MSS
172	5.404493	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092458012 Ack=1674013341 Win=64256 Len=0
173	*REF*	192.168.0.116	149.112.121.10	TLSv1.3	688	Client Hello
174	0.027275	149.112.121.10	192.168.0.116	TCP	60	443 → 52872 [ACK] Seq=1674013341 Ack=4092458646 Win=61440 Len=0
175	0.027284	149.112.121.10	192.168.0.116	TLSv1.3	288	Server Hello, Change Cipher Spec, Encrypted Extensions, Finished
176	0.027294	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092458646 Ack=1674013575 Win=64128 Len=0
177	0.027707	192.168.0.116	149.112.121.10	TLSv1.3	118	Change Cipher Spec, Finished
178	0.028269	192.168.0.116	149.112.121.10	HTTP2	224	Magic, SETTINGS[0], WINDOW_UPDATE[0], PRIORITY[3], PRIORITY[5], PRIORITY
179	0.028289	192.168.0.116	149.112.121.10	DoH	281	Standard query 0x0000 A cbc.ca OPT
180	0.028676	192.168.0.116	149.112.121.10	DoH	314	Standard query 0x0000 A cbc.ca OPT
181	0.089104	192.168.0.116	149.112.121.10	TLSv1.3	314	Application Data, Application Data, Application Data, Application Data
185	0.095359	149.112.121.10	192.168.0.116	TLSv1.3	293	New Session Ticket
186	0.095398	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092459367 Ack=1674013814 Win=64000 Len=0
187	0.095418	149.112.121.10	192.168.0.116	TCP	60	443 → 52872 [ACK] Seq=1674013814 Ack=4092459107 Win=66560 Len=0
188	0.095431	149.112.121.10	192.168.0.116	TCP	60	443 → 52872 [ACK] Seq=1674013814 Ack=4092459367 Win=68608 Len=0
191	0.114660	149.112.121.10	192.168.0.116	DoH	470	Standard query response 0x0000 A cbc.ca A 23.6.170.182 OPT
192	0.114688	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092459367 Ack=1674014230 Win=63616 Len=0
195	0.119905	149.112.121.10	192.168.0.116	TCP	66	443 → 52872 [ACK] Seq=1674014230 Ack=4092459367 Win=68608 Len=0 SLE=4092
227	0.381449	192.168.0.116	149.112.121.10	HTTP2	114	HEADERS[21]: POST /dns-query

```
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 52872, Seq: 1674013814, Ack: 4092459367, Len: 416
▾ Transport Layer Security
  ▸ TLSv1.3 Record Layer: Application Data Protocol: http2
  ▸ TLSv1.3 Record Layer: Application Data Protocol: http2
  ▸ TLSv1.3 Record Layer: Application Data Protocol: http2
  ▸ TLSv1.3 Record Layer: Application Data Protocol: http2
▸ HyperText Transfer Protocol 2
▾ HyperText Transfer Protocol 2
  ▸ Stream: HEADERS, Stream ID: 17, Length 31, 200 OK
  ▸ Stream: DATA, Stream ID: 17, Length 107
  ▾ Domain Name System (response)
      Transaction ID: 0x0000
    ▸ Flags: 0x8180 Standard query response, No error
      Questions: 1
      Answer RRs: 0
      Authority RRs: 1
      Additional RRs: 1
    ▾ Queries
      ▾ cbc.ca: type Unknown (65), class IN
          Name: cbc.ca
          [Name Length: 6]
          [Label Count: 2]
          Type: Unknown (65)
          Class: IN (0x0001)
    ▾ Authoritative nameservers
      ▾ cbc.ca: type SOA, class IN, mname a1-29.akam.net
          Name: cbc.ca
          Type: SOA (Start Of a zone of Authority) (6)
          Class: IN (0x0001)
          Time to live: 130 (2 minutes, 10 seconds)
          Data length: 52
          Primary name server: a1-29.akam.net
          Responsible authority's mailbox: hostmaster.nm.cbc.ca
          Serial Number: 2015021305
          Refresh Interval: 3600 (1 hour)
```

	Time	Source	Destination	Protocol	Length	Info
170	5.379957	192.168.0.116	149.112.121.10	TCP	74	52872 → 443 [SYN] Seq=4092458011 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2885954676 TSecr=0 WS=128
171	5.404472	149.112.121.10	192.168.0.116	TCP	66	443 → 52872 [SYN, ACK] Seq=1674013340 Ack=4092458012 Win=30660 Len=0 MSS=1460 SACK_PERM=1 WS=1024
172	5.404493	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092458012 Ack=1674013341 Win=64256 Len=0
173	5.406121	192.168.0.116	149.112.121.10	TLSv1.3	688	Client Hello
174	5.433396	149.112.121.10	192.168.0.116	TCP	60	443 → 52872 [ACK] Seq=1674013341 Ack=4092458646 Win=61440 Len=0
175	5.433405	149.112.121.10	192.168.0.116	TLSv1.3	288	Server Hello, Change Cipher Spec, Encrypted Extensions, Finished
176	5.433415	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092458646 Ack=1674013575 Win=64128 Len=0
177	5.433828	192.168.0.116	149.112.121.10	TLSv1.3	118	Change Cipher Spec, Finished
178	5.434390	192.168.0.116	149.112.121.10	HTTP2	224	Magic, SETTINGS[0], WINDOW_UPDATE[0], PRIORITY[3], PRIORITY[5], PRIORITY[7], PRIORITY[9], PRIORITY[11], PRIORITY[13]
179	5.434410	192.168.0.116	149.112.121.10	DoH	281	Standard query 0x0000 A cbc.ca OPT
180	5.434797	192.168.0.116	149.112.121.10	DoH	314	Standard query 0x0000 A cbc.ca OPT
181	5.495225	192.168.0.116	149.112.121.10	TLSv1.3	314	Application Data, Application Data, Application Data, Application Data
185	5.501480	149.112.121.10	192.168.0.116	TLSv1.3	293	New Session Ticket
186	5.501519	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092459367 Ack=1674013814 Win=64000 Len=0
187	5.501539	149.112.121.10	192.168.0.116	TCP	60	443 → 52872 [ACK] Seq=1674013814 Ack=4092459107 Win=66560 Len=0
188	5.501552	149.112.121.10	192.168.0.116	TCP	60	443 → 52872 [ACK] Seq=1674013814 Ack=4092459367 Win=68608 Len=0
191	5.520781	149.112.121.10	192.168.0.116	DoH	470	Standard query response 0x0000 A cbc.ca A 23.6.170.182 OPT
192	5.520809	192.168.0.116	149.112.121.10	TCP	54	52872 → 443 [ACK] Seq=4092459367 Ack=1674014230 Win=63616 Len=0
195	5.526026	149.112.121.10	192.168.0.116	TCP	66	443 → 52872 [ACK] Seq=1674014230 Ack=4092459367 Win=68608 Len=0 SLE=4092459107 SRE=4092459367
227	5.787570	192.168.0.116	149.112.121.10	HTTP2	114	HEADERS[21]: POST /dns-query

```
▾ Transport Layer Security
  ▾ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 629
    ▾ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 625
        Version: TLS 1.2 (0x0303)
        Random: aa539b90a4d97e409420e2ae03f565987ab26736b0af2650…
        Session ID Length: 32
        Session ID: 0457f28226ff7dbd5171530104f95ef6702289bf6f1e4494…
        Cipher Suites Length: 36
      ▸ Cipher Suites (18 suites)
        Compression Methods Length: 1
      ▸ Compression Methods (1 method)
        Extensions Length: 516
      ▾ Extension: server_name (len=35)
          Type: server_name (0)
          Length: 35
        ▾ Server Name Indication extension
            Server Name list length: 33
            Server Name Type: host_name (0)
            Server Name length: 30
            Server Name: private.canadianshield.cira.ca
      ▾ Extension: extended_master_secret (len=0)
```



We are **CIRA**.

We manage .CA domains on behalf of all Canadians and work every day to build a trusted internet for Canadians.

- **Summary of CIRA Canadian Shield DNS resolver addresses**

	Features	IPv4	IPv6	DoH	DoT
Private	DNS resolution only	149.112.121.10 149.112.122.10	2620:10A:80BB::10 2620:10A:80BC::10	https://private.canadianshield.cira.ca/dns-query	private.canadianshield.cira.ca
Protected	Malware and phishing protection	149.112.121.20 149.112.122.20	2620:10A:80BB::20 2620:10A:80BC::20	https://protected.canadianshield.cira.ca/dns-query	protected.canadianshield.cira.ca
Family	*Protected* + blocking pornographic content	149.112.121.30 149.112.122.30	2620:10A:80BB::30 2620:10A:80BC::30	https://family.canadianshield.cira.ca/dns-query	family.canadianshield.cira.ca

It is recommended that you use both IP addresses in configuring your DNS settings for extra redundancy. We do not recommend using a third resolver because if your device has a problem accessing the CIRA Canadian Shield service then you might be browsing unprotected ***without*** your knowledge.

Even with HTTPS between you and your resolver,
is it still secure?
What can go wrong?
How can a shared resolver help security?