Privacy in the Domain Name System (DNS):

DNS Privacy in Practice

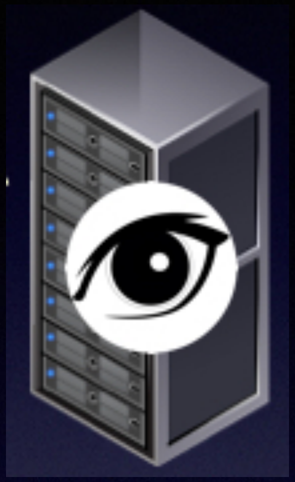
Sara Dickinson sara@sinodun.com
https://sinodun.com
@SinodunCom

Overview

- First - lets look at your DNS queries!
- Desktop DoT stub resolvers (client) (Stubby)
- Set up your own DoT recursive (Unbound) - decrypt DoT
- DoH - Clients & Browsers (Firefox) - decrypt DoH
- Mobile Apps
- DNS Libraries (getdns)
- Routers

Firefox DoH Decryption is easier….

dnsprivacy.org



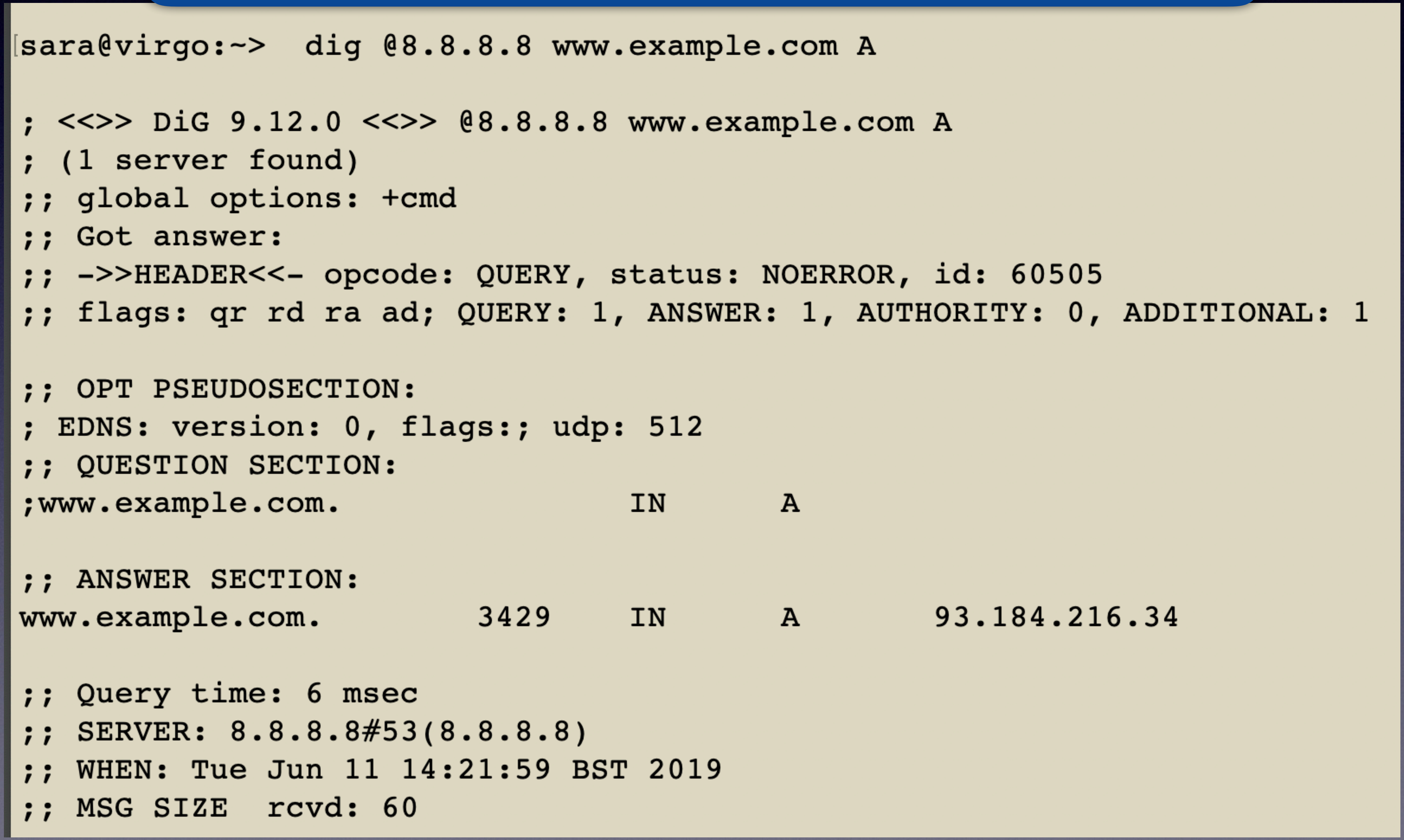
- DNS Privacy Clients
- DNS Privacy Servers setup guides
- DNS Privacy Test and Public resolvers
- DNS Privacy Monitoring
- DNS Privacy Current work

Reference material here for most setups and recursive resolvers

DNS Basics

DNS Basics - A UDP query

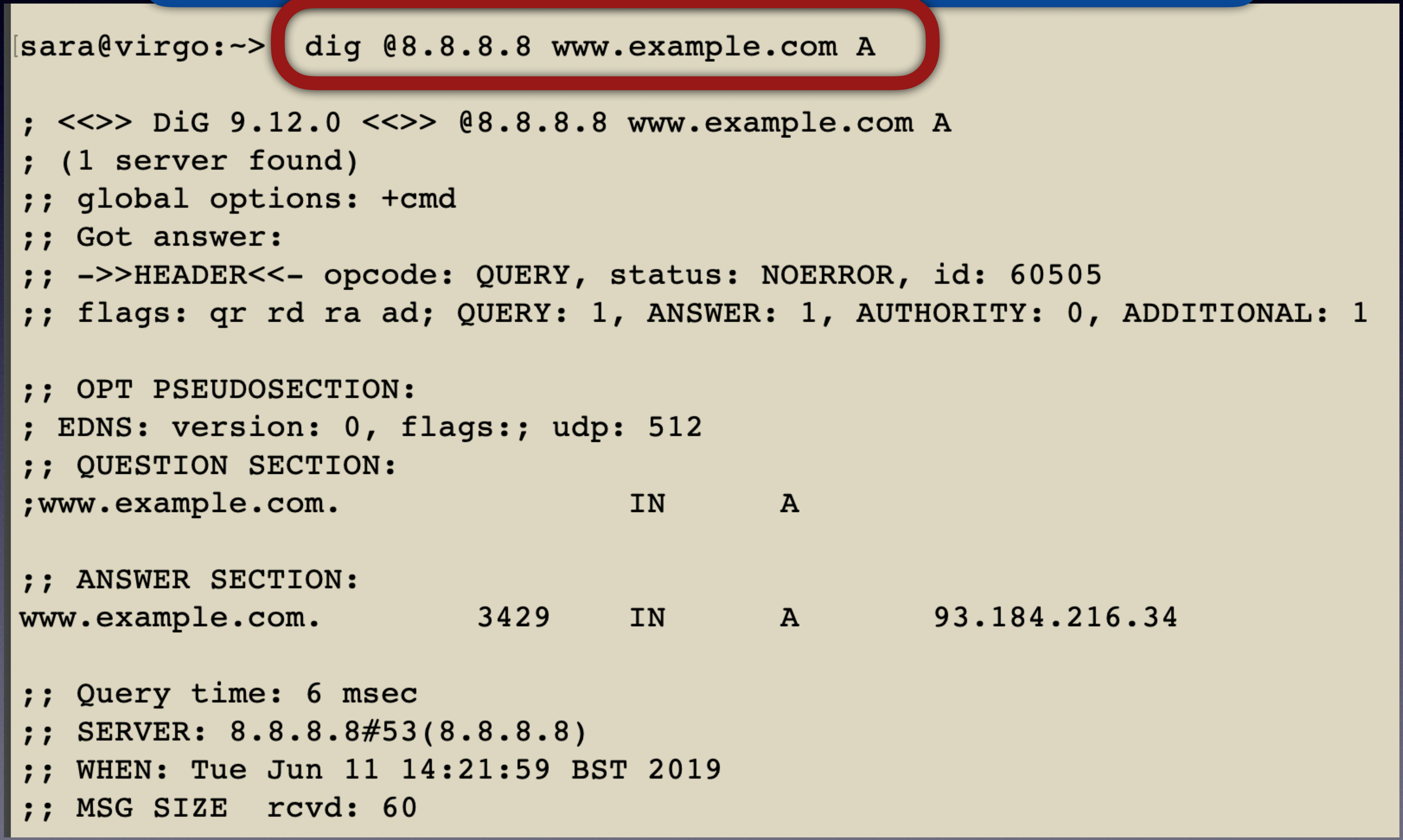
‘dig’ is available on most *nix systems (or ‘drill’)



```
sara@virgo:~>  dig @8.8.8.8 www.example.com A

; <<>> DiG 9.12.0 <<>> @8.8.8.8 www.example.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60505
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3429    IN      A       93.184.216.34

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Jun 11 14:21:59 BST 2019
;; MSG SIZE  rcvd: 60
```

DNS Basics - A UDP query

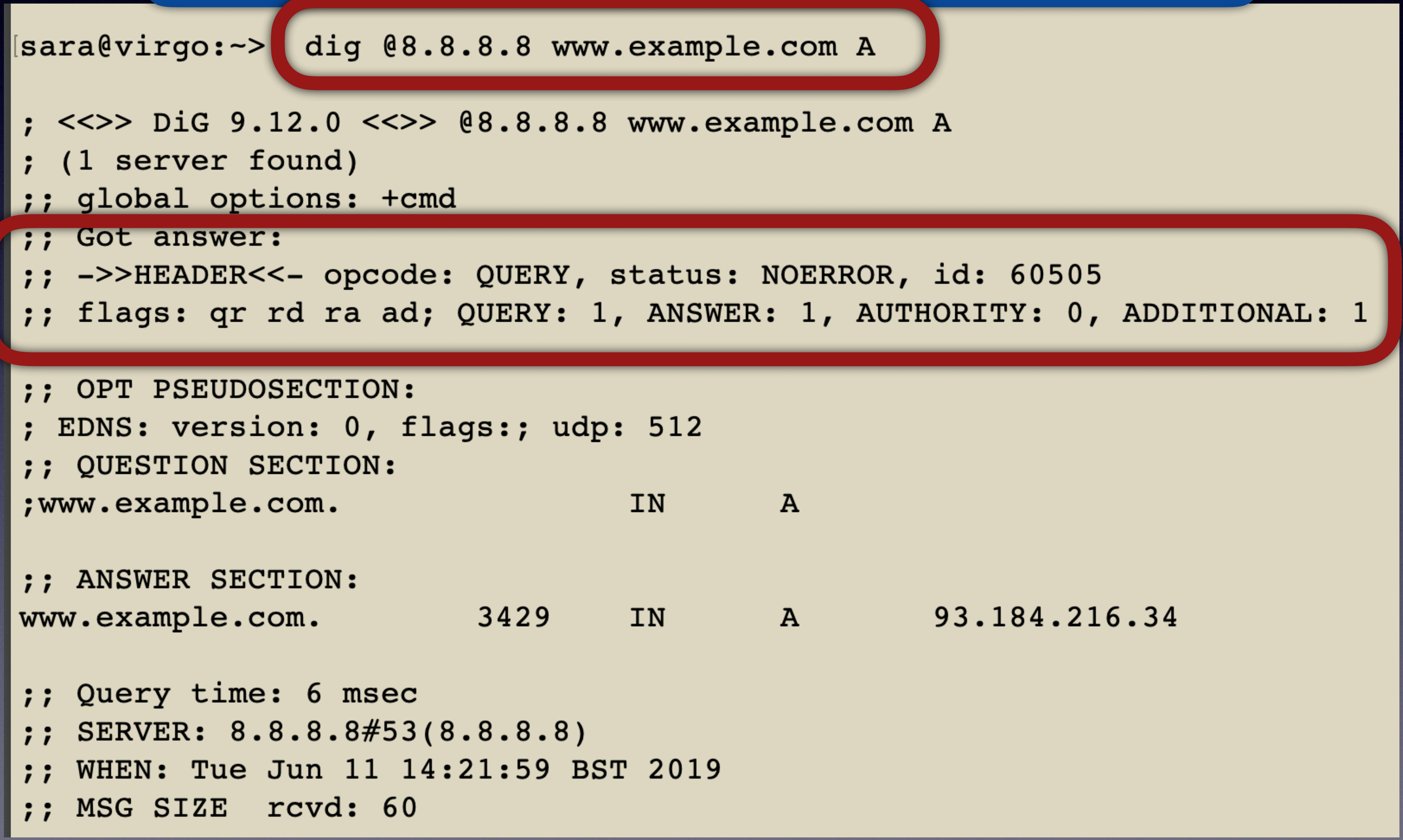
'dig' is available on most *nix systems (or 'drill')



```
sara@virgo:~> dig @8.8.8.8 www.example.com A

; <<>> DiG 9.12.0 <<>> @8.8.8.8 www.example.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60505
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3429    IN      A       93.184.216.34

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Jun 11 14:21:59 BST 2019
;; MSG SIZE  rcvd: 60
```

DNS Basics - A UDP query

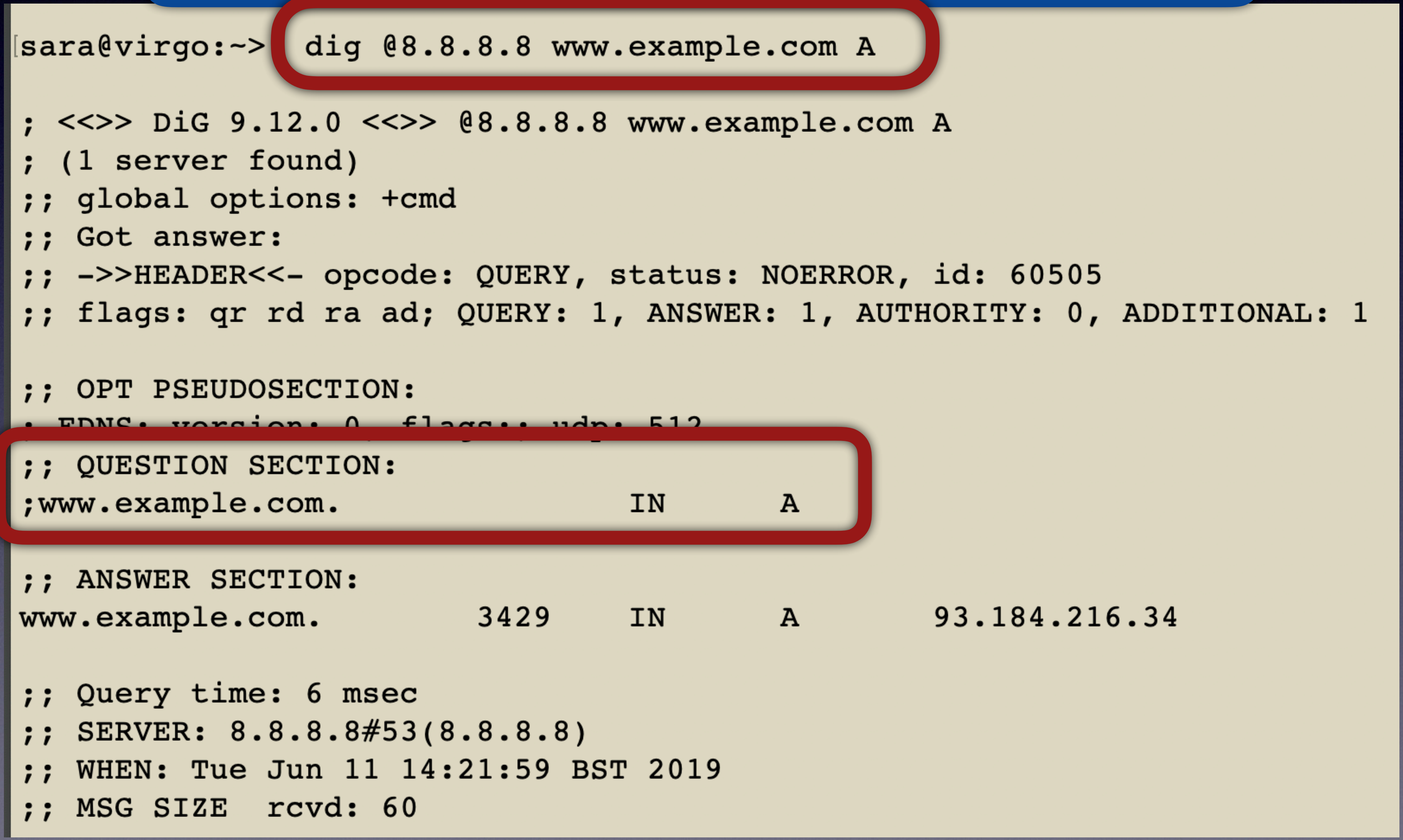
‘dig’ is available on most *nix systems (or ‘drill’)



```
sara@virgo:~> dig @8.8.8.8 www.example.com A

; <<>> DiG 9.12.0 <<>> @8.8.8.8 www.example.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60505
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3429    IN      A       93.184.216.34

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Jun 11 14:21:59 BST 2019
;; MSG SIZE  rcvd: 60
```

DNS Basics - A UDP query

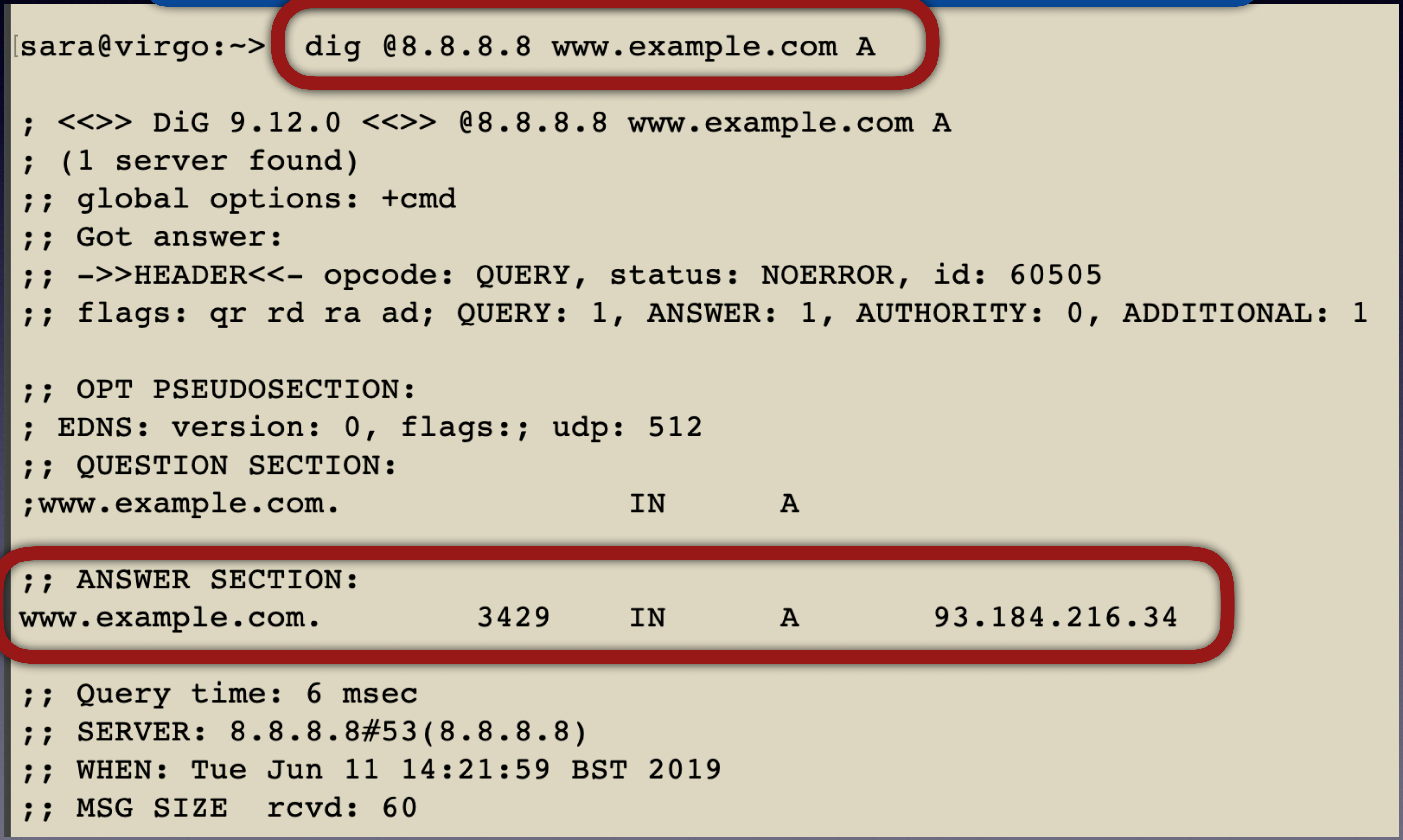
‘dig’ is available on most *nix systems (or ‘drill’)



```
sara@virgo:~> dig @8.8.8.8 www.example.com A

; <<>> DiG 9.12.0 <<>> @8.8.8.8 www.example.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60505
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3429    IN      A       93.184.216.34

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Jun 11 14:21:59 BST 2019
;; MSG SIZE  rcvd: 60
```

DNS Basics - A UDP query

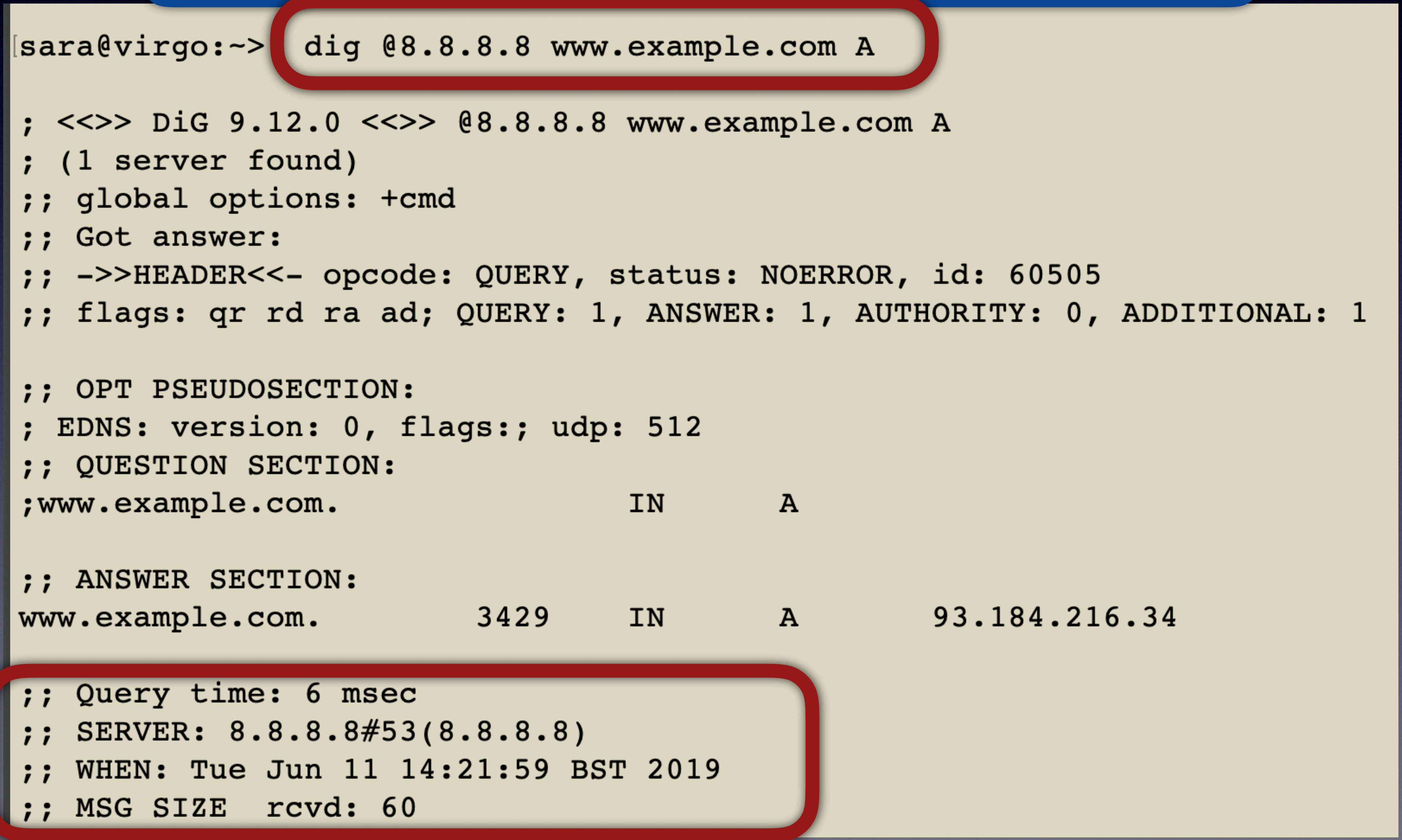
‘dig’ is available on most *nix systems (or ‘drill’)



```
sara@virgo:~> dig @8.8.8.8 www.example.com A

; <<>> DiG 9.12.0 <<>> @8.8.8.8 www.example.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60505
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3429    IN      A       93.184.216.34

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Jun 11 14:21:59 BST 2019
;; MSG SIZE  rcvd: 60
```

DNS Basics - A UDP query

'dig' is available on most *nix systems (or 'drill')

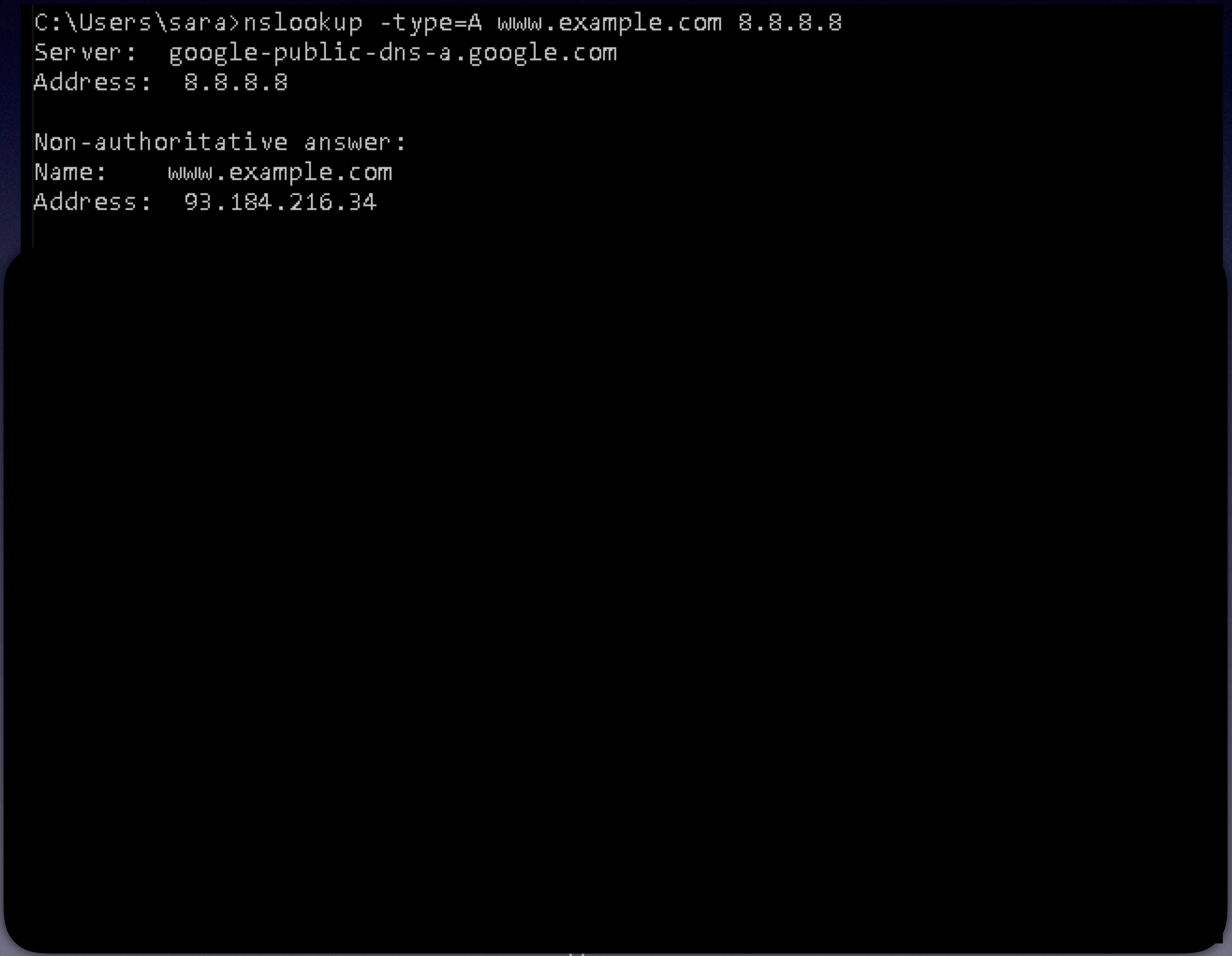


```
sara@virgo:~> dig @8.8.8.8 www.example.com A

; <<>> DiG 9.12.0 <<>> @8.8.8.8 www.example.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60505
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        3429    IN      A       93.184.216.34

;; Query time: 6 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Jun 11 14:21:59 BST 2019
;; MSG SIZE  rcvd: 60
```

DNS Basics - A UDP query

‘nslookup’ is available on Windows

order is important!

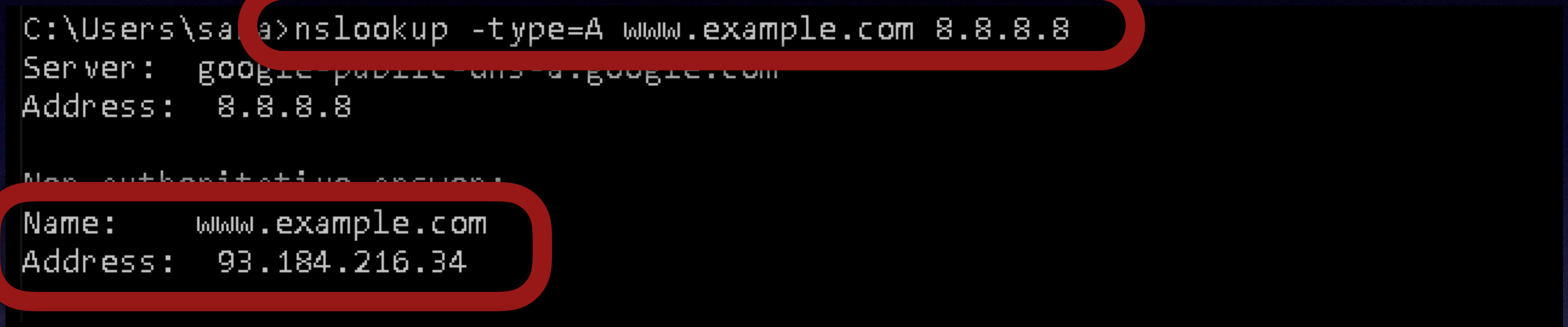


```
C:\Users\sara>nslookup -type=A www.example.com 8.8.8.8
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.example.com
Address:  93.184.216.34
```

DNS Basics - A UDP query

'nslookup' is available on Windows

order is important!



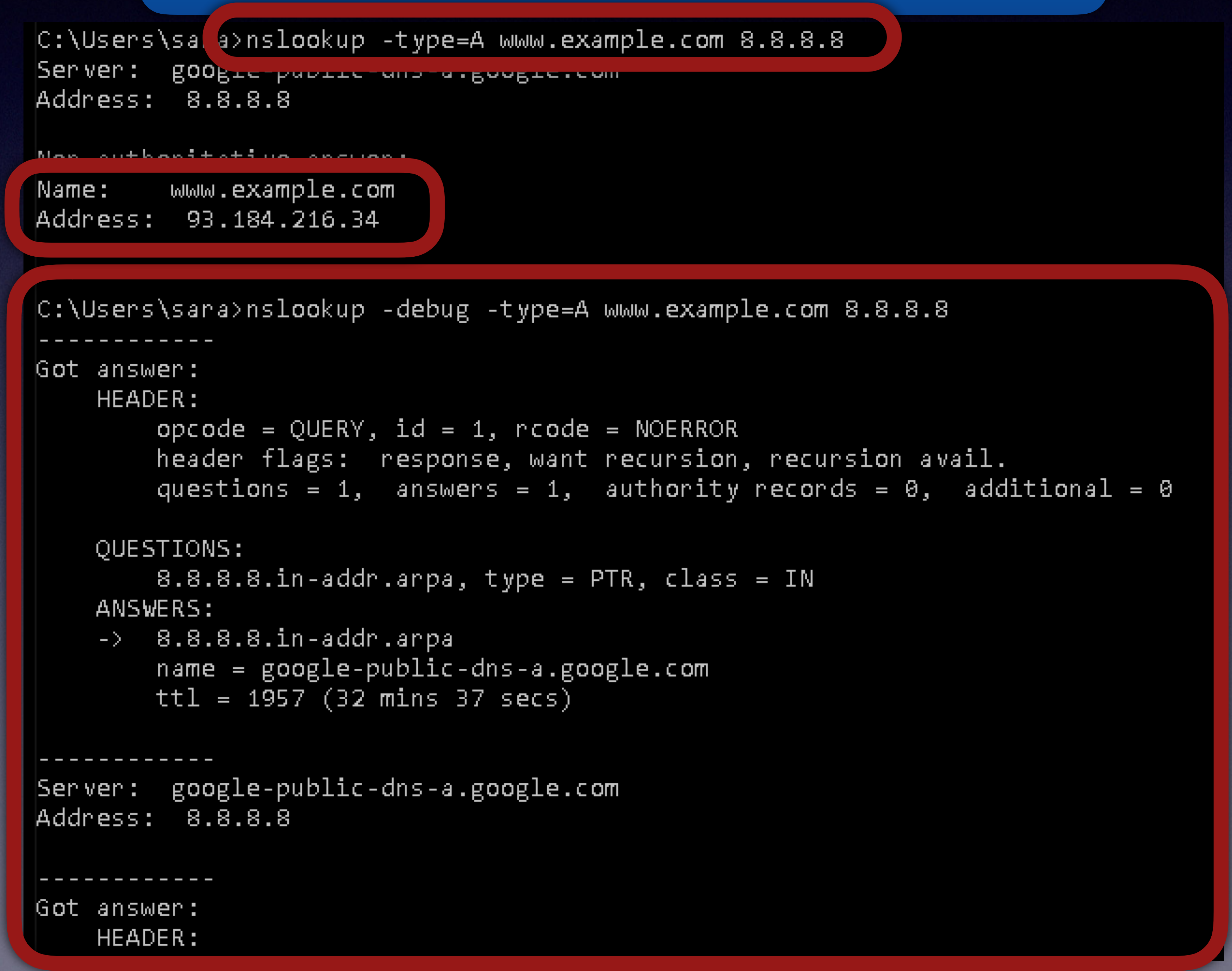
```
C:\Users\sa a>nslookup -type=A www.example.com 8.8.8.8
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.example.com
Address:  93.184.216.34
```

[redacted]

DNS Basics - A UDP query

‘nslookup’ is available on Windows

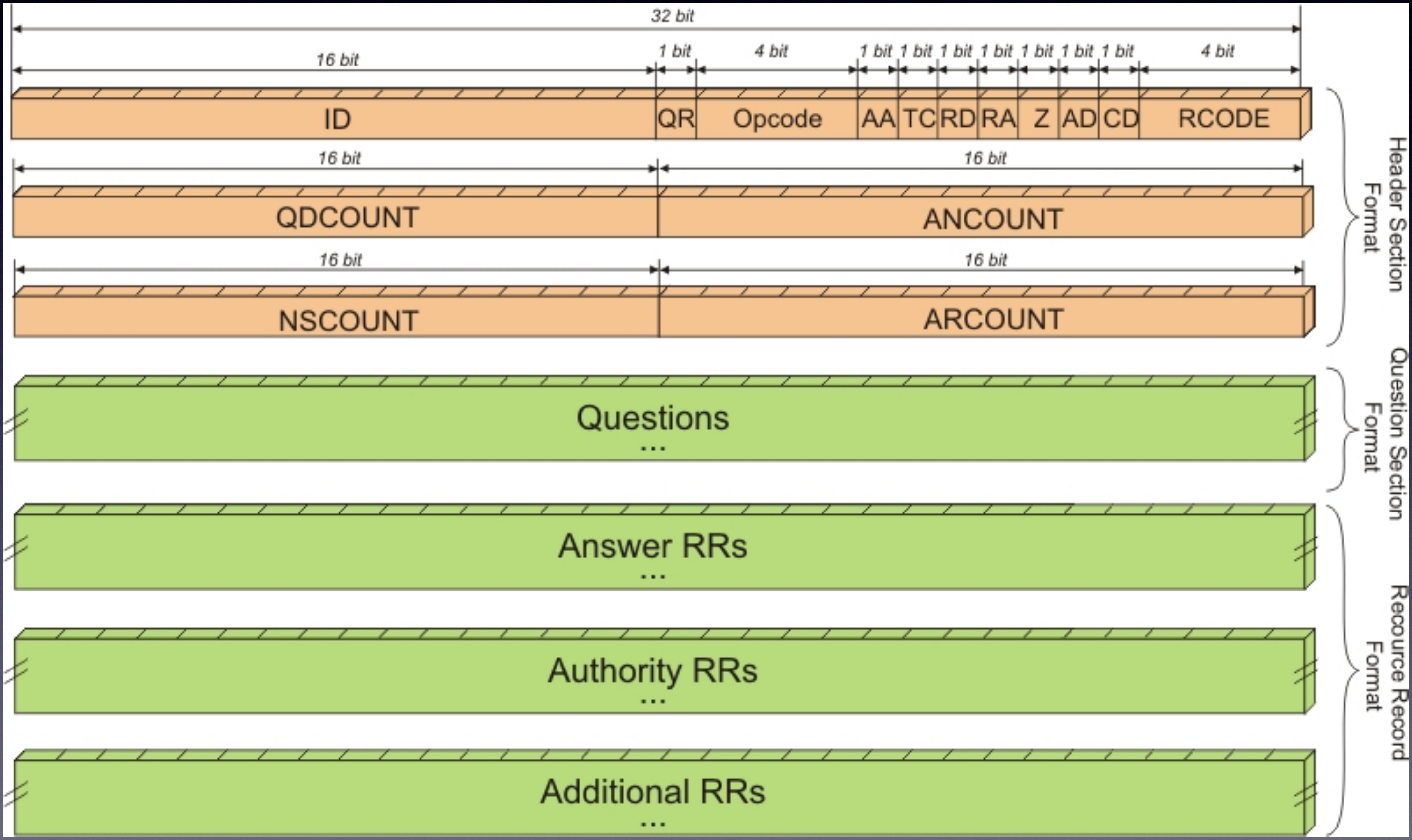
order is important!



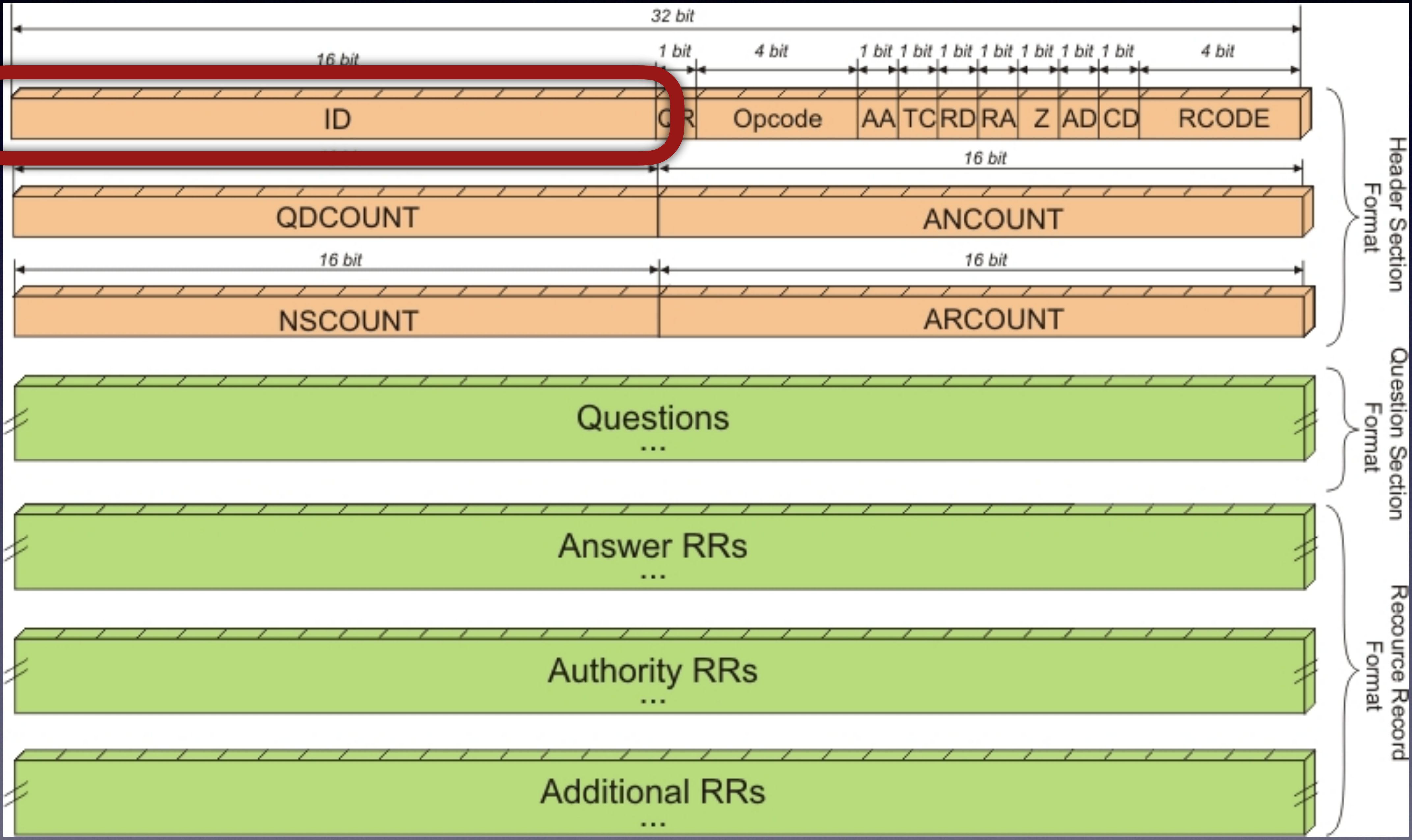
```
C:\Users\sara>nslookup -type=A www.example.com 8.8.8.8
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.example.com
Address:  93.184.216.34
```

```
C:\Users\sara>nslookup -debug -type=A www.example.com 8.8.8.8
------------
Got answer:
    HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  response, want recursion, recursion avail.
        questions = 1,  answers = 1,  authority records = 0,  additional = 0

    QUESTIONS:
        8.8.8.8.in-addr.arpa, type = PTR, class = IN
    ANSWERS:
    ->  8.8.8.8.in-addr.arpa
        name = google-public-dns-a.google.com
        ttl = 1957 (32 mins 37 secs)

------------
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

------------
Got answer:
    HEADER:
```

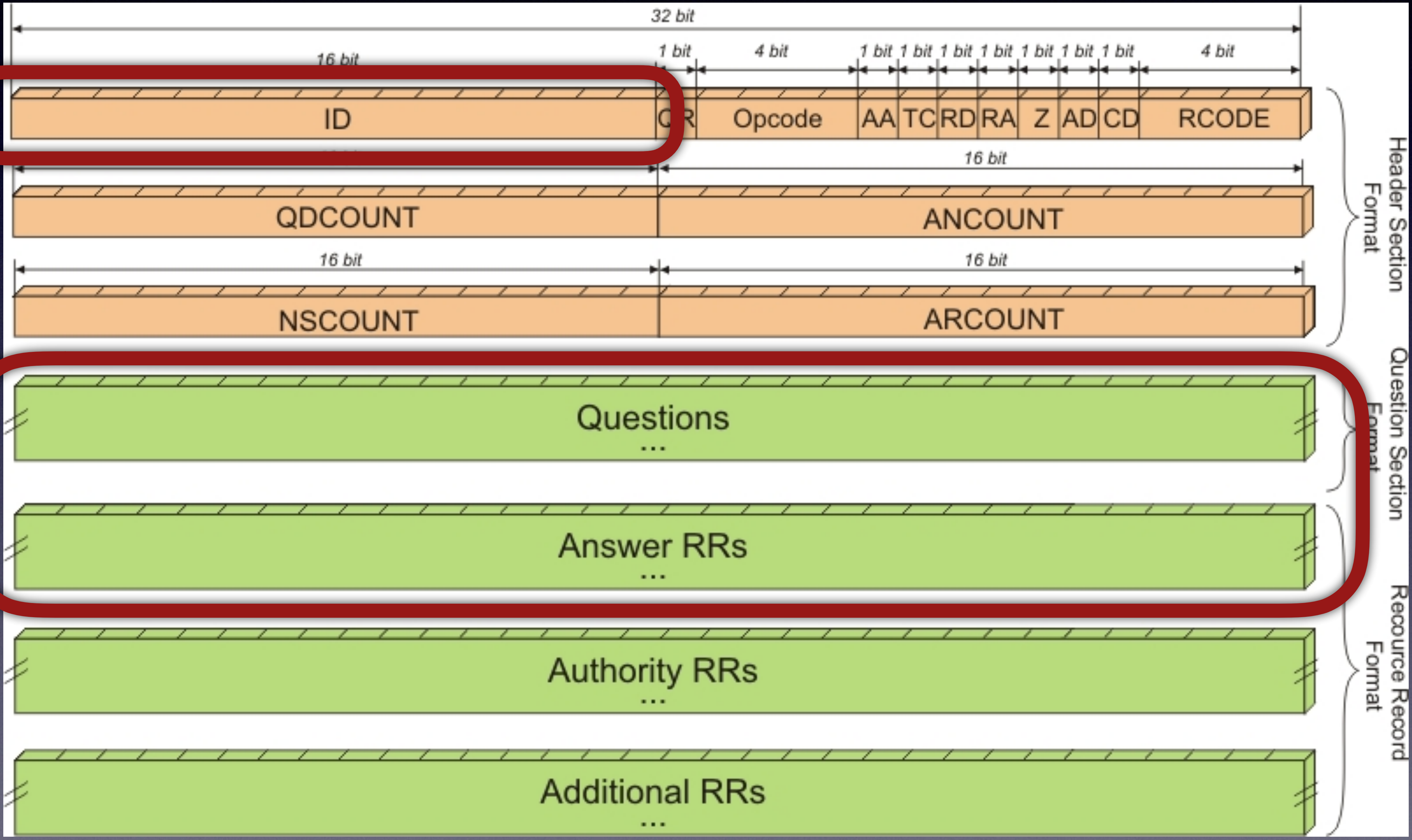
DNS Packet structure



DNS Packet structure



DNS Packet structure



Exercise - use dig

- Do a dig for a domain name and also try these options

  ```
  dig @8.8.8.8 www.example.com A
  ```

  - +short (just IP)
  - +qr (also print query)
  - +trace (trace delegations from the root - shows auth servers)
  - +tcp (but alas, ‘dig’ doesn’t do TLS, more on that later)

Exercise - look at your DNS Settings

- Do ‘dig’ again, without the @8.8.8.8 - what IP was used?
- Look at system settings via a GUI or command line
  - Note there are usually multiple settings from command line
- See next slides for OS specifics
  - *nix systems
  - Windows

Finding your DNS settings - *nix

- **macOS GUI**:
  - Settings->Network->Advanced->DNS

- **Command line:** (Most *nix distros don't directly use /etc/resolv.conf now)
  - systemd-resolved.service: `` `resolvectl status` `` (Global)
  - for macOS run `‘scutil —-dns`` ` and look at 'lan'

- Flush the cache:
  - macOS 10.14: `‘sudo killall -HUP mDNSResponder`` `
  - systemd: `` `sudo systemd-resolve --flush-caches` ``

Finding your DNS settings - Windows

- Open the Control Panel
- Choose 'Network and Internet'
- Choose 'View network status and tasks' under 'Network and Sharing Center'
- Choose 'Change adapter settings' from the left hand menu
- Then choose your interface - most likely either 'Wi-fi'
- In the dialog that appears, click on the 'Properties' button at the bottom
- Double click on 'Internet Protocol Version 4 (TCP/IPv4)' at the bottom
- Repeat for 'Internet Protocol Version 6 (TCP/IPv6)'

- Flush DNS cache: from a terminal run `ipconfig /flushdns`

Exercise - DNS traffic inspection

Need Wirshark 3.0.2

- Install Wireshark (GUI) or tcpdump (command line)
- Close all your apps
- Flush the local DNS cache (see previous slides)

- Start a capture with WS:
  - Choose the Wifi (or Ethernet) interface
  - Add a capture filter of 'port 53'
  - Hit the Blue fin
  - Add a filter of 'ip.addr == 8.8.8.8'
- `tcpdump -i eth0 host 8.8.8.8 and port 5 -n <-v> <-X>`

- Do a few 'dig's to 8.8.8.8 and look at the packets
(capture bytes for later on!)

Exercise - DNS traffic inspection

- Close all your apps
- Flush the local DNS cache (see previous slides)
- Then open one by one to see the DNS queries….
- Change your DNS setting and send all your queries to Google/Cloudfare/Quad9 (if you dare!)

What DNS reveals

- Mail clients - email hosting (server name!), which client
- Chat services - jabber server and Slack channels
- Calendars - where hosted
- Apps - often check for updates when opened
- Browsers - which client, which plugins. Open tabs, most visited, favourites, …. Then your browsing…

DoT on the Desktop

kdig & getdns_query

- ‘kdig’ comes as part of the ‘knot’ package (no Windows package)
    - syntax is exactly like ‘dig’ but…
    - +tls
- getdns_query - comes as part of the Stubby packages,
    - syntax is similar to dig, but different and output format is very different!

Lets do DoT on the desktop!

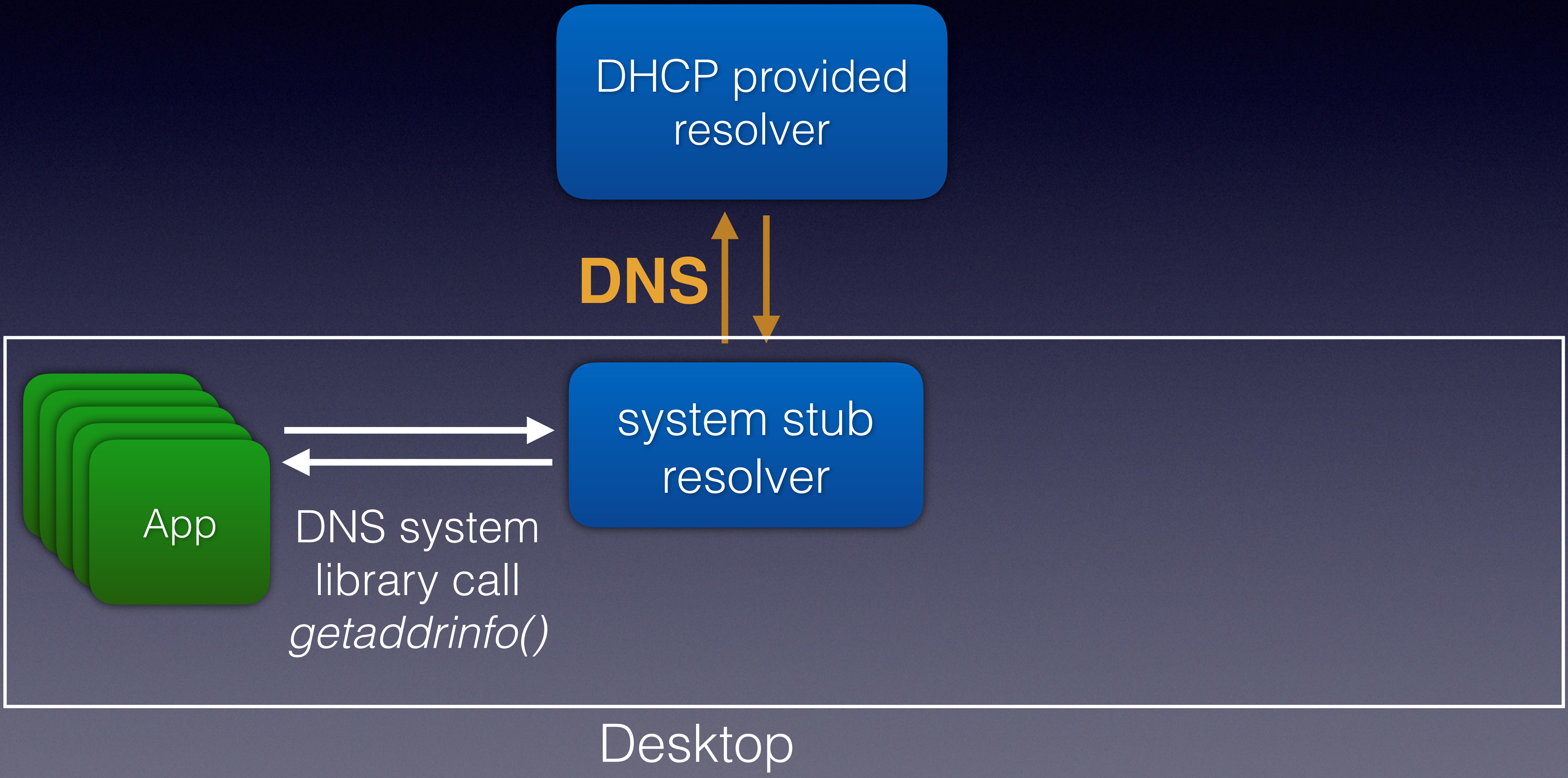
Lets do DoT on the desktop!

Software	See DNS Privacy Clients for details
Stubby	• Specifically designed as a privacy stub • Best for upstream performance (pipelines queries) + privacy features (but no cache yet)
Unbound	• Can use as a caching forwarder • But uses a new connection for each query (poor performance) • Can also configure stub zones
BIND	• Does not do DoT natively, but can be set up with a TLS proxy to forward queries over TLS
Knot resolver	• Similar to Unbound but less well known
systemd	• Native support but very 'systemd'-like….(only Opportunistic)

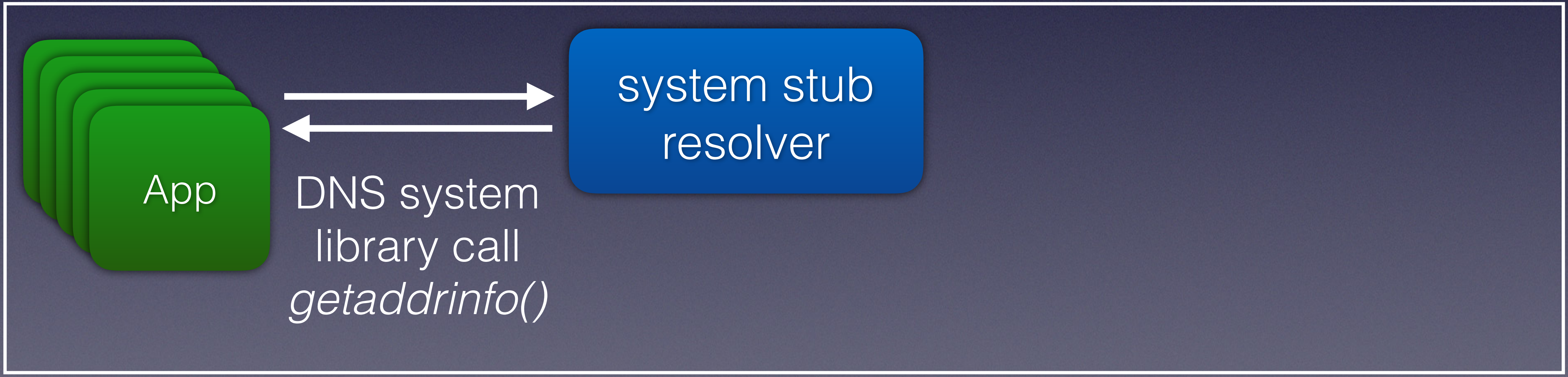
Lets do DoT on the desktop!

Software	See DNS Privacy Clients for details
Stubby	• Specifically designed as a privacy stub • Best for upstream performance (pipelines queries) + privacy features (but no cache yet)
Unbound	• Can use as a caching forwarder • But uses a new connection for each query (poor performance) • Can also configure stub zones
BIND	• Does not do DoT natively, but can be set up with a TLS proxy to forward queries over TLS
Knot resolver	• Similar to Unbound but less well known
systemd	• Native support but very 'systemd'-like….(only Opportunistic)

How does Stubby work?

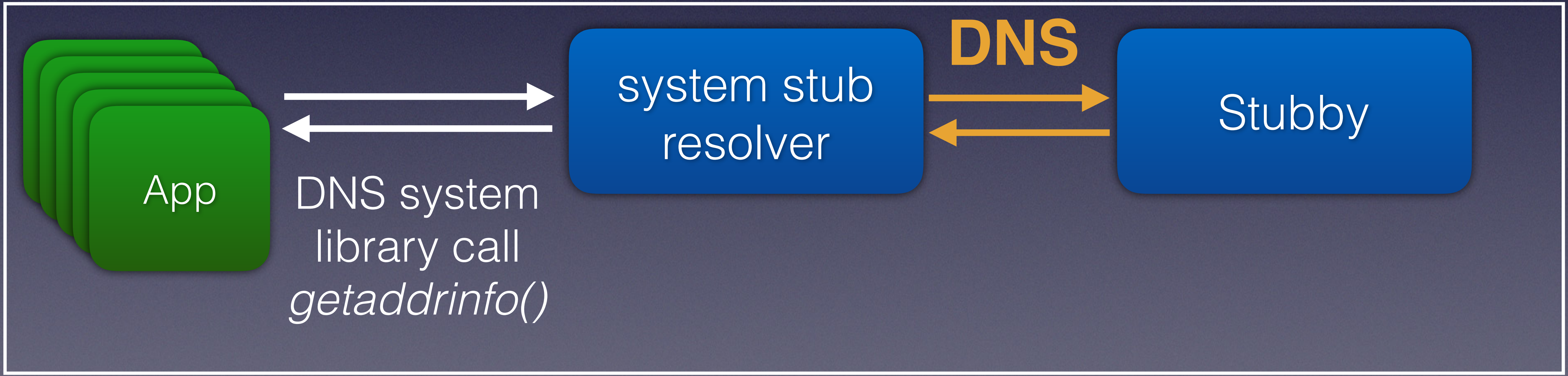


How does Stubby work?



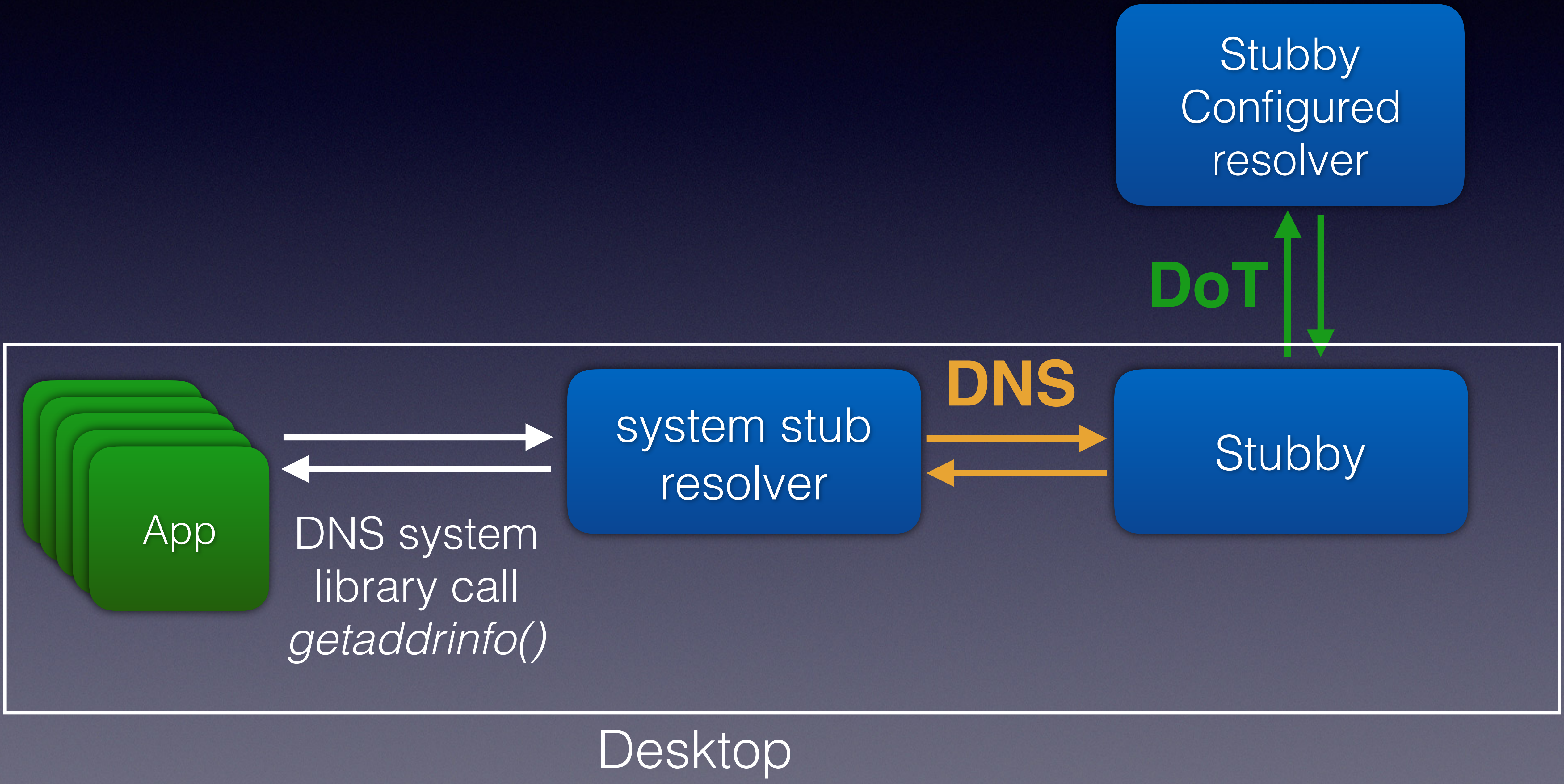
Desktop

How does Stubby work?



Desktop

How does Stubby work?

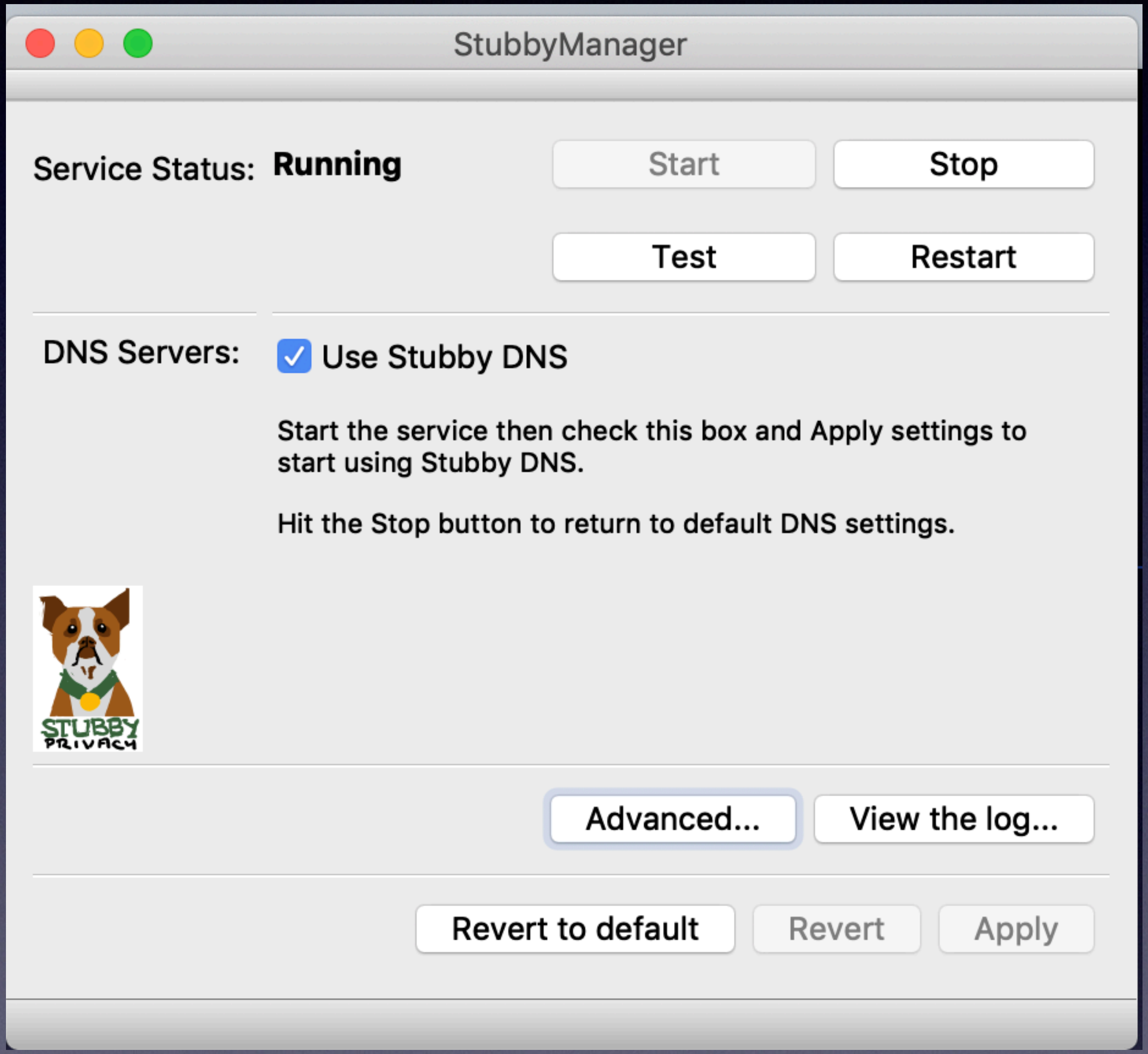


Stubby - Installing

- Stubby Homepage - built on the getdns library
- Stubby Installation Guide
  - Linux distros
    - Packages (NOTE: debian package version is wrong!)
    - Build from source
  - macOS
    - Homebrew
    - Prototype StubbyManager GUI for macOS
  - Windows
    - Windows installer (MSI, zip)
    - Chocolatey package

Install Stubby!

Stubby GUI



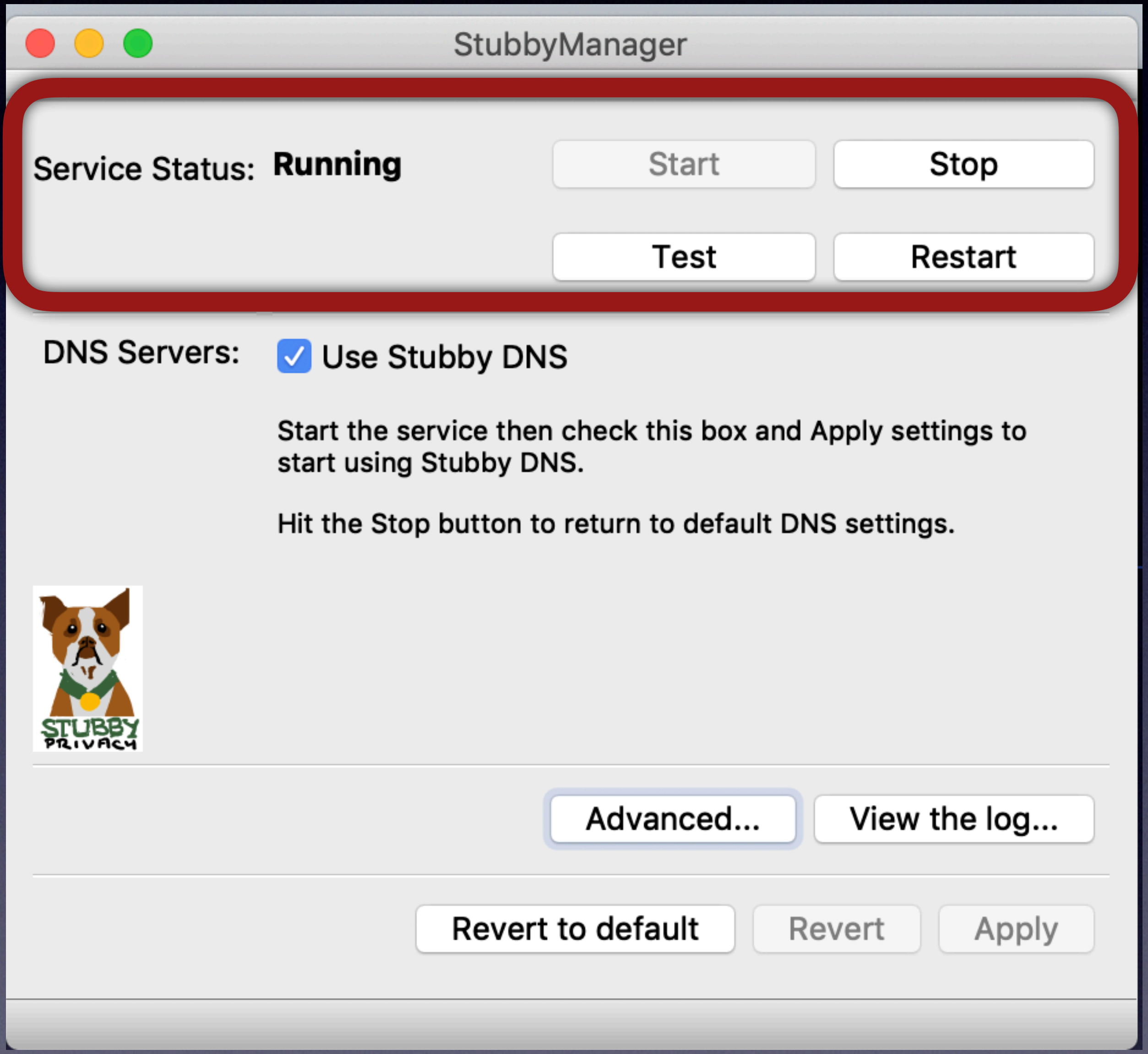
Service Status: **Running**

DNS Servers: ☑ Use Stubby DNS

Start the service then check this box and Apply settings to start using Stubby DNS.

Hit the Stop button to return to default DNS settings.

Stubby GUI



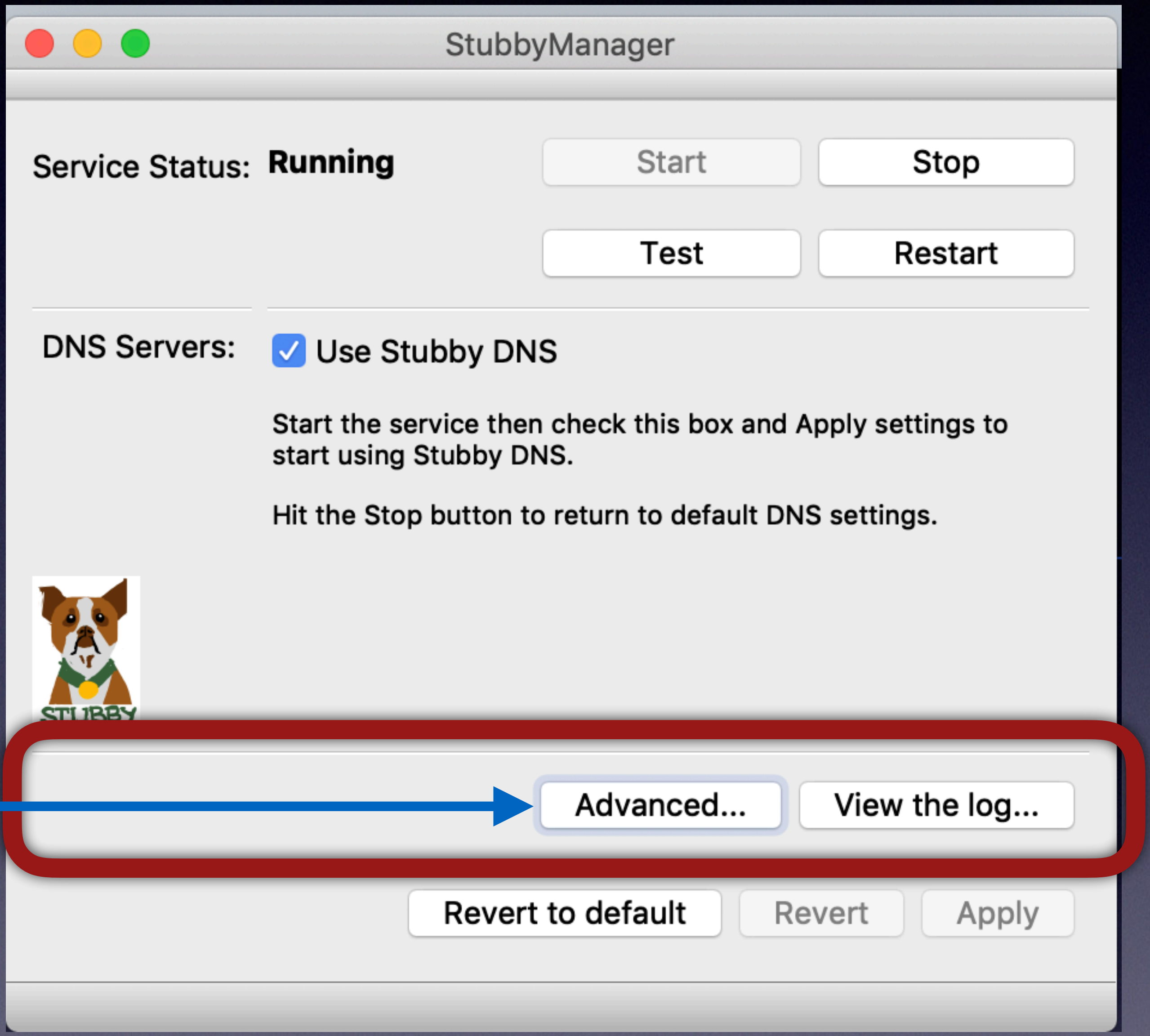
Service Status: **Running**

DNS Servers: ✓ Use Stubby DNS

Start the service then check this box and Apply settings to start using Stubby DNS.

Hit the Stop button to return to default DNS settings.

Stubby GUI

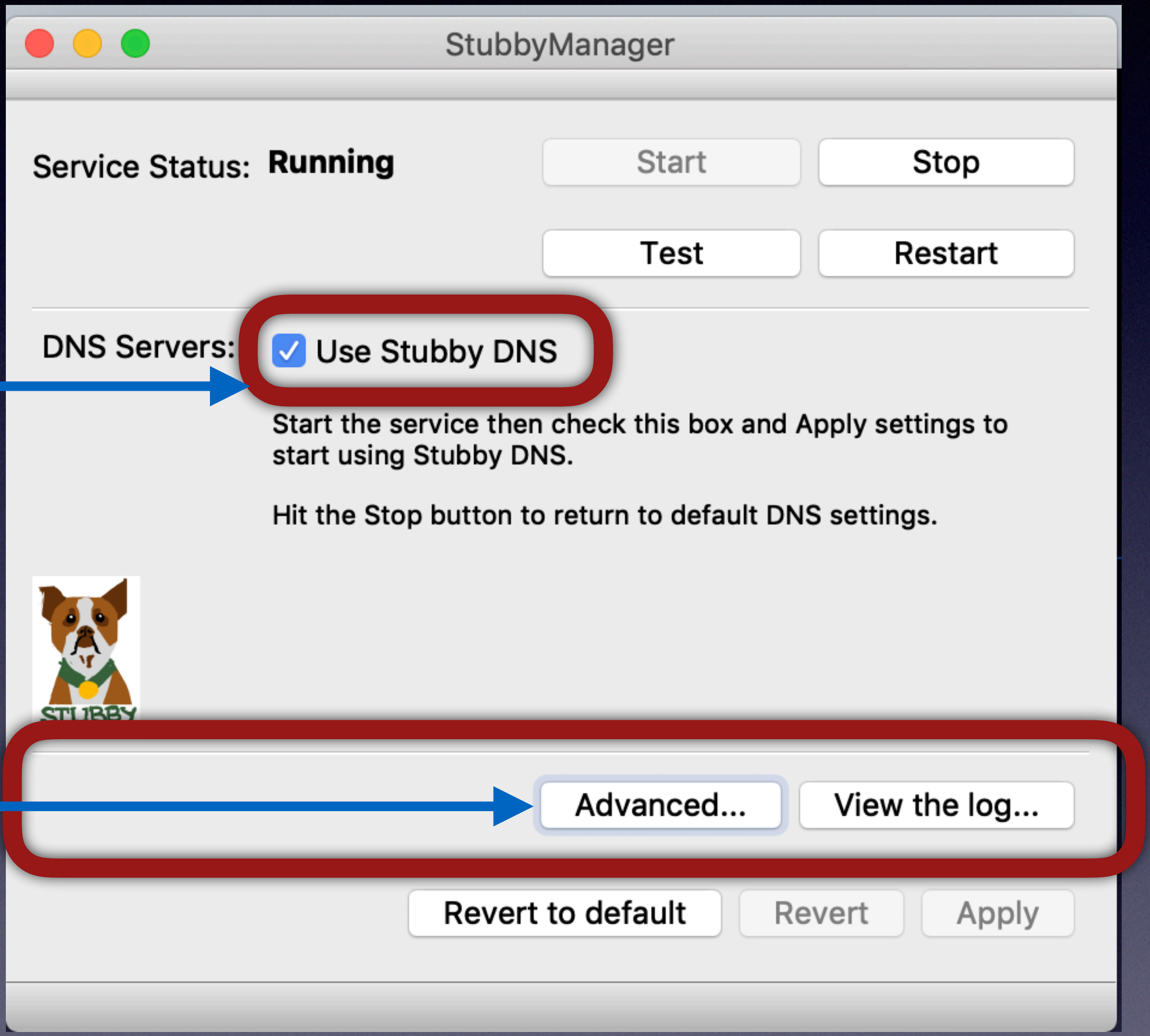


Start the service then check this box and Apply settings to start using Stubby DNS.

Hit the Stop button to return to default DNS settings.

Edit the config file

Stubby GUI



Service Status: **Running**

DNS Servers: [✓ Use Stubby DNS]

Start the service then check this box and Apply settings to start using Stubby DNS.

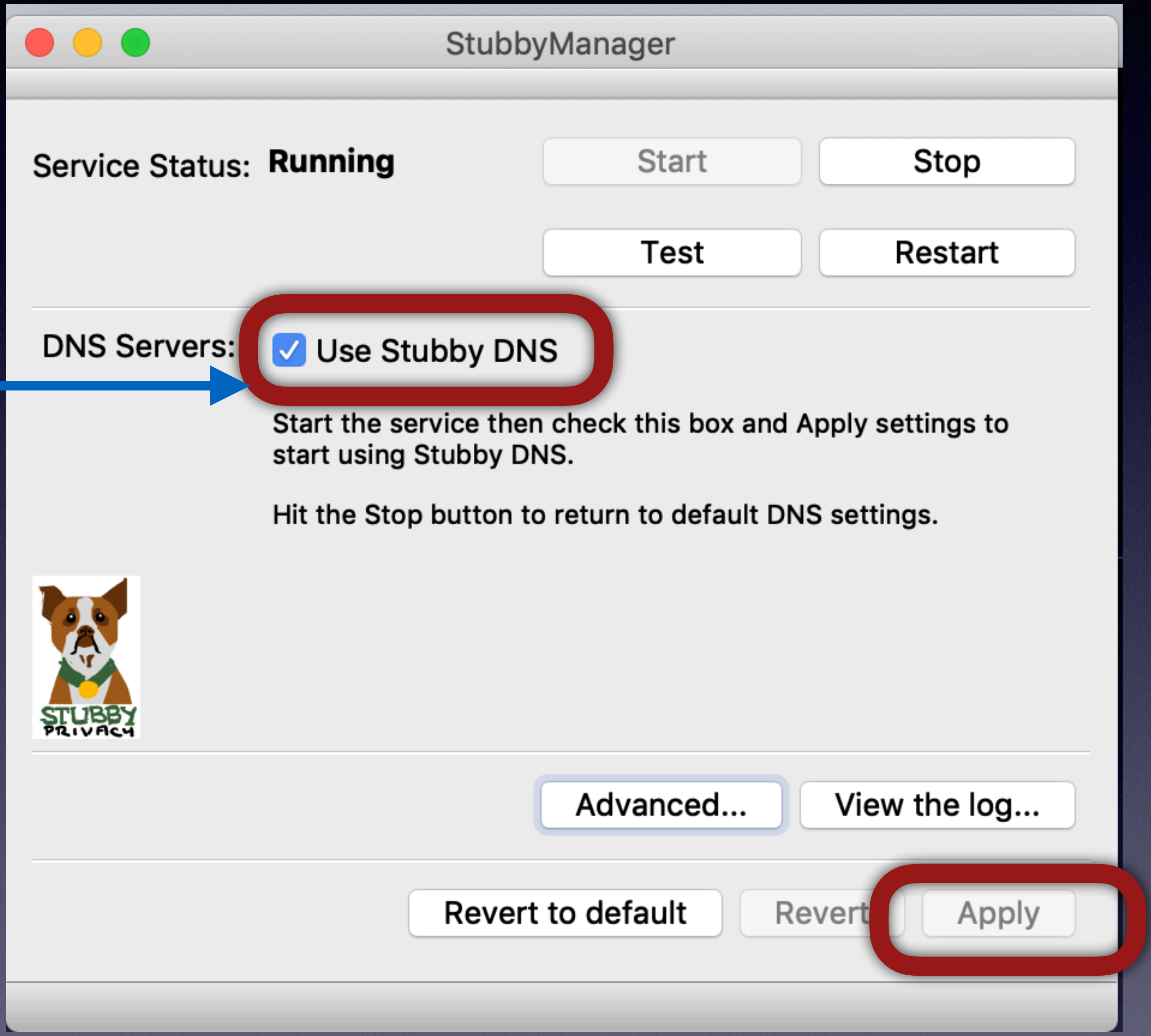
Hit the Stop button to return to default DNS settings.

Update system resolver

Edit the config file

Stubby GUI

Update system resolver


StubbyManager

Service Status: **Running**

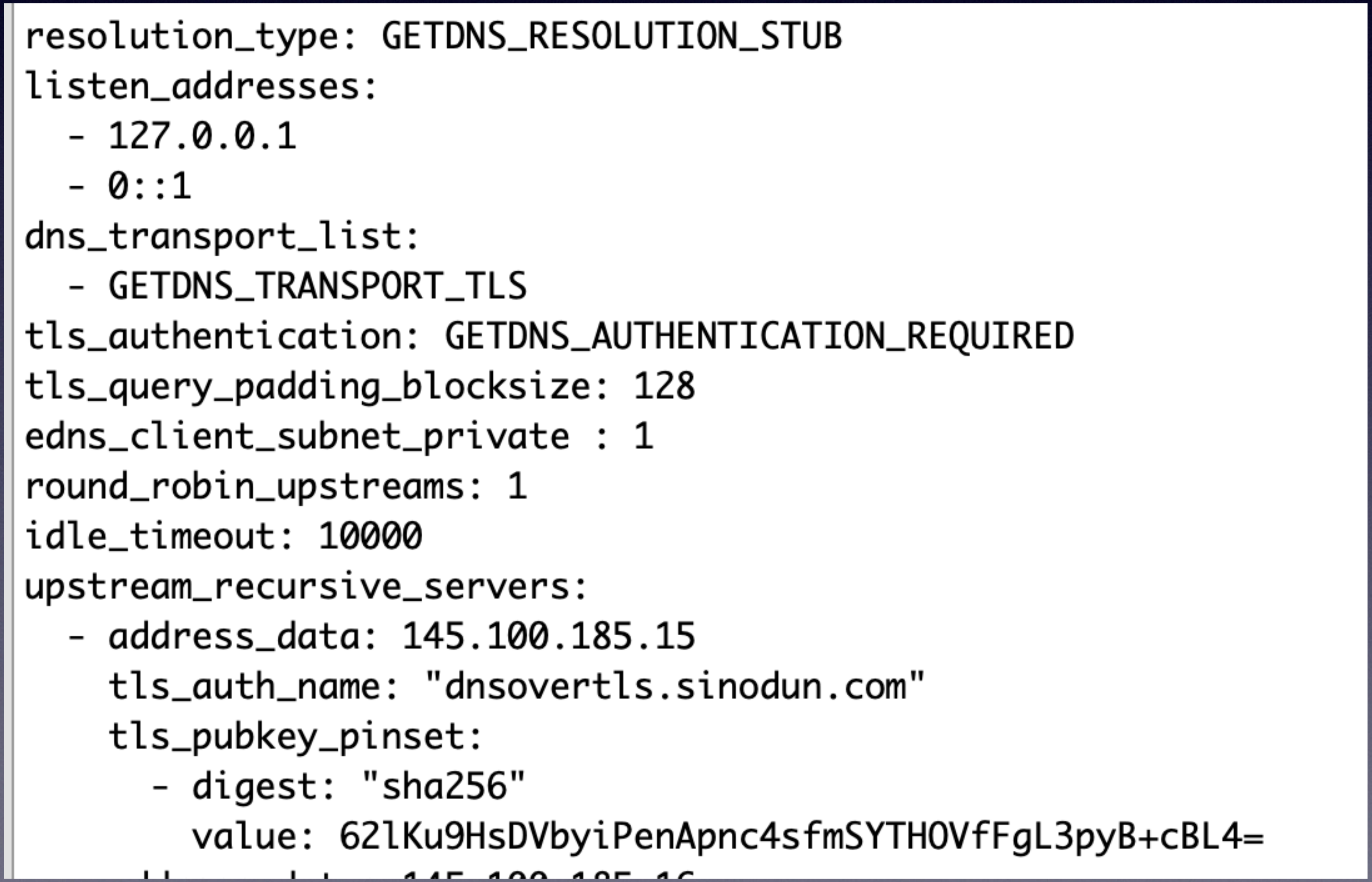
DNS Servers: [✓ Use Stubby DNS]

Start the service then check this box and Apply settings to start using Stubby DNS.

Hit the Stop button to return to default DNS settings.

Stubby - Configuring

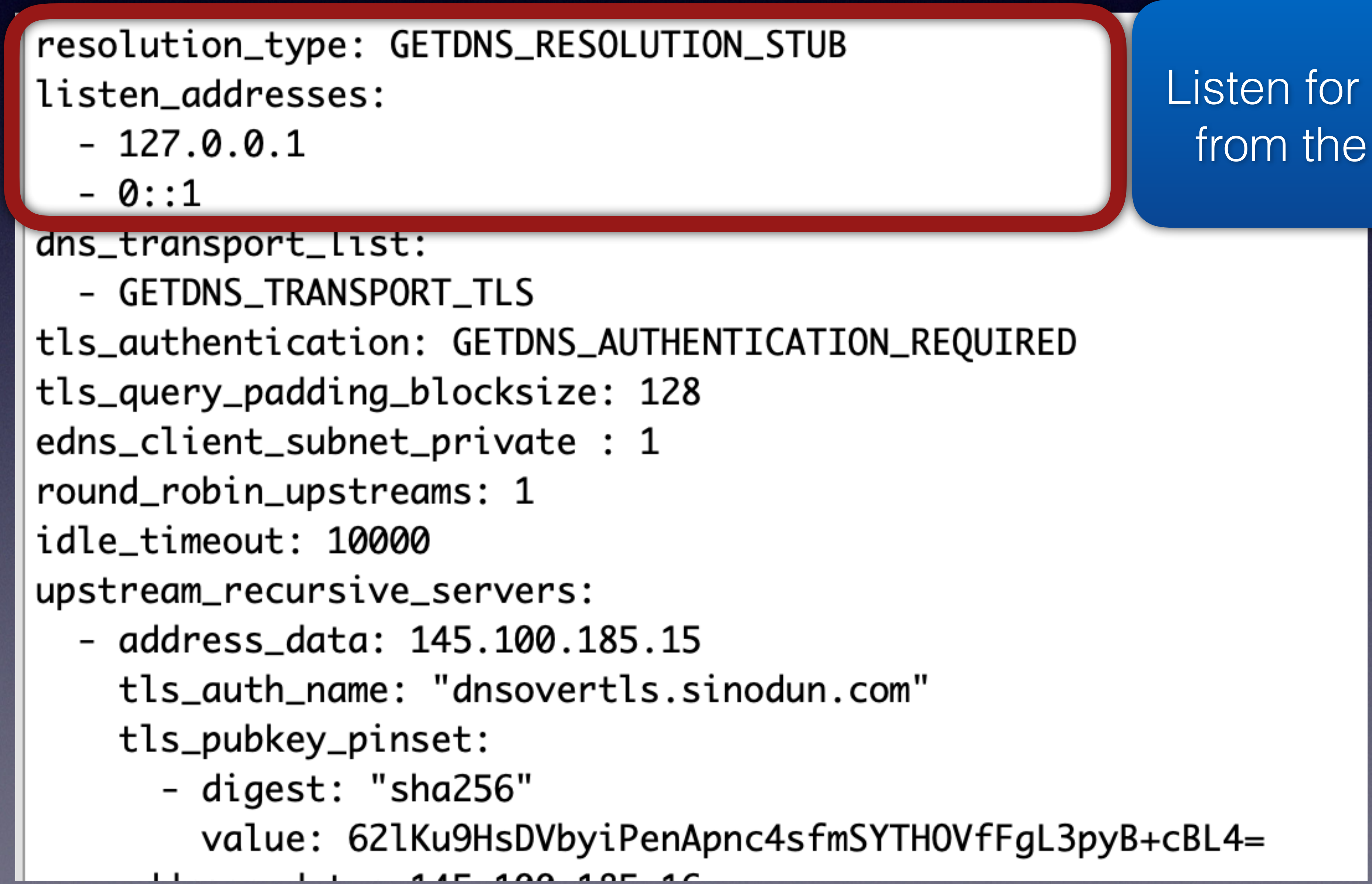
- Stubby has a stubby.yml config file - defaults:



```
resolution_type: GETDNS_RESOLUTION_STUB
listen_addresses:
  - 127.0.0.1
  - 0::1
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
tls_authentication: GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize: 128
edns_client_subnet_private : 1
round_robin_upstreams: 1
idle_timeout: 10000
upstream_recursive_servers:
  - address_data: 145.100.185.15
    tls_auth_name: "dnsovertls.sinodun.com"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 62lKu9HsDVbyiPenApnc4sfmSYTHOVfFgL3pyB+cBL4=
```

Stubby - Configuring

- Stubby has a stubby.yml config file - defaults:

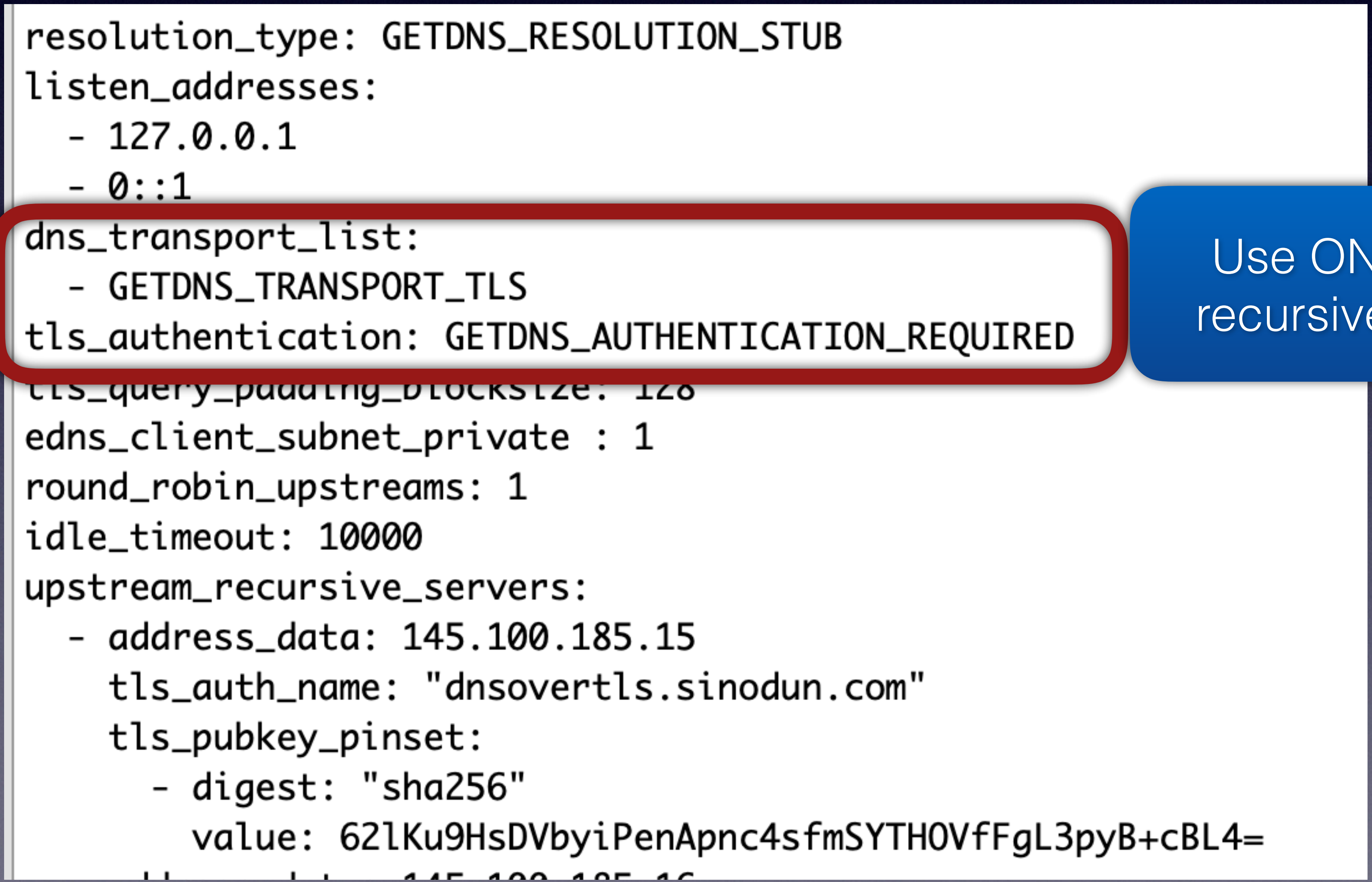


```
resolution_type: GETDNS_RESOLUTION_STUB
listen_addresses:
  - 127.0.0.1
  - 0::1
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
tls_authentication: GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize: 128
edns_client_subnet_private : 1
round_robin_upstreams: 1
idle_timeout: 10000
upstream_recursive_servers:
  - address_data: 145.100.185.15
    tls_auth_name: "dnsovertls.sinodun.com"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 62lKu9HsDVbyiPenApnc4sfmSYTHOVfFgL3pyB+cBL4=
```

Listen for queries coming from the local machine

Stubby - Configuring

- Stubby has a stubby.yml config file - defaults:

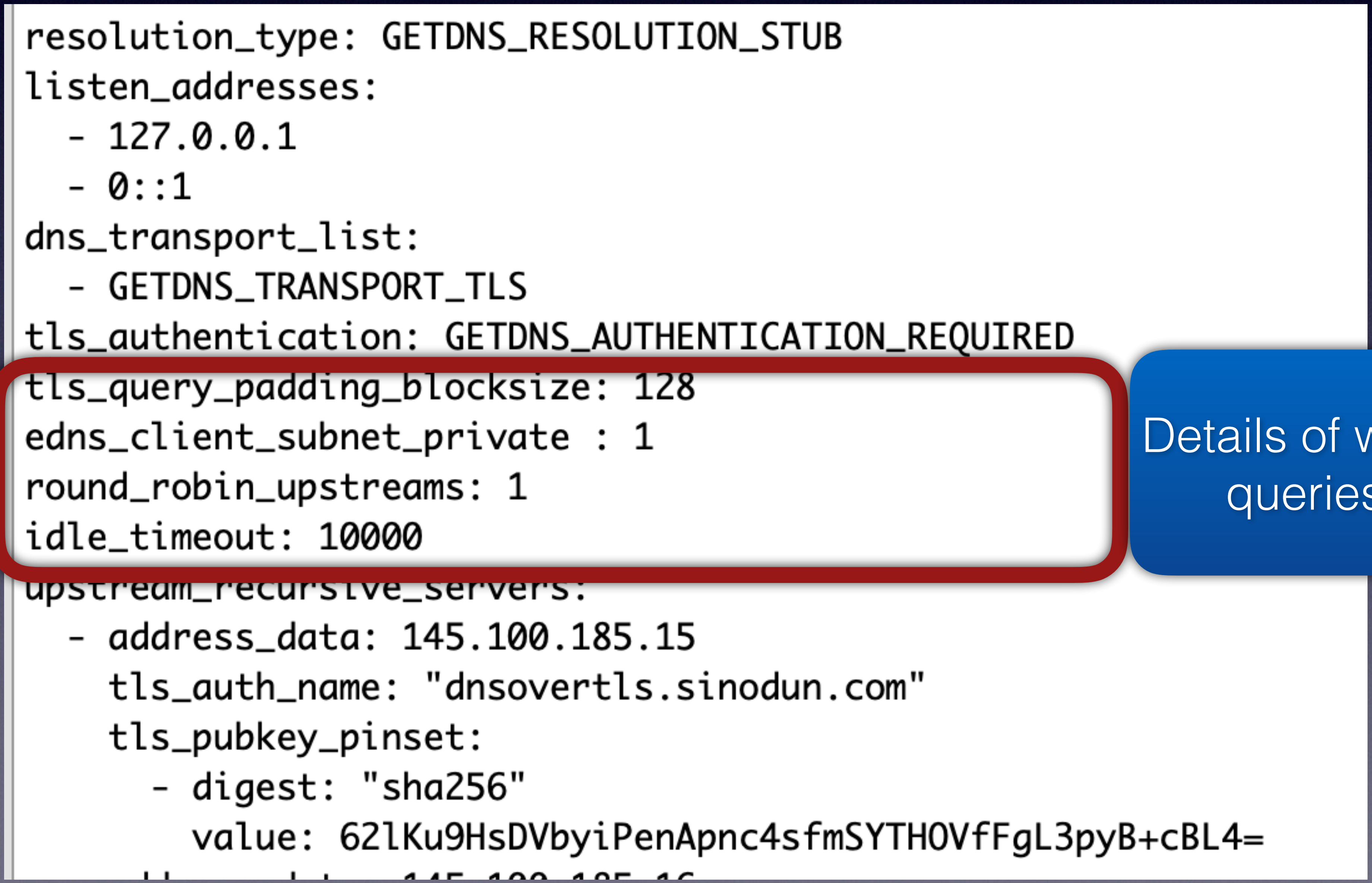


```
resolution_type: GETDNS_RESOLUTION_STUB
listen_addresses:
  - 127.0.0.1
  - 0::1
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
tls_authentication: GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize: 128
edns_client_subnet_private : 1
round_robin_upstreams: 1
idle_timeout: 10000
upstream_recursive_servers:
  - address_data: 145.100.185.15
    tls_auth_name: "dnsovertls.sinodun.com"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 62lKu9HsDVbyiPenApnc4sfmSYTHOVfFgL3pyB+cBL4=
  - address_data: 145.100.185.16
```

Use ONLY TLS to the recursive, require auth

Stubby - Configuring

- Stubby has a stubby.yml config file - defaults:

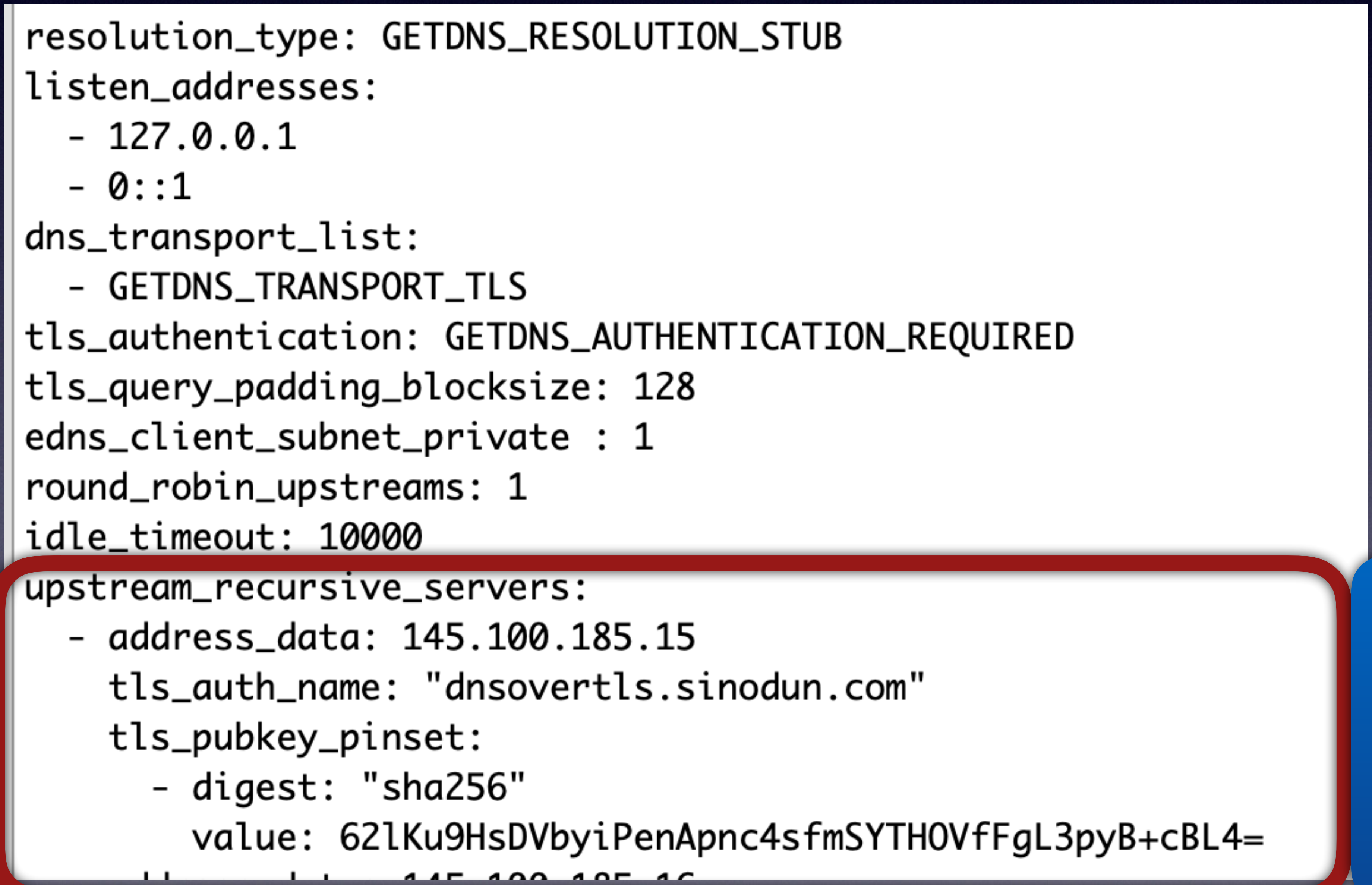


```
resolution_type: GETDNS_RESOLUTION_STUB
listen_addresses:
  - 127.0.0.1
  - 0::1
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
tls_authentication: GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize: 128
edns_client_subnet_private : 1
round_robin_upstreams: 1
idle_timeout: 10000
upstream_recursive_servers:
  - address_data: 145.100.185.15
    tls_auth_name: "dnsovertls.sinodun.com"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 62lKu9HsDVbyiPenApnc4sfmSYTHOVfFgL3pyB+cBL4=
```

Details of what/how to send queries to recursive

Stubby - Configuring

- Stubby has a stubby.yml config file - defaults:



```
resolution_type: GETDNS_RESOLUTION_STUB
listen_addresses:
  - 127.0.0.1
  - 0::1
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
tls_authentication: GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize: 128
edns_client_subnet_private : 1
round_robin_upstreams: 1
idle_timeout: 10000
upstream_recursive_servers:
  - address_data: 145.100.185.15
    tls_auth_name: "dnsovertls.sinodun.com"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 62lKu9HsDVbyiPenApnc4sfmSYTHOVfFgL3pyB+cBL4=
```

Which recursive + auth details (default is Stubby servers)

Stubby - Run it

- Run from the command line to start e.g.

```
sudo <path_to_exe>stubby -l <-C stubby.conf>
```

- Look at log - reports config

- Test Stubby. Open a new terminal and do a query directly to Stubby

```
dig @127.0.0.1 www.example.com
```

Run Stubby!

Stubby - Use for all DNS

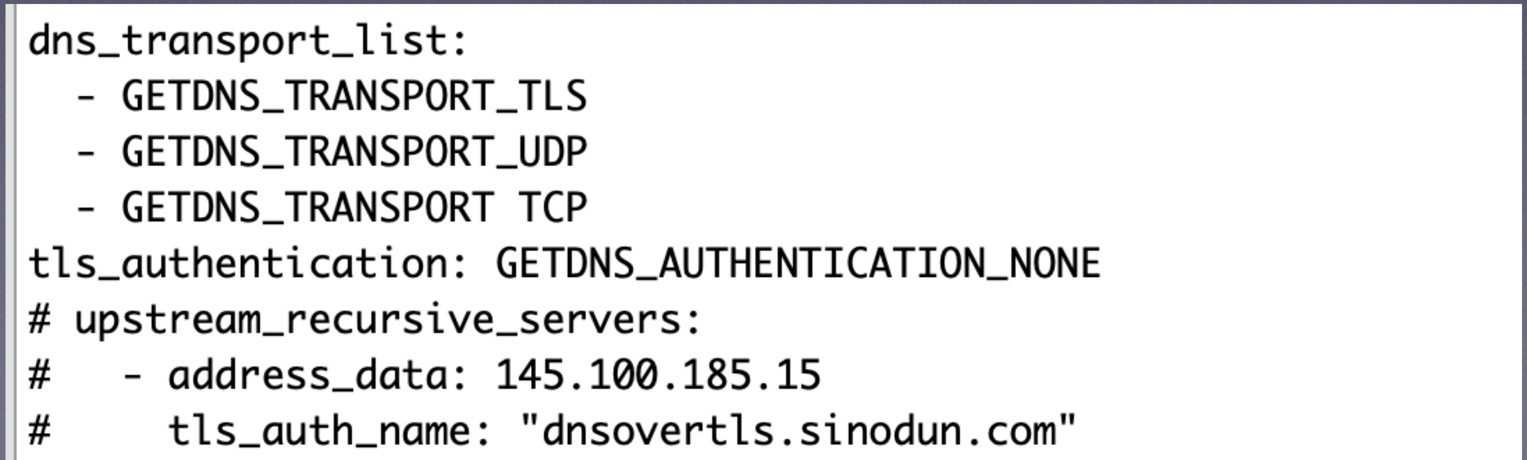
- Need to update your system resolver settings (and hit ‘Apply’): [systemd probably edit /etc/systemd/resolved.conf but…]
    - 127.0.0.1
    - 0::1
- Stubby log: lots of TLS connections (or a few long-lived)
- Look in Wireshark
    - No queries on port 53
    - The switch to port 853 - your queries are hidden in TLS sessions (note Wireshark doesn’t recognise DoT)!

How to decrypt TLS traffic?

- Client session keys - need a way to export them from the client (works for all TLS versions/cipher suites)
  - Stubby/Unbound do not support this yet
  - (Later we will see that Firefox does)
- Server private RSA keys - need access to private keys, normally only admins have this
  - Does not work with PFS ciphers (TLS 1.3, some TLS 1.2)
  - We can set up our own server to access the keys but have to restrict server to ‘weak’ ciphers

Stubby - other options

- Depending on your OS, you can configure Stubby to run as a service
  - Note you might have issues as you change network or hit a captive portal….
- Probably have getdns_query installed ('dig' like but with DoT)
  - `getdns_query @8.8.8.8~dns.google www.example.com -Lm +return_call_reporting`
- If you want to do Opportunistic DoT to the local resolver, reset your system resolvers and update 3 items in your config:



```
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
  - GETDNS_TRANSPORT_UDP
  - GETDNS_TRANSPORT TCP
tls_authentication: GETDNS_AUTHENTICATION_NONE
# upstream_recursive_servers:
#   - address_data: 145.100.185.15
#     tls_auth_name: "dnsovertls.sinodun.com"
```

Stubby - as a Sophisticated Service

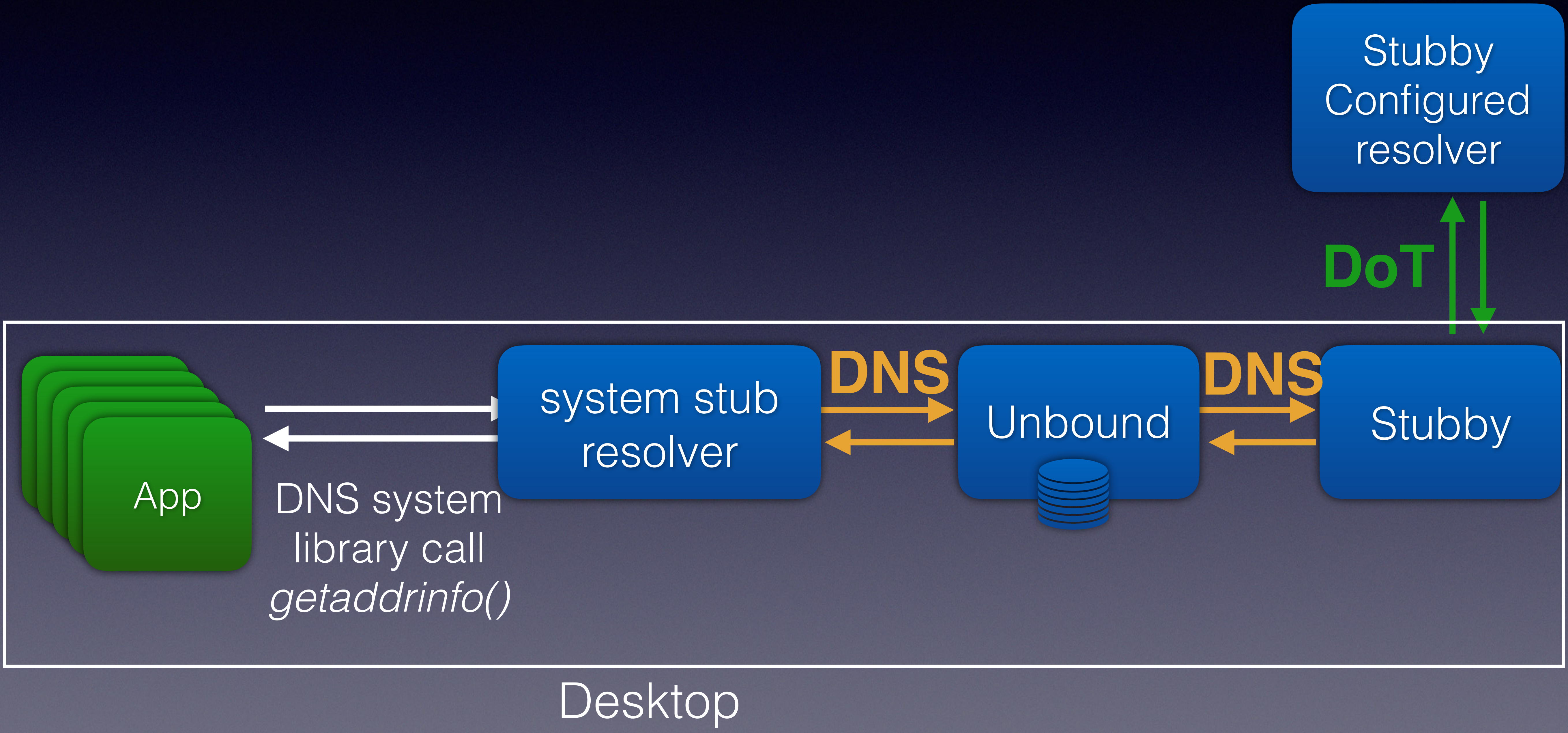
- Ideally - you could configure which networks you 'trust' and which you don't and use different configurations
- REALLY need a nice GUI with visual indicators to help users understand the state (GUI icon in menu: Bold=Strict)
- BUT - think about the usability here. 'Usable Security'
    - Green lock in HTTPS is per website (still confusing)
    - DNS is per network… and users don't understand DNS!

Unbound as a DoT stub

Unbound as a DoT stub

- Actually a recursive resolver but can be configured to cache locally and just forward all queries to another recursive resolver.
- Download from: https://nlnetlabs.nl/projects/unbound/download/
  - Packages, homebrew, Windows installers
- Example config here: Unbound config and in git repo
  - Must specify path to CA bundle to do authentication (Windows requires `tls-win-cert:` instead). May need to change user…
  - Can set up stub-zones for local queries
- To run in foreground: `sudo unbound -c <conf_file> -d <-vvvv>`
  - Unbound uses a new TCP connection for _every_ query (inefficient)

Bonus points: Unbound+Stubby


Stubby Configured resolver
DoT
App
DNS system library call *getaddrinfo()*
system stub resolver
DNS
Unbound
DNS
Stubby
Desktop

Running a DoT recursive resolver

Running a DoT Recursive

- Overview is here: [Running a DNS Privacy Resolver]
- Several open source DNS implementations do DoT natively (some do DoH too)
- Big differences to ‘normal DNS resolver’
  - Need to have a valid certificate for name (LE is good option)
  - Need a bit more configuration
  - Need to think about data handling (Best practices)

Unbound as a recursive DoT resolver

- Download from: https://nlnetlabs.nl/projects/unbound/download/
  - Packages, homebrew, Windows installers
- Full example config is here: Unbound server config
  - Next slide (and git repo) has suggested config for this lab
- Need to create a self-signed certificate using openssl & update paths in config file (need openssl 1.1.1):

```
>openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 7
>openssl rsa -in key.pem -out key_rsa.pem
```

- Sample cert and key files are in the git repo

unbound_rec.conf

```
server:
  directory: "/etc/unbound"
  username: unbound
  chroot: "/etc/unbound"
  logfile: "" # logging will be to stdout.
  pidfile: "/etc/unbound/unbound.pid"
  # verbosity: 1 # uncomment and increase to get more logging.
  # listen on localhost on port 853, answer queries from the local subnet.
  interface: 127.0.0.1@853
  interface: 0::1@853

  tls-service-key: “<path>/key_rsa.pem”
  tls-service-pem: “<path>/cert.pem"
  tls-port: 853
  incoming-num-tcp: 100 # Number of simultaneous incoming TCP connections

  # Listen on UDP but still issues queries upstream over UDP.
  # Only available in 1.6.7 and later
  udp-upstream-without-downstream: yes
  qname-minimisation: yes # Enable QNAME minimisation

  # Force a weak cipher suite to allow decryption
  # NEVER USE IN PRODUCTION!!!
  tls-ciphers: "RSA"
```

Unbound as a recursive DoT resolver

- To run in foreground: `sudo unbound -c <conf_file> -d  <-v>`
- Now lets point Stubby at this recursive resolver

```
tls_authentication: GETDNS_AUTHENTICATION_NONE
upstream_recursive_servers:
 - address_data: 127.0.0.1
```

- Or use ‘kdig’ or ‘getdns_query’ to do individual queries
- Look in Wireshark again on the ‘loopback’ interface with capture filter 'port 853’ to see the traffic from Stubby to Unbound
- Look on port 53 to see the traffic from Unbound out the authoritative servers

Decrypt local DoT traffic

- In Wireshark go to Preferences
- Expand ‘Protocols’ and select the word ‘Protocol’
- Start typing ‘TLS’ - this will jump you to the TLS settings
- Click on the Edit button next ‘RSA keys list’
- Add an entry for 127.0.0.1, 853, tls, <path the ‘key_rsa.pem’ file>
- Then hit OK twice to save the settings

Decrypt local DoT traffic

- Now look at the TLS -> Server Hello in Wireshark and you will see a Cipher containing RSA
- Select a packet now marked as ‘Unknown Ignored Packet’ - Wireshark doesn’t support DoT directly
- Click on the ‘Encrypted Application Data’, then at the very bottom select the pane marked ‘Decrypted TLS bytes’ - you should see what looks like a domain name!

DoH on the Desktop

DoH for Desktop

- Clients:
  - Cloudflare have release two tools to provide DOH clients, see Cloudflared
  - Frank Denis has a dnscrypt-proxy (client proxy) that supports DoH.
- ‘dig’ like tool for DoH:
  - Curl also supports DoH https://github.com/curl/doh

```
./doh www.example.com https://cloudflare-dns.com/dns-query
```

DoH in Browsers

Browsers

- Desktop:
  - Firefox (DoH)
  - [Chrome (DoH)]
  - Yandex (DNSCrypt)
- Mobile browser:
  - Bromite - based on Chrome (DoH)
  - Tenta for Android (DoT)

Browsers

- Desktop:
  - Firefox (DoH)
  - [Chrome (DoH)]
  - Yandex (DNSCrypt)

- Mobile browser:
  - Bromite - based on Chrome (DoH)
  - Tenta for Android (DoT)

Firefox

- Download the latest Firefox Nightly (or Firefox)

Need Latest Firefox Nightly to decrypt DoH

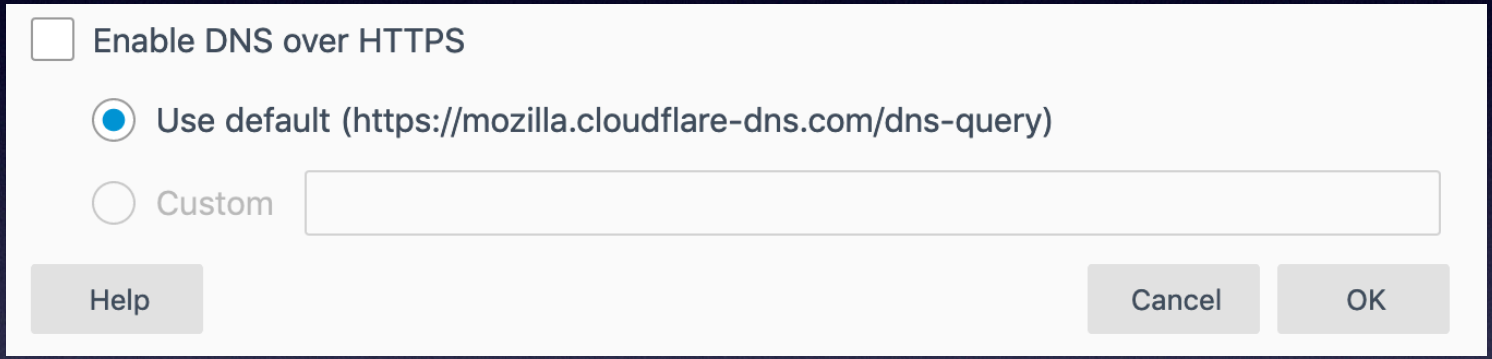
- Close your other apps.

- See DNS activity via `about:networking` tab
    - Select 'DNS Lookup' to do individual queries (use refresh)
    - Select 'DNS' to see queries (Note TRR=False)

- 2 levels of config
    - 'Easy' via a Preferences GUI option
    - Low-level via `about:config`

Firefox - DoH in the GUI

- Firefox->Preferences. Scroll to bottom 'Network Settings' click on 'Settings'
- Scroll the bottom to find the DoH checkbox



☐ Enable DNS over HTTPS

[✓ Use default (https://mozilla.cloudflare-dns.com/dns-query)]

Custom

- Check the box and open a tab, look again at about:networking
- Look in Wireshark again… Nothing on port 53 or 853.
  Look on port 443 - can you tell the DNS from the HTTP?
  Note: all DoH goes to Cloudflare, use a filter expression to see only DoH:

```
host cloudflare-dns.com
```

```
ip.addr == 108.61.201.119 or ip.addr == 104.16.248.249 or ipv6.addr ==
2606:4700::6810:f9f9 or ipv6 == 2606:4700::6810:f8f9
```

Firefox - DoH in the GUI

- Easy right? And you don't need to be an Admin on the machine…
  - And remember - in the next release this may be on by default!
  - Too easy? Use it without knowing (back to Informed Consent)
- Extreme scenarios
  - Great to avoid DNS snooping and interference in untrusted network
  - But so easy a child in a house where parental controls are enabled can do it with no-one knowing
- But all your other DNS queries are still clear text….

Firefox - DoH via config

- Use the about:config tab and in the search box type ‘trr’
- Will see a variety of settings…
    - ‘trr.mode’ = 2. This will fall back to cleartext DNS via the system resolver if it can’t talk to the DoH resolver (like Op DoT)
    - ‘trr.mode’ = 3 fails instead (like Strict DoT)
- Go back to your DoH setting and ‘mis-type’ the URI and apply
    - Now all the traffic has TRR=false
    - Did you get a warning?

GUIs for DNS settings are hard….

Decrypt DoH traffic

- Great SharkFest presentation on this
- For long captures start Wireshark with capture filter
  `` `port 443 and host cloudflare-dns.com` ``
- Close then re-start Firefox Nightly set up to export session keys:
  - Linux/macOS - close Firefox and relaunch from command line:
    ```
    SSLKEYLOGFILE="$PWD/keys.txt" <path>/firefox -no-remote -profile /tmp/ff
    ```
  - Firefox on Windows, create start-fx.cmd file, without quotes in the set line:
    ```
    set SSLKEYLOGFILE=C:\Users\User\Desktop\keys.txt
    start firefox
    ```

Decrypt DoH traffic

- In Wireshark go to Preferences
- Expand ‘Protocols’ and select the word ‘Protocol’
- Start typing ‘TLS’ - this will jump you to the TLS settings
- (Pre)-Master-Secret log filename - click browse and select the file exported by Firefox. Click OK
- Hey presto! Wireshark decodes packets as DoH.

DoH

- HTTP request to https://dnsserver.example.net/dns-query
- Query can use POST or GET:

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query?
dns=AAABAAABAAAAAAAAA3d3dwdleGFtcGxlA2NvbQAAAQAB
accept = application/dns-message
```

Query

```
:status = 200
content-type = application/dns-messag
content-length = 61
cache-control = max-age=3709
<61 bytes represented by the following hex encoding>
00 00 81 80 00 01 00 01  00 00 00 00 03 77 77 77
```

Response

DoT/DoH for your mobile phone

Mobile DNS encryption

Mobile OS	Options
Android	**Android supports DNS-over-TLS in Android Pie** Opportunistic by default to system resolver, also user override. Talk by the Android developers: Video, Slides
	• App called 'Intra' which can be used to send all queries from the device over DoH to a user configured resolver • Cloudflare has an App: 1.1.1.1 • Quad 9 has an App: Quad9 Connect • Other apps are available….
iOS	• Cloudflare has an App: 1.1.1.1 • Other apps are available….
	Work started on a Stubby app but stalled

Mobile traffic inspection is not straightforward…

Mobile DNS encryption

Mobile OS	Options
Android	**Android supports DNS-over-TLS in Android Pie** Opportunistic by default to system resolver, also user override. Talk by the Android developers: Video, Slides
	• App called 'Intra' which can be used to send all queries from the device over DoH to a user configured resolver • Cloudflare has an App: 1.1.1.1 • Quad 9 has an App: Quad9 Connect • Other apps are available….
iOS	• Cloudflare has an App: 1.1.1.1 • Other apps are available….
	Work started on a Stubby app but stalled

Both do DoT and DoH but only to one resolver

Mobile traffic inspection is not straightforward…

DNS Privacy Libraries

DNS Privacy libraries

Language	Library
C (Nodejs, python)	getdns
Go	GoDNS
Rust	trust-dns

getdns

- getdns Modern, asynchronous DNS library with DNSSEC and DoT
  - Specifically designed to be used by developers
  - Implements new DNS features quickly (experimental)
  - Significantly more useful than libc DNS functions
- Written in C but has Python and nodejs bindings
  - Quick start guide to C library
  - Deeper tutorial: Slides, Video
- Comes with a 'dig' like tool: getdns_query

```
getdns_query @8.8.8.8~dns.google www.example.com -Lm
+return_call_reporting
```

Routers

- DNS-over-TLS forwarding on a Turris router
- OpenWRT (LEDE)
- Asuswrt-Merlin