To actually visit a website, you must know the IP address of the server that the website lives on, the web browser doesn’t know the IP address, and humans don’t memorise IP addresses for the websites they want to visit.

Since we don’t know the IP address that the website lives on, that’s where DNS comes in. DNS servers map domain names with their respective IP addresses. So, when you enter a website URL, your web browser says to the DNS server ‘hey what’s the IP address of this URL?’ Then the DNS server will respond with the IP address then you can visit the website. Without DNS, everything on the internet would break.

When you type a website (academy.site.com) in the browser, your computer firstly use a stub resolver which basically checks your cache to see if you have visited that website before because if it has then the IP address would be there. Lets say it was not in the cache, your computer has a DNS server, googles DNS server (8.8.8.8) is the most common DNS server. The stub resolver will send a DNS query saying, ‘you know the IP address of academy .site.com right?’. Googles DNS server would say ‘no’ because googles DNS server is recursive which means it may not know every IP address of every website, but it knows other DNS servers that do know the IP address of academy.site.com. If googles DNS server has cache, you can access the website but let’s say it didn’t. So, since googles DNS does not know the IP address for academy.site.com, googles DNS server will reach out to one of the 13 root servers.

A screenshot of a computer

AI-generated content may be incorrect.

These root servers are like the high tier boss DNS servers, there are 13 of them. So googles DNS will reach out to one of these root servers. But these root servers don’t know exactly where the IP address of academy.site.com. The root servers only know about the top-level domains (TLD).

When google talks to one of these root servers it is only asking for the TLDs, the TLD is like the .com part or the .net. These root servers only handle the TLDs meaning that they maintain a list of other DNS servers that can help you with the TLDs. So, our google DNS server is only asking about the .com in academy.site.com. Then the root server will return a list of other DNS servers that are responsible for the .com domain. And those DNS servers that are responsible for the .com domain or any TLD domain, they are also the boss of their respective TLD management.

So, the root server will respond to googles DNS server and say hey go to a.gtld-servers.net, he knows about .com. So then googles DNS server will send another query to the .com TLD server, but the a.gtld-servers.net server (DNS server that manages .com stuff) does not know the IP address of academy.site.com. Googles DNS server is only asking the .com server regarding the ‘site’ part in academy.site.com. The ‘site’ part is called the second level domain (SLD). Googles DNS is asking the .com server ‘who manages site.com?’. That’s what TLD servers do, they keep a list of authoritative servers for website domains. Then the .com server will respond with an authoritative server(s) which we shall call Pablo for site.com (the SLD). Then googles DNS will send a query to Pablo asking if it knows the IP address for academy.site.com. And Pablo knows it. Pablo knows everything about the site.com, then Pablo will respond with the IP address of the domain academy.site.com. Then googles DNS will save it in its cache and then google DNS will tell our computers stub resolver the IP address for academy.site.com. Also, the ‘academy’ part in academy.site.com is referred to as the sub-domain.

Also, whenever our stub resolver queries a DNS server, it will by default do it by UDP Port 53, this is done by plain text meaning it is not encrypted. This means that a hacker can intercept and look at it because it is plain text. The hacker is able to see all the websites I am visiting.

To make DNS secure, there is DoH which is DNS over HTTPS. We know that HTTPS is the protocol we use to access websites securely. So when DNS goes over HTTPS, it also becomes a secure connection such as encryption. There are other secure ways to use DNS such as DoT, DNSCrypt, DNSSEC, Quad9.

DNS is more than just domain to IP address mapping.

A record (address record) – maps a domain name to an IPv4 address:

* example.com. 3600 IN A 93.184.216.34
* when someone visits example.com, DNS tells them to go to IP 93.184.216.34
* every website needs at least one A record
* TTL (3600) is time-to-live in seconds (1 hour)

AAAA record (IPv6 address record) – maps a domain name to an IPv6 address:

* example.com. 3600 IN AAAA 2606:2800:220:1:248:1893:25c8:1946
* same as A record but for IPv6
* you can have both A and AAAA for dual-stack websites.

NEXT PAGE

CNAME (canonical name record) – makes one domain an alias of another

* www.example.com. 3600 IN CNAME example.com.
* when a user visits [www.example.com](http://www.example.com), DNS says ‘actually use example.com instead’
* you cannot have a CNAME and any other record (A, MX, etc) on the same name
* great for pointing subdomains to another domain.

MX – (mail exchange record) – specified which server handles email for the domain

* example.com. 3600 IN MX 10 mail1.example.com.
* example.com. 3600 IN MX 20 mail2.example.com.
* email sent to @example.com goes first to mail1, then mail2 if mail1 fails
* lower number = higher priority

NS (name server record) – specifies which DNS servers are authoritative for the domain

* example.com. 86400 IN NS ns1.example.net.
* example.com. 86400 IN NS ns2.example.net.
* these servers hold the real DNS zone for the domain
* usually managed by your registrar or DNS host (cloudfare, route53, etc)

SOA (start of authority record) – stores administrative information about the zone

example.com. IN SOA ns1.example.net. admin.example.com. (

2025101201 ; serial

7200 ; refresh (2 hours)

3600 ; retry (1 hour)

1209600 ; expire (14 days)

86400 ; minimum TTL (1 day)

)

* primary name server: ns1.example.net
* admin email: [admin@example.com](mailto:admin@example.com) (note the dot instead of @)
* serial: zone version number (increment when you make changes)

TXT (text record) – stores arbitrary text, often for verification, SPF, DKIM, or security policies

* example SPF record: example.com. 3600 IN TXT "v=spf1 include:\_spf.google.com ~all"
* example domain verification: example.com. 3600 IN TXT "google-site-verification=xyz123"
* you can have multiple TXT records
* used heavily for email authentication

PTR (pointer record) – reverse DNS, maps an IP address to domain name

* 150.89.253.206.in-addr.arpa. 3600 IN PTR slayingthedragon.io.
* When you look up the IP 206.253.89.150, DNS returns the domain name
* Used mainly by mail servers and logs
* Set by whoever owns the IP block (your hosting provider)

SRV (service record) – specifies location (hostname + port) of specific services (XMPP, Minecraft)

* \_service.\_protocol.name. TTL IN SRV priority weight port target.
* \_sip.\_tcp.example.com. 3600 IN SRV 10 60 5060 sipserver.example.com.
* For SIP over TCP, use sipserver.example.com on port 5060
* Priority: lower means try first
* Weight: load balancing between same-priority servers

CAA (certification authority authorisation) – specified which certificate authorities can issue SSL certificates for your domain

* example.com. 3600 IN CAA 0 issue "letsencrypt.org"
* only lets encrypt is allowed to issue SSL certificates for this domain

DNSSEC records (advanced security) – used when DNSEC (DNS Security Extensions) is enabled.

* DNSKEY: public key for the zone
* RRSIG: digital signature for the other records
* DS: delegation signer
* NSEC/NSEC3: prove nonexistence of records securely
* Those all protect against DNS spoofing.