**DNS**

DNS (Domain Name System) is like an address book for the internet. It helps computers find the right addresses (IP addresses) for websites, email servers, and other online services.

There are different types of DNS records, like different sections in the address book, each serving a specific purpose:

**A Record**

* This record maps a website's domain name (like [www.example.com](http://www.example.com)) to its actual IP address (like 192.168.1.100) for regular internet connections (IPv4).

**AAAA Record**

* Similar to the A record, but for the newer type of internet addresses called IPv6 (like 2001:db8:3333:4444:5555:6666:7777:8888).

**CNAME Record**

* This record is like a nickname. It allows one domain name (like store.example.com) to point to another domain name (like example.shopify.com).

**MX Record**

* This record tells your computer where to send emails for a particular domain (like example.com). It lists the email server addresses and their priorities.

**TXT Record**

* This record can hold any text-based information. It's often used to verify domain ownership or list authorized email servers to prevent spam.

So, DNS records act like different sections in an address book, helping computers find the right addresses for various online services by providing the necessary information in different formats.

**TLD (Top-Level Domain)**

* Most righthand part of a domain name (e.g., .com in tryhackme.com)
* Two types: gTLD (Generic Top Level) and ccTLD (Country Code Top Level Domain)
* gTLD historically indicated purpose (e.g., .com for commercial, .org for organizations)
* ccTLD indicated geography (e.g., .ca for Canada, .co.uk for United Kingdom)
* Over 2000 new gTLDs available (e.g., .online, .club, .website, .biz)

**Second-Level Domain**

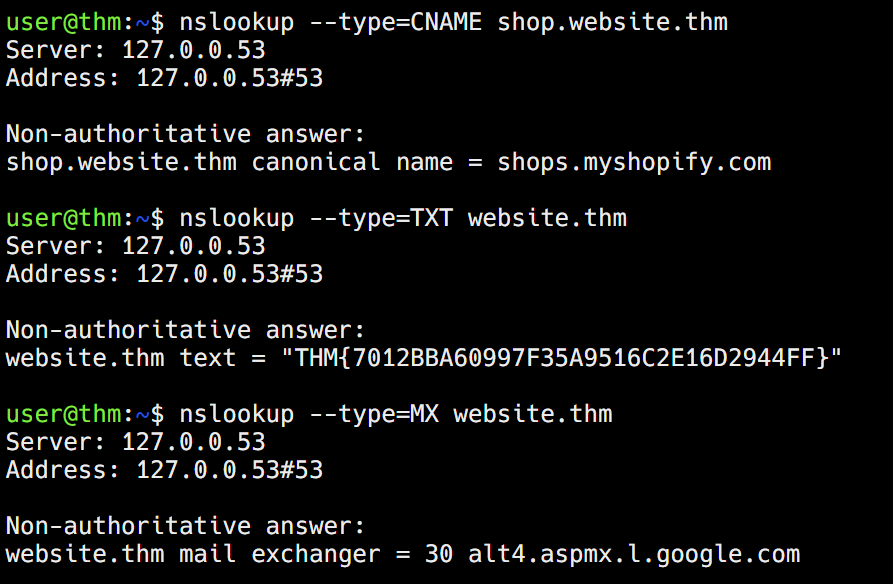
* In tryhackme.com, "tryhackme" is the Second-Level Domain
* Limited to 63 characters + TLD
* Can only use a-z, 0-9, and hyphens (no consecutive hyphens or starting/ending with hyphens)

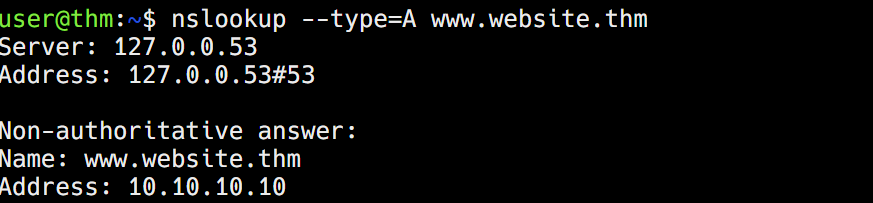
**Subdomain**

* Sits on the left-hand side of the Second-Level Domain (e.g., admin in admin.tryhackme.com)
* Same creation restrictions as Second-Level Domain (63 characters, a-z, 0-9, hyphens)
* Can use multiple subdomains separated by periods (e.g., jupiter.servers.tryhackme.com)
* Length must be kept to 253 characters or less
* No limit to the number of subdomains

**What Happens When You Make a DNS Request**

1. Client checks local cache for requested domain
   * If found, uses cached IP address
   * If not found, sends request to recursive DNS server (usually ISP's DNS server)
2. Recursive DNS server checks its own cache
   * If found, returns IP address to client
   * If not found, queries root DNS servers
3. Root servers provide address of Top Level Domain (TLD) nameservers for requested domain's TLD (.com, .org, etc.)
4. Recursive server queries TLD nameservers for address of authoritative nameservers for requested domain
5. Authoritative nameservers provide requested domain's IP address to recursive server
6. Recursive server caches response and returns IP address to client
7. Client uses IP address to connect to requested domain

* DNS responses cached based on Time to Live (TTL) value to minimize future lookups
* Multiple redundant authoritative nameservers used for reliability



# **HTTP (HyperText Transfer Protocol)**

* Set of rules for communicating with web servers
* Transmits webpage data (HTML, images, videos, etc.)
* Developed by Tim Berners-Lee (1989-1991)

# **HTTPS (HyperText Transfer Protocol Secure)**

* Secure version of HTTP
* Data is encrypted
* Prevents eavesdropping and provides server authentication

# **URLs (Uniform Resource Locators)**

* Provide instructions on how to access resources on the internet
* Components of a URL:
  + **Scheme**: This instructs on what protocol to use for accessing the resource such as HTTP, HTTPS, FTP (File Transfer Protocol).
  + **User**: Some services require authentication to log in, you can put a username and password into the URL to log in.
  + **Host**: The domain name or IP address of the server you wish to access.
  + **Port**: The Port that you are going to connect to, usually 80 for HTTP and 443 for HTTPS, but this can be hosted on any port between 1 - 65535.
  + **Path**: The file name or location of the resource you are trying to access.
  + **Query String:** Extra bits of information that can be sent to the requested path. For example, /blog?id=1 would tell the blog path that you wish to receive the blog article with the id of 1.
  + **Fragment:** This is a reference to a location on the actual page requested. This is commonly used for pages with long content and can have a certain part of the page directly linked to it, so it is viewable to the user as soon as they access the page.

**Example Request:**

****

To breakdown each line of this request:

**Line 1:** This request is sending the GET method ( more on this in the HTTP Methods task ), request the home page with / and telling the web server we are using HTTP protocol version 1.1.

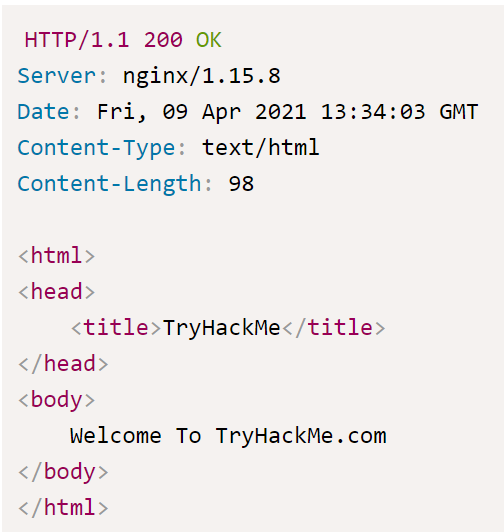
**Line 2:** We tell the web server we want the website tryhackme.com

**Line 3:** We tell the web server we are using the Firefox version 87 Browser

**Line 4:** We are telling the web server that the web page that referred us to this one is https://tryhackme.com

**Line 5:** HTTP requests always end with a blank line to inform the web server that the request has finished.

**Example Response:**

****

**Line 1:** HTTP 1.1 is the version of the HTTP protocol the server is using and then followed by the HTTP Status Code in this case "200 Ok" which tells us the request has completed successfully.

**Line 2:** This tells us the web server software and version number.

**Line 3:** The current date, time and timezone of the web server.

**Line 4:** The Content-Type header tells the client what sort of information is going to be sent, such as HTML, images, videos, pdf, XML.

**Line 5:** Content-Length tells the client how long the response is, this way we can confirm no data is missing.

**Line 6:** HTTP response contains a blank line to confirm the end of the HTTP response.

**Lines 7-14:** The information that has been requested, in this instance the homepage.

**GET Request-** This is used for getting information from a web server

**POST Request-** This is used for submitting data to the web server and potentially creating new records

**PUT Request-** This is used for submitting data to a web server to update information

**DELETE Request-** This is used for deleting information/records from a web server.

Web Servers

* Software that listens for incoming connections and delivers web content using HTTP
* Common examples: Apache, Nginx, IIS, NodeJS
* Serves files from root directory (e.g. /var/www/html on Linux, C:\inetpub\wwwroot on Windows)

Virtual Hosts

* Allow web servers to host multiple websites with different domain names
* Check hostname in HTTP headers and match to virtual host configuration files
* Can map different domain names to different root directories

Static vs Dynamic Content

* Static: Never changes (images, JS, CSS, static HTML)
* Dynamic: Changes with different requests (blog posts, search results)
* Dynamic content generated by backend using programming/scripting languages

Backend Languages

* PHP, Python, Ruby, NodeJS, Perl, etc.
* Interact with databases, call external services, process user data
* Example: PHP code <?php echo $\_GET["name"]; ?> to display value from URL parameter

Frontend vs Backend

* Frontend: What user sees in browser
* Backend: Processing happening behind the scenes, not visible in HTML source