DNS

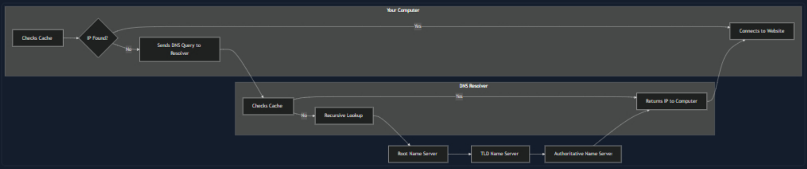
The `Domain Name System` (`DNS`) acts as the internet's GPS, guiding your online journey from memorable landmarks (domain names) to precise numerical coordinates (IP addresses). Much like how GPS translates a destination name into latitude and longitude for navigation, DNS translates human-readable domain names (like `www.example.com`) into the numerical IP addresses (like `192.0.2.1`) that computers use to communicate.

Imagine navigating a city by memorizing the exact latitude and longitude of every location you want to visit. It would be incredibly cumbersome and inefficient. DNS eliminates this complexity by allowing us to use easy-to-remember domain names instead. When you type a domain name into your browser, DNS acts as your navigator, swiftly finding the corresponding IP address and directing your request to the correct destination on the internet.

Without DNS, navigating the online world would be akin to driving without a map or GPS – a frustrating and error-prone endeavour.

How DNS Works

Imagine you want to visit a website like `www.example.com`. You type this friendly domain name into your browser, but your computer doesn't understand words – it speaks the language of numbers, specifically IP addresses. So, how does your computer find the website's IP address? Enter DNS, the internet's trusty translator.



1. `Your Computer Asks for Directions (DNS Query)`: When you enter the domain name, your computer first checks its memory (cache) to see if it remembers the IP address from a previous visit. If not, it reaches out to a DNS resolver, usually provided by your internet service provider (ISP).
2. `The DNS Resolver Checks its Map (Recursive Lookup)`: The resolver also has a cache, and if it doesn't find the IP address there, it starts a journey through the DNS hierarchy. It begins by asking a root name server, which is like the librarian of the internet.
3. `Root Name Server Points the Way`: The root server doesn't know the exact address but knows who does – the Top-Level Domain (TLD) name server responsible for the domain's ending (e.g., .com, .org). It points the resolver in the right direction.
4. `TLD Name Server Narrows It Down`: The TLD name server is like a regional map. It knows which authoritative name server is responsible for the specific domain you're looking for (e.g., `example.com`) and sends the resolver there.
5. `Authoritative Name Server Delivers the Address`: The authoritative name server is the final stop. It's like the street address of the website you want. It holds the correct IP address and sends it back to the resolver.
6. `The DNS Resolver Returns the Information`: The resolver receives the IP address and gives it to your computer. It also remembers it for a while (caches it), in case you want to revisit the website soon.
7. `Your Computer Connects`: Now that your computer knows the IP address, it can connect directly to the web server hosting the website, and you can start browsing.

The Hosts File

The `hosts` file is a simple text file used to map hostnames to IP addresses, providing a manual method of domain name resolution that bypasses the DNS process. While DNS automates the translation of domain names to IP addresses, the `hosts` file allows for direct, local overrides. This can be particularly useful for development, troubleshooting, or blocking websites.

The `hosts` file is located in `C:\Windows\System32\drivers\etc\hosts` on Windows and in `/etc/hosts` on Linux and MacOS. Each line in the file follows the format:

Code: txt

```
<IP Address>    <Hostname> [<Alias> ...]
```

For example:

Code: txt

```
127.0.0.1       localhost
192.168.1.10    devserver.local
```

To edit the `hosts` file, open it with a text editor using administrative/root privileges. Add new entries as needed, and then save the file. The changes take effect immediately without requiring a system restart.

Common uses include redirecting a domain to a local server for development:

Code: txt

```
127.0.0.1       myapp.local
```

testing connectivity by specifying an IP address:

Code: txt

Cheat Sheet

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

```
192.168.1.20    testserver.local
```

or blocking unwanted websites by redirecting their domains to a non-existent IP address:

Code: txt

```
0.0.0.0       unwanted-site.com
```

It's Like a Relay Race

Think of the DNS process as a relay race. Your computer starts with the domain name and passes it along to the resolver. The resolver then passes the request to the root server, the TLD server, and finally, the authoritative server, each one getting closer to the destination. Once the IP address is found, it's relayed back down the chain to your computer, allowing you to access the website.

Key DNS Concepts

In the `Domain Name System` (`DNS`), a `zone` is a distinct part of the domain namespace that a specific entity or administrator manages. Think of it as a virtual container for a set of domain names. For example, `example.com` and all its subdomains (like `mail.example.com` or `blog.example.com`) would typically belong to the same DNS zone.

The zone file, a text file residing on a DNS server, defines the resource records (discussed below) within this zone, providing crucial information for translating domain names into IP addresses.

To illustrate, here's a simplified example of what a zone file, for `example.com` might look like:

Code: zone

```
$TTL 3600 ; Default Time-To-Live (1 hour)
@       IN SOA   ns1.example.com. admin.example.com. (
                2024060401 ; Serial number (YYYYMMDDNN)
                3600       ; Refresh interval
                900        ; Retry interval
                604800     ; Expire time
                86400 )    ; Minimum TTL

@       IN NS    ns1.example.com.
@       IN NS    ns2.example.com.
@       IN MX 10 mail.example.com.
www     IN A     192.0.2.1
mail    IN A     198.51.100.1
ftp     IN CNAME www.example.com.
```

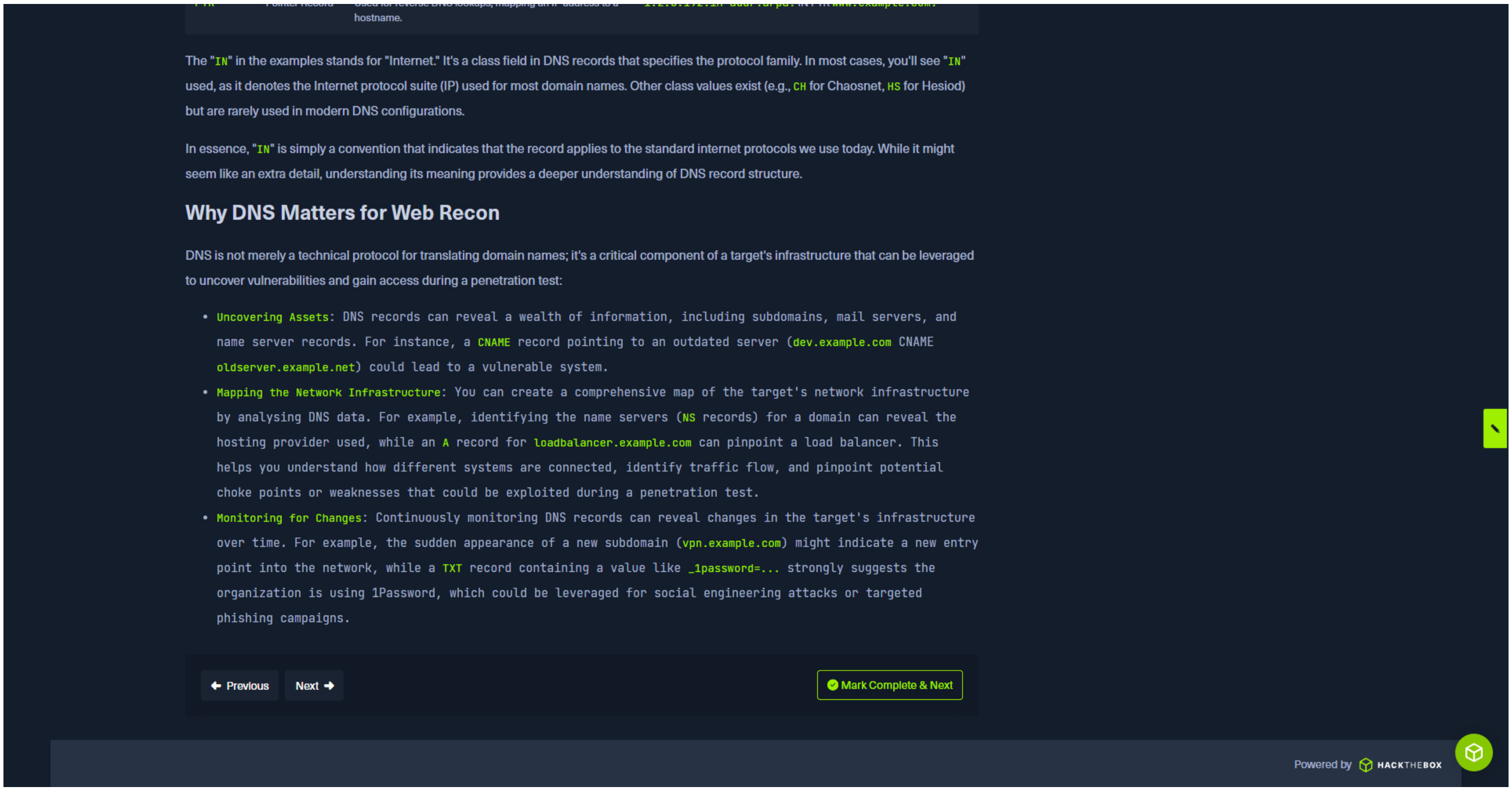
This file defines the authoritative name servers (`NS` records), mail server (`MX` record), and IP addresses (`A` records) for various hosts within the `example.com` domain.

DNS servers store various resource records, each serving a specific purpose in the domain name resolution process. Let's explore some of the most common DNS concepts:

DNS Concept	Description	Example
`Domain Name`	A human-readable label for a website or other internet resource.	`www.example.com`
`IP Address`	A unique numerical identifier assigned to each device connected to the internet.	`192.0.2.1`
`DNS Resolver`	A server that translates domain names into IP addresses.	Your ISP's DNS server or public resolvers like Google DNS (`8.8.8.8`)
`Root Name Server`	The top-level servers in the DNS hierarchy.	There are 13 root servers worldwide, named A-M: `a.root-servers.net`
`TLD Name Server`	Servers responsible for specific top-level domains (e.g., .com, .org).	Verisign for `.com`, PIR for `.org`
`Authoritative Name Server`	The server that holds the actual IP address for a domain.	Often managed by hosting providers or domain registrars.
`DNS Record Types`	Different types of information stored in DNS.	A, AAAA, CNAME, MX, NS, TXT, etc.

Now that we've explored the fundamental concepts of DNS, let's dive deeper into the building blocks of DNS information – the various record types. These records store different types of data associated with domain names, each serving a specific purpose:

Record Type	Full Name	Description	Zone File Example
`A`	Address Record	Maps a hostname to its IPv4 address.	`www.example.com.` IN A `192.0.2.1`
`AAAA`	IPv6 Address Record	Maps a hostname to its IPv6 address.	`www.example.com.` IN AAAA `2001:db8:85a3::8a2e:370:7334`
`CNAME`	Canonical Name Record	Creates an alias for a hostname, pointing it to another hostname.	`blog.example.com.` IN CNAME `webserver.example.net.`
`MX`	Mail Exchange Record	Specifies the mail server(s) responsible for handling email for the domain.	`example.com.` IN MX 10 `mail.example.com.`
`NS`	Name Server Record	Delegates a DNS zone to a specific authoritative name server.	`example.com.` IN NS `ns1.example.com.`
`TXT`	Text Record	Stores arbitrary text information, often used for domain verification or security policies.	`example.com.` IN TXT `"v=spf1 mx -all"` (SPF record)
`SOA`	Start of Authority Record	Specifies administrative information about a DNS zone, including the primary name server, responsible person's email, and other parameters.	`example.com.` IN SOA `ns1.example.com. admin.example.com. 2024060301 10800 3600 604800 86400`
`SRV`	Service Record	Defines the hostname and port number for specific services.	`_sip._udp.example.com.` IN SRV 10 5 5060 `sipserver.example.com.`
`PTR`	Pointer Record	Used for reverse DNS lookups, mapping an IP address to a	`1.2.0.192.in-addr.arpa.` IN PTR `www.example.com`



`PTR`	Pointer Record	Used for reverse DNS lookups, mapping an IP address to a hostname.	`1.2.0.192.in-addr.arpa. IN PTR www.example.com.`

The "`IN`" in the examples stands for "Internet." It's a class field in DNS records that specifies the protocol family. In most cases, you'll see "`IN`" used, as it denotes the Internet protocol suite (IP) used for most domain names. Other class values exist (e.g., `CH` for Chaosnet, `HS` for Hesiod) but are rarely used in modern DNS configurations.

In essence, "`IN`" is simply a convention that indicates that the record applies to the standard internet protocols we use today. While it might seem like an extra detail, understanding its meaning provides a deeper understanding of DNS record structure.

Why DNS Matters for Web Recon

DNS is not merely a technical protocol for translating domain names; it's a critical component of a target's infrastructure that can be leveraged to uncover vulnerabilities and gain access during a penetration test:

- `Uncovering Assets`: DNS records can reveal a wealth of information, including subdomains, mail servers, and name server records. For instance, a `CNAME` record pointing to an outdated server (`dev.example.com` CNAME `oldserver.example.net`) could lead to a vulnerable system.
- `Mapping the Network Infrastructure`: You can create a comprehensive map of the target's network infrastructure by analysing DNS data. For example, identifying the name servers (`NS` records) for a domain can reveal the hosting provider used, while an `A` record for `loadbalancer.example.com` can pinpoint a load balancer. This helps you understand how different systems are connected, identify traffic flow, and pinpoint potential choke points or weaknesses that could be exploited during a penetration test.
- `Monitoring for Changes`: Continuously monitoring DNS records can reveal changes in the target's infrastructure over time. For example, the sudden appearance of a new subdomain (`vpn.example.com`) might indicate a new entry point into the network, while a `TXT` record containing a value like `_1password=...` strongly suggests the organization is using 1Password, which could be leveraged for social engineering attacks or targeted phishing campaigns.