Welcome to Homelab Club!

- Please sign in at the clip board
  - Attendance is used for the following purposes:
    - Please include your **UMD username / directory ID** if you wish to be eligible to vote in the officer elections near the end of the semester
    - Please include your **Discord username** if you wish to be given the @Member role for meeting and other member-relevant updates
    - If you do not wish to tie your real name/UMD directory ID to your discord username, feel free to include them as separate line items

Introduction to Network Management & Security

•••

Homelab Club @ UMD Meeting (2026/02/16)

Intro to DNS

What does DNS mean? → **(D)omain (N)ame (S)ystem**

What does it do? → **Translates for the Internet!**

Why do we DNS? → **Too many IPs! Its faster! It provides abstraction!**

Breaking Down URLs

URL: https://suddenlysixam.club/projects/dns.html

https:// - protocol

suddenlysixam.club - domain

.club - top level domain

/projects/dns.html - path

DNS Queries

Ask a resolver for some information

Forward Lookup:

- Name -> IP (e.g. umiacs.umd.edu -> 128.8.120.33)

Reverse Lookup:

- IP -> Name (e.g. 128.8.120.33 -> umiacs.umd.edu)

Types of DNS servers

Recursive resolvers → server that accepts user queries and makes additional requests

Root name server → top of hierarchy, determines where to search

Top Level Domain server → determines where to search at the domain level (e.x. .com)

Authoritative name server → gives IP address for requested address

DNS Hierarchy

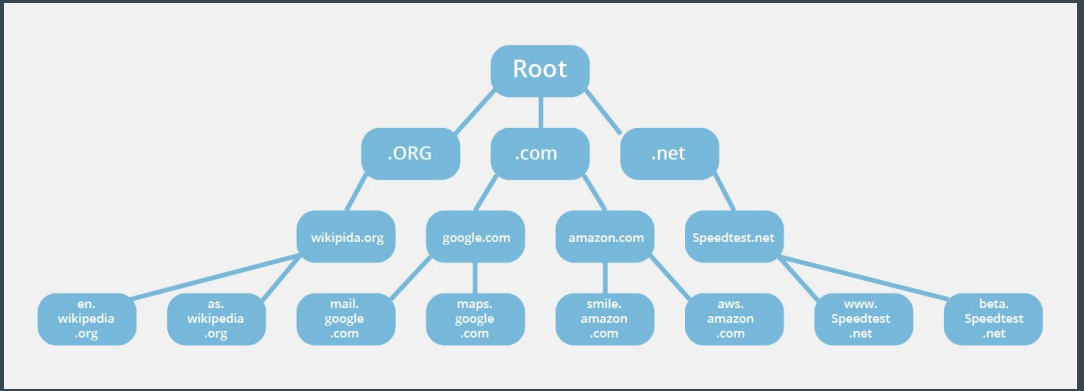
DNS is hierarchical, not a single giant database

Root servers (.)

Top Level Domain (TLD) servers (e.g. .com, .org, .edu, etc.)

Authoritative servers

Resolvers



from: https://www.cloudflare.com/learning/dns/glossary/dns-root-server/

What happens when you make a query?

1. First check DNS cache for IP info
2. Send out a request that is received by a resolver
3. The resolver queries a root name server
4. The root name server returns which TLD server to search
5. The resolver sends another request to the TLD server specified
6. TLD server responds with which authoritative name server to search
7. Resolver sends request to name server
8. Name server gives the IP info
9. DNS resolver sends IP info to the original request location (e.x. Web browser)
10. Then it can do whatever it wants like make a web page request

DNS Records

NAME	TTL	CLASS	TYPE	DATA
faculty	3600	IN	CNAME	umiacswww-vip.umiacs.umd.edu.

The name is the record you are looking up.

TTL (Time To Live) is how long this record may be cached for in seconds

Class is set to IN for internet queries

Type is the type of record

The data is type dependent but it's going to be the data you were requesting

Putting it all together, time for an example!

```
[bash-3.2$ dig @128.8.120.19 +trace suddenlysixam.club
; <<>> DiG 9.10.6 <<>> @128.8.120.19 +trace suddenlysixam.club
; (1 server found)
;; global options: +cmd
.                       92370   IN      NS      l.root-servers.net.
.                       92370   IN      NS      b.root-servers.net.
.                       92370   IN      NS      c.root-servers.net.
.                       92370   IN      NS      f.root-servers.net.
.                       92370   IN      NS      m.root-servers.net.
.                       92370   IN      NS      e.root-servers.net.
.                       92370   IN      NS      j.root-servers.net.
.                       92370   IN      NS      h.root-servers.net.
.                       92370   IN      NS      i.root-servers.net.
.                       92370   IN      NS      k.root-servers.net.
.                       92370   IN      NS      d.root-servers.net.
.                       92370   IN      NS      g.root-servers.net.
.                       92370   IN      NS      a.root-servers.net.
.                       265180  IN      RRSIG   NS 8 0 518400 20251003170000 20250920160000 46441 . NUN/F5FYGbdJnW2uBIvlW5VnNc360mA8oUewHxweM6WYUdi/bP3Utbnz BLA/Iycg2
zm8l1jU6qKneSRttSZe4aSLnhzHW9Gm dNrrfhOxPIPl9gKLRwRNb5XpLz/gahUSfLDio4fXZFwlPdi1qwpS+DtX kadWNFFRFj3Csa8idaY/RrCOCqI+rL+gWOrNwaUcCKCIZrvnrHX9uBr+ VqTDadb+lr4N7mwy7oBC
;; Received 1109 bytes from 128.8.120.19#53(128.8.120.19) in 0 ms

club.                   172800  IN      NS      ns3.dns.nic.club.
club.                   172800  IN      NS      ns1.dns.nic.club.
club.                   172800  IN      NS      b.nic.club.
club.                   172800  IN      NS      ns2.dns.nic.club.
club.                   172800  IN      NS      c.nic.club.
club.                   172800  IN      NS      a.nic.club.
club.                   86400   IN      DS      54682 8 2 4FC0DBB4F048E413BA1C0B1E92F4C5F0CCEBF7856370E20671AF6417 499DB258
club.                   86400   IN      RRSIG   DS 8 1 86400 20251006170000 20250923160000 46441 . J+DdVuTLptoggGLADNhQIyJecUcHXbPa6HgH7xAmJA0FFj1BlEJJcEDk rVc44BOsLc
ApdiEp/Ji9JaUY62n1eNE3AmCGdvrl JcypneaijbXhUeL/m4oPQfZNscuVIF1n+Hl0g8EBJlM/GI/9zpK+LDwD FOl/b/yS5Nfv1jloBxyy0U/6OiGrKjHBhrk7Pst9646HRC6gv9/FHFRG 5Vottt3TDQib+jTd0iiAC
;; Received 756 bytes from 198.41.0.4#53(a.root-servers.net) in 5 ms

suddenlysixam.club.     3600    IN      NS      derek.ns.cloudflare.com.
suddenlysixam.club.     3600    IN      NS      sima.ns.cloudflare.com.
p7ngjc7oaadjm89746jjos176dv5g8f9.club. 900 IN NSEC3 1 1 0 - P7NNM7LDMDSVGHPFLL7BLLI1SP6KVP0G  NS SOA RRSIG DNSKEY NSEC3PARAM CDS CDNSKEY TYPE65534
77i4h6d4308bs38cmtbvo6ccaqbnisjl.club. 900 IN NSEC3 1 1 0 - 77OGCUTB91SKR9ENJR7TFLD72L69EUOV  NS DS RRSIG
77i4h6d4308bs38cmtbvo6ccaqbnisjl.club. 900 IN RRSIG NSEC3 8 2 900 20251007031234 20250923021442 15345 club. 0CllzdFdpqeFN4uBevdm2J48Dv/+EhGQfFAyei7CvzA3AYOxoRuplVUW b
IGa3nHGRrNs/2PUjAjqs4kjuUuX5a9JWOukT9Ky Wg9dkT4kSUJconcC5yE5+q67UlI/ip4RVmSzHiI+7LI3jg==
p7ngjc7oaadjm89746jjos176dv5g8f9.club. 900 IN RRSIG NSEC3 8 2 900 20251005073304 20250921063755 15345 club. CNmd8r7+Ej7cfU/dIHQCkdIDJ1Wn1CwBW07JmO5KtZDj90Y9ODCbvfJo 2
5OqRhJd+e5tuoR+5yP4QBxDyLNAP9QUcOyRm5o/ G+jGywT8tvvtpp0slJz1c0esLe3gAGFi7iNFBqL0Bp7/Pw==
;; Received 689 bytes from 156.154.144.215#53(ns1.dns.nic.club) in 6 ms

suddenlysixam.club.     300     IN      A       104.21.53.110
suddenlysixam.club.     300     IN      A       172.67.212.56
;; Received 79 bytes from 173.245.59.154#53(derek.ns.cloudflare.com) in 7 ms
```

dns server to query

record we are looking for

Step 1: Our dns server gives us the root servers & tells us to ask them

Step 2: A root server gives us the TLD servers for the .club domain, and tells us to ask them

Step 3: The TLD server gives us the authoritative servers for suddenlysixam.club, and tells us to ask them

answer!

Step 4: The authoritative server gives us the record we are looking for

Common DNS Attacks

Spoofing / Cache Poisoning

- injects fake DNS data to a resolver makes queries redirect to fake sites

Distributed Denial of Service (DDoS)

- Utilizing DNS resolvers to overload a network

DNS Tunneling

- Using DNS queries to sneak information

Firewalls

What are they? → Acts as a filter for network data

Why use them? → Prevents malicious actors and helps control traffic

How do they work? → Defining rules and inspecting data

https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html

https://www.cloudflare.com/learning/security/what-is-a-firewall/

Types of Firewalls

Packet Filtering → inspects packet information and filters based on predefined criteria

Proxies → serves as “gateway” from one network to another item to prevent direct connections

Stateful Inspection → uses past information to monitor and make filtering decisions

Next Generation → all features of older firewalls + application awareness, user integration, and more

https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html

https://www.cloudflare.com/learning/security/what-is-a-firewall/

Where do firewalls exist?

Software → Installed on hosts that inspect traffic for its specific device

Hardware → Physical appliance that stands between your network and the rest of the internet

Cloud → Virtual, is a firewall based from your provider

https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html

https://www.cloudflare.com/learning/security/what-is-a-firewall/

Firewall Example (Raspberry Pi / UFW) - Rules

> To start, I have allowed connections to port 22 (SSH)

> I am then able to SSH to the Pi

> I then added a rule to deny incoming traffic from my laptop's IP address

> Which causes attempts to SSH to time out

(Please note that I did have to reorder the rules by editing /etc/ufw/user.rules)

Some common ports: 22 (SSH), 53 (DNS), 80 (HTTP), 443 (HTTPS)

```
labclub@fighter:~ $ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22                         ALLOW IN    Anywhere
22 (v6)                    ALLOW IN    Anywhere (v6)
```

```
meganst@meganstlap00 ~ % ssh labclub@10.70.57.180
labclub@10.70.57.180's password:
Linux fighter 6.12.47+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.47-1+rpt1 (2025-09-16) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 16 14:35:08 2026 from 10.70.57.180
labclub@fighter:~ $
```

```
labclub@fighter:~ $ sudo ufw deny in on eth0 from 10.70.57.163 to any port 22
Rule added
```

```
labclub@fighter:~ $ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22 on eth0                 DENY IN     10.70.57.163
22                         ALLOW IN    Anywhere
22 (v6)                    ALLOW IN    Anywhere (v6)
```

```
meganst@meganstlap00 ~ % ssh labclub@10.70.57.180
ssh: connect to host 10.70.57.180 port 22: Operation timed out
meganst@meganstlap00 ~ %
```

Firewall Example (Raspberry Pi / UFW) - Logging

> For the previous example, we can also check the logs to see the allowed or denied traffic:

```
[labclub@fighter:~ $ sudo ufw status verbose
Status: active
Logging: on (medium)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22 on eth0                 DENY IN     10.70.57.163               (log)
22                         ALLOW IN    Anywhere
22 (v6)                    ALLOW IN    Anywhere (v6)
```

```
2026-02-16T15:39:29.320450-05:00 fighter kernel: [UFW AUDIT] IN=eth0 OUT= MAC=d8:3a:dd:d7:bf:16:9c:eb:e8:3b:6d:31:08:00 SRC=10.70.57.163 DST=10.70.57.180 LEN=64 TOS=0x08 PREC=0x40 TTL=64 ID=0 DF PROTO=TCP SPT=63851 DPT=22 WINDOW=65535 RES=
0x00 SYN URGP=0
2026-02-16T15:39:29.320496-05:00 fighter kernel: [UFW BLOCK] IN=eth0 OUT= MAC=d8:3a:dd:d7:bf:16:9c:eb:e8:3b:6d:31:08:00 SRC=10.70.57.163 DST=10.70.57.180 LEN=64 TOS=0x08 PREC=0x40 TTL=64 ID=0 DF PROTO=TCP SPT=63851 DPT=22 WINDOW=65535 RES=
0x00 SYN URGP=0
```

(Please note that I turned on logging for this specific rule, otherwise we only get the AUDIT entry which is less clear for the purposes of this demonstration.)

Some common ports: 22 (SSH), 53 (DNS), 80 (HTTP), 443 (HTTPS)

Access Control Lists (ACLs)

- Set of rules that specify what packets to accept/deny at a network interface
- Checks rules sequentially (top down)
- Are not stateful, does not care about context
- Commonly used elsewhere as well though for different contexts

How DNS + Firewalls work together

- DNS requests must pass through firewalls
- Firewalls can examine requests and potentially deny unsafe requests or redirect to a safe page
- If deep packet inspection is allowed, can combat DNS tunneling
- Firewall can log suspicious requests
- However, DNS requests may be encrypted, would need next gen firewalls with capabilities to examine encrypted traffic

Pi-hole

- https://pi-hole.net/
- Acts as a recursive DNS resolver
- Can redirect/sinkhole requests based on domain
- Blocks Ads from being downloaded onto your network
- Combines ideas from DNS and Firewalls

Pi-hole vs. Browser Add-on Ad Blockers vs. Firewall

Firewall

- Allow/deny
- Inbound and outbound traffic
- Port(s)
- Protocol
- IP address(es)
- Interface(s)

Pi-hole

- Acts as your DNS resolver
- Filters DNS requests
- Works network wide, including on phones, smart TVs, streaming devices, game consoles, etc.

Browser Add-on Ad Blockers

- Inspects website content & traffic
- Works for that specific device, in that specific browser