[Skip to main content](https://lms.alnafi.com/xblock/block-v1:alnafi+DCCS102+2025_DCCS+type@vertical+block@2887eb97abc049fea78b6bd57cc8aa3c?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#main)

**DNS**

1: What are DNS & DNS RECORDS?

What is a DNS record?

DNS records (aka zone files) are instructions that live in authoritative DNS servers and provide information about a domain including what IP address is associated with that domain and how to handle requests for that domain. These records consist of a series of text files written in what is known as DNS syntax. DNS syntax is just a string of characters used as commands that tell the DNS server what to do. All DNS records also have a ‘TTL’, which stands for time-to-live, and indicates how often a DNS server will refresh that record.

You can think of a set of DNS records like a business listing on Yelp. That listing will give you a bunch of useful information about a business such as its location, hours, services offered, etc. All domains are required to have at least a few essential DNS records for a user to be able to access their website using a domain name, and there are several optional records that serve additional purposes.

What are the most common types of DNS records?

* A record - The record that holds the IP address of a domain. Learn more about the A record.
* AAAA record - The record that contains the IPv6 address for a domain (as opposed to A records, which list the IPv4 address). Learn more about the AAAA record.
* CNAME record - Forwards one domain or subdomain to another domain, does NOT provide an IP address. Learn more about the CNAME record.
* MX record - Directs mail to an email server.
* TXT record - Lets an admin store text notes in the record. These records are often used for email security.
* NS record - Stores the name server for a DNS entry.
* SOA record - Stores admin information about a domain.
* SRV record - Specifies a port for specific services.
* PTR record - Provides a domain name in reverse-lookups.

What are some of the less commonly used DNS records?

* AFSDB record - This record is used for clients of the Andrew File System (AFS) developed by Carnegie Melon. The AFSDB record functions to find other AFS cells.
* APL record - The ‘address prefix list’ is an experiment record that specifies lists of address ranges.
* CAA record - This is the ‘certification authority authorization’ record, it allows domain owners state which certificate authorities can issue certificates for that domain. If no CAA record exists, then anyone can issue a certificate for the domain. These records are also inherited by subdomains.
* DNSKEY record - The ‘DNS Key Record’ contains a public key used to verify Domain Name System Security Extension (DNSSEC) signatures.
* CDNSKEY record - This is a child copy of the DNSKEY record, meant to be transferred to a parent.
* CERT record - The ‘certificate record’ stores public key certificates.
* DCHID record - The ‘DHCP Identifier’ stores info for the Dynamic Host Configuration Protocol (DHCP), a standardized network protocol used on IP networks.
* DNAME record - The ‘delegation name’ record creates a domain alias, just like CNAME, but this alias will redirect all subdomains as well. For instance, if the owner of ‘example.com’ bought the domain ‘website.net’ and gave it a DNAME record that points to ‘example.com’, then that pointer would also extend to ‘blog.website.net’ and any other subdomains.
* HIP record - This record uses ‘Host identity protocol’, a way to separate the roles of an IP address; this record is used most often in mobile computing.
* IPSECKEY record - The ‘IPSEC key’ record works with the Internet Protocol Security (IPSEC), an end-to-end security protocol framework and part of the Internet Protocol Suite (TCP/IP).
* LOC record - The ‘location’ record contains geographical information for a domain in the form of longitude and latitude coordinates.
* NAPTR record - The ‘name authority pointer’ record can be combined with an SRV record to dynamically create URI’s to point to based on a regular expression.
* NSEC record - The ‘next secure record’ is part of DNSSEC, and it’s used to prove that a requested DNS resource record does not exist.
* RRSIG record - The ‘resource record signature’ is a record to store digital signatures used to authenticate records in accordance with DNSSEC.
* RP record - This is the ‘responsible person record and it stores the email address of the person responsible for the domain.
* SSHFP record - This record stores the ‘SSH public key fingerprints’; SSH stands for Secure Shell and it’s a cryptographic networking protocol for secure communication over an unsecured network.

2 : Tool You Need To Enumerate DNS

DNS servers are the heart and soul of the Internet. Without them, we couldn't resolve hostnames and domain names into IP addresses.

However, DNS is also one of the most frequently attacked protocols, where different types of DNS attacks are spread from home users to small, mid and large companies.

That's why, in the information-gathering process, the most common practice is to create a full inventory of all internet-connected devices and domain names from the company you're investigating.

We all know that DNS servers are basically computers connected to the Internet, and that helps us to resolve hostnames into IP addresses. They're in charge of managing and processing DNS requests from clients that need to fetch fresh domain name information, along with DNS records.

That's where the weak link shows up, thanks to the way the DNS was built. It's a bit vulnerable, which allows us to perform DNS enumeration (also known as DNS recon) easily.

But, for those who are just starting out in this OSINT world, let's first find out what DNS enumeration is, and the different techniques and DNS search tools that can help us on this journey.

Top  DNS enumeration tools

Dig

Host

DNSenum

Nmap

DNS Recon

**What's DNS enumeration?**

DNS enumeration is one of the most popular reconnaissance tasks there is for building a profile of your target.

In plain English, it's the act of detecting and enumerating all possible DNS records from a domain name. This includes hostnames, DNS record names, DNS record types, TTLs, IP addresses, and a bit more, depending on how much information you're looking for.

With effective DNS enumeration, you can clone DNS zones manually, using scripts or by exploiting DNS zone transfer vulnerabilities, known as AXFR (Asynchronous Transfer Full Range) Transfer. This latter type of DNS transfer takes place when an attacker detects a misconfigured DNS server that is actually responding to AXFR requests.

**Impact**

Once DNS enumeration is completed, unauthenticated users may use this information to observe internal network records, grabbing useful DNS information that provides the attacker access to a full DNS map. This allows him to explore the attack surface area of any company, so he can later scan it, collect data, and—while he's at it—exploit it if there's an open opportunity.

In the past, we've seen a bit of DNS enumeration, such as in the How to Find Subdomains article. However, that was only focused on subdomains. Today we'll go one step forward and show you how to perform full DNS enumeration.

**Top  DNS enumeration tools**

There are plenty of DNS recon and DNS enumeration tools around, from Bash to Python scripts; however, the easiest DNS enumeration can be performed with a single system command.

Let's explore the best ways to perform a DNS enumeration.

**Dig**

Once again, our beloved dig command comes to the rescue, helping us perform DNS enumeration by querying popular types of DNS records. Here's how it's done.

To perform a simple domain lookup to fetch A records:

dig securitytrails.com +short

Expected output:

[research@securitytrails.com ~]$ dig securitytrails.com +short

151.139.243.5

[research@securitytrails.com ~]$

Now let's grab some mail server information adding -t mx parameters:

[research@securitytrails.com ~]$ dig securitytrails.com -t mx +short

10 aspmx2.googlemail.com.

1 aspmx.l.google.com.

5 alt2.aspmx.l.google.com.

5 alt1.aspmx.l.google.com.

10 aspmx3.googlemail.com.

[research@securitytrails.com ~]$

That's right, securitytrails.com uses Google Apps for email services.

The same goes for NS, CNAME, and other records:

dig securitytrails.com -t ns +short

Output:

[research@securitytrails.com ~]$ dig securitytrails.com -t ns +short

ns07.domaincontrol.com.

ns08.domaincontrol.com.

[research@securitytrails.com ~]$

Great! With this last command, we now have the authoritative name servers.

Of course, dig allows DNS transfers and this leads us to use the AXFR argument against the ns08.domaincontrol.com NS, as you see below:

[researcg@securitytrails.com ~]$ dig axfr securitytrails.com ns08.domaincontrol.com

; <<>> DiG 9.11.10-RedHat-9.11.10-1.fc30 <<>> axfr securitytrails.com ns08.domaincontrol.com

;; global options: +cmd

; Transfer failed.

[research@securitytrails.com ~]$

And the transfer failed because our DNS servers are well-secured.

Recon tip: sometimes a particular name server may be configured to reject AXFR requests. However, some others may not—that's why we suggest you launch AXFR queries against all the authoritative NS.

**Host**

The host is a command used to resolve the IP address of any given domain name.

As an example, we'll use it to get all the public DNS records from securitytrails.com. See below:

host securitytrails.com

Expected output:

[research@securitytrails.com ~]$ host securitytrails.com

securitytrails.com has address 151.139.243.5

securitytrails.com mail is handled by 5 alt2.aspmx.l.google.com.

securitytrails.com mail is handled by 5 alt1.aspmx.l.google.com.

securitytrails.com mail is handled by 10 aspmx3.googlemail.com.

securitytrails.com mail is handled by 1 aspmx.l.google.com.

securitytrails.com mail is handled by 10 aspmx2.googlemail.com.

The default host command retrieves A, AAAA and MX records.

If you want to specify any specific type of DNS record, you can use the -t option. For example:

host -t ns securitytrails.com

Expected output:

[research@securitytrails.com ~]$ host -t ns securitytrails.com

securitytrails.com name server ns08.domaincontrol.com.

securitytrails.com name server ns07.domaincontrol.com.

What if you want to grab MX records? Just use -t mx, as shown here:

[research@securitytrails.com ~]$ host -t mx securitytrails.com

securitytrails.com mail is handled by 10 aspmx3.googlemail.com.

securitytrails.com mail is handled by 5 alt2.aspmx.l.google.com.

securitytrails.com mail is handled by 10 aspmx2.googlemail.com.

securitytrails.com mail is handled by 1 aspmx.l.google.com.

securitytrails.com mail is handled by 5 alt1.aspmx.l.google.com.

The -t option can be used to request any recognized query type such as: CNAME, NS, SOA, TXT, DNSKEY, AXFR, etc. If no query type is specified, host by default will look for A, AAAA and MX records.

Having said that, let's try to perform a full DNS transfer:

[research@securitytrails.com ~]$ host -t axfr securitytrails.com ns08.domaincontrol.com

Trying "securitytrails.com"

Host securitytrails.com not found: 9(NOTAUTH)

; Transfer failed.

[research@securitytrails.com ~]$

No luck :), our DNS server doesn't allow AXFR transfers by default.

In the case of a successful DNS transfer, you should be able to get the full DNS zone for the given domain name, as you see below—notice this time we are using -l option, which is another way to list all DNS records from a domain name—while testing the vulnerable site zonetransfer.me:

[research@securitytrails.com ~]$ host -l zonetransfer.me nsztm1.digi.ninja

Using domain server:

Name: nsztm1.digi.ninja

Address: 81.4.108.41#53

Aliases:

zonetransfer.me has address 5.196.105.14

zonetransfer.me name server nsztm1.digi.ninja.

zonetransfer.me name server nsztm2.digi.ninja.

14.105.196.5.IN-ADDR.ARPA.zonetransfer.me domain name pointer www.zonetransfer.me.

asfdbbox.zonetransfer.me has address 127.0.0.1

canberra-office.zonetransfer.me has address 202.14.81.230

dc-office.zonetransfer.me has address 143.228.181.132

deadbeef.zonetransfer.me has IPv6 address dead:beaf::

email.zonetransfer.me has address 74.125.206.26

home.zonetransfer.me has address 127.0.0.1

internal.zonetransfer.me name server intns1.zonetransfer.me.

internal.zonetransfer.me name server intns2.zonetransfer.me.

intns1.zonetransfer.me has address 81.4.108.41

intns2.zonetransfer.me has address 167.88.42.94

office.zonetransfer.me has address 4.23.39.254

ipv6actnow.org.zonetransfer.me has IPv6 address 2001:67c:2e8:11::c100:1332

owa.zonetransfer.me has address 207.46.197.32

alltcpportsopen.firewall.test.zonetransfer.me has address 127.0.0.1

vpn.zonetransfer.me has address 174.36.59.154

www.zonetransfer.me has address 5.196.105.14

[research@securitytrails.com ~]$

**¶DNSenum**

DNSEnum is a great script specifically designed for DNS recon activities. Written in Perl, it can help you create a full DNS map of any domain name on the Internet.

Available in most distros, including Ubuntu, Fedora, and of course, Kali Linux, it offers an easy syntax for all who are performing reconnaissance tasks.

What can I do with DNSenum?

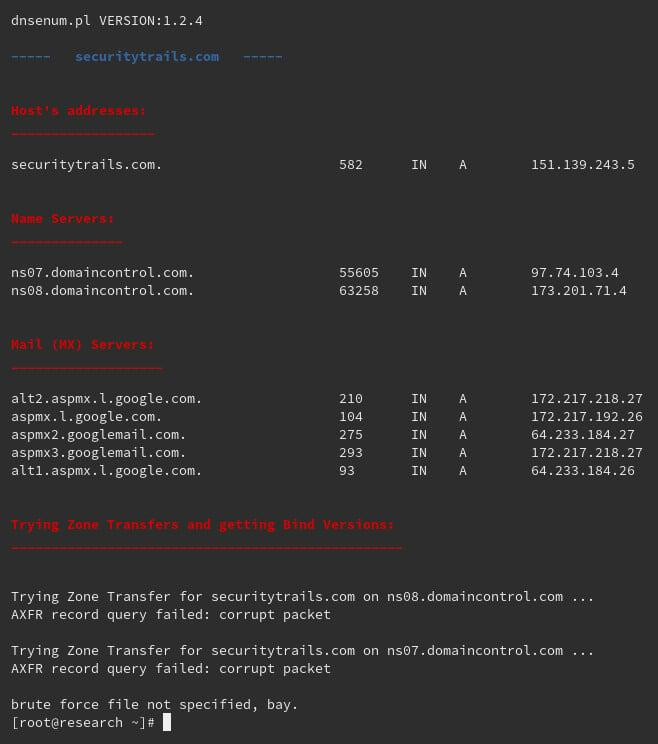
Fetch NS, MX, AXFR and A records, as well as remote BIND version from the DNS server. Apart from that, it also allows you to perform Google scraping using Google dorks such as allinurl: -www site:domain, launch brute force subdomain reconnaissance attacks using word lists, and get a full list of C class domain network ranges. It also allows WHOIS queries on each one of them and reverse DNS lookups against netranges.

How does it work?

Let's look at an example where we'll be avoiding reverse lookup (--noreverse) and saving the output into a file.xml (-o) while querying securitytrails.com:

[root@research ~]# dnsenum --noreverse -o file.xml securitytrails.com

Expected output:



As shown in the result, DNSenum was able to get A records for the main hostname, as well as NS and MX records, while performing a DNS zone transfer by querying the AXFR record.

DNSenum also allows you to use the Google search engine to "scrape" the results and get a list of subdomains. We'll now combine this with the "-p" argument, which specifies the number of pages searched on Google that will be processed (by default 5 pages). The "-s" option defines the maximum number of subdomains that will be extracted from Google (default is 15).

dnsenum --dnsserver ns3.p16.dynect.net github.com -p 10 -s 50



**¶Nmap**

Nmap was our #1 choice when we reviewed the best port scanners, but it's really more than that. This time it will help us reveal DNS information from a remote domain name.

By using the dns-brute script, Nmap will attempt to enumerate DNS hostnames by brute forcing popular subdomain names. In this case, we did it against microsoft.com and this was the result:

[root@research ~]# nmap -T4 -p 53 --script dns-brute microsoft.com

Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-02 11:56 -03

Nmap scan report for microsoft.com (40.113.200.201)

Host is up (0.21s latency).

Other addresses for microsoft.com (not scanned): 40.76.4.15 104.215.148.63 40.112.72.205 13.77.161.179

PORT STATE SERVICE

53/tcp filtered domain

Host script results:

| dns-brute:

| DNS Brute-force hostnames:

| admin.microsoft.com - 13.107.9.156

| admin.microsoft.com - 2620:1ec:4:0:0:0:0:156

| ads.microsoft.com - 23.100.75.192

| alerts.microsoft.com - 40.112.72.205

| apps.microsoft.com - 23.44.181.123

| id.microsoft.com - 13.107.6.190

| info.microsoft.com - 192.28.149.178

| test.microsoft.com - 104.211.31.212

|\_ news.microsoft.com - 192.237.225.141

Nmap done: 1 IP address (1 host up) scanned in 10363.72 seconds

[root@research ~]

While this isn't the most complete DNS recon option, it does help as an alternative source of information during your data collection process.

**¶DNS Recon**

DNSRecon is another great script that can help you discover DNS data from any given domain name.

It allows you to enumerate all types of DNS records, including A, AAAA, SPF, TXT, SOA, NS and MX, and also includes a brute force technique for grabbing subdomain and host A and AAAA records based on a wordlist.

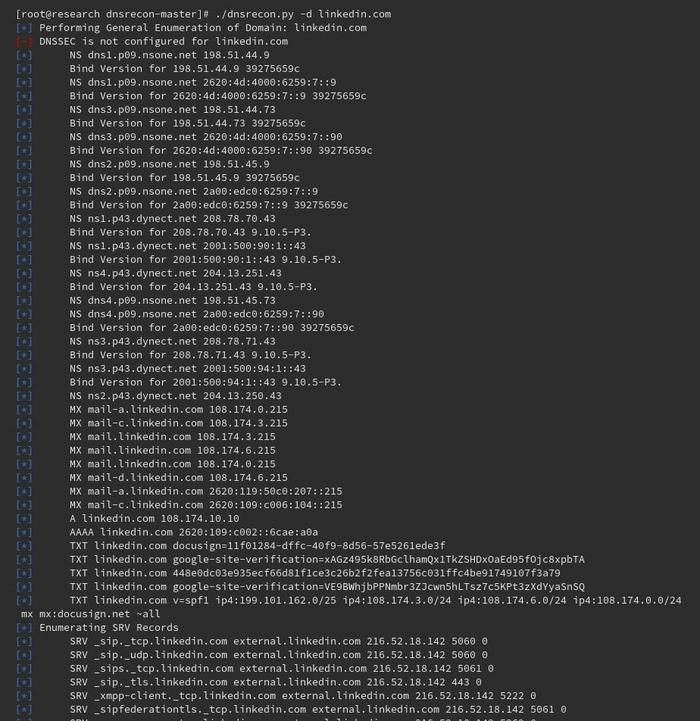
A cool thing we noticed is that it supports checking for cached A and AAAA DNS records on the DNS servers, as well as local DNS enumeration capabilities.

How can I perform DNS exploration with DNSRecon?

The easiest way is by using the -d parameter, as you see below:

dnsrecon -d domain.com

Here we performed this dns enumeration against linkedin.com, and this was the result:



As shown, it was able not only to fetch multiple records (MX, A, AAAA, TXT, SRV and NS), but also to find some exposed bind versions from their DNS servers.

**¶Fierce**

Fierce is another great DNS reconnaissance tool. Written in Perl, Fierce offers numerous options for performing DNS enumeration by scanning domains in just minutes. Its syntax is pretty easy, as you can see:

fierce -dns securitytrails.com

Expected output:

root@securitytrails-kali:~# fierce -dns securitytrails.com

DNS Servers for securitytrails.com:

ns08.domaincontrol.com

ns07.domaincontrol.com

Trying zone transfer first...

Testing ns08.domaincontrol.com

Request timed out or transfer not allowed.

Testing ns07.domaincontrol.com

Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)

Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...

Nope. Good.

Now performing 2280 test(s)...

151.139.243.5 blog.securitytrails.com

151.139.243.5 ftp.securitytrails.com

Subnets found (may want to probe here using nmap or unicornscan):

151.139.243.0-255 : 2 hostnames found.

Done with Fierce scan: http://ha.ckers.org/fierce/

Found 2 entries.

Have a nice day.

root@securitytrails-kali:~#

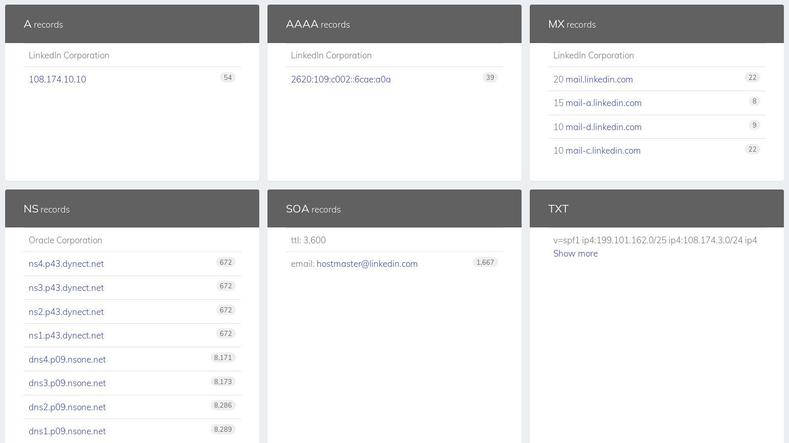
Fierce was able to discover a few subdomains, along with NS records, and attempted to run a DNS transfer, which obviously failed.

By adding the --wide, you can also extend the fierce scan to the entire class C after finding any matching hostnames in that class C. This can take a lot of time to finish, especially on networks with a lot of hosts, so keep that in mind.

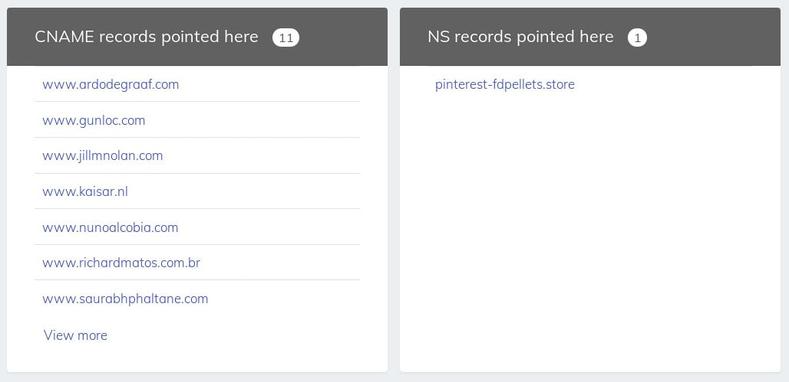
**¶SecurityTrails advanced DNS enumeration**

DNS record detection is one of the core features in all of our products, from the free app to our passive DNS API to our enterprise-grade products.

In particular, our free app offers great results when it comes to building a DNS map of all possible DNS records from a given domain name, as shown here:



For this purpose, the SecurityTrails free app offers both current and historical DNS records for A, AAAA, NS, MX, SOA and TXT records, letting you build an existing DNS structure quickly.



It also includes full subdomain enumeration capabilities, which allows you to discover all existing subdomains for the target domain. In this case, linkedin.com showed that it has around 1235 subdomains, quite an extensive list.

Where other DNS enumeration tools fail to retrieve all the existing subdomains, or require to launch brute-force techniques, our intelligence tool does it all, in a matter of seconds.

However, if you're ready to jump into the real thing, that's where our enterprise-grade SurfaceBrowser product comes in.

SurfaceBrowser is no mere DNS exploration tool, but a complete OSINT utility that will give you a look at any attack surface area by combining DNS information, IP addresses, associated domains, SSL certificates and SSL ports.

Unlike the free app, SurfaceBrowser doesn't limit the amount of DNS information, and gives you instant access to full DNS enumeration like no other tool on the Internet.

Want to see how it works? Check it out:

* First, log in from https://securitytrails.com/app/sb/ (if you don't have an account, schedule a demo with our sales team).
* Now, analyze a domain—let's say sputniknews.com, a popular russian news service.

http://172.30.2.67:8000

http://172.30.2.67:8001

http://172.30.2.67:8003

3: DNSDumpster

No brute force subdomain enumeration is used as is common in dns recon tools that enumerate subdomains. We use open source intelligence resources to query for related domain data. It is then compiled into an actionable resource for both attackers and defenders of Internet facing systems.

More than a simple DNS lookup this tool will discover those hard to find sub-domains and web hosts. The search relies on data from our crawls of the Alexa Top 1 Million sites, Search Engines, Common Crawl, Certificate Transparency, Max Mind, Team Cymru, Shodan and scans.io.

http://172.30.2.67:8000

[http://172.30.2.67:8001](http://172.30.2.67:8001/)

http://172.30.2.67:8003

4 ; Subdomain Enumration

In the Domain Name System (DNS) hierarchy, a subdomain is a domain that is a part of another (main) domain.[1] For example, if a domain offered an online store as part of their website example.com, it might use the subdomain shop.example.co

The Domain Name System (DNS) has a tree structure or hierarchy, which includes nodes on the tree being a domain name. A subdomain is a domain that is part of a larger domain. Each label may contain from 1 to 63 octets. The full domain name may not exceed a total length of 253 ASCII characters in its textual representation.[2]

Subdomains are defined by editing the DNS zone file pertaining to the parent domain. However, there is an ongoing debate over the use of the term "subdomain" when referring to names which map to the Address record A (host) and various other types of zone records which may map to any public IP address destination and any type of server. Network Operations teams insist that it is inappropriate to use the term "subdomain" to refer to any mapping other than that provided by zone NS (name server) records and any server-destination other than that.

According to RFC 1034, "a domain is a subdomain of another domain if it is contained within that domain". Based on that definition, a host cannot be a subdomain, only a domain can be a subdomain. A subdomain will also have a separate zone file with a SOA record (Start of Authority).

Most domain registries only allocate a two-level domain name. Hosting services typically provide DNS Servers to resolve subdomains within that master domain.



Example of subdomain

A fully qualified domain name consists of multiple parts. For example, take the English Wikipedia domain en.wikipedia.org. The en is a subdomain of wikipedia.org. Although wikipedia.org is usually considered to be the domain name, wikipedia is actually a sub-domain of the org TLD (top level domain). Any fully qualified domain name can be a host or a subdomain.

A domain name that does not include any subdomains is known as an apex domain, root domain, or bare domain.[3] For example, wikipedia.org is the apex domain of Wikipedia, which redirects to the subdomain www.wikipedia.org.

Subdomains are often used by internet service providers supplying web services. They allocate one (or more) subdomains to their clients who do not have their own domain name. This allows independent administration by the clients over their subdomain.

Subdomains are also used by organizations that wish to assign a unique name to a particular department, function, or service related to the organization. For example, a university might assign "cs" to the computer science department, such that a number of hosts could be used inside that subdomain, such as www.cs.example.edu.

There are some widely recognized subdomains such as WWW and FTP. This allows for a structure where the domain contains administrative directories and files including the FTP directories and webpages. The FTP subdomain could contain logs and the web page directories, while the WWW subdomain contains the directories for the webpages. Independent authentication for each domain provides access control over the various levels of the domain.

http://172.30.2.67:8000

[http://172.30.2.67:8001](http://172.30.2.67:8001/)

http://172.30.2.67:8003