The Canonical Name Record

The *Canonical Name (CNAME) record* points one domain at another. Listing 2-4 shows a CNAME record response. CNAME records can make administration a bit easier. For example, you can create a CNAME record named *mail.yourdomain.com* and direct it to Gmail's login page. This not only is easier for your users to remember but also gives you the flexibility of pointing the CNAME at another email provider in the future without having to inform your users.

```
$ dig mail.google.com. a
-- snip --
;QUESTION
mail.google.com. IN A
;ANSWER
❶ mail.google.com. 21599 IN CNAME ❷googlemail.l.google.com.
googlemail.l.google.com. 299 IN A 172.217.3.229
-- snip --
```

Listing 2-4: DNS answer of the mail.google.com *CNAME resource record*

Notice that you ask the domain name server for the A record of the subdomain *mail.google.com*. But in this case, you receive a CNAME instead. This tells you that *googlemail.l.google.com* ❷ is the canonical name for *mail.google.com* ❶. Thankfully, you receive the A record for *googlemail.l.google.com* with the response, alleviating you from having to make a second query. You now know your destination IP address is 172.217.3.229. Google's domain name server was able to return both the CNAME answer and the corresponding Address answer in the same reply because it is an authority for the CNAME answer's domain name as well. Otherwise, you would expect only the CNAME answer and would then need to make a second query to resolve the CNAME answer's IP address.

The Mail Exchange Record

The *Mail Exchange (MX) record* specifies the mail server hostnames that should be contacted when sending email to recipients at the domain. Remote mail servers will query the MX records for the domain portion of a recipient's email address to determine which servers should receive mail for the recipient. Listing 2-5 shows the response a mail server will receive.

```
$ dig google.com. mx
-- snip --
;QUESTION
google.com. IN MX
;ANSWER
google.com. 599 IN MX ❶10 aspmx.l.google.com.
google.com. 599 IN MX 50 alt4.aspmx.l.google.com.
google.com. 599 IN MX 30 alt2.aspmx.l.google.com.
google.com. 599 IN MX 20 alt1.aspmx.l.google.com.
google.com. 599 IN MX 40 alt3.aspmx.l.google.com.
-- snip --
```

Listing 2-5: DNS answer of the google.com *MX resource records*

In addition to the domain name, TLL value, and record type, MX records contain the *priority field* ❶, which rates the priority of each mail server. The lower the number, the higher the priority of the mail server. Mail servers attempt to deliver emails to the mail server with the highest priority, then resort to the mail servers with the next highest priority if necessary. If more than one mail server shares the same priority, the mail server will pick one at random.

The Pointer Record

The *Pointer (PTR) record* allows you to perform a reverse lookup by accepting an IP address and returning its corresponding domain name. Listing 2-6 shows the reverse lookup for 8.8.4.4.

```
$ dig 4.4.8.8.in-addr.arpa. ptr
-- snip --
;QUESTION
❶ 4.4.8.8.in-addr.arpa. IN PTR
;ANSWER
4.4.8.8.in-addr.arpa. 21599 IN PTR ❷google-public-dns-b.google.com.
-- snip --
```

Listing 2-6: DNS answer of the 8.8.4.4 PTR resource record

To perform the query, you ask the domain name server for the IPv4 address in reverse order ❶ with the special domain *in-addr.arpa* appended because the reverse DNS records are all under the *.arpa* top-level domain. For example, querying the pointer record for the IP 1.2.3.4 means you need to ask for *4.3.2.1.in-addr.arpa*. The query in Listing 2-6 tells you that the IPv4 address 8.8.4.4 reverses to the domain name *google-public-dns-b.google.com* ❷. If you were performing a reverse lookup of an IPv6 address, you'd append the special domain *ip6.arpa* to the reversed IPv6 address as you did for the IPv4 address.

NOTE *See Wikipedia for more information on reverse DNS lookup:* https://en.wikipedia.org/wiki/Reverse_DNS_lookup.

The Text Record

The *Text (TXT) record* allows the domain owner to return arbitrary text. These records can contain values that prove domain ownership, values that remote mail servers can use to authorize email, and entries to specify which IP addresses may send mail on behalf of the domain, among other uses. Listing 2-7 shows the text records associated with *google.com*.

```
$ dig google.com. txt
-- snip --
;QUESTION
google.com. IN TXT
;ANSWER
```

```
google.com. 299 IN TXT
  ❶"facebook-domain-verification=22rm551cu4k0ab0bxsw536tlds4h95"
google.com. 299 IN TXT "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com. 299 IN TXT "v=spf1 include:_spf.google.com ~all"
google.com. 299 IN TXT
    "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
-- snip --
```

Listing 2-7: DNS answer of the google.com TXT resource records

The domain queries and answers should start to look familiar by now. The last field in a TXT record is a string of the TXT record value ❶. In this example, the field has a Facebook verification key, which proves to Facebook that Google's corporate Facebook account is who they say they are and has the authority to make changes to Google's content on Facebook. It also contains *Sender Policy Framework* rules, which inform remote mail servers which IP addresses may deliver email on Google's behalf.

NOTE *The Facebook for Developers site has more information about domain verification at* https://developers.facebook.com/docs/sharing/domain-verification/.

Multicast DNS

Multicast DNS (mDNS) is a protocol that facilitates name resolution over a local area network (LAN) in the absence of a DNS server. When a node wants to resolve a domain name to an IP address, it will send a request to an IP multicast group. Nodes listening to the group receive the query, and the node with the requested domain name responds to the IP multicast group with its IP address. You may have used mDNS the last time you searched for and configured a network printer on your computer.

Privacy and Security Considerations of DNS Queries

DNS traffic is typically unencrypted when it traverses the internet. A potential exception occurs if you're connected to a virtual private network (VPN) and are careful to make sure all DNS traffic passes through its encrypted tunnel. Because of DNS's unencrypted transport, unscrupulous ISPs or intermediate providers may glean sensitive information in your DNS queries and share those details with third parties. You can make a point of visiting HTTPS-only websites, but your DNS queries may betray your otherwise secure browsing habits and allow the DNS server's administrators to glean the sites you visit.

Security is also a concern with plaintext DNS traffic. An attacker could convince your web browser to visit a malicious website by inserting a response to your DNS query. Considering the difficulty of pulling off such an attack, it's not an attack you're likely to experience, but it's concerning nonetheless. Since DNS servers often cache responses, this attack usually takes place between your device and the DNS server it's configured to use. RFC 7626 (*https://tools.ietf.org/html/rfc7626/*) covers these topics in more detail.

Domain Name System Security Extensions

Generally, you can ensure the authenticity of data sent over a network in two ways: authenticating the content and authenticating the channel. *Domain Name System Security Extensions (DNSSEC)* is a method to prevent the covert modification of DNS responses in transit by using digital signatures to authenticate the response. DNSSEC ensures the authenticity of data by authenticating the content. DNS servers cryptographically sign the resource records they serve and make those signatures available to you. You can then validate the responses from authoritative DNS servers against the signatures to make sure the responses aren't fraudulent.

DNSSEC doesn't address privacy concerns. DNSSEC queries still traverse the network unencrypted, allowing for passive observation.

DNS over TLS

DNS over TLS (DoT), detailed in RFC 7858 (*https://tools.ietf.org/html/rfc7858/*), addresses both security and privacy concerns by using *Transport Layer Security (TLS)* to establish an encrypted connection between the client and its DNS server. TLS is a common protocol used to provide cryptographically secure communication between nodes on a network. Using TLS, DNS requests and responses are fully encrypted in transit, making it impossible for an attacker to eavesdrop on or manipulate responses. DoT ensures the authenticity of data by authenticating the channel. It does not need to rely on cryptographic signatures like DNSSEC because the entire conversation between the DNS server and the client is encrypted.

DoT uses a different network port than does regular DNS traffic.

DNS over HTTPS

DNS over HTTPS (DoH), detailed in RFC 8484 (*https://tools.ietf.org/html/rfc8484/*) aims to address DNS security and privacy concerns while using a heavily used TCP port. Like DoT, DoH sends data over an encrypted connection, authenticating the channel. DoH uses a common port and maps DNS requests and responses to HTTP requests and responses. Queries over HTTP can take advantage of all HTTP features, such as caching, compression, proxying, and redirection.

What You've Learned

We covered a lot of ground in this chapter. You learned about IP addressing, starting with the basics of IPv4 multicasting, broadcasting, TCP and UDP ports, socket addresses, network address translation, and ARP. You then learned about IPv6, its address categories, and its advantages over IPv4.

You learned about the major network-routing protocols, ICMP and DNS. I'll again recommend the *TCP/IP Guide* by Charles M. Kozierok (No Starch Press, 2005) for its extensive coverage of the topics in this chapter.