DNSSEC

DNS

Domain Name Systems are what translate human readable romain names (ex: “google.com” -> 172.217.19.14). A DNS reads a domain name from right to left and traverses a network of servers to locate the corresponding ip. This network is made up of 4 main layers: the Recursive Name server, the Root servers, the TLD servers, and the Authoritative Name servers.

Recursive Name Server

Also called the Recursive Resolver, the Recursive Name server receives a client’s URL search, communicates with the other servers, and returns the resulting ip to the client.

These servers are usually managed by an ISP, and can cache frequent or recent searches to improve efficiency.

Root Name Server

These servers don’t have the ip we’re looking for, but can redirect us to the appropriate TLD server. There are 13 Root server networks around the world. They are overseen by a non-profit called the Internet Corporation of Assigned Names and Numbers (ICANN) and managed by various other groups (ex: NASA and VeriSign Inc.).

Top-Level Domain Server

These servers also don’t have the ip we’re looking to, but are responsible for all domains within a certain top-level domain (“.ca”, .com”, “.org” are top-level domains). These servers are overseen by a branch of ICANN called the Internet Assigned Numbers Authority (IANA).

IANA splits TLD servers into 2 main groups:

- Generic top-level domains (.com .org .net …)
- Country code top-level domains (.ca .us .uk .jp …)

Authoritative Name Server

These servers maintain records on IP’s belonging to its domain (example: IP’s of servers belonging to “utoronto.ca”). They are also responsible for its sub-domains (like “mail.utoronto.ca”). These servers are the ones that will be providing us with an actual ip relating to our searched URL.

Different domain names can point to the same IP, even if we travel through different servers to find them. (“example.com” and “example.net” point to the same IP)

Authoritative names servers can also return an alias instead of an IP, in which case the Recursive Resolver starts a new search using the new alias. When this occurs, and the DNS finally finds an IP, it returns that IP and caches it with the original domain name searched.

For example, searching for “example.net” will direct us to “example.com”’s server, but as far as the client sees this is still example.net.

The Problem

Security wasn’t the primary concern when DNS was designed back in the 80s. DNS traffic is not encrypted and the RNS has no way of authenticating responses it gets from the servers.

DNS Poisoning

Since the RNS can’t verify the validity of a DNS response, it may cache a malicious IP returned by an attacker to URL of a valid and commonly used site. Future users seeking this domain will continue to be redirected to this IP for as long as is remains in the cache.

DNS servers can also “share” their cached information with other servers, spreading the exploit further.

DNSSEC

Domain Name **S**ystem **Sec**urity Extension. Adds a layer of security on top of DNS which signs responses.

Note: this does not encrypt the responses.

RRSets

A grouping of DNS records (also called Resource Records).

Records of the same type get grouped into these sets. For example if “mail.example.com” and “example.com” both have an IPv4 record (the ‘A’ records) they will be grouped together.

Each authoritative name server has a pair of keys: a private key and and public key (the public key is stored in the DNSKEY entry on the server). These are the Zone-Signing Key pair. Now we can sign the RRSet with the private key, creating the signed RR called RRSIG. If we send RRSet, RRSig and the public zone signing key, the RNS can validate the response.

But what if the DNSKEY was compromised? At this point we have no way of knowing that the keys haven't been tampered with.

There's another set of keys, calle the key-signing keys, which we use to validate the ZSK public key. To do this, the server signs both the public ZSK and the public KSK using its private KSK, creating a new signature for the DNSKEY RRset.

Now, when a resolver requests an RRset, it also receives a RRsig for that set. To verify that signature, it requests the DNSKEY's RRset and also receives a RRsig. It verifies the DNSKEY with the public KSK, then verifies the requested RRset with the now verified public ZSK.

This way, the zone-signing keys can be more compact and still offer some level of security. They are also easier to replace when old or compromised.

But how can we trust the Key-Signing Keys now?

Chain of Trust

A DNS server hashes its public KSK and gives it to its parent server. This has gets published as a DS record (delegation signer record). Now whenever the RNS gets referred to a child server by its parent server, that parent server also provides the DS record. This way the resolver can hash the KSK and compare it to the provided DS record to check its validity.

This means that as long as we trust the parent zone, we can trust the child zone. I.e. if we trust ".ca" we can trust "utoronto.ca", then 'mail.utoronto.ca" and so on.

But what about the Root Servers? There's no parent zone for us to trust from here.

The Root Signing Ceremony

Several selected individuals from around the world are selected to come together and sign the Root Server's DNSKEY RRset, producing an RRsig that can be used for the next few months. This is done in a public and highly audited way.

The whole process can be read about here (from the perspective of a member of Cloudflare who participated in this ceremony):
https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/

The process is very long and detailed, but interesting. And as long as we trust that the signatures are valid, then we can trust the Root servers, then the TLD servers and so on.