DNS

Domain Name System

1. I ask my local resolver, "what is the IP for iana.org?"
2. ... ???
3. My local resolver gives me the right answer

... but how?

Let's take a look at some DNS traffic using the Wireshark packet capture and dissection tool.

DNS details

UDP datagrams to port 53

Resource Records

- A, AAAA, CNAME, MX, NS, TXT...
- static or dynamic

Authoritative name servers

Common DNS record types include:

Type	Meaning
A	IPv4 address
AAAA	IPv6 address
CNAME	"Canonical name" (like a symbolic link)
MX	Mail eXchange server
NS	Nameserver (authoritative nameservers for a domain)
TXT	Arbitrary text; now often used for crypto protocols

A DNS server can be *authoritative* for a domain, meaning that it is the canonical server with the right to tell you about domain records ______________________, not via ________________.

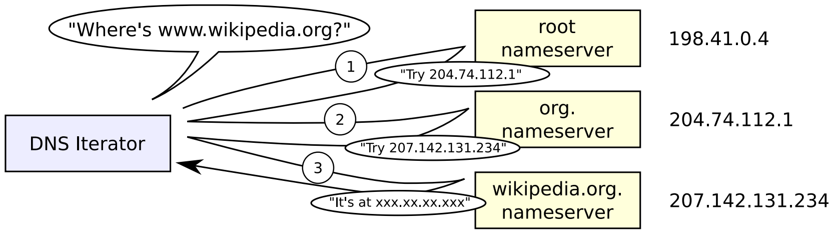
Root servers

Server	Operator	Server	Operator
a	VeriSign, Inc.	h	US Army (Research Lab)
b	USC (ISI)	i	Netnod
c	Cogent Communications	j	VeriSign, Inc.
d	University of Maryland	k	RIPE NCC
e	NASA Ames	l	ICANN
f	ISC	m	WIDE Project
g	US DoD (NIC)		

Much of the foundation of the Internet depends on a few key DNS servers, most of which are located in — or at least run by entities in — the United States. This makes sense given the history of ARPAnet, but what might seem natural and fitting to us may seem like something else to others!

Each "server" is actually a bunch of root server instances rather than a single point of failure, (see root-servers.org), but each is controlled by one organization. So, the "A" server might actually be dozens of DNS servers reachable at `198.41.0.4` or `2001:503:ba3e::2:30`, but they're all run by Verisign.

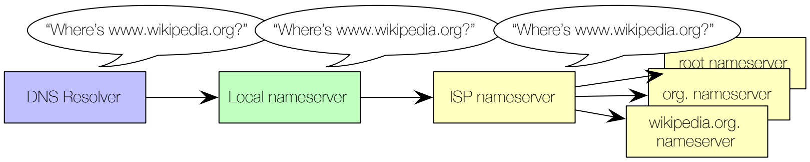
Iterative lookup



Source: Wikimedia Commons

DNS resolution can be done *iteratively*. In this case, we start by resolving a domain's ________ ______________________, then the second-level domain, etc.

Recursive lookup


"Where's www.wikipedia.org?"
"Where's www.wikipedia.org?"
"Where's www.wikipedia.org?"
DNS Resolver
Local nameserver
ISP nameserver
root nameserver
org. nameserver
wikipedia.org. nameserver

UDP and trust

Most of the time, however, we don't want to do all of those round trips ourselves. Rather, we ask questions of a _________________________, which forwards queries to other servers and memoizes their results.

- **Q:** how does UDP work (vs TCP)?
- **Q:** what trust assumptions are made?
- **Q:** how could you violate those assumptions?
- **A:** connectionless protocol... can just send data!

Spoofing DNS

What could you do if you could trick DNS resolvers?

What information is needed to trick DNS resolvers?

- need: **what** query was issued + **when** + from **whence** it came
- … unless the nameserver *is* the attacker (more on this later)

Impractical to fool a resolver directly, but…

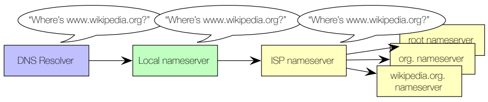
If you could trick DNS resolvers, you could ______________________________. If everyone were using always-on TLS all the time, making pessimistic assumptions about Dolev-Yao attackers in the network, that wouldn't be so bad. However, there is a lot of HTTP going on out there!

If you ask the DNS server a question, it will recursively attempt to find the answer and cache it. If you can know when a request is made and send a fake answer to the server ______________ ______________, you can trick the resolver.

Fooling DNS resolvers is tricky: you need to know __________ a query is issued, __________ the query is and ________________ it came (e.g., port number).

DNS cache poisoning

(a.k.a., DNS spoofing)


"Where's www.wikipedia.org?"
"Where's www.wikipedia.org?"
"Where's www.wikipedia.org?"
DNS Resolver
Local nameserver
ISP nameserver
root nameserver
org. nameserver
wikipedia.org. nameserver

Cause DNS server to make request(s)

Send spoofed "responses"

One thing an attacker *can* do is *cache poisoning*. Maybe you can't know when a request will be made by your intended victim, but what if you could pre-load your attack data on the DNS resolver?

To carry out this attack, you need to send queries to the DNS resolver for ________________ ________________________. One easy way to do this is to make requests for domains that don't exist, e.g., `b37deac230.mun.ca`.

Then, the attacker can send "responses" to the server saying, "You were asking about `b37deac230.mun.ca`? OK, well `mun.ca` says that you should use this ________________ ______________ over here to resolve its names."

How hard is this?

Difficulties:

- guessing TXID (16b)
- guessing port (16b)

Even though the attacker controls the content and timing of the request to the cacheing DNS resolver, they still have to guess some things in order to make the attack successful.

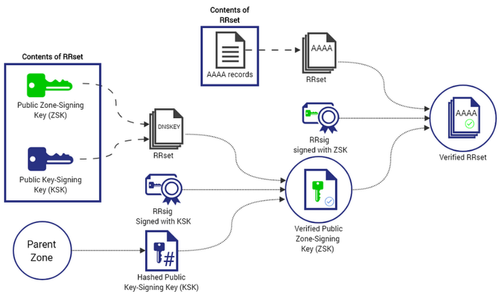
When a DNS resolver queries its upstream resolver, it generates a transaction ID that the upstream resolver will use to send back a response. If that ID were just a serial number, an attacker could have an easy time of guessing it. If it's a random value (as it should be), given that it's a 16b number, we would expect the attacker to have to send 2^{15} responses with random transaction IDs in order to fool the resolver. However, it's a bit easier than that due to the ____________ ____________: just $2^{\frac{16}{2}} = 2^8 = 256$ transactions are required when we can cause lots of queries to be made.

Things are a bit harder for the attacker now that the source port for the resolver's request isn't always port 53 any more. Still, adding another 16b random number as the source address means that the attacker should expect to succeed after $2^{\frac{16+16}{2}} = 2^1 6 = 65536$ tries. This isn't at all an inconceivable number of transactions to try.

Preventing DNS spoofing

DNSSEC

- only accept signed responses
- **Q:** signed by what?
- **A:** signed by a public key?
- **Q:** why do we trust that public key?



Source: iocscan.io

This is a critical question to ask if we're going to avoid security snake oil. Just as we saw with code signing, having a digital signature by itself proves nothing. What matters is whether we can link that digital signature to ______________________________ like an ________________.

DNSSEC chain of trust

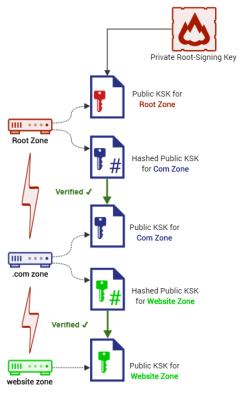
Root signing key

Root KSK

Root signs .com KSK

.com signs foo.com KSK, etc.

Neat visualization:
http://dnsviz.net/d/verisign.com/dnssec



Source: iocscan.io

What does this all look like? Kinda like a chain of trust to a ______________________! But we know that those aren't always super-great, so... how do we know that we can trust these root signatures?

DNSSEC root of trust

But how can we trust the root signatures?

Partial answer: security *ceremonies*

- Generalization of security *protocols*
- Anyone can volunteer to be an external witness
- You can watch @ https://iana.org/dnssec/ceremonies
  April 26th: opening the safe @ 0:14:02, opening HSM @ 0:40:00,
  KSK @ 4:08:15, locking up safes @ 7:52:16, ...

Security protocols don't truly involve humans named Alice and Bob: they involve Alice and Bob's __________________. A security ceremony is a protocol that involves people, with specific actions that people have to take as part of the ceremony.

When watching the ceremony online, you'll see things happen like people holding up labels in front of cameras and reading off serial numbers before opening sealed bags containing keys, etc. It's slightly surreal, but kind of cool. There are a couple of readable writeups about these ceremonies here:

https://www.cloudflare.com/en-ca/dns/dnssec/root-signing-ceremony

https://blog.apnic.net/2021/10/12/dns-security-and-key-ceremonies

DNS hijacking

The adversary *is* the legitimate nameserver (hunh?)

- captive portals
- ad-greedy ISPs

Q: security of *whose interests*?

This is another example of security technologies needing to be designed to protect someone's interests, and the *someone* depending on who creates the technology! Given that DNSSEC allows end users to validate information that a DNS provides them, it gives users more power at the expense of intermediaries like ISPs. Do you think of that as a good thing, a bad thing or a neutral thing?

Summary

DNS assumptions

DNS cache poisoning

DNSSEC

DNS hijacking