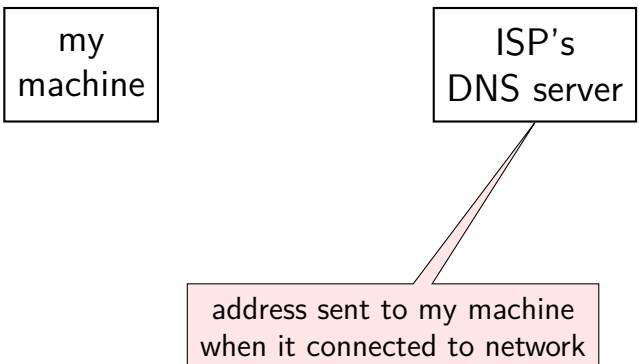
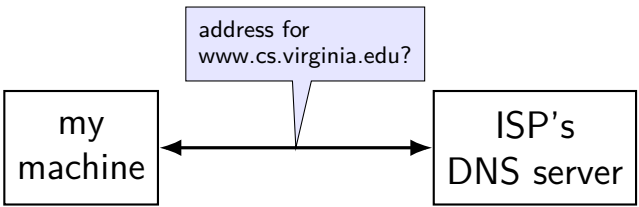
names and addresses

name	address
logical identifier	location/how to locate
DNS name `www.virginia.edu`	IPv4 address `128.143.22.36`
DNS name `mail.google.com`	IPv4 address `216.58.217.69`
DNS name `mail.google.com`	IPv6 address `2607:f8b0:4004:80b::2005`
DNS name `reiss-t3620.cs.virginia.edu`	IPv4 address `128.143.67.91`

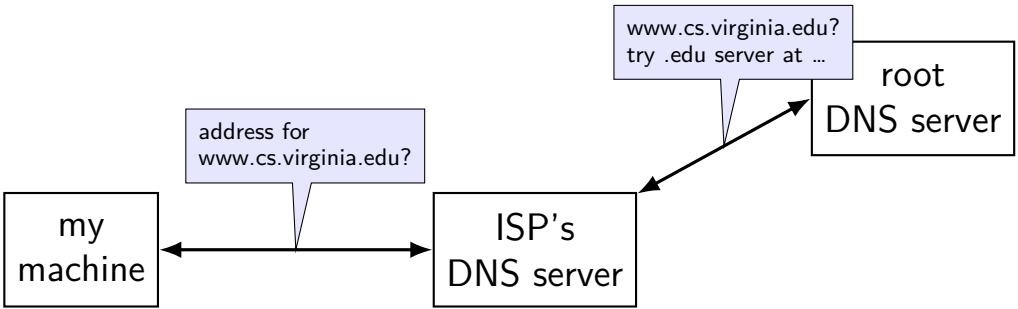
DNS: distributed database



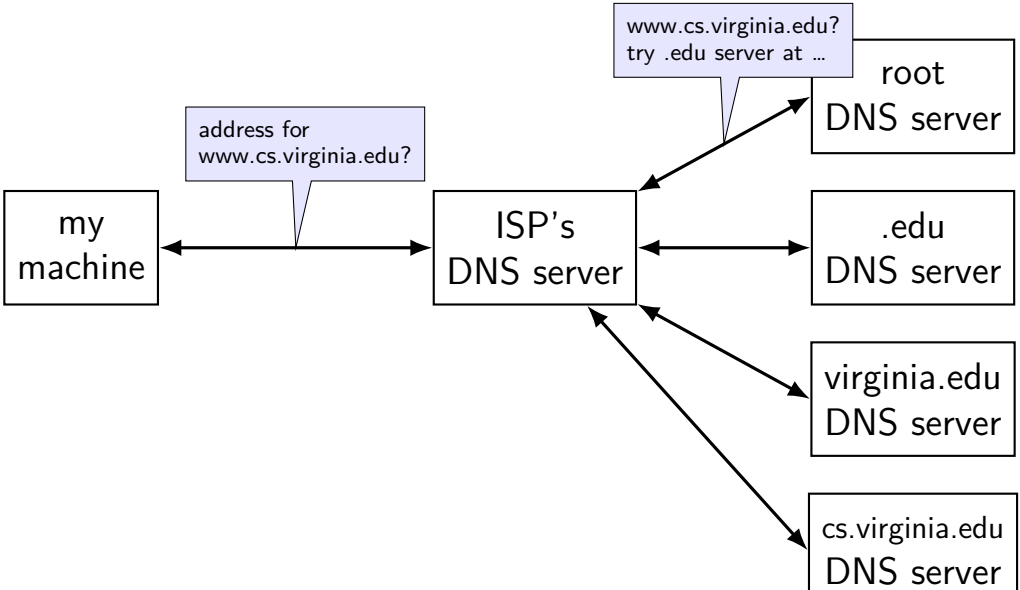
DNS: distributed database



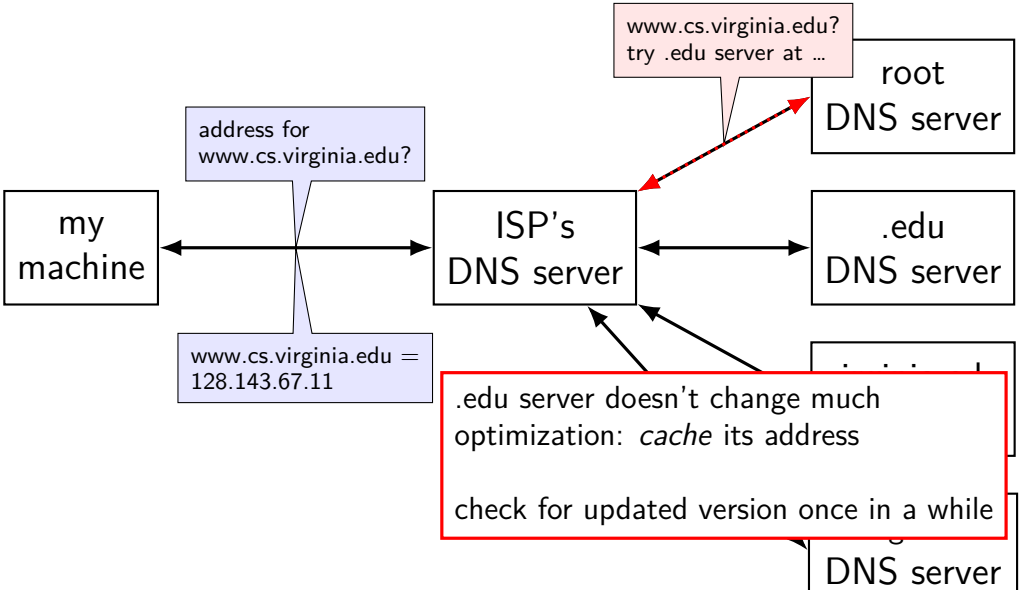
DNS: distributed database



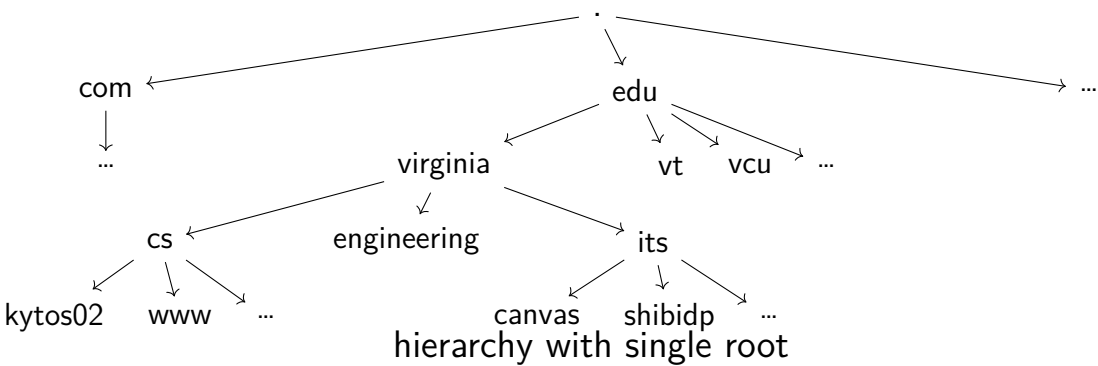
DNS: distributed database



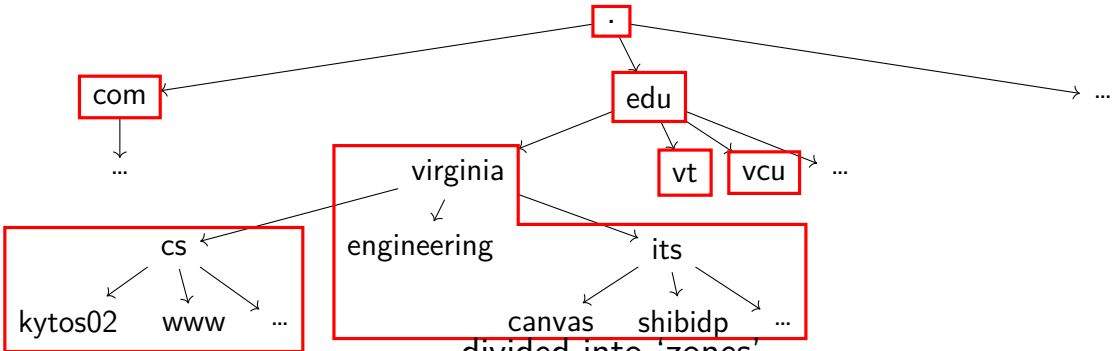
DNS: distributed database



DNS hierarchy



DNS hierarchy



divided into ‘zones’

each with own set of authoritative servers

terms: authority, recursive

DNS server is *authoritative* for X.Y.Z?

> (claims to be) official source of information for X.Y.Z
> not giving a cached version obtained from elsewhere

DNS server is a *recursive resolver*?

> will contact other DNS servers to get answer
> typically caches everything time-to-live allows
> type of DNS server one gets from DHCP, etc.

choice of whoever configures DNS server

usually (but not always) mutually exclusive

querying the root

```
$ dig +trace +all www.cs.virginia.edu
...
edu.                    172800      IN      NS      b.edu-servers.net.
edu.                    172800      IN      NS      f.edu-servers.net.
edu.                    172800      IN      NS      i.edu-servers.net.
edu.                    172800      IN      NS      a.edu-servers.net.
...
b.edu-servers.net.      172800      IN      A       191.33.14.30
b.edu-servers.net.      172800      IN      AAAA       2001:503:231d::2:30
f.edu-servers.net.      172800      IN      A       192.35.51.30
f.edu-servers.net.      172800      IN      AAAA       2001:503:d414::30
...
;; Received 843 bytes from 198.97.190.53#53(h.root-servers.net) in 8 ms
...
```

querying the edu

```
$ dig +trace +all www.cs.virginia.edu
...
virginia.edu.                 172800        IN        NS        nom.virginia.edu.
virginia.edu.                 172800        IN        NS        uvaarpa.virginia.edu.
virginia.edu.                 172800        IN        NS        eip-01-aws.net.virginia.edu.
nom.virginia.edu.        172800        IN        A        128.143.107.101
uvaarpa.virginia.edu.         172800        IN        A        128.143.107.117
eip-01-aws.net.virginia.edu. 172800 IN        A        44.234.207.10
;; Received 165 bytes from 192.26.92.30#53(c.edu-servers.net) in 40 ms
...
```

querying virginia.edu+cs.virginia.edu

```
$ dig +trace +all www.cs.virginia.edu
...
cs.virginia.edu.        3600        IN        NS        coresrv01.cs.virginia.edu.
coresrv01.cs.virginia.edu. 3600        IN        A        128.143.67.11
;; Received 116 bytes from 44.234.207.10#53(eip-01-aws.net.virginia.edu) in 72 ms

www.cs.Virginia.EDU.        172800        IN        A        128.143.67.11
cs.Virginia.EDU.        172800        IN        NS        coresrv01.cs.Virginia.EDU.
coresrv01.cs.Virginia.EDU. 172800 IN        A        128.143.67.11
;; Received 151 bytes from 128.143.67.11#53(coresrv01.cs.virginia.edu) in 4 ms
```

querying typical ISP's resolver

```
$ dig www.cs.virginia.edu
...
;; ANSWER SECTION:
www.cs.Virginia.EDU.          7183        IN        A        128.143.67.11
..
```

DNS presentation format

DNS uses compact binary format we'll talk about later

also has standardized 'presentation format' that `dig` uses (roughly)

format is used for DNS server configuration files

DNS time-to-live

need some way to remove out-of-date DNS records

DNS solution: time-to-live

number of seconds record is valid for

typically set to fixed value by authoritative server

- configured by system administrator

decremented when returned from cache by recursive resolvers

DNS exercise (1)

"www.cs.virginia.edu is 128.148.67.11 for next 86400 seconds"

(given record above) if sysadmin changes IP address DNS server returns for www.cs.virginia.edu, then what will happen to machines accessing website?

A. they'll start using the new address after 86400 seconds, and use the old one before then.

B. different machines will use the new address at different times, but no longer than 86400 seconds from when it changes

C. machines will start using the new address almost immediately, but after some small delay after it is changed

D. machines may keep using the old address until they are rebooted

E. something else?

DNS exercise (2)

if sysadmin wants to change the IP address of www.cs.virginia.edu, how do they do this without downtime?

they can change the IP address the server returns and/or the time-to-live?

what should they change and when to smoothly transition to a new address?

DNS exercise (3)

suppose initially

*.foo.com DNS server ('nameserver') = 10.2.3.4, valid 200 s

www.foo.com = 10.1.2.3, valid 100 s

if at time 0 seconds, changed to:

*.foo.com DNS server = 10.3.4.5, valid 100 s

www.foo.com DNS server = 10.3.5.1, valid 400 s

ex 0: when will new DNS server/www.foo.com start being used?

ex 1: when can we shut down old DNS server?

ex 2: when can we shut down old www.foo.com?

root servers (1)

(from IANA's website)

List of Root Servers

HOSTNAME	IP ADDRESSES	OPERATOR
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	170.247.170.2, 2801:1b8:10::b	University of Southern California, Information Sciences Institute
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

root servers (2)

(from root-servers.org)

anycast root servers

many root servers have multiple separate sites

...but one IPv4/IPv6 address

which one you get = which one routed to

multiple sites with BGP announcements for IP

- often with care taken to limit how far route goes

idea called 'anycast'

- get to 'any' of several servers

also used for some public recursive DNS servers

- 1.1.1.1 (CloudFlare), 8.8.8.8 (Google), 208.67.222.222 (Cisco), ...

DNS registries

IANA and affiliates publish root zone file

- same organization as for IP address, AS numbers
- (but domain names are lot more politically active)

For usable top-level domains, there are *registries*:

- VeriSign (VA company) for `COM`, `NET`
- Educause (non-profit) for `EDU`
- Registry Services, LLC (under contract with the US Dept of Commerce) for `US`
- ...

many registries allow for many *registrars* who can sell new domains

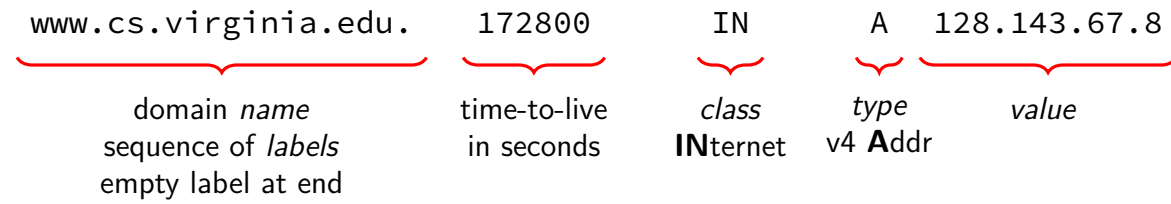
DNS resource records

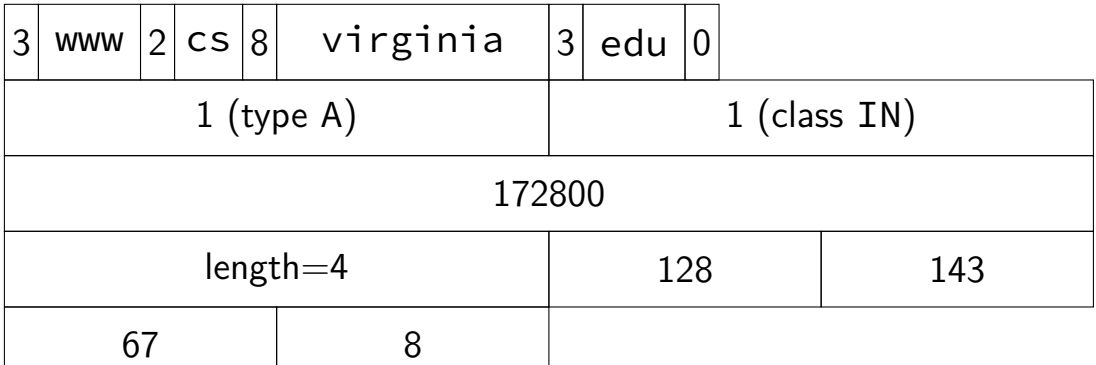
resource record (RR) = single DNS database ‘entry’

has standard text and binary representation

- text used for server config files and display
- binary representation used for network protocol

DNS record example (text)





DNS name format

zero or more: length of $K > 0$ followed by K character case-insensitive *label*

- labels limited to 64 bytes

then either:

- length of 0 to indicate end-of-name *or*
  "pointer" to earlier name in message
  - pointer encoded as `0xc000` + byte number in message (big endian)
  - upper two bits being set prevents confusion with length

selected RR types

text	id	purpose	data format
A	1	IPv4 address	32-bit integer (big-endian)
AAAA	28	IPv6 address	128-bit integer (big-endian)
NS	2	authoritative name server	domain name
CNAME	5	‘canonical name’	domain name
TXT	16	text string	arbitrary string
SRV	33	service location	priority, weight, domain name, port

CNAME

CNAME = delegate name to another name

output of `dig canvas.its.virginia.edu a`:

```
canvas.its.virginia.edu. 2711   IN      CNAME
    universityofvirginia-vanity.instructure.com.
```

canvas.its.virginia.edu = universityofvirginia.vanity.instructure.com

```
universityofvirginia-vanity.instructure.com. 238 IN CNAME \
    canvas-pdx-prod-c354-1908777142.us-west-2.elb.amazonaws.com.
```

universityofvirginia.vanity.instructure.com = canvas-pdx-prod-c354-1908777142.us-west-2.elb.amazonaws.com.

```
canvas-pdx-prod-c354-1908777142.us-west-2.elb.amazonaws.com. 2 IN A 34.208.211.157
canvas-pdx-prod-c354-1908777142.us-west-2.elb.amazonaws.com. 2 IN A 44.238.73.244
canvas-pdx-prod-c354-1908777142.us-west-2.elb.amazonaws.com. 2 IN A 52.26.9.245
```

NS

```
virginia.edu. 3600    IN      NS      uvaarpa.virginia.edu.
virginia.edu. 3600    IN      NS      nom.virginia.edu.
virginia.edu. 3600    IN      NS      eip-01-aws.net.virginia
```

these three servers are authoritative for virginia.edu

(but need their A or AAAA records to actually contact them)

SRV

```
_ldap._tcp.virginia.edu. 600    IN      SRV     0 100 389 vadc
_ldap._tcp.virginia.edu. 600    IN      SRV     0 100 389 vadc
```

virginia.edu's LDAP servers are vadcv4.virginia.edu port 389 and vadc5.virginia.edu port 389

(LDAP = Lightweight Directory Access Protocol)

example: lookup email ID by name

SRV records were late...

SRV records seem like something we'd use all the time...

but only used with a few protocols

why?

SRV records added late (1996–2000)

sending email already had its own record type (MX)

poor support for querying them in standard networking libraries

hard to get access to add them in many organizations

dig virginia.edu txt

```
virginia.edu. 3548 IN TXT "google-site-verification=zEwuk4FIG8_vtv2BZJOD6IWzg9JbNiJPH9mQdPNC
virginia.edu. 3548 IN TXT "cisco-ci-domain-verification=268b991de3589451b38fcfeaa99473f8e4fb
virginia.edu. 3548 IN TXT "miro-verification=9f3238c881466b3ccb99c4347b99e5504eafc118"
virginia.edu. 3548 IN TXT "MS=ms40126609"
virginia.edu. 3548 IN TXT "onetrust-domain-verification=d901adf09116474c89790a9752e0046c"
virginia.edu. 3548 IN TXT "google-site-verification=rePHmrM7FEfSw62DHq9TpuMFUw66J6hgpFFh5cq8
virginia.edu. 3548 IN TXT "v=spf1 a mx ip4:52.254.56.82 ip4:52.137.91.139 ip4:128.143.125.90
virginia.edu. 3548 IN TXT "docusign=6547d080-1c40-427c-8736-fafa466ff73f"
virginia.edu. 3548 IN TXT "apple-domain-verification=zllSamqSFAw4lNQo"
virginia.edu. 3548 IN TXT "v=msv1 t=537499f256ea46bd386f7543d18d28"
virginia.edu. 3548 IN TXT "atlassian-domain-verification=zeQ45p2Vl1fXekKYCzmq4ONEpDEkGKx3Y8Q
virginia.edu. 3548 IN TXT "00256316"
virginia.edu. 3548 IN TXT "yYMwnn+VY4w6aVf+cE4888ppz+MDXBKnaI0m5eteiMGgwWIBTOgA6aTSJM4QYRG6s
virginia.edu. 3548 IN TXT "apple-domain-verification=mROwKUEMfwLBfltC"
virginia.edu. 3548 IN TXT "e2ma-verification=rx8eb"
virginia.edu. 3548 IN TXT "ZOOM_verify_PuU-eQzyQ7ONrij_vTP72A"
virginia.edu. 3548 IN TXT "sending_domain1023271=f4a69653389ff4543a64e982f1cbc2e9db30eab95cc
```

TXT record usage

TXT records ‘just’ hold arbitrary strings

lots of machine-based usage of TXT records

probably in part because it’s too slow/hard to add new record types

TXT example: SPF

```
virginia.edu. 3548    IN      TXT "v=spf1 a mx
    ip4:52.254.56.82 ip4:52.137.91.139
    ip4:128.143.125.90 ip4:128.143.125.91
    include:spf.protection.outlook.com
    include:_spf.google.com
    include:spf.elluciancloud.com ~all"
```

SPF protocol for specifying who can send email from domain

why in TXT record instead of its own type?

RFC 6686, Appendix A excerpt

At the time of SPF's initial development, the prospect of getting an RRTYPE allocated for SPF was not seriously considered, partly because doing so had high barriers to entry....

Later, after RRTYPE 99 was assigned ...a plan was put into place to effect a gradual transition to using RRTYPE 99 instead of using RRTYPE 16. This plan failed to take effect for four primary reasons:...

1....existing nameservers (and, in fact, DNS-aware firewalls) would drop or reject requests for unknown RRTYPEs ...

2. many DNS provisioning tools ...were, and still are, typically lethargic about adding support for new RRTYPEs

...

primary and secondary servers

usually have multiple DNS servers for each zone

- needed to handle outages

DNS protocol supports *zone transfers* to synchronize them

- DNS query for type AXFR returns all RRs as special case
- (most public DNS servers block this from 'normal'users)

one server designed as 'primary'; others are 'secondary'

special SOA (start of authority) record provides metadata about zone

- name of primary server to contact
- serial number for current version (to quickly check if update)
- how long to wait between updates, etc.

DNS messages (1)

client sends query to server

server responds with response

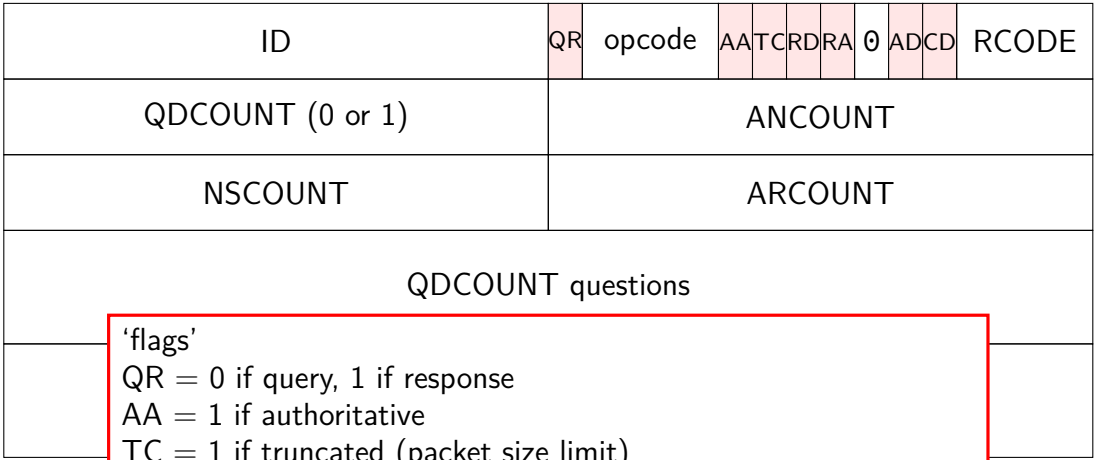
response and query have same format

- but different fields set/used

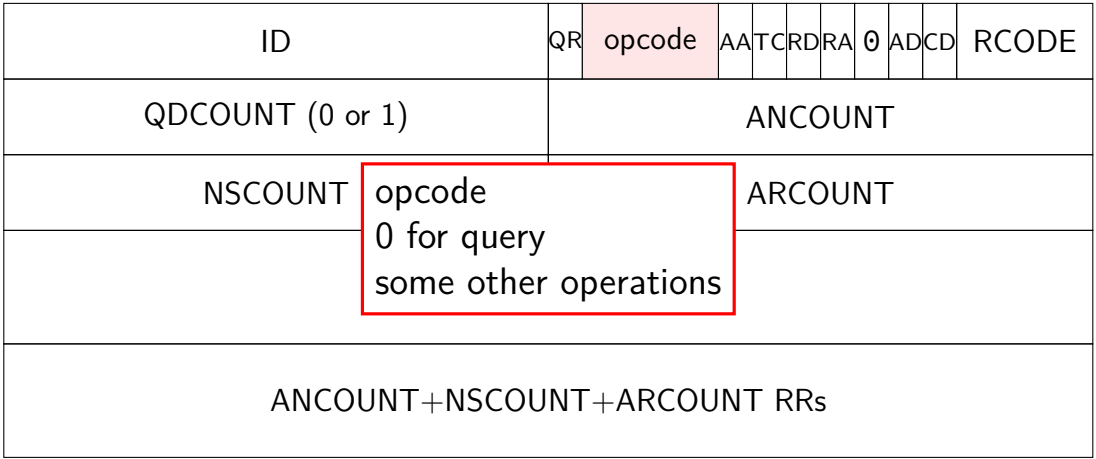
DNS header

<table>
<tr><td colspan="8">ID</td><td>QR</td><td>opcode</td><td>AA</td><td>TC</td><td>RD</td><td>RA</td><td>0</td><td>AD</td><td>CD</td><td>RCODE</td></tr>
<tr><td colspan="8">QDCOUNT (0 or 1)</td><td colspan="8">ANCOUNT</td></tr>
<tr><td colspan="8">NSCOUNT</td><td colspan="8">ARCOUNT</td></tr>
<tr><td colspan="16">QDCOUNT questions</td></tr>
<tr><td colspan="16">ANCOUNT+NSCOUNT+ARCOUNT RRs</td></tr>
</table>

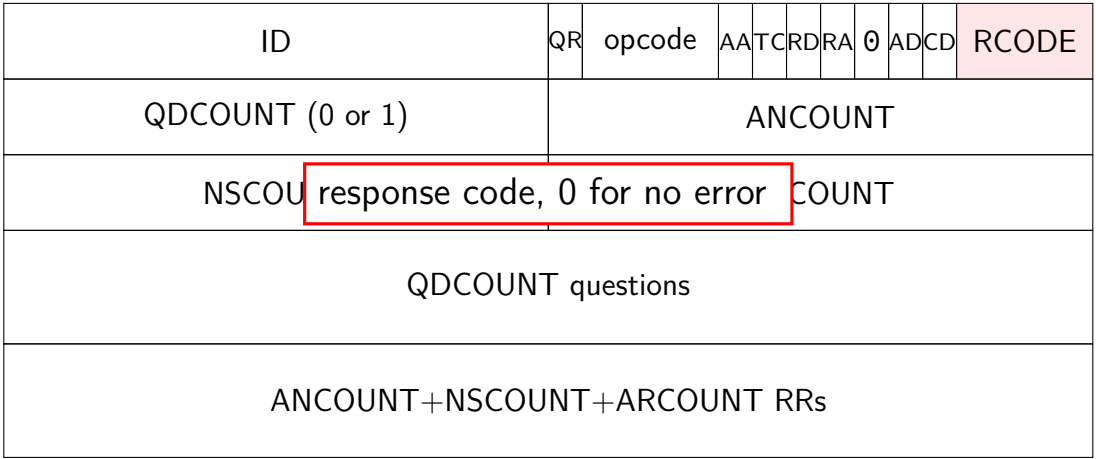
DNS header


ID
QR
opcode
AA
TC
RD
RA
0
AD
CD
RCODE
QDCOUNT (0 or 1)
ANCOUNT
NSCOUNT
ARCOUNT
QDCOUNT questions
'flags'
QR = 0 if query, 1 if response
AA = 1 if authoritative
TC = 1 if truncated (packet size limit)
RD = 1 if query wants recursion (contact other servers)
RA = 1 if server supports recursion

DNS header


ID
QR
opcode
AA
TC
RD
RA
0
AD
CD
RCODE
QDCOUNT (0 or 1)
ANCOUNT
NSCOUNT
ARCOUNT
opcode
0 for query
some other operations
ANCOUNT+NSCOUNT+ARCOUNT RRs

DNS header


ID
QR
opcode
AA
TC
RD
RA
0
AD
CD
RCODE
QDCOUNT (0 or 1)
ANCOUNT
NSCOU
response code, 0 for no error
COUNT
QDCOUNT questions
ANCOUNT+NSCOUNT+ARCOUNT RRs

caching ‘domain name does not exist’

when domain does not exist (RCODE=3, “NXDOMAIN”):

(RFC 2308)

response should include SOA record

- recall: indicates primary server for zone and how often secondaries should update

can cache NXDOMAIN based on minimum of TTL and SOA record’s update frequency

DNS message

header

0 or 1 question

- domain name + class + type
- (RR without TTL or value)
- special type values for ANY, some other things

ANCOUNT RRs in ‘answer section’:

- CNAMEs (even if that’s not the question type)
- matches for domain name (or CNAME) + class + type

NSCOUNT+ARCOUNT RRs in ‘authority section’ and ‘additional section

- not always included; usually NS or SOA records plus corresponding A/AAAA recods

DNS over ...

UDP port 53

- send DNS message
- reply with DNS message back

TCP port 53

- send length (2B, big endian) + message
- reply with length (2B, big endian) + message
- (+ repeat)

over HTTPS (RFC 8484)

- using `https://server/dns-query`

DNS UDP size limit

```
$ dig virginia.edu txt +notcp +all
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.18.28-0ubuntu0.22.04.1-Ubuntu <<>> virginia.edu txt +notcp +all
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53043
;; flags: qr rd ra; QUERY: 1, ANSWER: 17, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;virginia.edu.                          IN        TXT

;; ANSWER SECTION:
virginia.edu. 3537 IN TXT "cisco-ci-domain-verification=268b991de3589451b38fcfeaa9
....
```

DNS UDP size limit

512 byte size limit for UDP

- if exceeded, set TC (truncate) flag and truncate response
- clients supposed to retry with TCP

extensions to DNS increase limit

- (but have to be opted into by clients)

explains why only 13 root server names

- 13 NS RRs + 13 A RRs $\approx$ 480 bytes (w/ name compression)

eDNS(0) (RFC 6891)

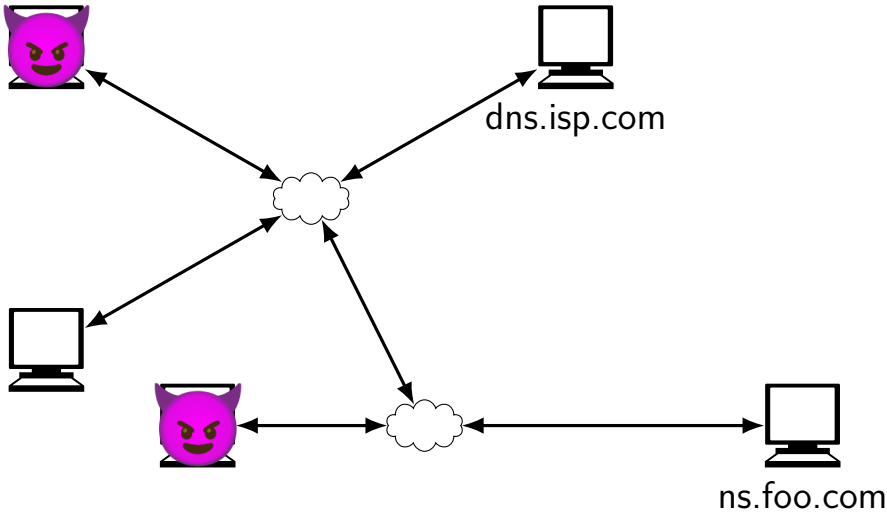
extend DNS protocol by sending psuedo-RR

- added in additoinal section of requests
- name `.` (empty)
- type OPT/44
- CLASS, TTL fields reused for other stuff

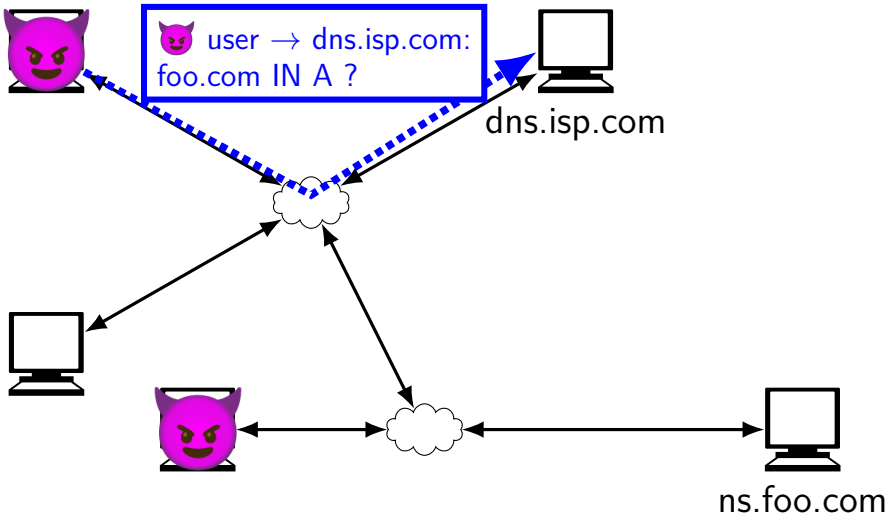
specifies:

- max size over UDP
- extra bits for RCODE
- additoinal, variable length data

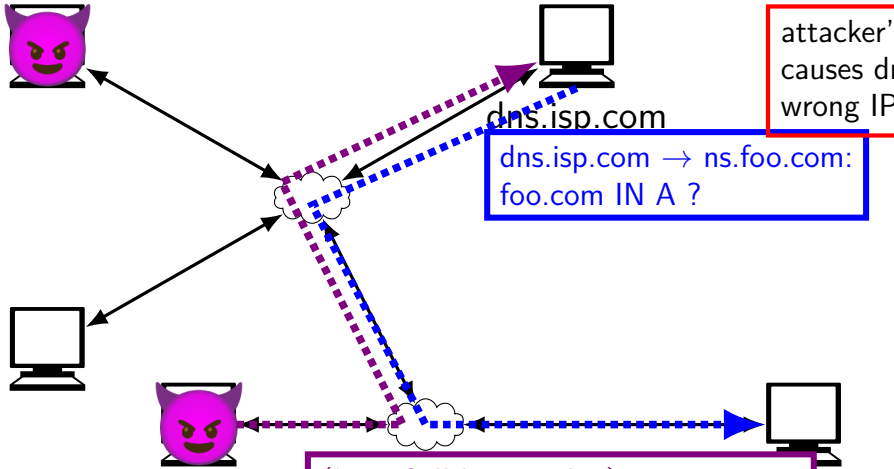
DNS cache poisoning



DNS cache poisoning

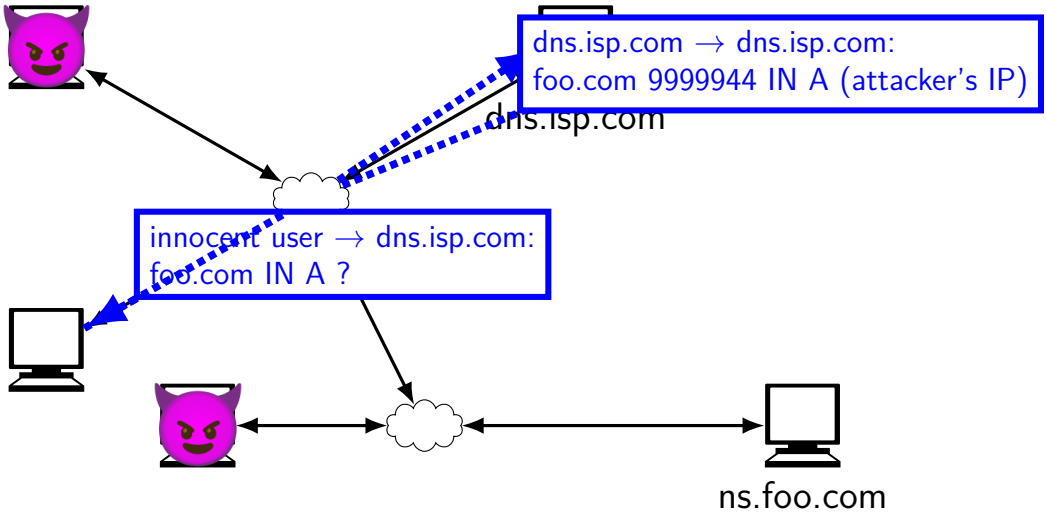


DNS cache poisoning



attacker’s ‘spoofed’ response causes dns.isp.com to record wrong IP

DNS cache poisoning



mitigating cache poisoning attacks

filter out packets with source address for where they come from?

- not feasible if real/spoofed packets forwarde through many other ISPs

use random port number for queries

- attacker can spoof many port numbers at once
- attacker can keep trying until they guess right

use random ID number in DNS query

- not good enough alone — attacker can guess often enough
- probably enough with random port?

add additional randomness to DNS query

- randomize capitalization (assuming it's returned the same in response)
- 'DNS cookie' extension (RFC 7873; assuming remote server supports it)

use TCP with random sequence number (slow)

DNSSEC

public key infrastructure for DNS

single set of root keys from ICANN/IANA

no certificate authorities like web PKI

each DNS zone has its own public key

all records within zone have a signature

delegation records have keys for subzones

digital signature

generate (public key, private key) such that

Sign(private key, message) = signature

Verify(public key, signature, message) = 1

and it's really hard to generate X s.t.
Verify(public key, X, signature) = 1
if you don't know the private key

distribute public key widely

DNSSEC offline signing

design goal: generate + sign zone

have digital signatures that can verify all records

means: DNS servers don't need to actually sign anything!

can have signing keys offline

- much lower security risk

RRSIG components

rr-type = type of resource record sign (example: NS, A, ...)
- covers ALL resource records for that type

sig-type = which digital signature algorithm

orig-ttl = original time-to-live

expiration, inception = dates; when this is considered valid

key tag + zone name = identifies which key is used
- zone name is something like 'com.' or 'example.com' or 'cs.virginia.edu.'
- key tag meant to distinguish between keys
- (but still might be multiple keys with same tag+zone name!)

signature = data from digital signature algorithm

awkwardness with TTLs

signature covers range of dates

attacker can always ‘replay’ record + signature within that range

TTL doesn’t really do anything about it

RRSIG example

```
cloudflare.com.         86400   IN      NS      ns3.cloudflare.com.
cloudflare.com.         86400   IN      NS      ns4.cloudflare.com.
cloudflare.com.         86400   IN      NS      ns5.cloudflare.com.
cloudflare.com.         86400   IN      NS      ns6.cloudflare.com.
cloudflare.com.         86400   IN      NS      ns7.cloudflare.com.
cloudflare.com.         86400   IN      RRSIG   NS 13 2 86400 \
    20241025022114 20241023002114 34505 cloudflare.com. \
    VtBeT5L8cznPZmXB81txqhj1SBs94CnI7ocA2cVsU7j3lChMYnpITUfNetWYTbu8go5OtKjL5HZG7r
```

RRSIG verifies all the NS records

from 2024-10-23 02:21:14 UTC to 2024-10-25 02:21:14 UTC

key tag 34505 for cloudflare.com.

VtBe...is digital signature data

pass to signature verification function with all the NS records to validate

DNSKEY / DS

DNSKEY records hold public keys

- gives zone key is intended for
- doesn't tell you the key is actually good

DS records delegate from key to another

- each DNSKEY needs corresponding DS record
- DS record contains hash of DNSKEY + related info

DS records signed using RRSIG records

- combination of DS + DNSKEY tells you key is good

multiple DNSKEYs

can/usually do have multiple keys per zone

typically “Key-Signing Key” (KSK) + “Zone-Signing Key” (ZSK)

goal: if ZSK is compromised, replace it

keep KSK protected much more heavily than ZS

DNSKEY/DS signing

```
;; records maintained by com. server:
cloudflare.com. 86400 IN DS     2371 13 2 329968....F6D6 3826F2B9
cloudflare.com. 86400 IN RRSIG  DS 13 2 86400 20241030011127 20241023000127 29942
cloudflare.com. 3600  IN DNSKEY 257 3 13 mdss...kHAeF+ KkxL...KGQ==

;; records maintained by cloudflare.com. servers:
cloudflare.com. 3600  IN DNSKEY 256 3 13 oJMRES...5ar0IRd8 KqXXF...hSA==
cloudflare.com. 3600  IN RRSIG DNSKEY ...
```

DS record signed by com. keyid 29942

'flags' of 257 = key-signing key, 256 = zone-signing key

not the key tag, that's derived from key itself

RRSIG verifies that zone-signing key is endorsed from key-signing key

DNSSEC root key

root key-signing key

- key material split between air-gapped safe and...
- designated ‘crpytographic officers’ (3 of 7 needed to do signing)
- cryptographic officers have smart card with some key material

- designated ‘recovery key share holders’ (5 of 7 can reconstruct keys if disaster)
- semi-public ‘key signing ceremonies’

periodically (approx 4x/year) new root zone-signing keys

DANE/TLSA (RFC 7671)

DANE — mechanism for authenticating websites/email servers/etc. with DNSSEC

not supported by any browser I know of

instead, authenticate websites [mostly] separate from DNS

DNSSEC and missing records

'easy' idea: sign "`randomjunk.cs.virginia. IN DOESNOTEXIST`"

problems:

violates model of signing everything 'offline'

potential way to overload server with lots of requests

do we really want to cache `randomjunk.cs.virginia.edu` does not exist?

...for each value of `randomjunk`?

DNSSEC and missing records

multiple options:

- signed ‘no result between W.Y.Z, type Q and Z.Y.Z, type A’ message (NSEC)
- signed ‘no result with hash(?) = A < hash(X) < B = hash(?)’ message (NSEC3)

can be generated in advance (with signing keys kept ‘offline’)

- can also generate dynamically to reveal less informationx

NSEC (‘next secure’)

```
$ dig +trace +dnssec weird.invalid
...
intuit. 86400   IN NSEC  investments. NS DS RRSIG NSEC
intuit. 86400   IN RRSIG NSEC ...
```

there are no NS, DS, RRSIG, NSEC recods between ‘intuit.’ and ‘investments.’

NSEC3

```
$ dig +trace +dnssec foo.example.com a
...
0qn0igs6chbcq47kevankt96i9obe5he.example.com. 3600 IN NSEC3 \
    1 0 5 A4196F45E2097176 DCKKHGFRAJB05JCM258PTCEHOVGMIPAN \
    A NS SOA MX TXT AAAA RRSIG DNSKEY NSEC3PARAM
0qn0igs6chbcq47kevankt96i9obe5he.example.com. 3600 IN RRSIG NSEC3 ...
```

A4196...is a ‘salt’ to make hash unique

- defense against ‘rainbow tables’

0qn0...and DCKK...hashes of names on either side of ‘foo’

- record proves: no names in between exist

DNSSEC deployment: validation

queries supporting validation: approx. 35%

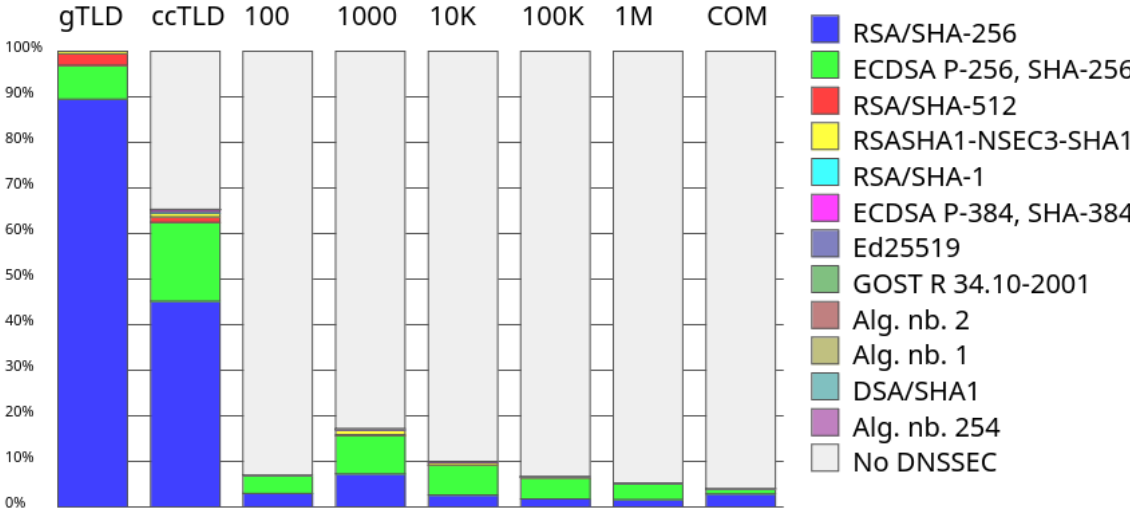
from `https://stats.labs.apnic.net/dnssec/`

approx. 45% recursive resolvers support

from `https://ithi.research.icann.org/`

DNSSEC deployment: signing

via `https://ithi.research.icann.org/graph-m11.html`:



reverse DNS (IPv4)

what's a domain name for IP 128.143.107.101?

special domain name: `101.107.143.128.in-addr.arpa`

...and PTR record type for this:

```
$ dig -x 128.143.107.101
...
101.107.143.128.in-addr.arpa. 3516 IN   PTR     eip-04-udc.net.virginia.edu.
...
$ dig eip-04-udc.net.virginia.edu a
...
eip-04-udc.net.virginia.edu. 3600 IN    A       128.143.107.101
...
```

might not be only name:

```
$ dig nom.virginia.edu a
...
nom.virginia.edu.       86400   IN      A       128.143.107.101
...
```

reverse DNS (IPv6)

```
$ dig -x 2607:f8b0:4004:c1d::65
...
5.6.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.1.c.0.4.0.0.4.0.b.8.f.7.0.6.2.ip6.arpa.
    3345 IN PTR ww-in-f101.1e100.net.
...
```

internationalized domain names

`https://日本レジストリサービス.jp/`

how does this work?

becomes:
`https://xn--vckfdb7e3c7hma3m9657c16c.jp/`

encoding scheme called *punycode*

IDN homograph attacks

bankofamerica.com

xn--bnkofamerica-x9j.com

a = U+0430 = CYRLLIC SMALL LETTER A

defenses against homograph attacks

at registries, restrict domain registration

- disallow mixed scripts (e.g. latin and cyllric)
- test if looks identical to registered domains

at browsers, restrict display in non-`xn--...` form

- allow-list for 'good' top-level domains (e.g. `.gr`, `.jp`, etc.)
- otherwise, only allow known non-confusing combinations

domain name system blocklists

historically common non-domain-name use of DNS

- I'm including to illustrate idea of 'other' DNS usage

database identifying (typically) IP addresses for filtering

used for spam/bot detection/prevention

- these days tend to expect payment for serious use

typical idea similar to reverse IP lookups

- use specific IP address to mark 'in' or 'not in' list

domain name system blocklists

example: Team Cymru's Bogon list identifies invalid addresses

```
# 248.1.1.1 is on blocklist
$ dig 1.1.1.248.v4.fullbogons.cymru.com a
...
1.1.1.248.bogons.cymru.com. 21600 IN    A       127.0.0.2
...

# 128.143.67.31 is not on blocklist
$ dig 31.67.143.128.v4.fullbogons.cymru.com a
...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 6153
...
```

backup slides