Domain Name System

- A giant distributed database that maps domain names to other information
  - Domain name = sequence of labels, separated by dots
    protocols.netlab.uky.edu.
    219.140.163.128.in-addr.arpa.
  - FQDN = Fully-qualified domain name (ends in “.” = “root”)
  - Other information = addresses, other domain names, other text
DNS’s three components

- Tree-structured namespace
  - Zones of authority define who is allowed to extend the tree
- **DNS servers** connected to the public Internet
  - Each zone must provide at least two “authoritative” name servers
  - Each server stores data for one or more zones
- Database
  - Logically: sequence of (FQDN, info) pairs
  - Stored as **Resource Records**
Resource Records (RRs)

- The protocol standard defines **details of representation** of the information in the RR's (i.e. how it is stored in the database)!
  - This ensures it is simple to go back and forth between database and wire format
- Example: an “A” RR has 32-bit IP address in network byte order

| ozark.netlab.uky.edu | A   | IN | TTL | 128 | 163 | 140 | 219 |
Domain Name Space

"Cut" = root of a zone of authority

"Zones of Authority" = subtrees controlled by different entities
Server Hierarchy

Root DNS Servers

com DNS servers
- yahoo.com DNS servers
- amazon.com DNS servers

org DNS servers
- pbs.org DNS servers

edu DNS servers
- poly.edu DNS servers
- uky.edu DNS servers

**Client wants IP for www.amazon.com (1st approximation):**

- Client queries a root server to find com DNS server
- Client queries com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com
DNS: Root name servers

13 root name servers worldwide

- a Verisign, Dulles, VA
- b USC-ISI Marina del Rey, CA
- c Cogent, Herndon, VA (also Los Angeles)
- d U Maryland College Park, MD
- e NASA Mt View, CA
- f Internet Software C. Palo Alto, CA (and 17 other locations)
- g US DoD Vienna, VA
- h ARL Aberdeen, MD
- i Autonomica, Stockholm (plus 3 other locations)
- j Verisign, (11 locations)
- k RIPE London (also Amsterdam, Frankfurt)
- l ICANN Los Angeles, CA
- m WIDE Tokyo
General DNS Message Format

<table>
<thead>
<tr>
<th>Query ID</th>
<th>Flags etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td># Questions</td>
<td># Answer RRs</td>
</tr>
<tr>
<td># Authority RRs</td>
<td># Add'l Info RRs</td>
</tr>
<tr>
<td>Questions</td>
<td></td>
</tr>
<tr>
<td>Answers</td>
<td></td>
</tr>
<tr>
<td>Authorities</td>
<td></td>
</tr>
<tr>
<td>(NS RRs for above Answers)</td>
<td></td>
</tr>
<tr>
<td>Additional Info</td>
<td></td>
</tr>
<tr>
<td>(A records for Nameserver RRs)</td>
<td></td>
</tr>
</tbody>
</table>

Query/Answer flag
Opcode
Authoritative Answer flag
Recursion Desired flag
Recursion available flag
Return code
Resolving Name to Address

- Goal: find IP address (v4) for ozark.netlab.uky.edu

```
oxly> ping ozark.netlab.uky.edu
PING ozark.netlab.uky.edu (128.163.140.219) 56 data bytes
64 bytes from 128.163.140.219: icmp_seq=0 ttl=61 time=2.097 ms
64 bytes from 128.163.140.219: icmp_seq=1 ttl=61 time=0.948 ms
64 bytes from 128.163.140.219: icmp_seq=2 ttl=61 time=0.945 ms
^C
--- ozark.netlab.uky.edu ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.945/1.330/2.097/0.542 ms
oxly>
```
How it works

ozark.netlab.uky.edu

My ISP’s local nameserver

Greater Internet

Root Name Server

EDU Name Server

UKY.EDU Name Server

NETLAB.UKY.EDU Name Server

My PC
How it works

My ISP’s local nameserver

My PC

ozark.netlab.uky.edu A?

ozark.netlab.uky.edu

Greater Internet

Root Name Server

EDU Name Server

UKY.EDU Name Server

NETLAB.UKY.EDU Name Server
How it works

My PC

ozark.netlab.uky.edu

My ISP's local nameserver

Root Name Server

EDU Name Server

UKY.EDU Name Server

NETLAB.UKY.EDU Name Server

Greater Internet

"ask the EDU Name Server at 11.24.32.11"

ozark.netlab.uky.edu A?
How it works

ozark.netlab.uky.edu

My ISP’s local nameserver

My PC

Root Name Server

EDU Name Server

UKY.EDU Name Server

NETLAB.EDU Name Server

ozark.netlab.uky.edu A?

“ask the UKY. EDU NS at 128.163.1.2”
How it works

My ISP's local nameserver

ozark.netlab.uky.edu

My PC

Greater Internet

Root Name Server

EDU Name Server

UKY.EDU Name Server

NETLAB.UKY.EDU Name Server

“ask the NETLAB.UKY. EDU NS at 128.163.140.42”

ozark.netlab.uky.edu A?
How it works

ozark.netlab.uky.edu

My ISP’s local nameserver

Greater Internet

Root Name Server

EDU Name Server

UKY.EDU Name Server

NETLAB.EDU.EDU Name Server

My PC

ozark.netlab.uky.edu A?

OZARK.NETLAB.UKY.EDU A 128.163.140.219"
Caching

DNS servers (like my ISP's local server) cache results for efficiency

- In the example, cached responses could be used to answer other queries about:
  - EDU
  - UKY.EDU
  - NETLAB.UKY.EDU
- Resource Records remain in the cache for a number of seconds specified in the RR
Cache Poisoning Attack

- Idea: get bogus information into the cache ➔ impersonate a target host
  - E.g. import a record: “www.fsnb.com A 128.163.140.219”
- How to do it?
  - In the “good [bad] old days”, some implementations would simply add the content of any response to their cache – whether they had a pending query or not!
  - Today, responses are checked for validity:
    - Sent to port from which the query was sent
    - Question matches the original query exactly
    - Query ID matches the original (16-bit number)
Cache Poisoning

Victim nameserver (cache to be poisoned)

www.fsnb.com A?

Bad guy’s Systems

Greater Internet

fake www.fsnb.com 192.168.1.1

Root Name Server

10.23.1.5

fsnb.com nameserver 10.1.200.7

www.fsnb.com 10.1.1.2
Cache Poisoning

Victim nameserver (cache to be poisoned)

Root Name Server
10.23.1.5

Greater Internet

www.fsnb.com A?

Bad guy’s Systems

fake www.fsnb.com
192.168.1.1

fsnb.com nameserver
10.1.200.7

www.fsnb.com
10.1.1.2
Cache Poisoning

Victim nameserver (cache to be poisoned)

Root Name Server
10.23.1.5

"Ask fsnb.com nameserver"

Greater Internet

fsnb.com nameserver
10.1.200.7

www.fsnb.com
10.1.1.2

Bad guy’s Systems

fake www.fsnb.com
192.168.1.1
Cache Poisoning

Victim nameserver (cache to be poisoned)

Greater Internet

Root Name Server
10.23.1.5

fsnb.com nameserver
10.1.200.7

www.fsnb.com
10.1.1.2

Bad guy’s Systems

fake www.fsnb.com
192.168.1.1

www.fsnb.com A?
Cache Poisoning

Victim nameserver (cache to be poisoned)

www.fsnb.com A 192.168.1.1

www.fsnb.com A 10.1.1.2

Source: 10.1.200.7
Port: <Query Port>
ID: 1.65535

Root Name Server

10.23.1.5

fsnb.com nameserver
10.1.200.7

www.fsnb.com
10.1.1.2

Bad guy’s Systems

thousands of bogus responses

fake www.fsnb.com 192.168.1.1
Cache Poisoning

Victim nameserver (cache to be poisoned)

www.fsnb.com A 192.168.1.1

Fake www.fsnb.com 192.168.1.1

Bad guy’s Systems

Root Name Server

10.23.1.5

fsnb.com nameserver 10.1.200.7

www.fsnb.com 10.1.1.2

Bad guy can tell if he’s gotten lucky by response to original query
“Improved” Cache Poisoning Attack

- Poison the cache with NS records \(\Rightarrow\) control an entire domain!
- Spoof the response to a query earlier in the process
  - Bad guy’s server can impersonate .COM name server, or .EDU, or google.com, or ...
  - Note: can’t spoof root servers – their addresses are built-in
Kaminsky Attack

Victim nameserver (cache to be poisoned)

www.fsnb.com A?

Bad guy’s Systems

Greater Internet

Root Name Server

10.23.1.5

fsnb.com nameserver
10.1.200.7

www.fsnb.com
10.1.1.2

fake fsnb.com nameserver
192.168.1.1
Kaminsky Attack

Victim nameserver (cache to be poisoned)

Greater Internet

Root Name Server
10.23.1.5

fsnb.com nameserver

www.fsnb.com 10.1.1.2

Bad guy’s Systems

fake fsnb.com nameserver 192.168.1.1

www.fsnb.com A?
Kaminsky Attack

Victim nameserver (cache to be poisoned)

Greater Internet

Root Name Server
10.23.1.5

fsnb.com nameserver
10.1.2.3

www.fsnb.com
10.1.1.2

Source:
10.23.1.5
Port = <Query Port>
ID = 1..65535

"Ask fsnb.com nameserver at 10.1.2.3"

"Ask fsnb.com nameserver at 192.168.1.1"

Bad guy's Systems

thousands of bogus responses
What it takes to succeed

- **Attacker must know:**
  - Which root (or TLD) server the victim will ask
  - Port number from which the victim’s request comes
    - Not difficult to find: does not typically change from query to query
  - Exact form of the question

- **Attacker must either:**
  - Predict the Query ID correctly
    - Why servers randomize their query IDs today!
  - Get lucky
    - Bogus response received before the correct one
    - Takes 64K tries to exhaust the space
    - Small chance ... but BIG payoff!
Kaminsky’s Attack Revelation

- Kaminsky alerted various Internet players before going public
  - Immediate strong concern: “We’ve got to do something!”
- Change to DNS protocol in process: expand Query ID to 32 bits
  - Increases space to ~4 billion IDs ➔ very difficult to get the right one through
  - Many (not all!) servers already patched for this change
- Servers probably take other steps (e.g., notice that lots of bogus responses are coming in, ignore all)