Network #2: DNS

(Most slides stolen from Dave Wagner)

Meme of the Day

Addressing on the Layers On The Internet

- Ethernet:
  - Address is 6B MAC address, Identifies a machine on the local LAN
- IP:
  - Address is a 4B (IPv4) or 16B (IPv6) address, Identifies a system on the Internet
- TCP/UDP:
  - Address is a 2B port number, Identifies a particular listening server/process/activity on the system
    - Both the client and server have to have a port associated with the communication
  - Ports 0-1024 are for privileged services
    - Must be root to accept incoming connections on these ports
    - Any thing can do an outbound request to such a port
  - Port 1025+ are for anybody
    - And high ports are often used ephemerally

UDP:
Datagrams on the Internet

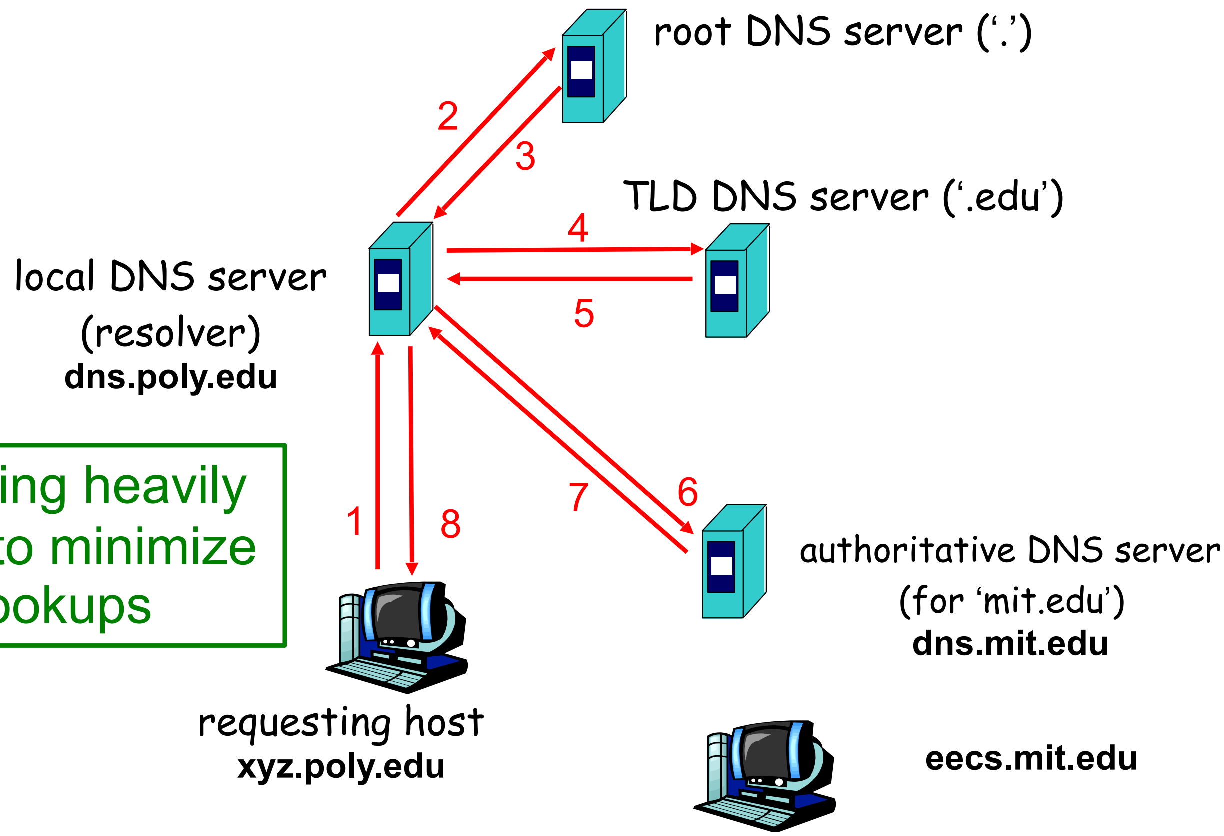
- UDP is a protocol built on the Internet Protocol (IP)
- It is an "unreliable, datagram protocol"
  - Messages may or may not be delivered, in any order
  - Messages can be larger than a single packet
    - IP will fragment these into multiple packets (mostly)
- Programs create a socket to send and receive messages
  - Just create a datagram socket for an ephemeral port
  - Bind the socket to a particular port to receive traffic on a specified port
  - Basic recipe for Python:
    https://wiki.python.org/moin/UdpCommunication

DNS Overview

- DNS translates www.google.com to 74.125.25.99
  - Turns a human abstraction into an IP address
  - Can also contain other data
- It's a performance-critical distributed database.
- DNS security is critical for the web.
  (Same-origin policy assumes DNS is secure.)
  - Analogy: If you don't know the answer to a question, ask a friend for help (who may in turn refer you to a friend of theirs, and so on).
- Based on a notion of hierarchical trust:
  - You trust . for everything, com. for any com, google.com. for everything google…

DNS Lookups via a *Resolver*

Host at **`xyz.poly.edu`** wants IP address for **`eecs.mit.edu`**



Caching heavily used to minimize lookups

Security risk #1: malicious DNS server

- Of course, if *any* of the DNS servers queried are malicious, they can lie to us and fool us about the answer to our DNS query
- (In fact, they used to be able to fool us about the answer to other queries, too. We'll come back to that.)

Security risk #2: on-path eavesdropper

- If attacker can eavesdrop on our traffic… we're hosed.
- Why? We'll see why.

Security risk #3: off-path attacker

- If attacker can't eavesdrop on our traffic, can he inject spoofed DNS responses?
- This case is especially interesting, so we'll look at it in detail.

DNS Threats

- DNS: path-critical for just about everything we do
  - Maps hostnames ⇔ IP addresses
  - Design only **scales** if we can minimize lookup traffic
    - #1 way to do so: caching
    - #2 way to do so: return not only answers to queries, but additional info that will likely be needed shortly
      - The "glue records"
- What if attacker eavesdrops on our DNS queries?
  - Then similar to DHCP, ARP, AirPwn etc, can spoof responses
- Consider attackers who *can't* eavesdrop - but still aim to manipulate us via *how the protocol functions*
- Directly interacting w/ DNS: **`dig`** program on Unix
  - Allows querying of DNS system
  - Dumps each field in DNS responses

```
dig eecs.mit.edu A
```

Use Unix “dig” utility to look up IP address (“A”) for hostname eecs.mit.edu via DNS

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

The question we asked the server

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           2160

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

A 16-bit **transaction identifier** that enables the DNS client (dig, in this case) to match up the reply with its original request

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode:
;; flags: qr rd ra; QUE                                    ONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.          21600    IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.               11088    IN      NS      BITSY.mit.edu.
mit.edu.               11088    IN      NS      W20NS.mit.edu.
mit.edu.               11088    IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.        126738   IN      A       18.71.0.151
BITSY.mit.edu.         166408   IN      A       18.72.0.3
W20NS.mit.edu.         126738   IN      A       18.70.0.160
```

"**Answer**" tells us the IP address associated with eecs.mit.edu is 18.62.1.6 and we can cache the result for 21,600 seconds

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                                                  du.
mit.edu.                                                  edu.

;; ADDITIONAL SECTION
STRAWB.mit.edu.
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

In general, a single Resource Record (RR) like this includes, left-to-right, a DNS name, a time-to-live, a family (IN for our purposes - ignore), a type (A here), and an associated value

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cm
;; Got answer:
;; ->>HEADER<<- opcode
;; flags: qr rd ra; QU                                        3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

"**Authority**" tells us the name servers responsible for the answer. Each RR gives the hostname of a different name server ("NS") for names in mit.edu. We should cache each record for 11,088 seconds.

If the "**Answer**" had been empty, then the resolver's next step would be to send the original query to one of these name servers.

dig eecs.mit.edu A

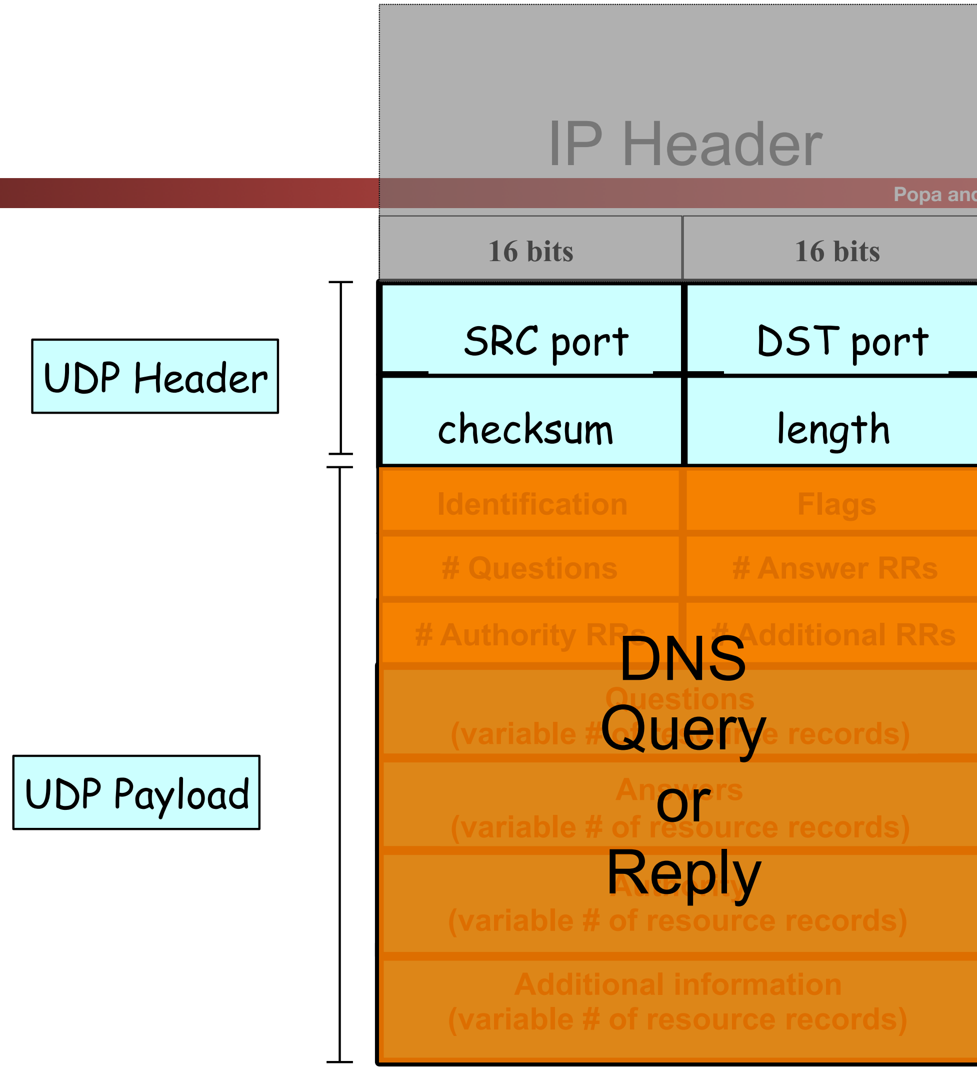
```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION
eecs.mit.edu.

;; AUTHORITY SECTI
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

"**Additional**" provides extra information to save us from making separate lookups for it, or helps with bootstrapping.

Here, it tells us the IP addresses for the hostnames of the name servers. We add these to our cache.

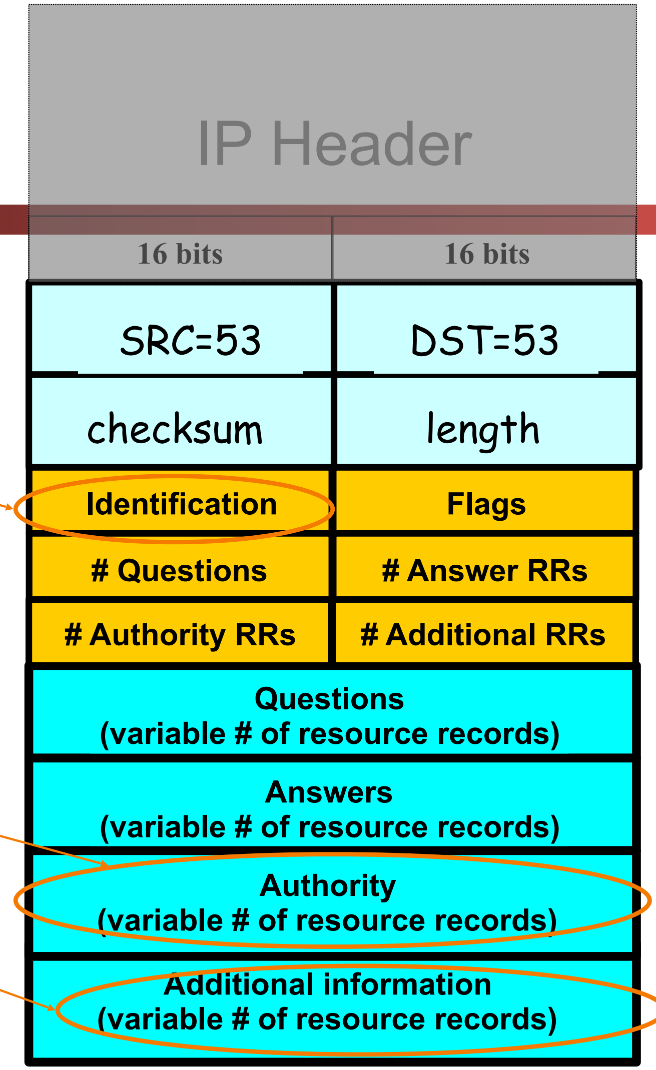
DNS Protocol

- Lightweight exchange of *query* and *reply* messages, both with same message format
- Primarily uses UDP for its transport protocol, which is what we'll assume
- Servers are on port 53 always
- Frequently, clients used to use port 53 but can use any port



Message header:

- Identification: 16 bit # for query, reply to query uses same #
- Along with repeating the Question and providing Answer(s), replies can include "**Authority**" (name server responsible for answer) and "**Additional**" (info client is likely to look up soon anyway)
- Each Resource Record has a Time To Live (in seconds) for **caching** (not shown)



dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY:                          ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.           216                                  .6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

What if the mit.edu server is untrustworthy? Could its operator steal, say, all of our web surfing to berkeley.edu's main web server?

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.            216                                              .6

;; AUTHORITY SECTION:
mit.edu.                 11088    IN      NS      BITSY.mit.edu.
mit.edu.                 11088    IN      NS      W20NS.mit.edu.
mit.edu.                 11088    IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.          126738   IN      A       18.71.0.151
BITSY.mit.edu.           166408   IN      A       18.72.0.3
W20NS.mit.edu.           126738   IN      A       18.70.0.160
```

Let's look at a flaw in the original DNS design (since fixed)

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      www.berkeley.edu.

;; ADDITIONAL SECTION:
www.berkeley.edu.       100000  IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

What could happen if the mit.edu server returns the following to us instead?

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      www.berkeley.edu.

;; ADDITIONAL SECTION:
www.berkeley.edu.       100000  IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

We'd dutifully store in our cache a mapping of www.berkeley.edu to an IP address under MIT's control. (It could have been any IP address they wanted, not just one of theirs.)

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.          2                              6

;; AUTHORITY SECTION:
mit.edu.               11088    IN      NS      BITSY.mit.edu.
mit.edu.               11088    IN      NS      W20NS.mit.edu.
mit.edu.               11088    IN      NS      www.berkeley.edu.

;; ADDITIONAL SECTION:
www.berkeley.edu.      100000   IN      A       18.6.6.6
BITSY.mit.edu.         166408   IN      A       18.72.0.3
W20NS.mit.edu.         126738   IN      A       18.70.0.160
```

In this case they chose to make the mapping last a long time. They could just as easily make it for just a couple of seconds.

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                30      IN      NS      www.berkeley.edu.

;; ADDITIONAL SECTION:
www.berkeley.edu.       30      IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

How do we fix such **cache poisoning**?

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cm
;; Got answer:
;; ->>HEADER<<- opcode
;; flags: qr rd ra; QU

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088   IN
mit.edu.                11088   IN
mit.edu.                11088   IN

;; ADDITIONAL SECTION:
www.berkeley.edu.       100000  IN
BITSY.mit.edu.          166408  IN
W20NS.mit.edu.          126738  IN
```

Don’t accept **Additional** records unless they’re for the domain we’re looking up

E.g., looking up eecs.mit.edu ⇒ only accept additional records from *.mit.edu

No extra risk in accepting these since server could return them to us directly in an **Answer** anyway.

This is called "***Bailiwick*** checking"

bail·i·wick

/ˈbāləˌwik/

noun

1. one's sphere of operations or particular area of interest.
   "you never give the presentations—that's my bailiwick"
2. LAW
   the district or jurisdiction of a bailie or bailiff.

DNS Resource Records and RRSETs

- DNS records (Resource Records) can be one of various types
    - Name TYPE Value
        - Also a “time to live” field: how long in seconds this entry can be cached for
  - Addressing:
    - A: IPv4 addresses
    - AAAA: IPv6 addresses
    - CNAME: aliases, “Name X should be name Y”
    - MX: “the mailserver for this name is Y”
  - DNS related:
    - NS: “The authority server you should contact is named Y”
    - SOA: “The operator of this domain is Y”
  - Other:
    - text records, cryptographic information, etc….
- Groups of records of the same type form RRSETs:
  - E.g. all the nameservers for a given domain.

The Many Moving Pieces In a DNS Lookup of www.isc.org

`? A www.isc.org`

User's ISP's Recursive Resolver

`? A www.isc.org`

Name	Type	Value	TTL

. Authority Server (the "root")

```
? A www.isc.org
Answers:
Authority:
org. NS a0.afilias-nst.info
Additional:
a0.afilias-nst.info A 199.19.56.1
```

The Many Moving Pieces In a DNS Lookup of www.isc.org

User's ISP's Recursive Resolver

`? A www.isc.org`

Name	Type	Value	TTL
org.	NS	a0.afilias-nst.info	172800
a0.afilias-nst.info.	A	199.19.56.1	172800

org. Authority Server

```
? A www.isc.org
Answers:
Authority:
isc.org. NS sfba.sns-pb.isc.org.
isc.org. NS ns.isc.afilias-nst.info.
Additional:
sfba.sns-pb.isc.org.      A 199.6.1.30
ns.isc.afilias-nst.info. A 199.254.63.254
```

The Many Moving Pieces
In a DNS Lookup of www.isc.org

User's ISP's Recursive Resolver **? A www.isc.org**

Name	Type	Value	TTL
org.	NS	a0.afilias-nst.info	172800
a0.afilias-nst.info.	A	199.19.56.1	172800
isc.org.	NS	sfba.sns-pb.isc.org.	86400
isc.org.	NS	ns.isc.afilias-net.info.	86400
sfbay.sns-pb.isc.org.	A	199.6.1.30	86400

isc.org. Authority Server

```
? A www.isc.org
Answers:
www.isc.org. A 149.20.64.42
Authority:
isc.org. NS sfba.sns-pb.isc.org.
isc.org. NS ns.isc.afilias-nst.info.
Additional:
sfba.sns-pb.isc.org.     A 199.6.1.30
ns.isc.afilias-nst.info. A 199.254.63.254
```

The Many Moving Pieces
In a DNS Lookup of **`www.isc.org`**

User's ISP's Recursive Resolver

`? A www.isc.org`
`Answers: www.isc.org A 149.20.64.42`

Name	Type	Value	TTL
org.	NS	a0.afilias-nst.info	172800
a0.afilias-nst.info.	A	199.19.56.1	172800
isc.org.	NS	sfba.sns-pb.isc.org.	86400
isc.org.	NS	ns.isc.afilias-net.info.	86400
sfbay.sns-pb.isc.org.	A	199.6.1.30	86400
www.isc.org	A	149.20.64.42	600

Stepping Through This With `dig`

- Some flags of note:
  - +norecurse: Ask directly like a recursive resolver does
  - +trace: Act like a recursive resolver without a cache

```
nweaver% dig +norecurse slashdot.org @a.root-servers.net

;  <<>> DiG 9.8.3-P1 <<>> +norecurse slashdot.org @a.root-servers.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26444
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 12

;; QUESTION SECTION:
;slashdot.org.                  IN      A

;; AUTHORITY SECTION:
org.                    172800  IN      NS      a0.org.afilias-nst.info.
...

;; ADDITIONAL SECTION:
a0.org.afilias-nst.info. 172800 IN      A       199.19.56.1
```

So in `dig` parlance

- So if you want to recreate the lookups conducted by the recursive resolver:
  - **`dig +norecurse www.isc.org @a.root-servers.net`**
  - **`dig +norecurse www.isc.org @199.19.56.1`**
  - **`dig +norecurse www.isc.org @199.6.1.30`**

Security risk #1: malicious DNS server

- Of course, if *any* of the DNS servers queried are malicious, they can lie to us and fool us about the answer to our DNS query…
- and they used to be able to fool us about the answer to other queries, too, using *cache poisoning*. Now fixed (phew).

Security risk #2: on-path eavesdropper

- If attacker can eavesdrop on our traffic…
  we’re hosed.
- Why?

Security risk #2: on-path eavesdropper

- If attacker can eavesdrop on our traffic… we're hosed.
- Why? They can see the query and the 16-bit transaction identifier, and race to send a spoofed response to our query.
  - China does this operationally:
  - **`dig www.benign.com @www.tsinghua.edu`**
  - **`dig www.facebook.com @www.tsinghua.edu`**

Security risk #3: off-path attacker

- If attacker can't eavesdrop on our traffic, can he inject spoofed DNS responses?
- Answer: It used to be possible, via *blind spoofing*. We've since deployed mitigations that makes this harder (but not totally impossible).

Blind spoofing

- Say we look up mail.google.com; how can an **off-path** attacker feed us a bogus A answer before the legitimate server replies?

- How can such a **remote** attacker even know we are looking up mail.google.com?

Suppose, e.g., we visit a web page under their control:

...<img src="http://mail.google.com" …> ...

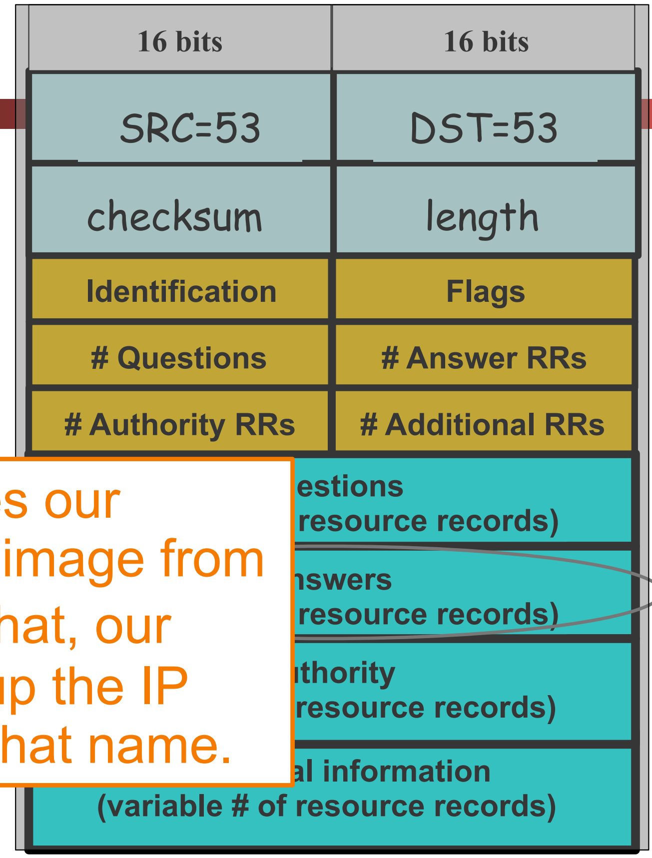
16 bits	16 bits
SRC=53	DST=53
checksum	length
Identification	Flags
# Questions	# Answer RRs
# Authority RRs	# Additional RRs
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	

Blind spoofing

- Say we look up mail.google.com; how can an **off-path** attacker feed us a bogus A answer before the legitim

- How c even l mail.g Suppose, e.g., we visit a web page under their control:

This HTML snippet causes our browser to try to fetch an image from mail.google.com. To do that, our browser first has to look up the IP address associated with that name.



...<img src="http://mail.google.com" …> ...

Blind spoofing

Fix?

16 bits	16 bits
SRC=53	DST=53
checksum	length
Identification	**Flags**
# Questions	**# Answer RRs**
# Authority RRs	**# Additional RRs**
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	

Once they know we're looking it up, they just have to guess the Identification field and reply before legit server.

How hard is that?

Originally, identification field incremented by 1 for each request. How does attacker guess it?

<img src="http://badguy.com" …> They observe ID k here

<img src="http://mail.google.com" …> So this will be k+1

DNS Blind Spoofing, cont.

Once we randomize the Identification, attacker has a 1/65536 chance of guessing it correctly.
Are we pretty much safe?

Attacker can send lots of replies, not just one …

However: once reply from legit server arrives (with correct Identification), it's **cached** and no more opportunity to poison it. Victim is innoculated!

16 bits	16 bits
SRC=53	DST=53
checksum	length
Identification	**Flags**
# Questions	**# Answer RRs**
# Authority RRs	**# Additional RRs**
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	

Unless attacker can send 1000s of replies before legit arrives, we're likely safe – phew! **?**

Enter Kaminski... Glue Attacks

- Dan Kaminski noticed something strange, however...
  - Most DNS servers would ***cache*** the in-bailiwick glue...
  - And then ***promote*** the glue
  - And will also ***update*** entries based on glue
- So if you first did this lookup...
  - And then went to a0.org.afilias-nst.info
  - there would be no other lookup!

```
nweaver% dig +norecurse slashdot.org @a.root-servers.net

;  <<>> DiG 9.8.3-P1 <<>> +norecurse slashdot.org @a.root-servers.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26444
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 12

;; QUESTION SECTION:
;slashdot.org.                  IN      A

;; AUTHORITY SECTION:
org.                    172800  IN      NS      a0.org.afilias-nst.info
...

;; ADDITIONAL SECTION:
a0.org.afilias-nst.info. 172800 IN      A       199.19.56.1
...

;; Query time: 128 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Tue Apr 16 09:48:32 2013
;; MSG SIZE  rcvd: 432
```

The Kaminski Attack In Practice

- Rather than trying to poison **`www.google.com`**...
- Instead try to poison **`a.google.com`**...
  And state that "**`www.google.com`**" is an authority
  And state that "**`www.google.com A 133.7.133.7`**"
  - If you succeed, great!
- But if you fail, just try again with b.google.com!
  - Turns "Race once per timeout" to "race until win"
- So now the attacker may still have to send lots of packets
  - In the 10s of thousands
- The attacker can keep trying until success

Defending Against Kaminski: Up the Entropy

- Also randomize the UDP source port
  - Adds 16 bits of entropy
- Observe that most DNS servers just copy the request directly
  - Rather than create a new reply
- So caMeLcase the NamE ranDomly
  - Adds only a few bits of entropy however, but it does help

Defend Against Kaminski: Validate Glue

- Don't blindly accept glue records...
  - Well, you ***have*** to accept them for the purposes of resolving a name
- But if you are going to cache the glue record...
- Either only use it for the context of a DNS lookup
  - No more promotion
- Or explicitly validate it with another fetch
- Unbound implemented this, bind did not
  - Largely a political decision: bind is heavily committed to DNSSEC (next week's topic)

Oh, and Profiting from Rogue DNS

- Suppose you take over a lot of home routers...
  - How do you make money with it?
- Simple: Change their DNS server settings
  - Make it point to yours instead of the ISPs
- Now redirect all advertising
  - And instead serve up ads for "Vimax" pills...

Extra Material

Summary of DNS Security Issues

- DNS threats highlight:
  - Attackers can attack opportunistically rather than eavesdropping
    - Cache poisoning only required victim to look up some name under attacker's control (has been fixed)
  - Attackers can often manipulate victims into vulnerable activity
    - E.g., IMG SRC in web page to force DNS lookups
  - Crucial for identifiers associated with communication to have sufficient entropy (= a lot of bits of unpredictability)
  - "Attacks only get better": threats that appears technically remote can become practical due to unforeseen cleverness
    - The introduction of glue-based poisoning turned race-once into race-until-win

Common Security Assumptions

- (Note, these tend to be pessimistic … but prudent)
- Attackers can interact with our systems without particular notice
  - *Probing* (poking at systems) may go unnoticed …
  - … even if highly repetitive, leading to crashes, and *easy to detect*
- It's easy for attackers to know general information about their targets
  - OS types, software versions, usernames, server ports, IP addresses, usual patterns of activity, administrative procedures

Common Assumptions

- Attackers can obtain access to a copy of a given system to measure and/or determine how it works
- Attackers can make energetic use of automation
  - They can often find clever ways to automate
- Attackers can pull off complicated coordination across a bunch of different elements/systems
- Attackers can bring large resources to bear if needed
  - Computation, network capacity
  - But they are *not* super-powerful (e.g., control entire ISPs)

Common Assumptions

- If it helps the attacker in some way, assume they can obtain privileges
  - But if the privilege gives everything away (attack becomes trivial), then we care about unprivileged attacks
- The ability to robustly *detect* that an attack has occurred does not replace desirability of preventing
- Infrastructure machines/systems are well protected (hard to directly take over)
  - So a vulnerability that requires infrastructure compromise is less worrisome than same vulnerability that doesn't

Common Assumptions

- Network routing is hard to alter … other than with physical access near clients (e.g., “coffeeshop”)
  - Such access helps fool clients to send to wrong place
  - Can enable *Man-in-the-Middle* (MITM) attacks
- We worry about attackers who are lucky
  - Since often automation/repetition can help “make luck”
- Just because a system does not have apparent value, it may still be a target
- Attackers are undaunted by fear of getting caught