Attacks on DNS

CS 161: Computer Security

Prof. David Wagner

March 3, 2013

Today

- Reminder: Project due tonight, 11:59pm
- Today, DNS: protocol for mapping hostnames to IP addresses, and attacks on DNS.

DNS Overview

- DNS translates www.google.com to 74.125.25.99
- 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).

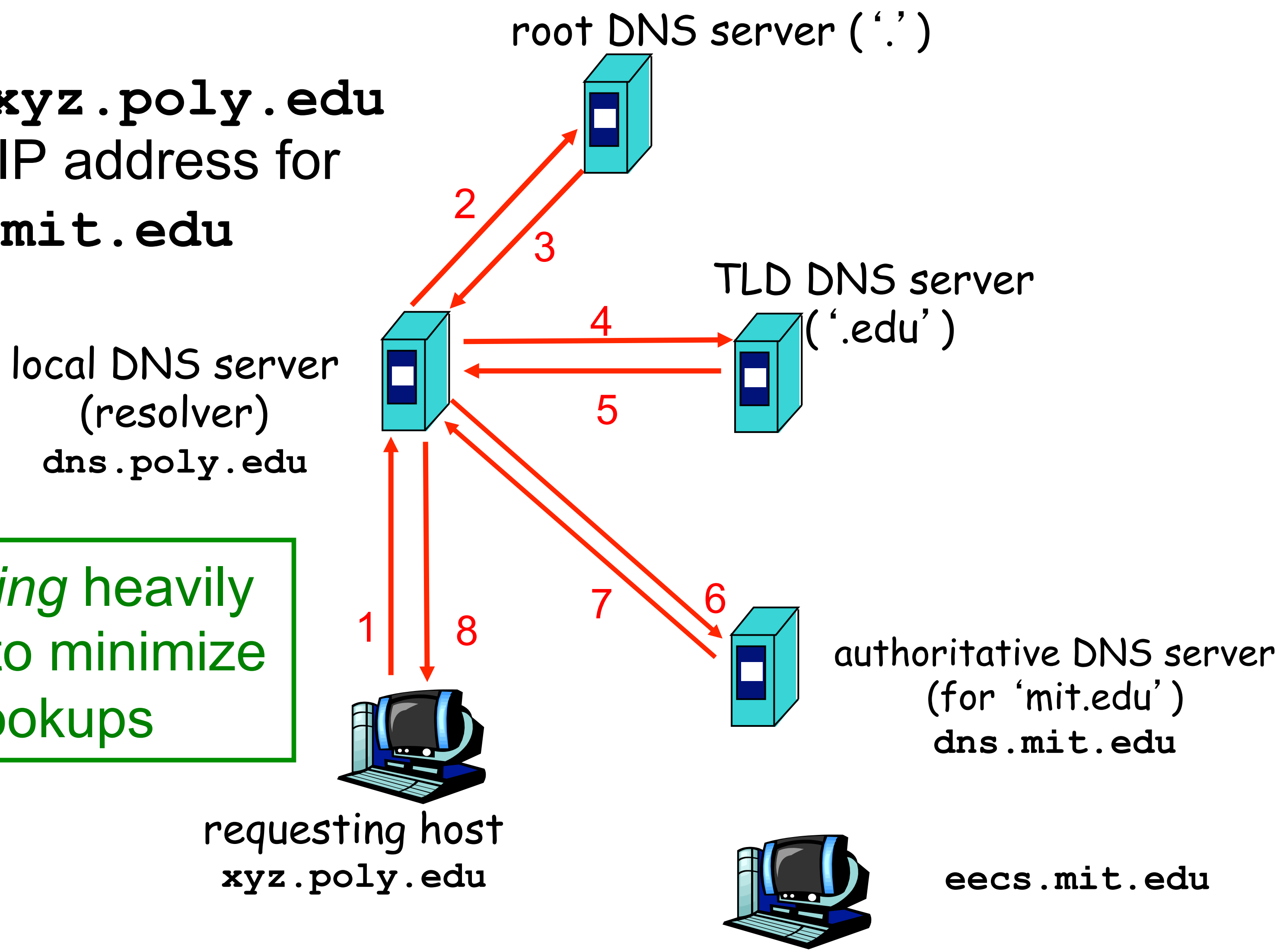
DNS Overview

- DNS translates www.google.com to 74.125.25.99
- 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).
- Security risks: friend might be malicious, communication channel to friend might be insecure, friend might be well-intentioned but misinformed

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
- What if attacker eavesdrops on our DNS queries?
  - Then similar to DHCP/TCP, 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.                                        RAWB.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; QU                                        IONAL: 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.                                                    edu.
mit.edu.                                                    edu.
mit.edu.                                                     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
```

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

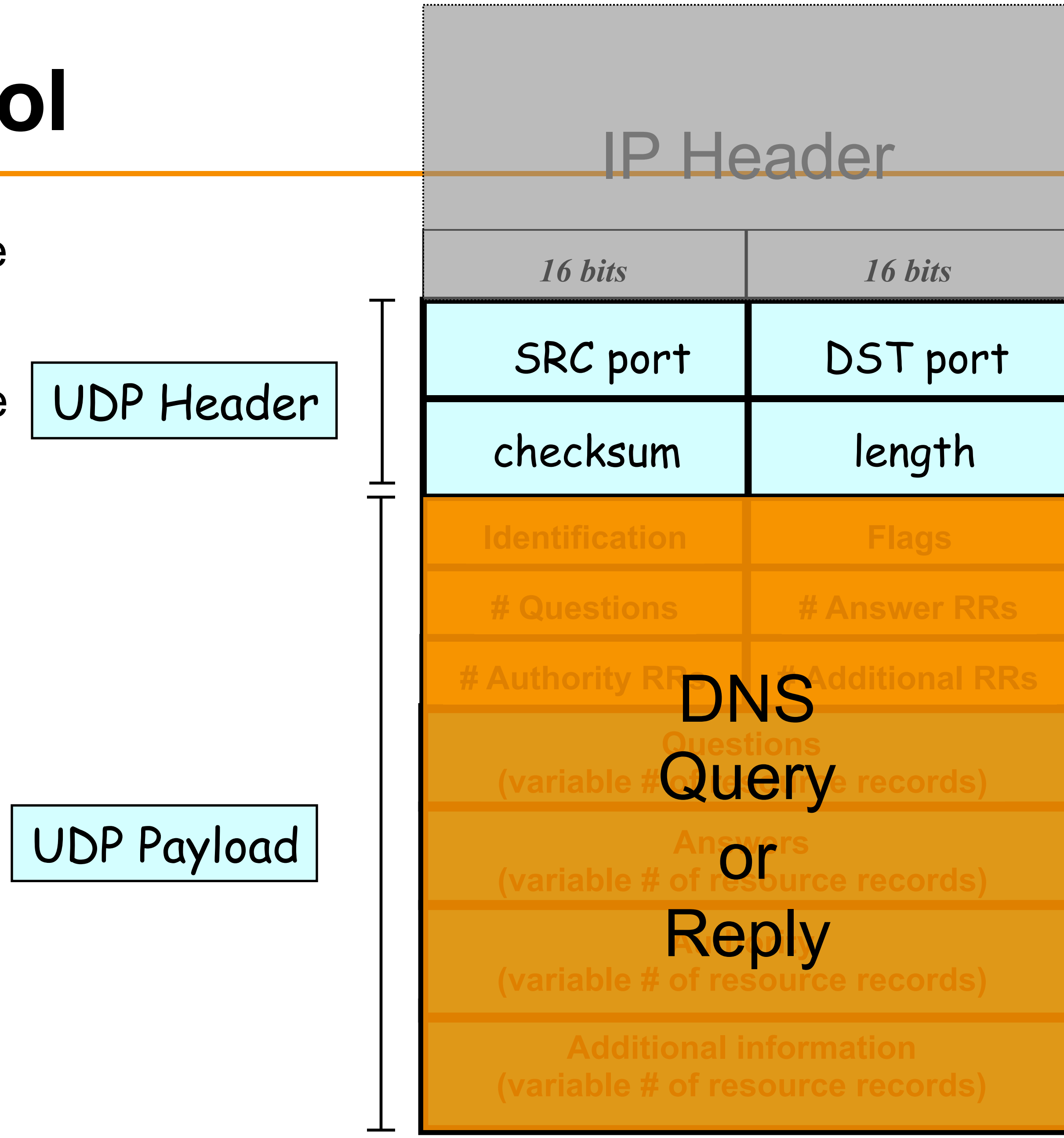
```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

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

;; ANSWER SECTION
eecs.mit.edu.

;; AUTHORITY 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
```

"**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.

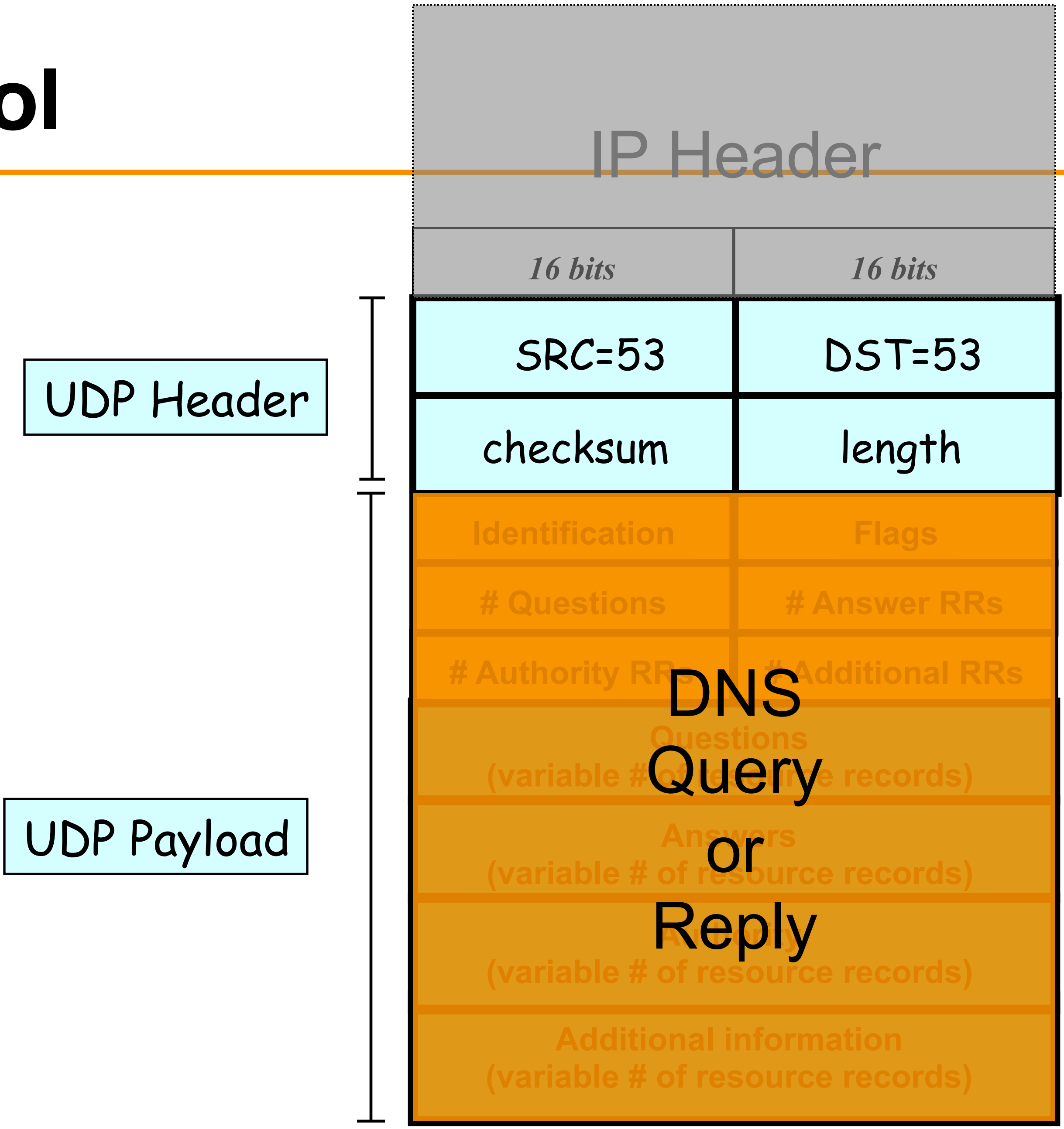
DNS Protocol

- 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
- Frequently, both clients and servers use port 53



DNS Protocol

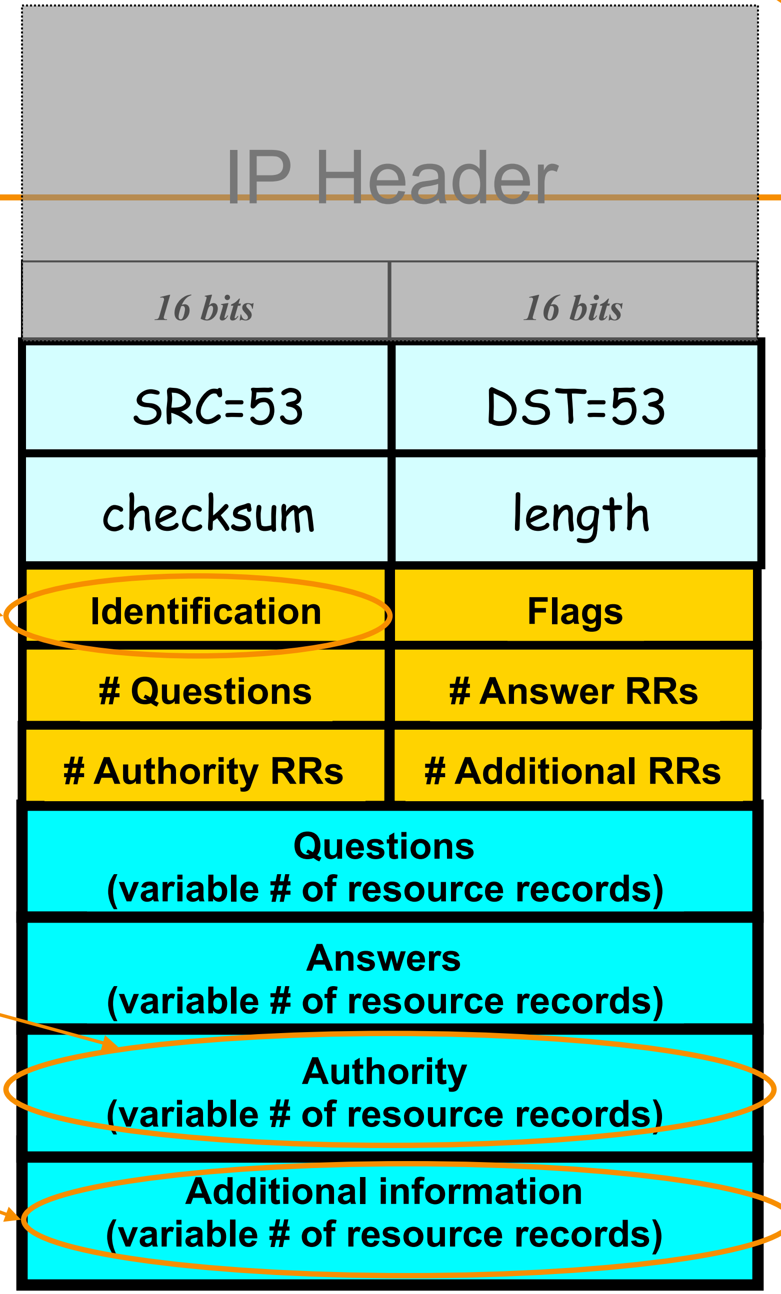
- Lightweight exchange of *query* and *reply* messages, both with same message format
- Primarily uses UDP for its transport protocol, which is what we'll assume
- Frequently, both clients and servers use port 53



DNS Protocol, cont.

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: QU                                    001
;; flags: qr rd ra; QUERY:                                    ADDITIONAL: 3

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

;; ANSWER SECTION:
eecs.mit.edu.            21                                   .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.            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
```

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.                 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
```

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.

;; 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
```

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.                                          6

;; 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
```

In this case they chose to make the mapping *disappear* after 30 seconds. They could have made it persist for weeks, or disappear even quicker.

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.
```

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

```
;; 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
```

dig eecs.mit.edu A

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

;; 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.               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
```

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.

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.

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).

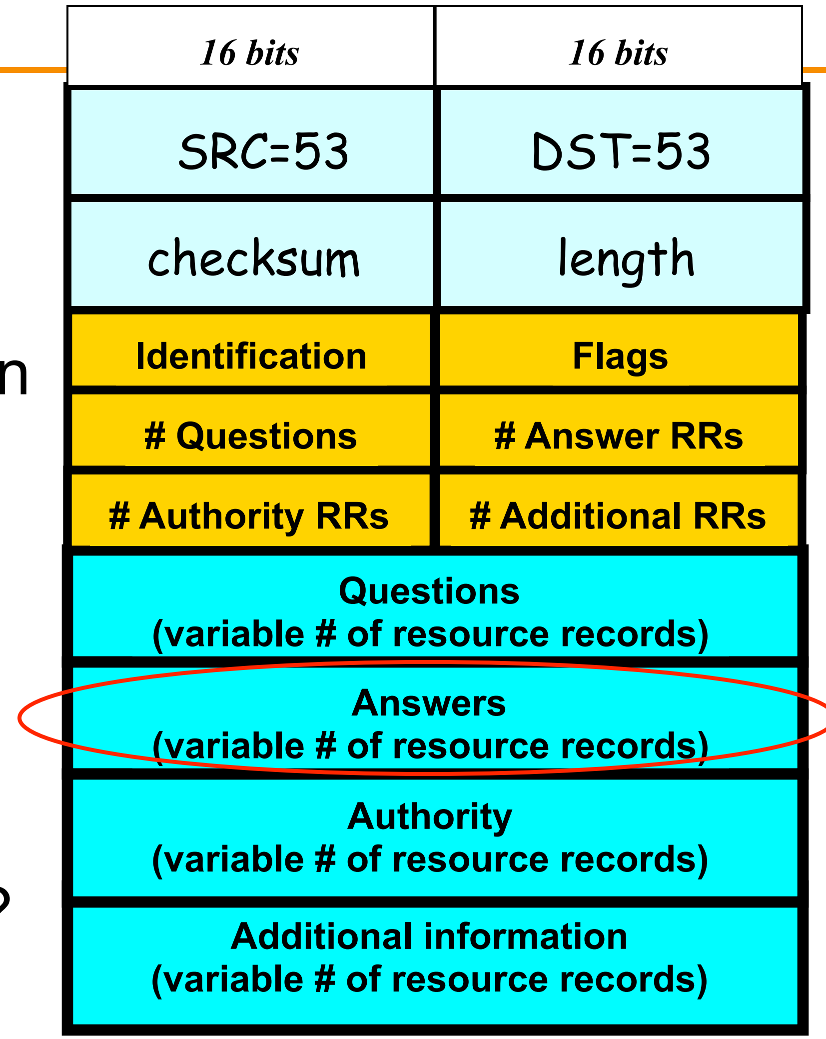
Blind spoofing

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

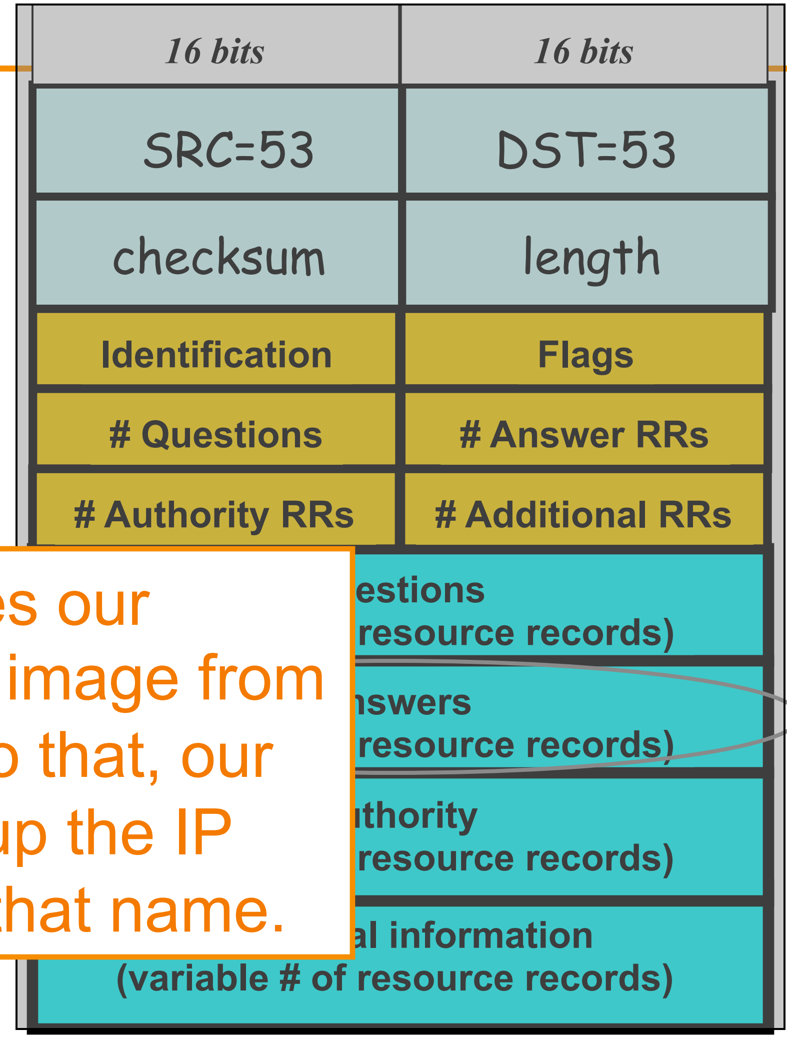
- How can such a **remote** attacker even know we are looking up `mail.google.com`?

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

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



Blind spoofing



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

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?

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)	

<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.

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 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!

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

DNS Blind Spoofing (Kaminsky 2008)

- Two key ideas:
  - Attacker can get around caching of legit replies by generating a series of different name lookups:

```
<img src="http://random1.google.com" …>
<img src="http://random2.google.com" …>
<img src="http://random3.google.com" …>
                    . . .
<img src="http://randomN.google.com" …>
```

  - Trick victim into looking up a domain you don't care about, use **Additional** field to spoof the domain you do

Kaminsky Blind Spoofing

For each lookup of *randomk*.google.com, attacker **spoofs** a bunch of records like this, each with a different Identifier

```
;; QUESTION SECTION:
;randomk.google.com.                IN      A

;; ANSWER SECTION:
randomk.google.com      21600       IN      A       doesn't matter

;; AUTHORITY SECTION:
google.com.             11088       IN      NS      mail.google.com

;; ADDITIONAL SECTION:
mail.google.com         126738      IN      A       6.6.6.6
```

Once they win the race, not only have they poisoned `mail.google.com` …

Kaminsky Blind Spoofing

For each lookup of *randomk*.google.com, attacker **spoofs** a bunch of records like this, each with a different Identifier

```
;; QUESTION SECTION:
;randomk.google.com.            IN      A

;; ANSWER SECTION:
randomk.google.com      21600   IN      A       doesn't matter

;; AUTHORITY SECTION:
google.com.             11088   IN      NS      mail.google.com

;; ADDITIONAL SECTION:
mail.google.com         126738  IN      A       6.6.6.6
```

Once they win the race, not only have they poisoned mail.google.com … but also the cached NS record for google.com's name server - so any **future** *X*.google.com lookups *go through the attacker's machine*

Defending Against Blind Spoofing

Central problem: all that tells a client they should accept a response is that it matches the Identification field.

With only 16 bits, it lacks sufficient entropy: even if truly random, the *search space* an attacker must *brute force* is too small.

Where can we get more entropy? (*Without* requiring a protocol change.)

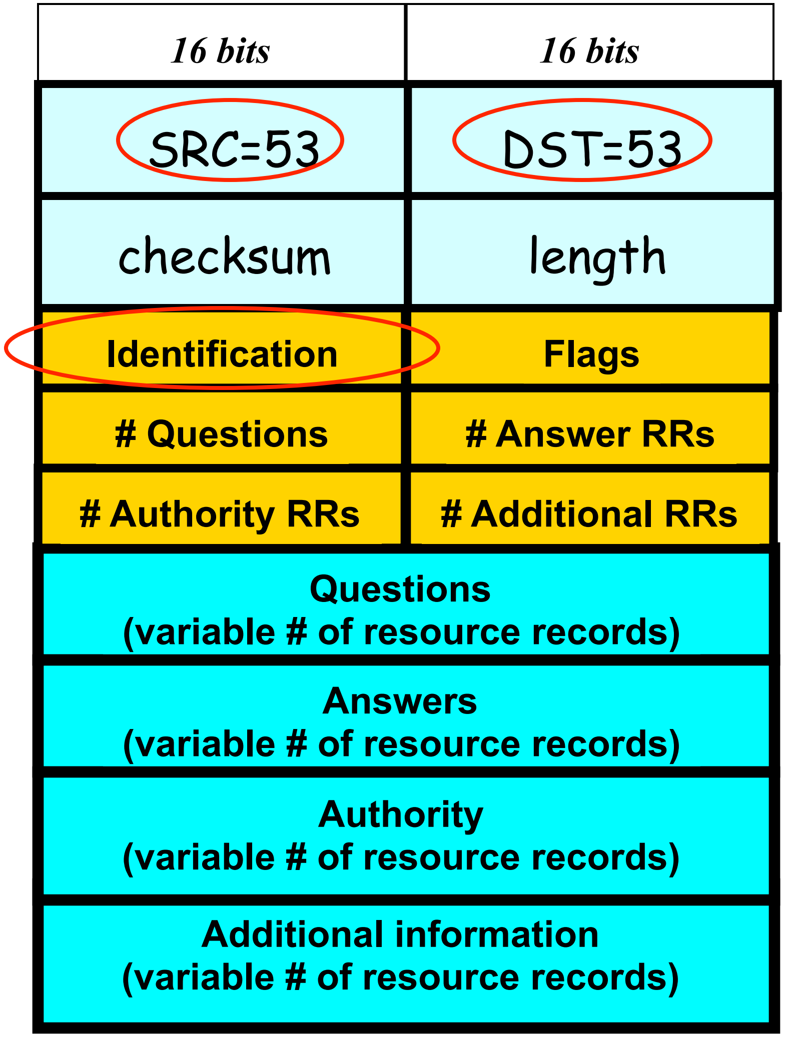
16 bits	16 bits
SRC=53	DST=53
checksum	length
Identification	**Flags**
# Questions	**# Answer RRs**
# Authority RRs	**# Additional RRs**
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable #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

Defending Against Blind Spoofing

Total *entropy*: 16 bits

For requestor to receive DNS reply, needs both correct Identification and correct ports.

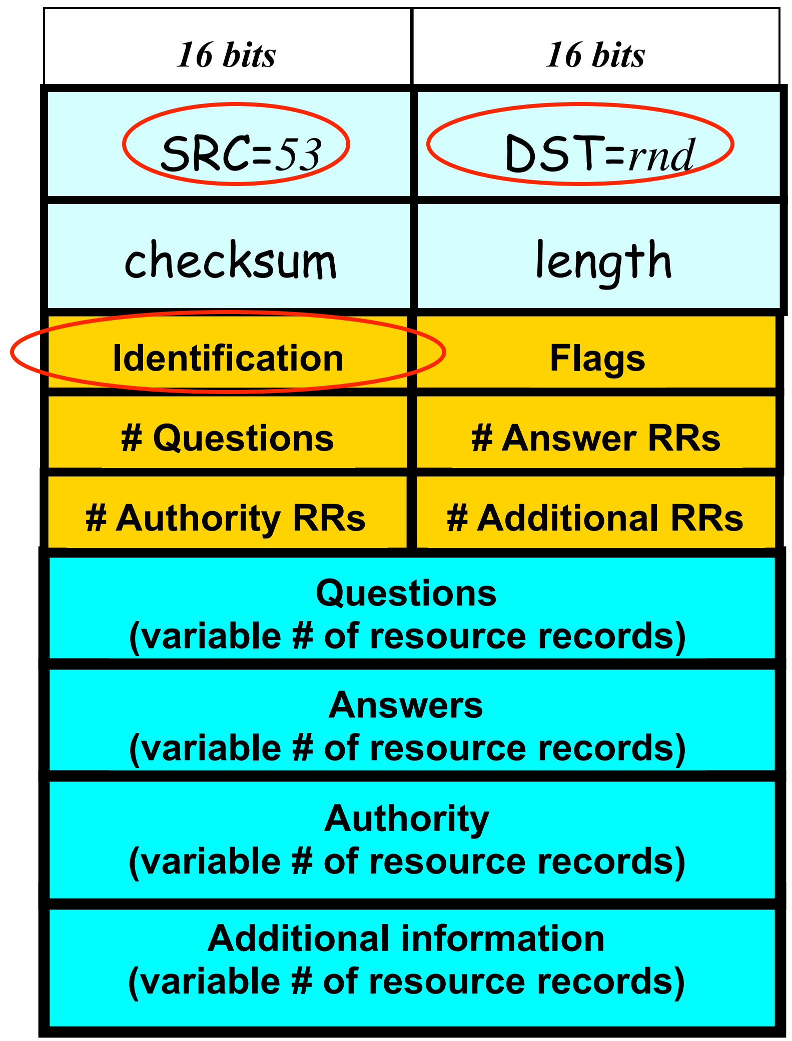
On a request, DST port = 53. SRC port usually also 53 - but not fundamental, just convenient.



Defending Against Blind Spoofing

Total *entropy*: ? bits

"Fix": client uses random source port ⇒ attacker doesn't know correct dest. port to use in reply

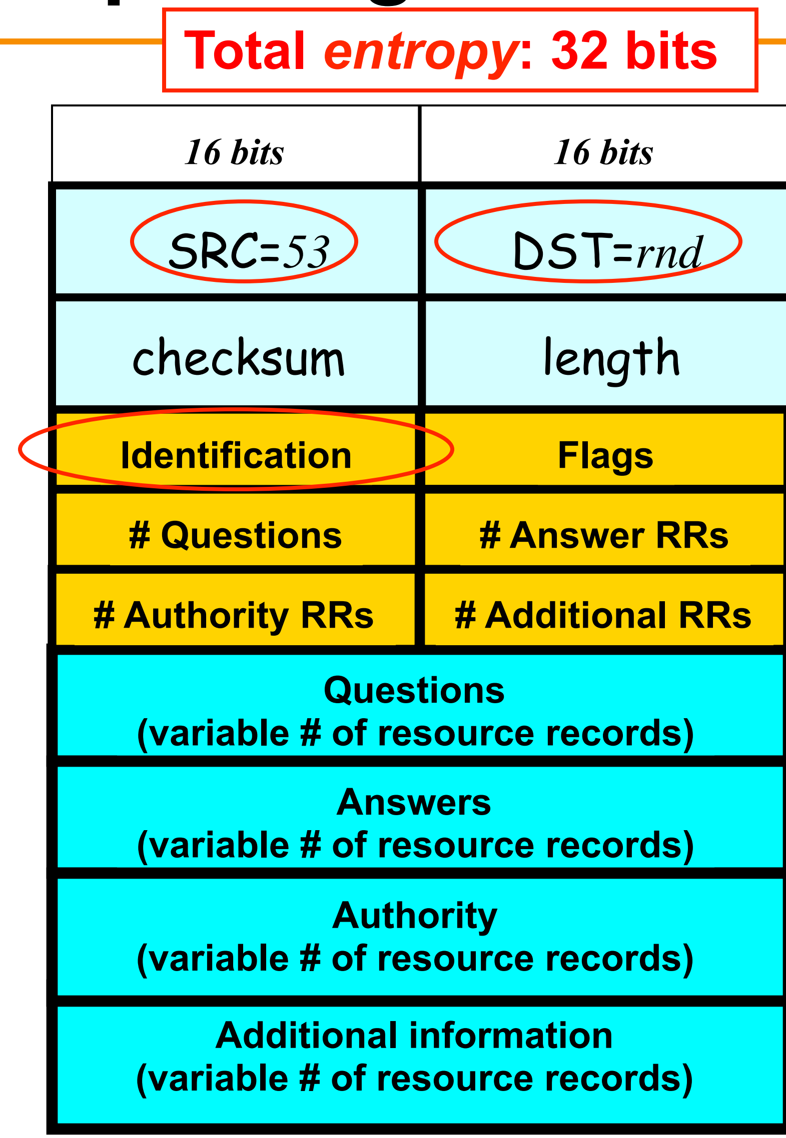


Defending Against Blind Spoofing

"Fix": client uses random source port ⇒ attacker doesn't know correct dest. port to use in reply

32 bits of entropy makes it orders of magnitude harder for attacker to guess all the necessary fields and dupe victim into accepting spoof response.

This is what primarily "secures" DNS against blind spoofing today.



Lessons learned

- Security risks: friend might be malicious
- Communication channel to friend might be insecure
- Friend might be well-intentioned but misinformed

Extra Material

Summary of DNS Security Issues

- DNS threats highlight:
  - Attackers can attack opportunistically rather than eavesdropping
    - o 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
    - o 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

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