

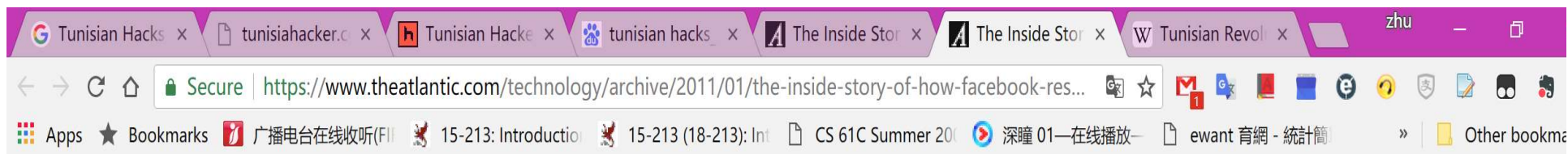
# **Attacks on DNS: Risks of Caching**

**May 10, 2019**

# The Inside Story of How Facebook Responded to Tunisian Hacks



It was on Christmas Day that Facebook's Chief Security Officer Joe Sullivan first noticed strange things going on in Tunisia. Reports started to trickle in that political-protest pages were being hacked. "We were getting anecdotal reports saying 'It looks like someone logged into my account and deleted it' " Sullivan said.



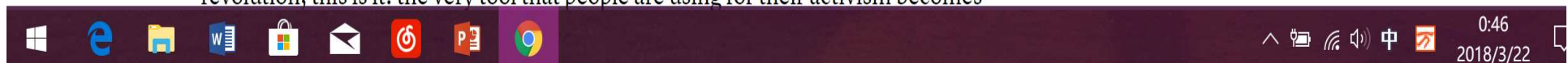
After more than ten days of intensive investigation and study, Facebook's security team realized something very, very bad was going on. The country's Internet service providers were running a malicious piece of code that was recording users' login information when they went to sites like Facebook.

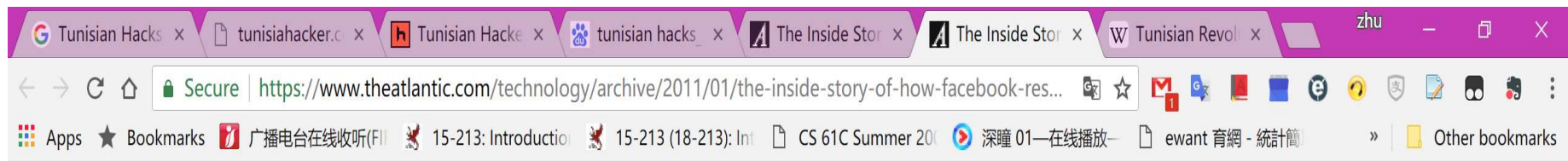
By January 5, it was clear that an entire country's worth of passwords were in the process of being stolen right in the midst of the greatest political upheaval in two decades. Sullivan and his team decided they needed a country-level solution -- and fast.

Though Sullivan said Facebook has encountered a wide variety of security problems and been involved in various political situations, they'd never seen anything like what was happening in Tunisia.

"We've had to deal with ISPs in the past who have tried to filter or block our site," Sullivan said. "In this case, we were confronted by ISPs that were doing something unprecedented in that they were being very active in their attempts to intercept user information."

If you need a parable for the potential and pitfalls of a social-media enabled revolution, this is it: the very tool that people are using for their activism becomes





The software was basically a country-level keystroke logger, with the passwords presumably being fed from the ISPs to the Ben Ali regime. As a user, you just logged into some part of the cloud, Facebook or your email, say, and it snatched up that information. If you stayed persistently logged in, you were safe. It was those who logged out and came back that were open to the attack.

Sullivan's team rapidly coded a two-step response to the problem. First, all Tunisian requests for Facebook were routed to an https server. The Https protocol encrypts the information you send across it, so it's not susceptible to the keylogging strategy employed by the Tunisian ISPs.

The second technical solution they implemented was a "roadblock" for anyone who had logged out and then back in during the time when the malicious code was running. Like Facebook's version of a "mother's maiden name" question to get access to your old password, it asks you to identify your friends in photos to complete an account login.

They rolled out the new solutions to 100% of Tunisia by Monday morning, five days after they'd realized what was happening. It wasn't a totally perfect solution. Most specifically, ISPs can force a downgrade of https to http, but Sullivan said



## Anonymous' most memorable hacks



13

### Open Letter from ANONYMOUS

January 21, 2011



DEAR CITIZENS OF TUNISIA,

Congratulations once again for your bravery in putting your lives on the line in the streets of Tunisia and refusing to accept the interim government dominated by the old regime who seem to be very good at saying that they are good people with no blood on their hands. Yet, they refuse to prove that they are good people by standing down to give you the genuine confidence that you deserve - confidence that the old regime is truly gone, and that you are safe. The fact that they don't care about your security - and your legitimate fear after all that was done to you - is why they must go. These people - without any sense of irony - have the audacity to ask the truly brave Tunisian citizen Slim Amamou "where is your tie?" rather than "has there been any progress in bringing the people who abused your human rights to justice yet, and is there anything I can do to expedite the process?". This shows their complete disrespect for human dignity. Added to the contempt they clearly have for the intelligence of the Tunisian people if they seriously believe that simply resigning (in name) from a dictator's party is a sufficient action at this crucial moment in Tunisia's history. You are on the streets right now saying this. We are in cyberspace echoing your thoughts.

Open

In re  
limit  
struc  
DDo  
cras

# What to Know About the Worldwide Hacker Group 'Anonymous'

By [GENEVA SANDS](#) • Mar 19, 2016, 1:58 AM ET

[Share with Facebook](#)

[Share with Twitter](#)



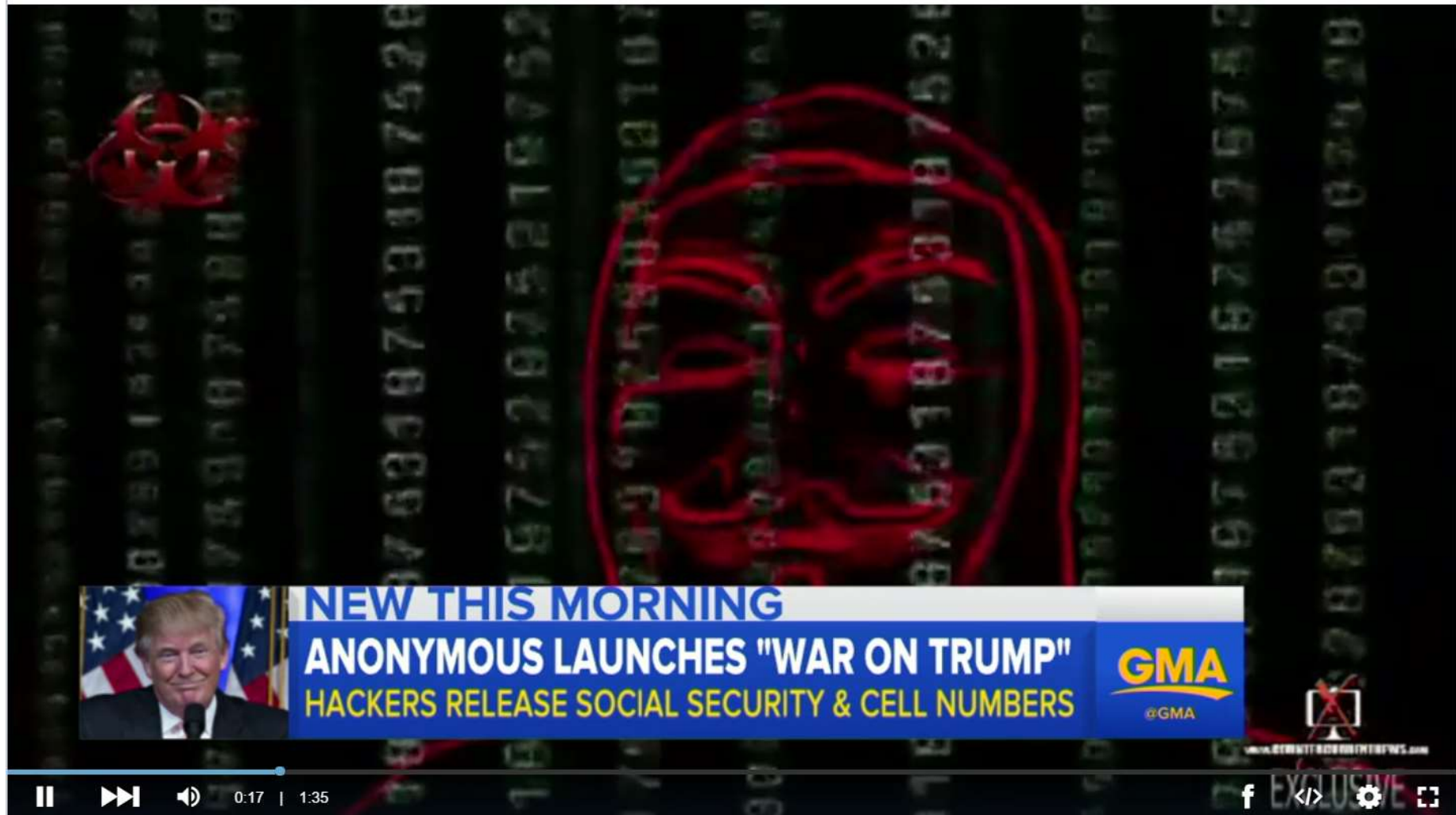
PYMCA/UG/Getty Images

# Anonymous Claims to Hack Donald Trump

By CANDACE SMITH • Mar 17, 2016, 9:35 PM ET

[f Share with Facebook](#)

[t Share with Twitter](#)



sidetheatre.com

on Donald Trump

# DNS Overview

- DNS translates [www.google.com](http://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).

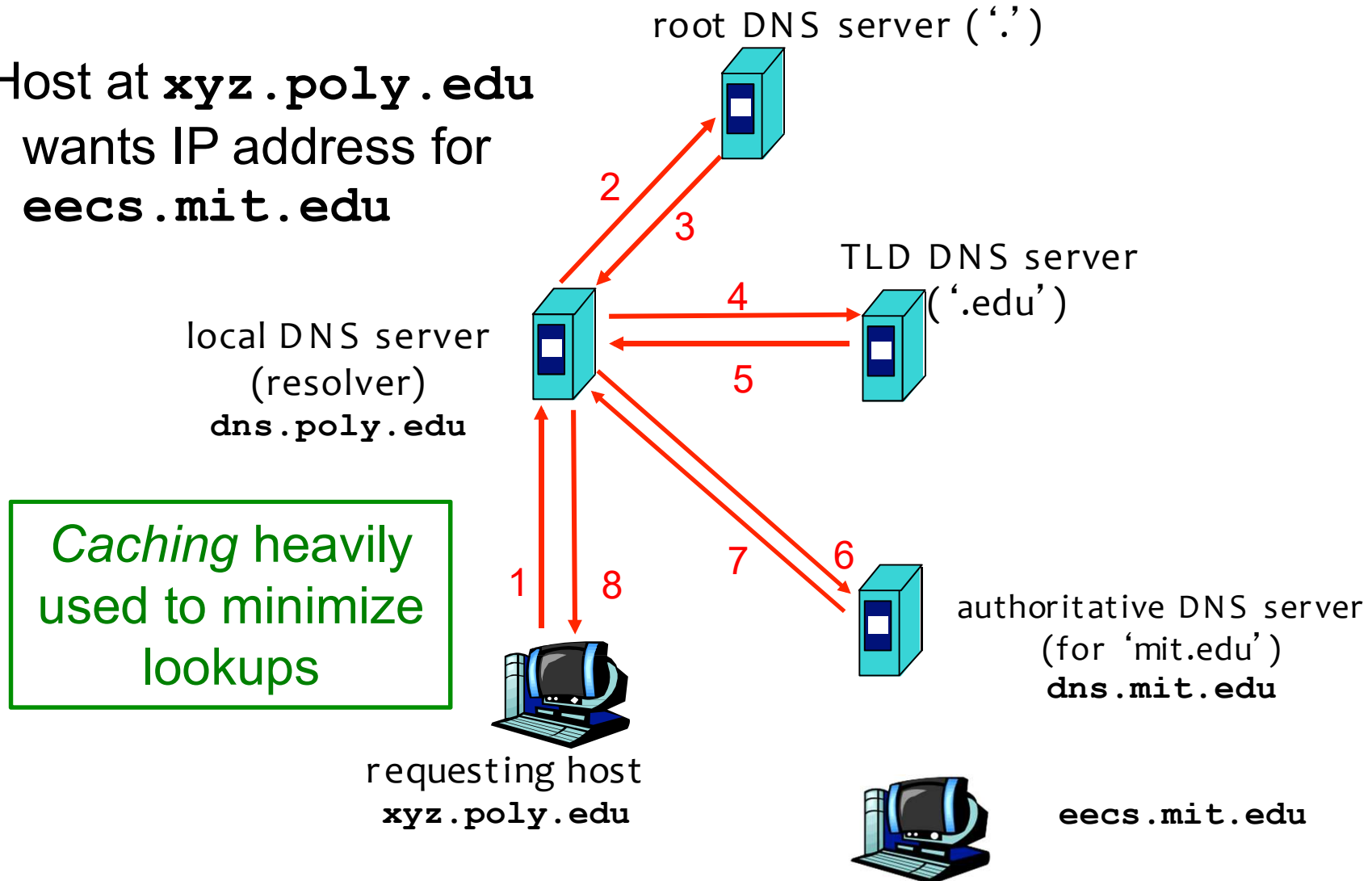


# DNS Overview

- DNS translates [www.google.com](http://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  
**eeecs.mit.edu**



# Group Discussion

- Please discuss the potential attacks towards DNS and illustrate it.

# 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  $\Leftrightarrow$  IP addresses
  - Design only **scales** if we can minimize lookup traffic
    - o #1 way to do so: **caching**
    - o #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

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

;; 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

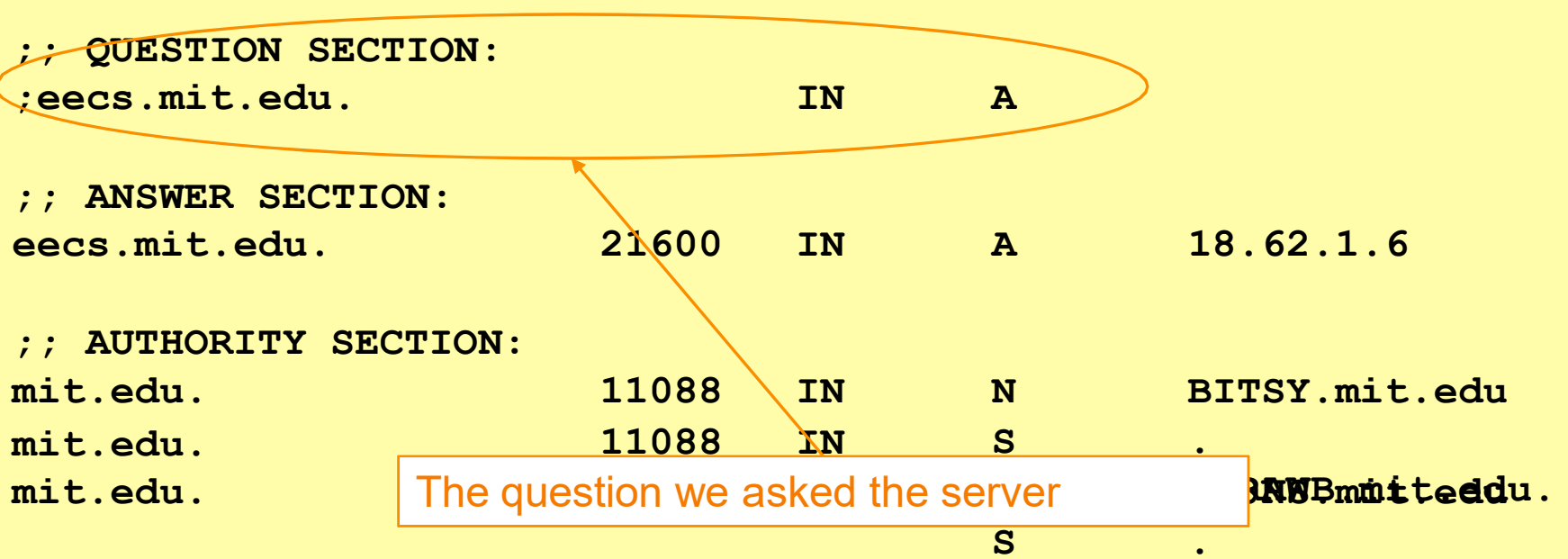
```
; ; <<>> DiG 9.6.0-APPLE- <<>> eecs.mit.edu a
P2 global options:
;; +cmd Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 19901          3
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL:

;; 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      N      BITSY.mit.edu
mit.edu.                    11088   IN      S      .
mit.edu.                    11088   IN      S      .

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

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

;; ANSWER SECTION:            216
eecs.mit.edu.                 0
                                A 16-bit transaction identifier that enables
                                the DNS client (dig, in this case) to match up
                                the reply with its original request

;; 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 e
```

```
;; flags: qr rd d u
```

```
ra;
```

```
;; QUESTION SECTION:
```

```
;eecs.mit.edu.
```

```
;; ANSWER SECTION:
```

```
eecs.mit.edu.
```

```
;; AUTHORITY SECTION:
```

```
mit.edu.
```

```
mit.edu.
```

```
mit.edu.
```

```
;; ADDITIONAL SECTION:
```

```
STRAWB.mit.edu.
```

```
BITSY.mit.edu.
```

```
W20NS.mit.edu.
```

"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

```
ONAL: 3
```

IN

A

21600

IN

A

18.62.1.6

11088

IN

NS

BITSY.mit.edu.

11088

IN

NS

W20NS.mit.edu.

11088

IN

NS

STRAWB.mit.edu.

126738

IN

A

18.71.0.151

166408

IN

A

18.72.0.3

126738

IN

A

18.70.0.160

## dig eecs.mit.edu A

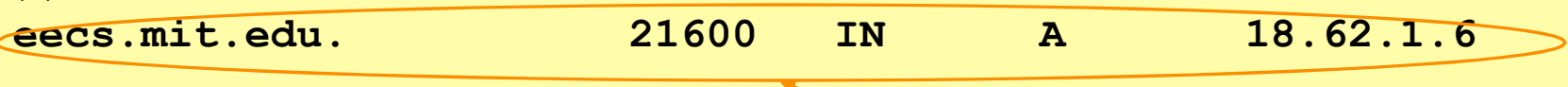
```
; ; <<>> DiG 9.6.0-APPLE- <<>> eecs.mit.edu a
P2 global options:
;; +cmd Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 19901          3
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL:

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

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

;; AUTHORITY SECTION:
mit.edu.
mit.edu.
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
```



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

du.  
du.  
edu  
.



# dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu
```

```
a; global options: +cn
```

```
; ; Got answer:
```

```
; ; ->>HEADER<<- opcode
```

```
; ; flags: qr rd ra; QU
```

```
; ; QUESTION SECTION:
```

```
; eecs.mit.edu.
```

```
; ; ANSWER SECTION:
```

```
; eecs.mit.edu.
```

```
; ; AUTHORITY SECTION:
```

```
; mit.edu.
```

```
; mit.edu.
```

```
; mit.edu.
```

```
; ; ADDITIONAL SECTION:
```

```
; STRAWB.mit.edu.
```

```
; BITSY.mit.edu.
```

```
; W20NS.mit.edu.
```

“**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.

21600 IN A 18.62.1.6

11088 IN NS  
11088 IN NS  
11088 IN NS

BITSY.mit.edu.  
W20NS.mit.edu.  
STRAWB.mit.edu

18.71.0.151

18.72.0.3

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.
```

```
;; ANSWER SECTION:
```

```
eecs.mit.edu.
```

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

```
mit.edu.      11088      IN      NS      W20NS.mit.edu.
```

```
mit.edu.      11088      IN      NS      STRAWB.mit.edu.
```

```
;; ADDITIONAL SECTION:
```

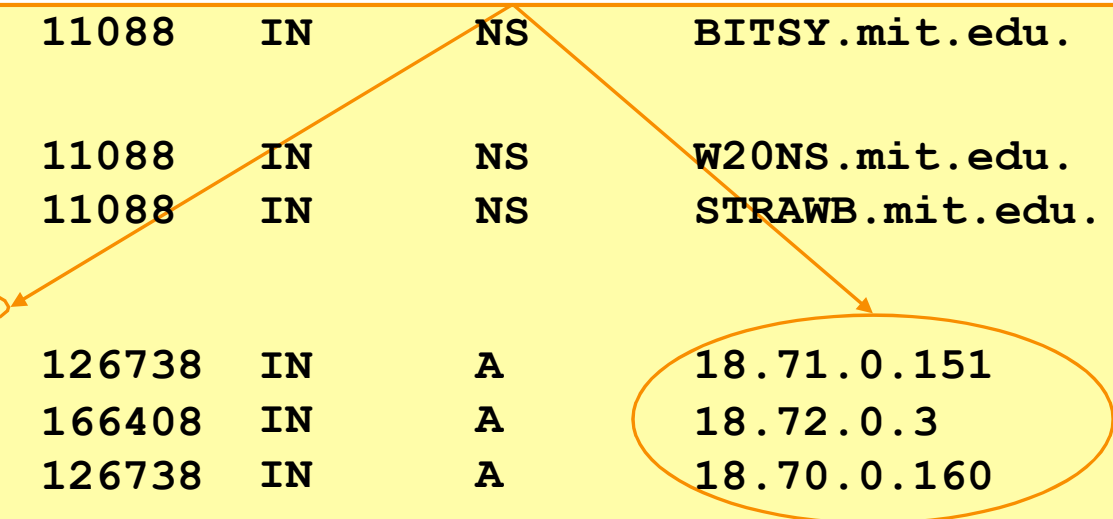
```
STRAWB.mit.edu.      126738      IN      A      18.71.0.151
```

```
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

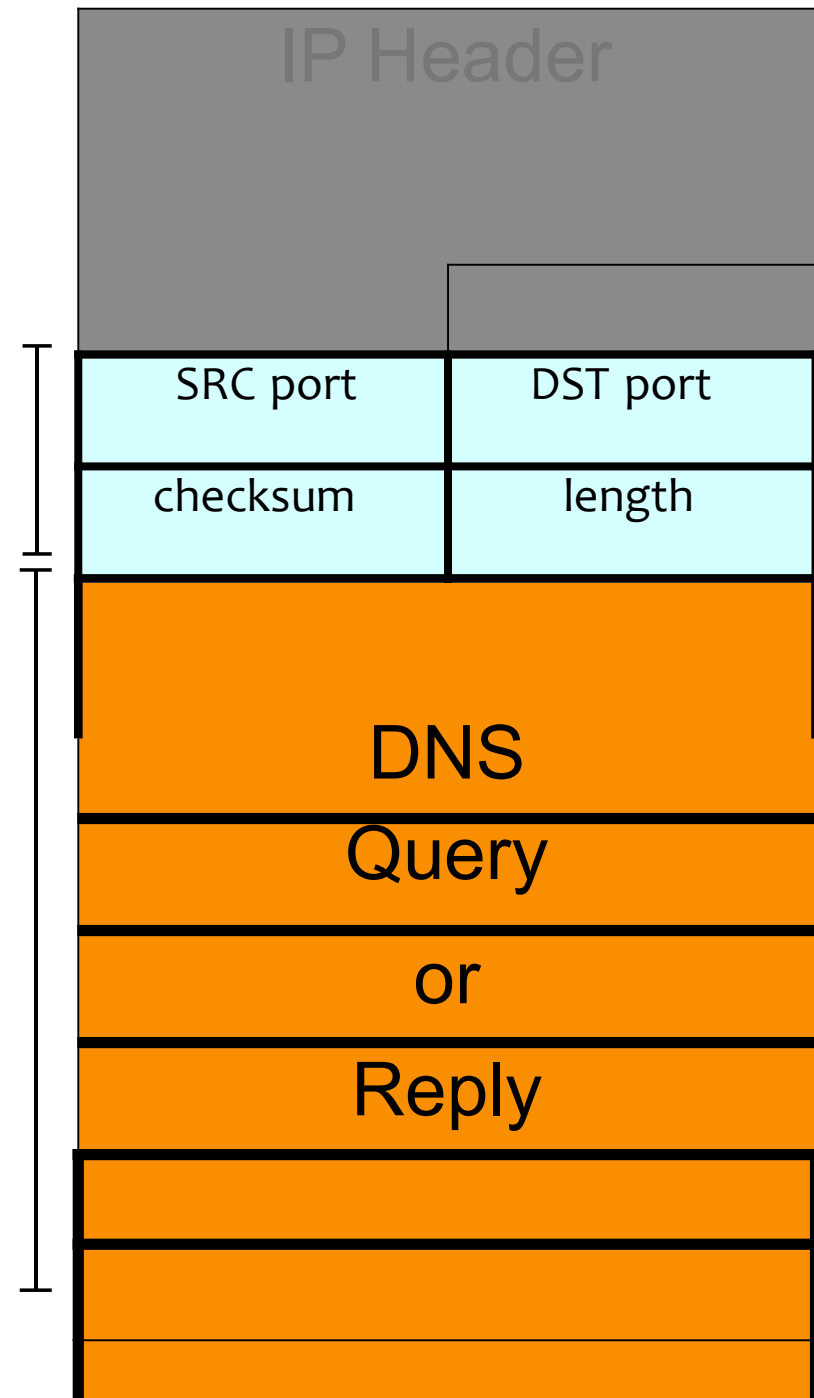
Lightweight exchange  
of *query* and *reply*  
messages, both  
with **same** message  
format

UDP Header

Primarily uses UDP  
for its transport  
protocol, which is  
what we'll assume

UDP Payload

Frequently, both  
clients and servers  
use port 53



# DNS Protocol

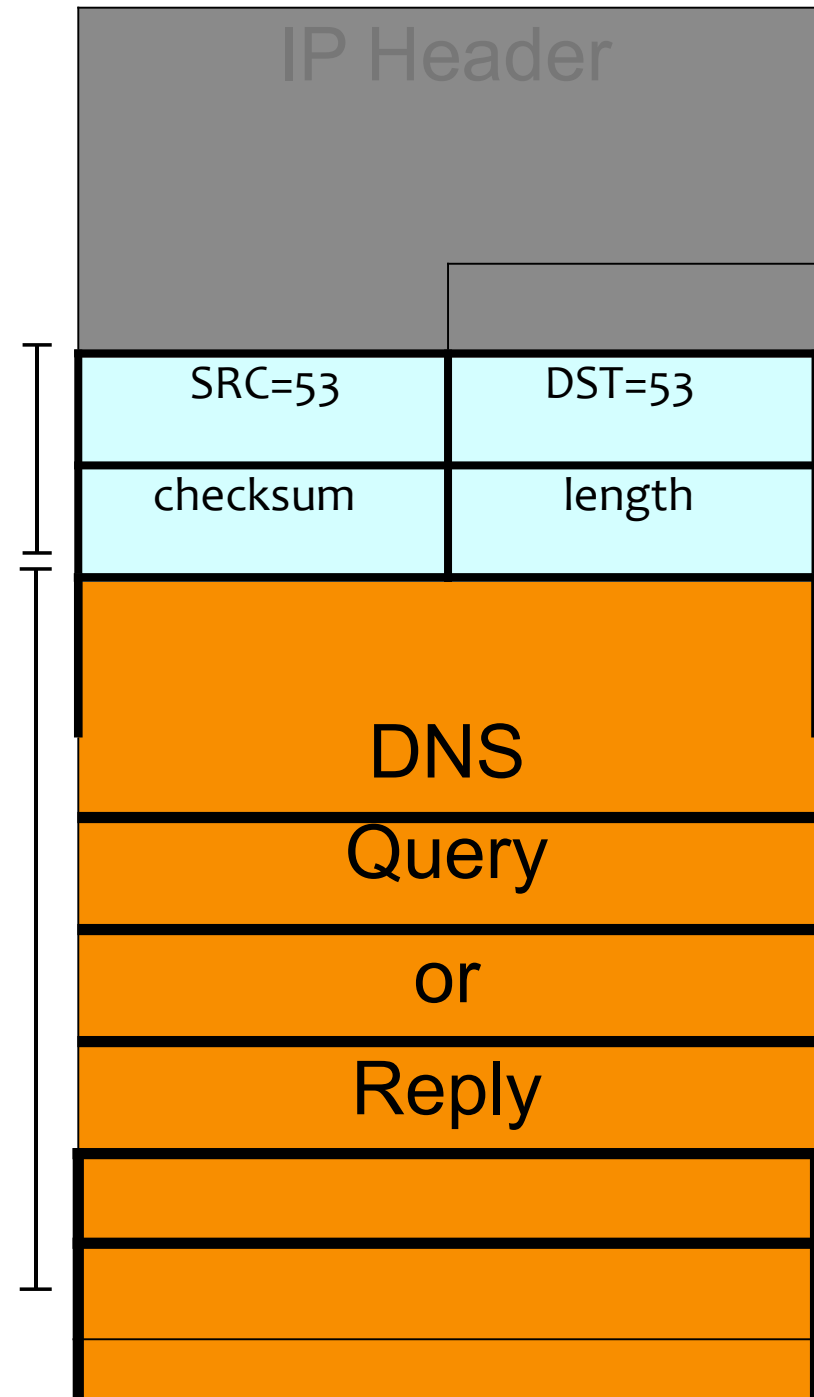
Lightweight exchange  
of *query* and *reply*  
messages, both  
with **same** message  
format

UDP Header

Primarily uses UDP  
for its transport  
protocol, which is  
what we'll assume

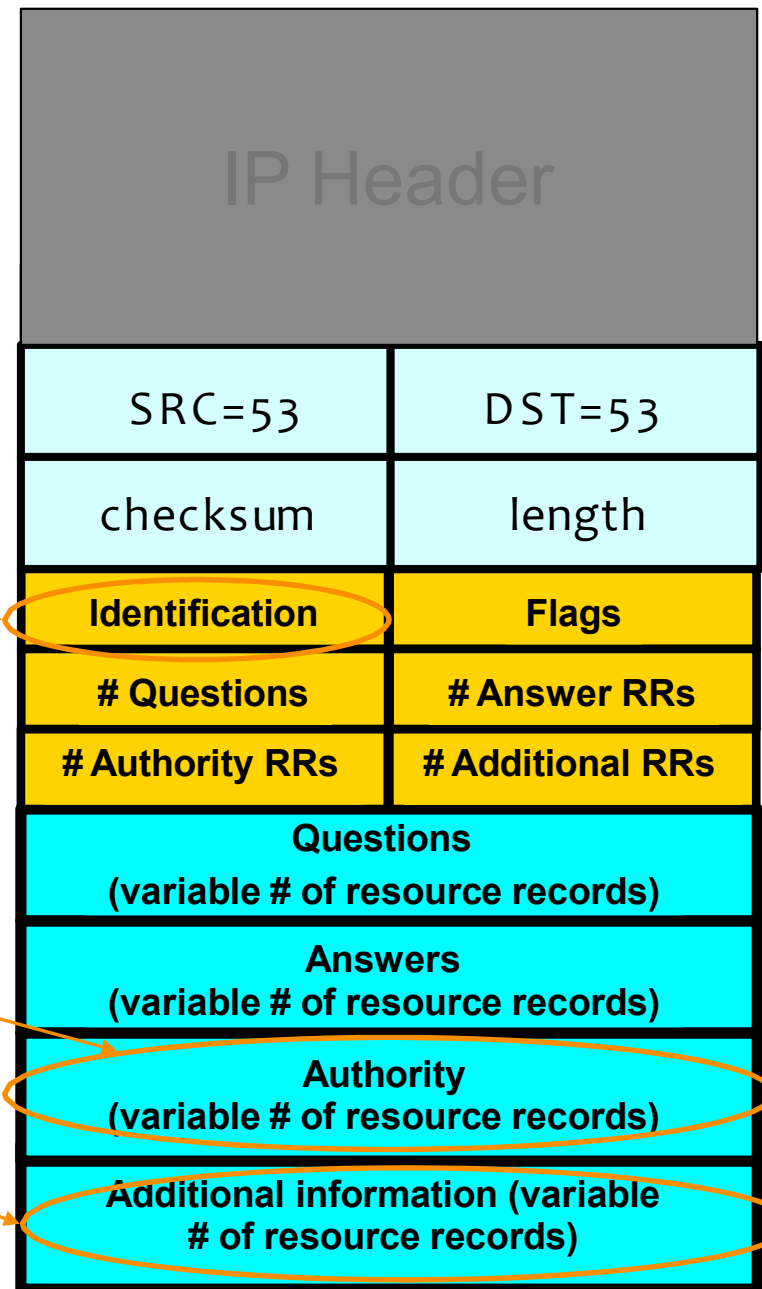
UDP Payload

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: QUERY status: NOERROR
;; flags: qr rd ra; QUERY: eecs.mit.edu. type: A
;; QUESTION SECTION:
;eecs.mit.edu. type: A

;; ANSWER SECTION:
eecs.mit.edu. 21 IN NS BITSY.mit.edu.
eecs.mit.edu. 11088 IN NS W20NS.mit.edu.
eecs.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          3
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL:

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

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

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

```
;; 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
```

## 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.
```

```
;; SECTION:
```

AUTHORITY	11088	IN	NS	BITSY.mit.edu.
mit.edu	11088	IN	NS	W20NS.mit.edu.
.	30	IN	NS	<u>www.berkeley.edu.</u>
mit.edu				
;; ADDITIONAL SECTION:				
<u>www.berkeley.edu.</u>	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](http://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- <<>> eecs.mit.edu a
P2 global options:
;; +cmd Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901          3
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL:

;; 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
```

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.

6

## 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	<u><a href="http://www.berkeley.edu">www.berkeley.edu</a></u> .

```
;; ADDITIONAL SECTION:
```

<u><a href="http://www.berkeley.edu">www.berkeley.edu</a></u> .	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<<- opcode
```

```
;; 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       N       BITSY.mit.edu.
```

```
mit.edu.               11088    IN       S       W20NS.mit.edu.
```

```
mit.edu.               30       IN       NS      www.berkeley.edu
```

```
NS  
.
```

```
;; 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`  $\Rightarrow$  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 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:

... . . .

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

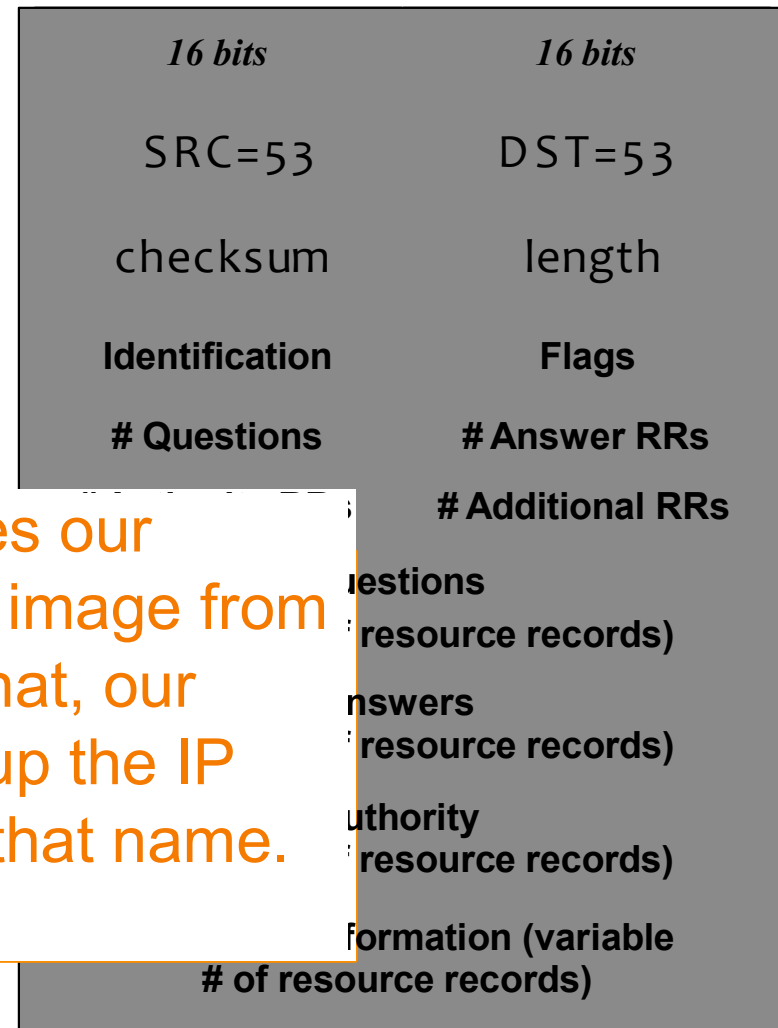
- Say we look up mail.google.com; how can an **off-path** attacker feed us a bogus

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.

- How even k mail.c

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

... . . .



# Blind spoofing

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?

**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)	



← They observe ID k here



← 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 inoculated!

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! ?

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

# 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