Network Attacks

CS 161 - Computer Security

Profs. Vern Paxson & David Wagner

TAs: John Bethencourt, Erika Chin, Matthew Finifter, Cynthia Sturton, Joel Weinberger

http://inst.eecs.berkeley.edu/~cs161/

Feb 10, 2010

Focus of Today’s Lecture

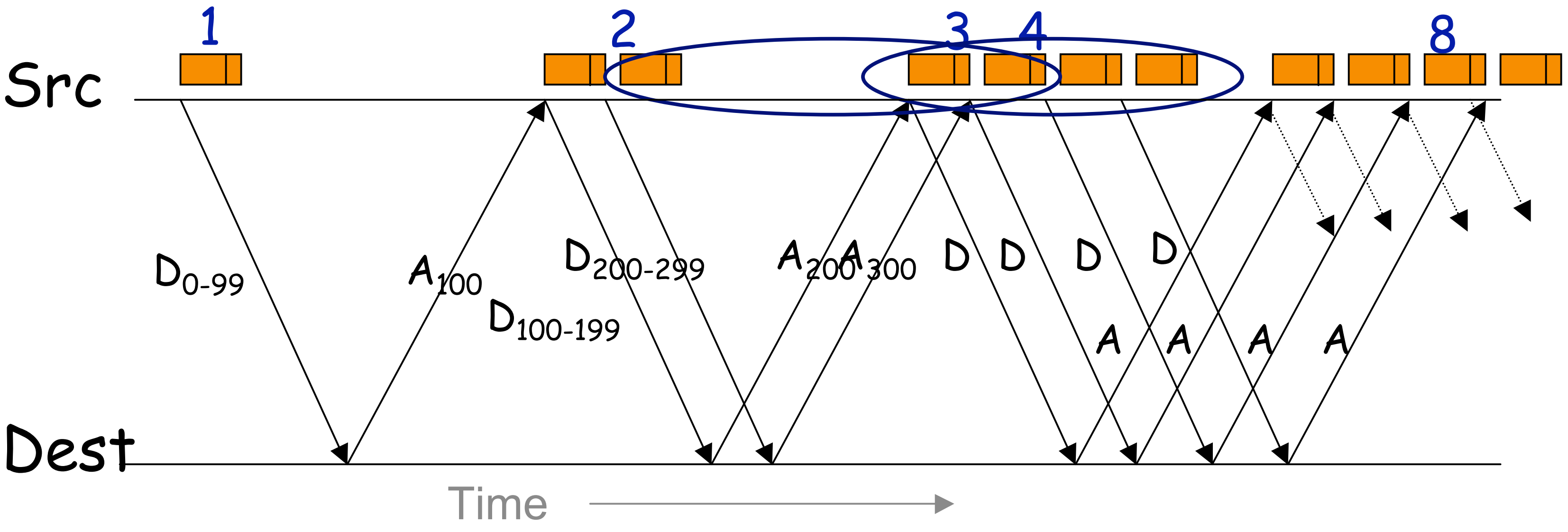
- Finish discussion of security threats in TCP
  - The problem of “cheaters” who exceed the allowed transmission rate
  - Summary of TCP issues/principles
- Security threats in DHCP and DNS
  - Summary of issues/principles

- Note that none of these threats concerns direct application threats. They all target

TCP's Rate Management

Unless there's loss, TCP doubles data in flight every "round-trip". All TCPs expected to obey ("fairness").

Mechanism: for each arriving ack for new data, increase allowed data by 1 maximum-sized packet

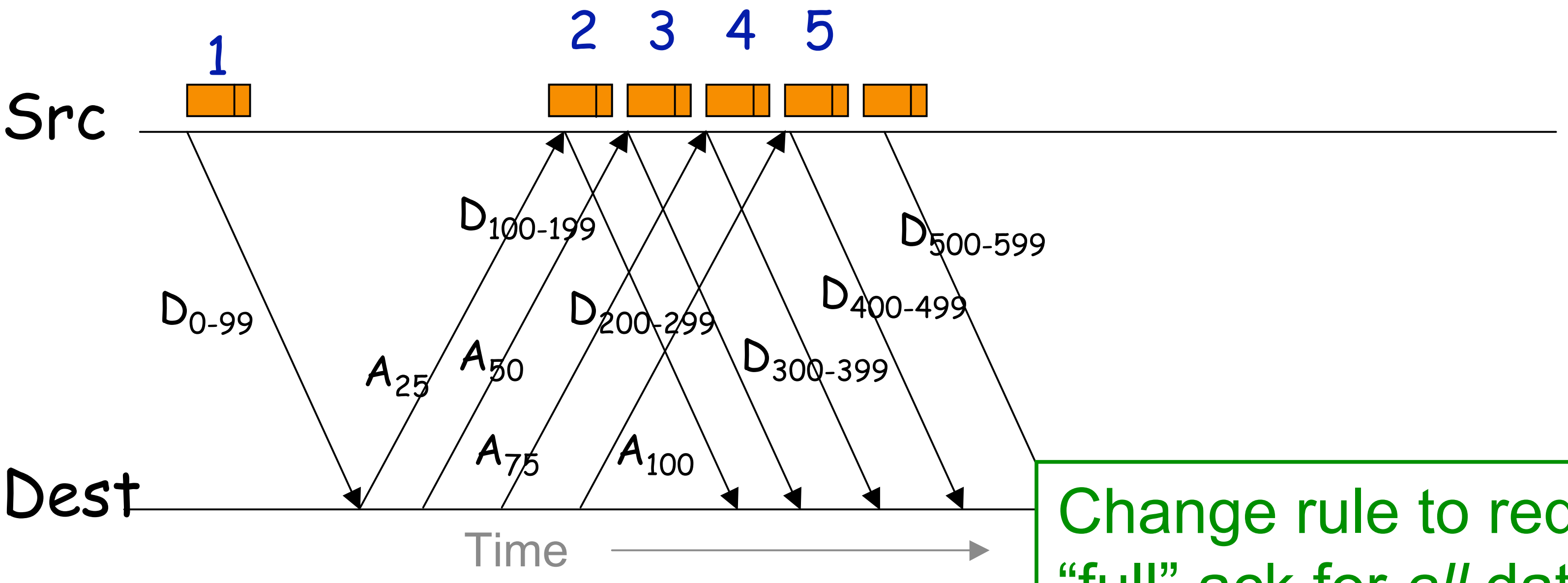


E.g., suppose maximum-sized packet = 100 bytes

TCP Threat: Cheating on Allowed Rate

How can the destination (receiver) get data to come to them faster than normally allowed?

ACK-Splitting: each ack, even though **partial**, increases allowed data by one maximum-sized packet



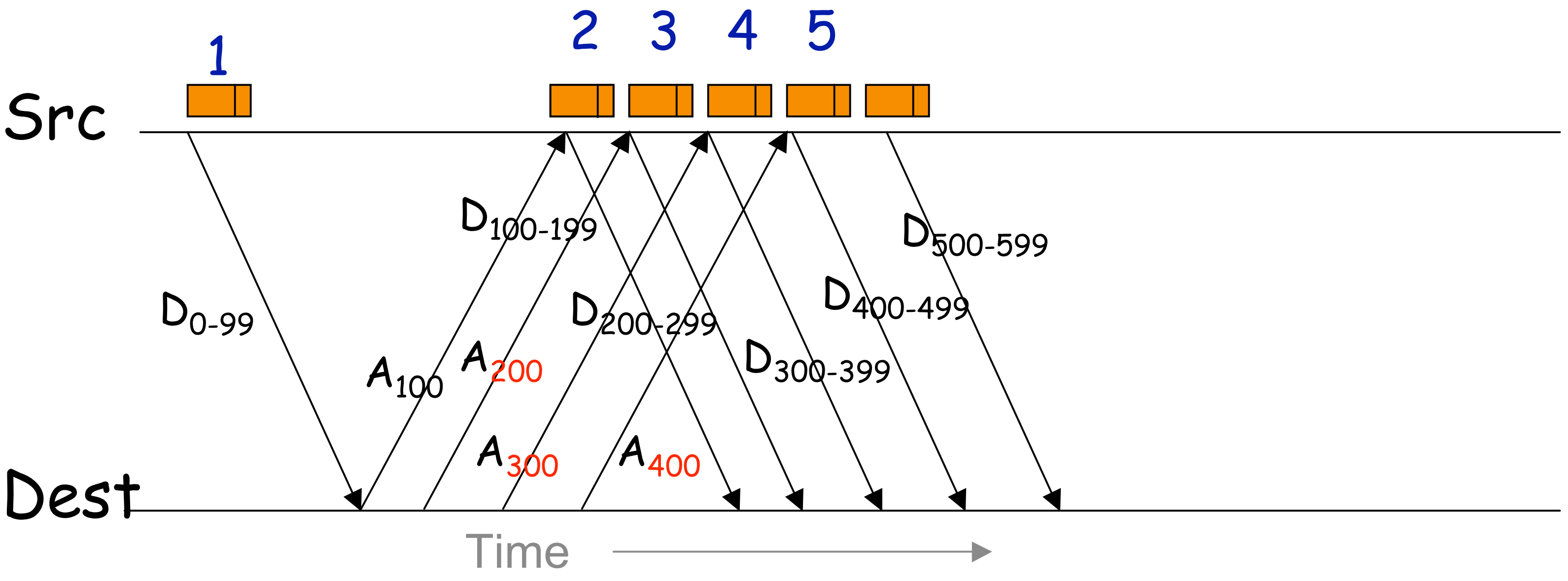
How do we defend against this?

Change rule to require "full" ack for *all* data sent in a packet

TCP Threat: Cheating on Allowed Rate

How can the destination (receiver) *still* get data to come to them faster than normally allowed?

Opportunistic ack'ing: acknowledge data not yet seen!



How do we defend against *this*?

Keeping Receivers Honest

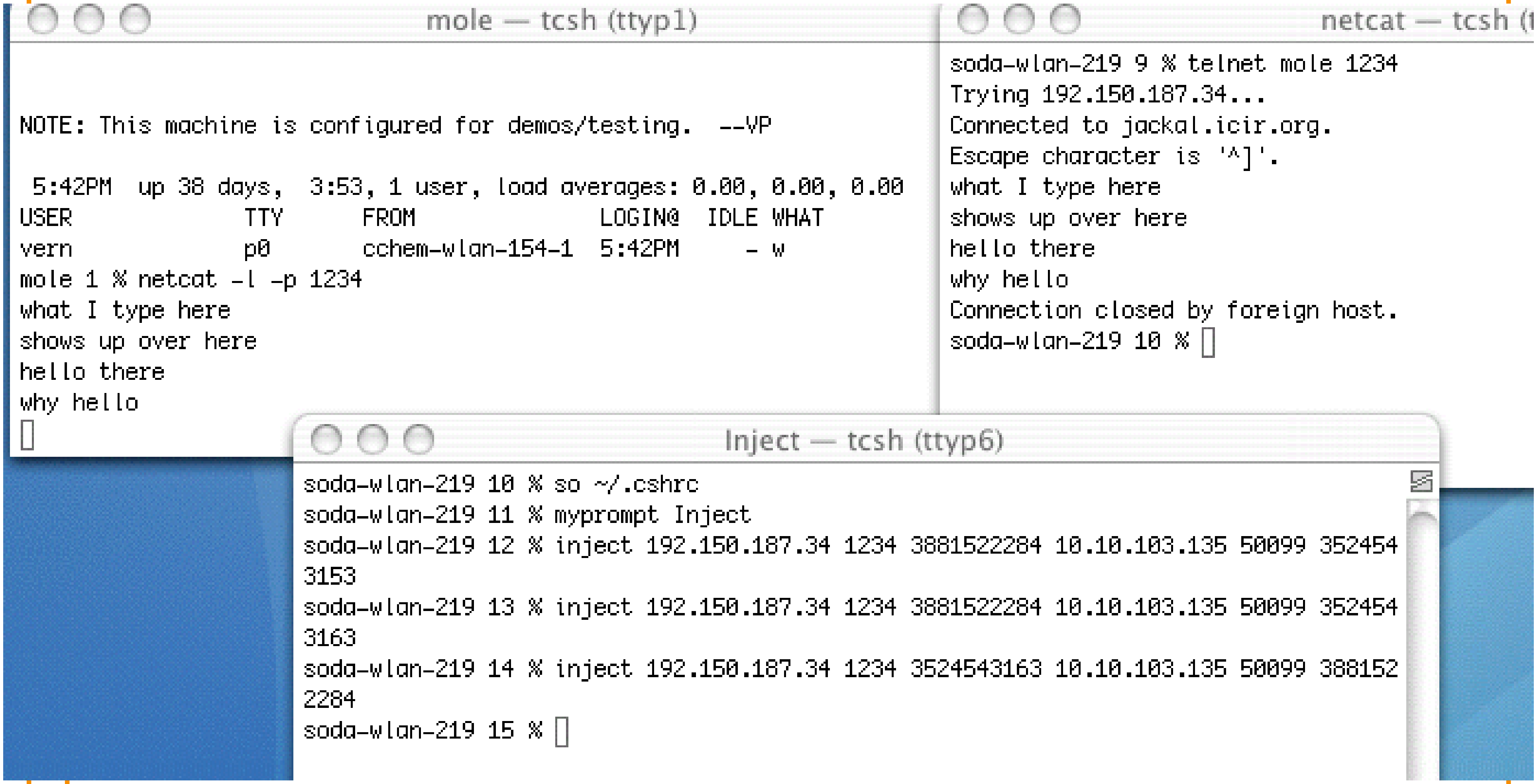
- Approach #1: if you receive an ack for data you haven’t sent, kill the connection
  - Works only if receiver acks too far ahead
- Approach #2: follow the “round trip time” (RTT) and if ack arrives too quickly, kill the connection
  - Flaky: RTT can vary a lot, so you might kill innocent connections
- Approach #3: make the receiver prove they received the data

Note: a *protocol* change

  - Add a **nonce** (“random” marker) & require receiver to include it in ack. Kill connections w/ incorrect nonces
    - (nonce could be function computed over payload, so sender doesn’t explicitly transmit, only implicitly)

Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully **terminate** by forging a RST packet
  - **Inject** data into either direction by forging data packets
  - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
  - *Remains a major threat today*



```
NOTE: This machine is configured for demos/testing.  --VP

 5:42PM  up 38 days,  3:53, 1 user, load averages: 0.00, 0.00, 0.00
USER             TTY      FROM              LOGIN@  IDLE WHAT
vern             p0       cchem-wlan-154-1  5:42PM     - w
mole 1 % netcat -l -p 1234
what I type here
shows up over here
hello there
why hello
```

```
soda-wlan-219 9 % telnet mole 1234
Trying 192.150.187.34...
Connected to jackal.icir.org.
Escape character is '^]'.
what I type here
shows up over here
hello there
why hello
Connection closed by foreign host.
soda-wlan-219 10 %
```

```
soda-wlan-219 10 % so ~/.cshrc
soda-wlan-219 11 % myprompt Inject
soda-wlan-219 12 % inject 192.150.187.34 1234 3881522284 10.10.103.135 50099 352454
3153
soda-wlan-219 13 % inject 192.150.187.34 1234 3881522284 10.10.103.135 50099 352454
3163
soda-wlan-219 14 % inject 192.150.187.34 1234 3524543163 10.10.103.135 50099 388152
2284
soda-wlan-219 15 %
```

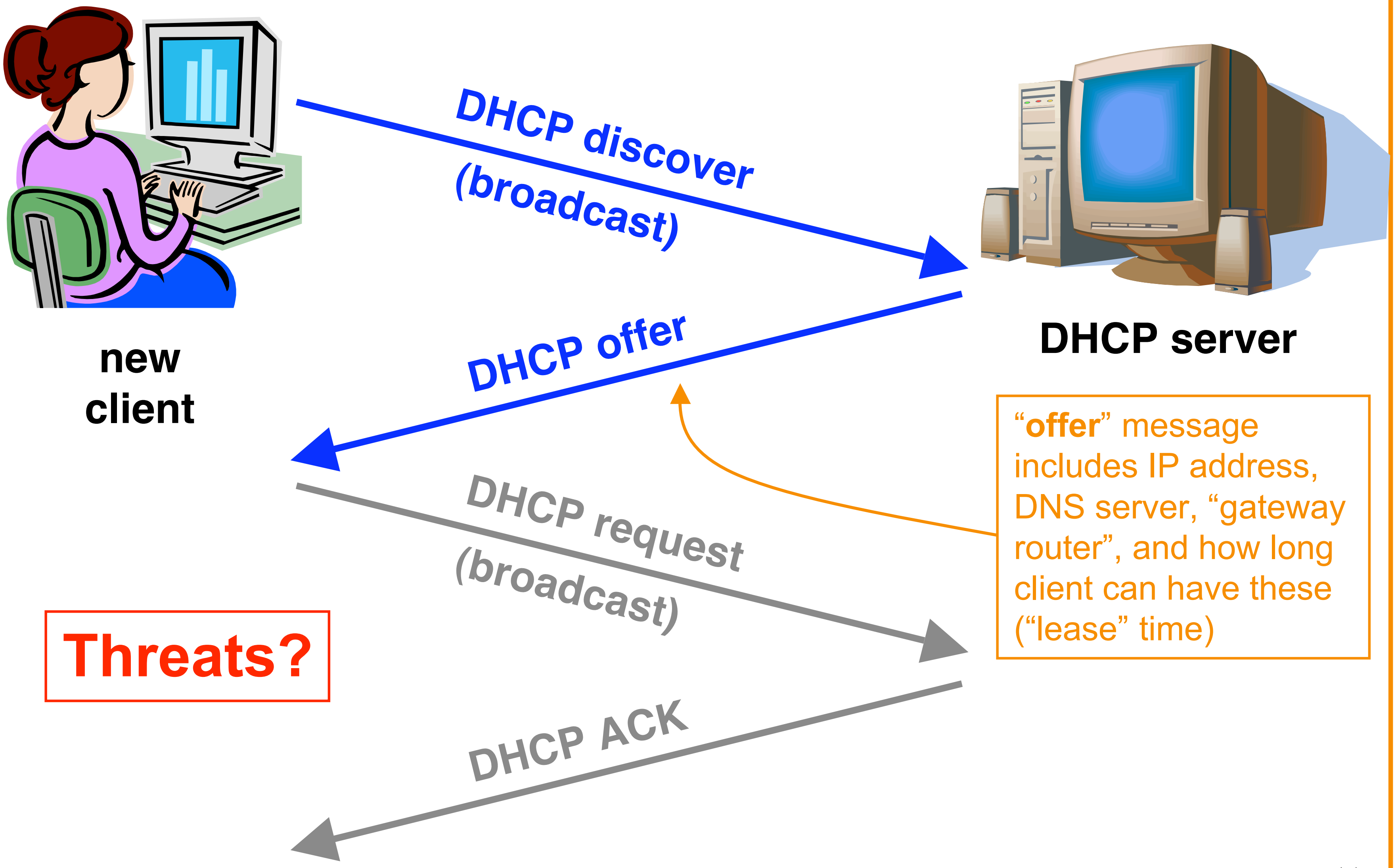
Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully **terminate** by forging a RST packet
  - **Inject** data into either direction by forging data packets
  - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
  - *Remains a major threat today*
- An attacker who can predict the ISN chosen by a server can "blind spoof" a connection to the server
  - Makes it appear that host ABC has connected, and has sent data of the attacker's choosing, when in fact it hasn't
  - *Undermines any security based on trusting ABC's IP address*
  - Allows attacker to "frame" ABC or otherwise avoid detection
  - **Fixed** today by choosing **random** ISNs
- Both highlight flawed "security-by-obscurity" assumption

TCP Security Issues, con't

- TCP limits the rate at which senders transmit:
  - TCP relies on endpoints behaving properly to achieve "fairness" in how network capacity is used
  - Protocol lacks a mechanism to prevent cheating
  - Senders can cheat by just not abiding by the limits
    - Remains a significant threat: essentially nothing today prevents
- Receivers can manipulate honest senders into sending too fast because senders trust that receivers are honest
  - To a degree, sender can validate (e.g., partial acks)
  - A **nonce** can force receiver to only act on data they've seen
  - Rate manipulation remains a threat today
- General observation: tension between ease/power of protocols that assume everyone follows vs. violating
  - Security problems persist due to difficulties of retrofitting …
  - … coupled with **investment in installed base**

Dynamic Host Configuration Protocol



new client

DHCP server

DHCP discover (broadcast)

DHCP offer

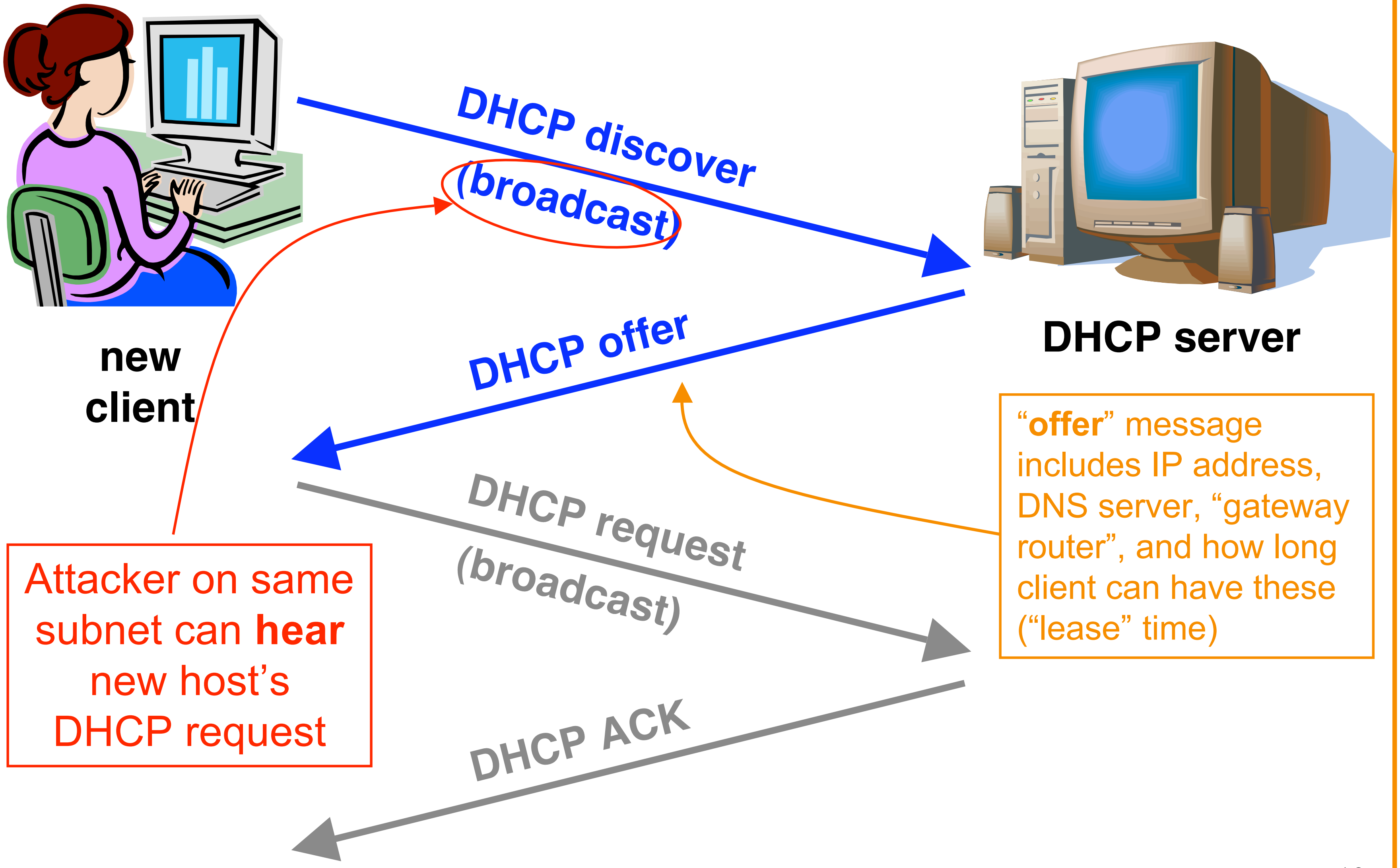
DHCP request (broadcast)

DHCP ACK

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

Threats?

Dynamic Host Configuration Protocol



new client

DHCP server

DHCP discover (broadcast)

DHCP offer

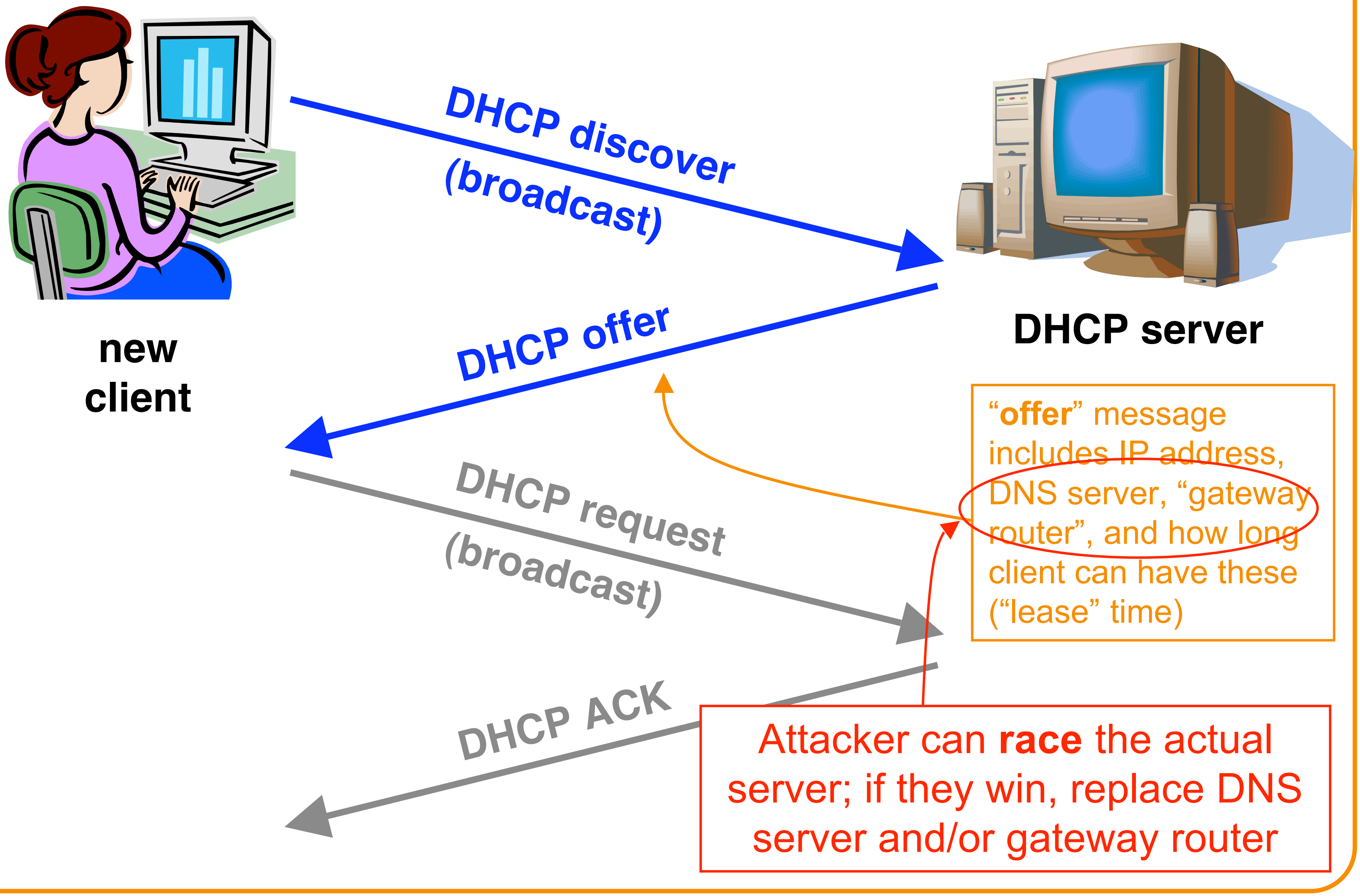
DHCP request (broadcast)

DHCP ACK

Attacker on same subnet can **hear** new host's DHCP request

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

Dynamic Host Configuration Protocol



new client

DHCP server

DHCP discover (broadcast)

DHCP offer

DHCP request (broadcast)

DHCP ACK

“**offer**” message includes IP address, DNS server, “gateway router”, and how long client can have these (“lease” time)

Attacker can **race** the actual server; if they win, replace DNS server and/or gateway router

DHCP Threats

- Substitute a fake DNS server
  - Redirect any of a host's lookups to a machine of attacker's choice
- Substitute a fake "gateway"
  - Intercept all of a host's off-subnet traffic
    - (even if not preceded by a DNS lookup)
  - Relay contents back and forth between host and remote server
    - Modify however attacker chooses
- An invisible "Man In The Middle" (MITM)
  - Victim host has no way of knowing it's happening
    - (Can't necessarily alarm on peculiarity of receiving multiple DHCP replies, since that can happen benignly)
- How can we fix this?

Non-Eavesdropping Threats: DNS

- DHCP attacks show brutal power of attacker who can eavesdrop
- Consider attackers who *can't* eavesdrop - but still aim to manipulate us via how protocols function
- 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
- 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 DNS address (“A”) for hostname `eecs.mit.edu`

```
; ; <<>> 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.

;; 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
```

These are just comments from `dig` itself with details of the request/response

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
```

Transaction identifier

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.                                    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
```

Here the server echoes back the question that it is answering

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; 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 its address 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.

;; 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 record gives the *hostname* of a different name server ("`NS`") for names in `mit.edu.`

We should cache each record for 11,088 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.

;; 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.

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.
```

What happens 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      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.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.                  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      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.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 dutifully store in our cache a mapping of `eecs.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       eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.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.

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

;; ADDITIONAL SECTION:
eecs.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: +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      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
~~eecs.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

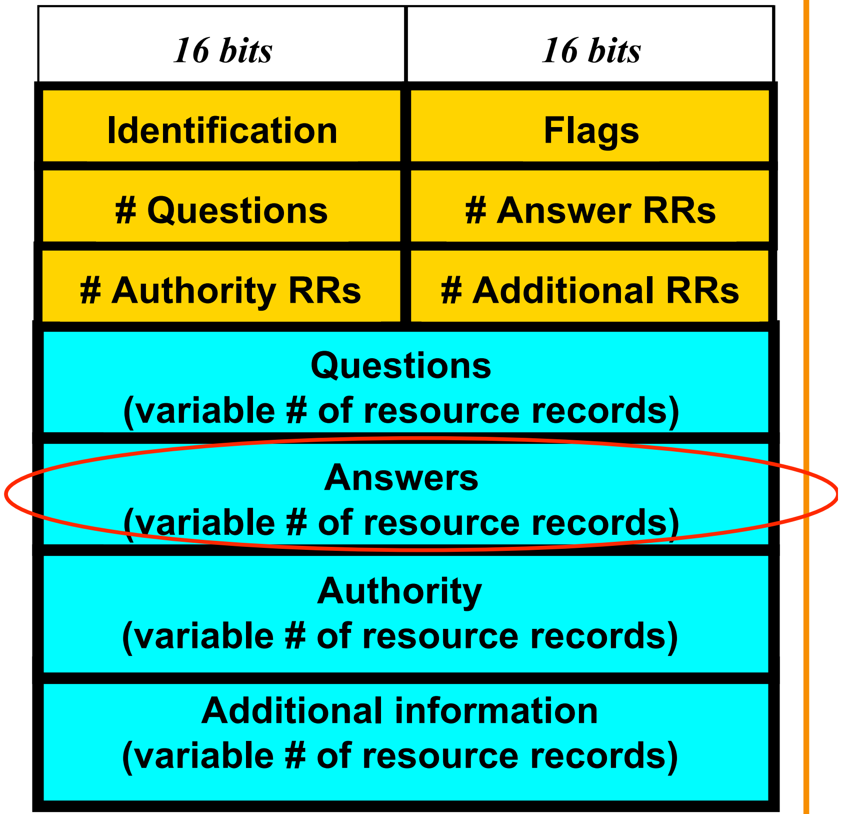
E.g., looking up `eecs.mit.edu` ⇒ only accept additional records from `*.mit.edu`

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

DNS Threats, con’t

What about *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 an attacker even know we are looking up `mail.google.com`?



16 bits | *16 bits*

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://mail.google.com" …>
```

DNS Blind Spoofing, con’t

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*
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)**	

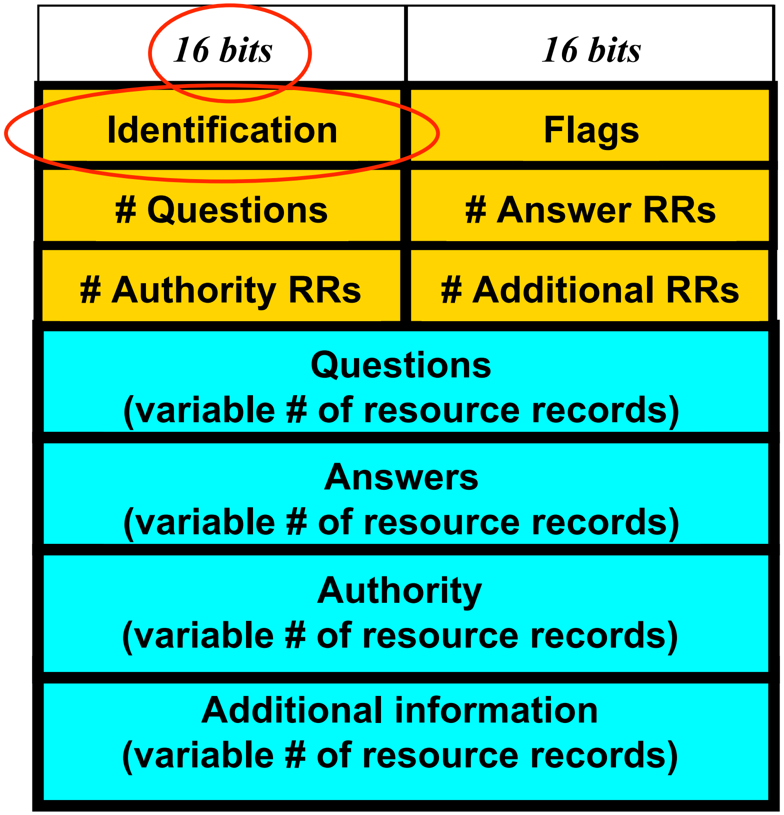
```
<img src="http://badguy.com" …>         ← They observe ID k here
<img src="http://mail.google.com" …>    ← So this will be k+1
```

DNS Blind Spoofing, con’t

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*
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! **?**