CS 161: Computer Security

Lecture 16

October 22, 2015

DNS

- Domain Name Service
- DNS translates domain names to IP addresses
- Performance critical distributed database.
- DNS security critical for the web.
    - (Same-origin policy assumes DNS is secure.)

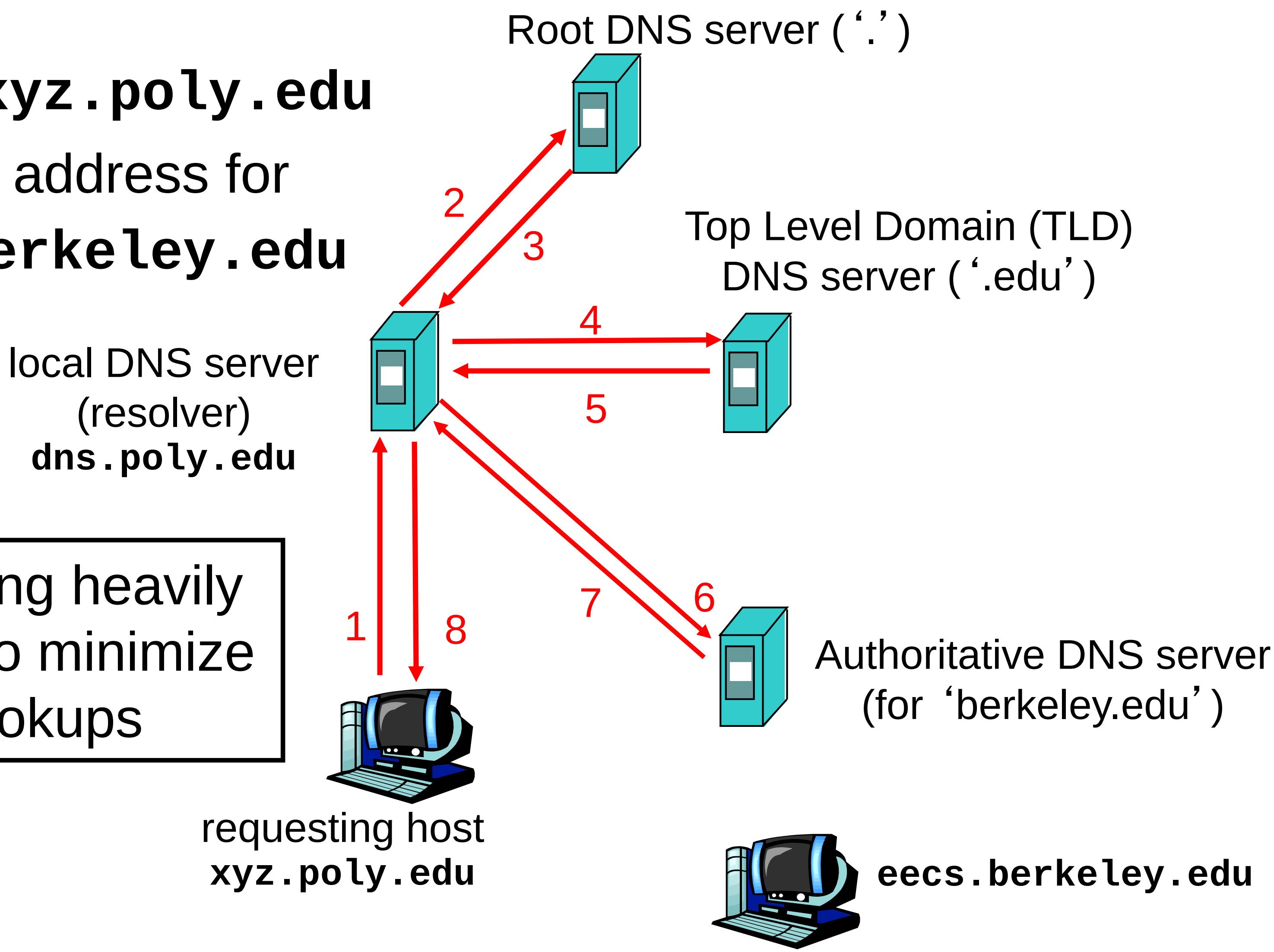
DNS Lookups via a *Resolver*

Host at **`xyz.poly.edu`**

wants IP address for

`eecs.berkeley.edu`

Caching heavily used to minimize lookups



DNS risks

- If *any* queried DNS servers are malicious, they may give incorrect answers
- Eavesdropping may lead to total control
- Spoofed off-path attacks

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400   IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
;; flags: qr aa rd ra; QUERY: 1, ANSWE

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400  IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400  IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400  IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400  IN      A       169.229.60.153
cgl.UCSF.edu.           86400  IN      A       169.230.27.20
adns1.berkeley.edu.     172800 IN      A       128.32.136.3
adns1.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800 IN      A       128.32.136.14
adns2.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::e
```

Use Unix "`dig`" utility to look up IP address for hostname `eecs.berkeley.edu` via DNS

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400   IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

The question we asked the server

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

;; QUESTION SECTION:
;eecs.berkeley.edu.            IN

;; ANSWER SECTION:
eecs.berkeley.edu.      86400  IN

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400  IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400  IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400  IN      A       169.229.60.153
cgl.UCSF.edu.           86400  IN      A       169.230.27.20
adns1.berkeley.edu.     172800 IN      A       128.32.136.3
adns1.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800 IN      A       128.32.136.14
adns2.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::e
```

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.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY,
;; flags: qr aa rd ra; QUERY: 1

;; QUESTION SECTION:
;eecs.berkeley.edu.

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400   IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

“Answer” tells us the IP address associated with `eecs.berkeley.edu` is `128.32.244.172` and we can cache the result for 86,400 seconds

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      8640

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     8640
ns.eecs.berkeley.edu.   8640
cgl.UCSF.edu.           8640
adns1.berkeley.edu.     1728
adns1.berkeley.edu.     3600
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

In general, a single Resource Record (RR) like this includes:

- a DNS name,
- a time-to-live,
- a family (IN for our purposes - ignore),
- a type (A here), and
- an associated value

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-Re
;; global options: +cm
;; Got answer:
;; ->>HEADER<<- opcode
;; flags: qr aa rd ra;

;; QUESTION SECTION:
;eecs.berkeley.edu.

;; ANSWER SECTION:
eecs.berkeley.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 `eecs.berkeley.edu.` We should cache each record for 86,400 seconds.

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

```
;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400   IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.

;; AUTHORITY SECTION:
eecs.berkeley.edu.
eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

"Additional" provides extra information: here, it tells us the IP addresses for the hostnames of the name servers. We add to our cache.

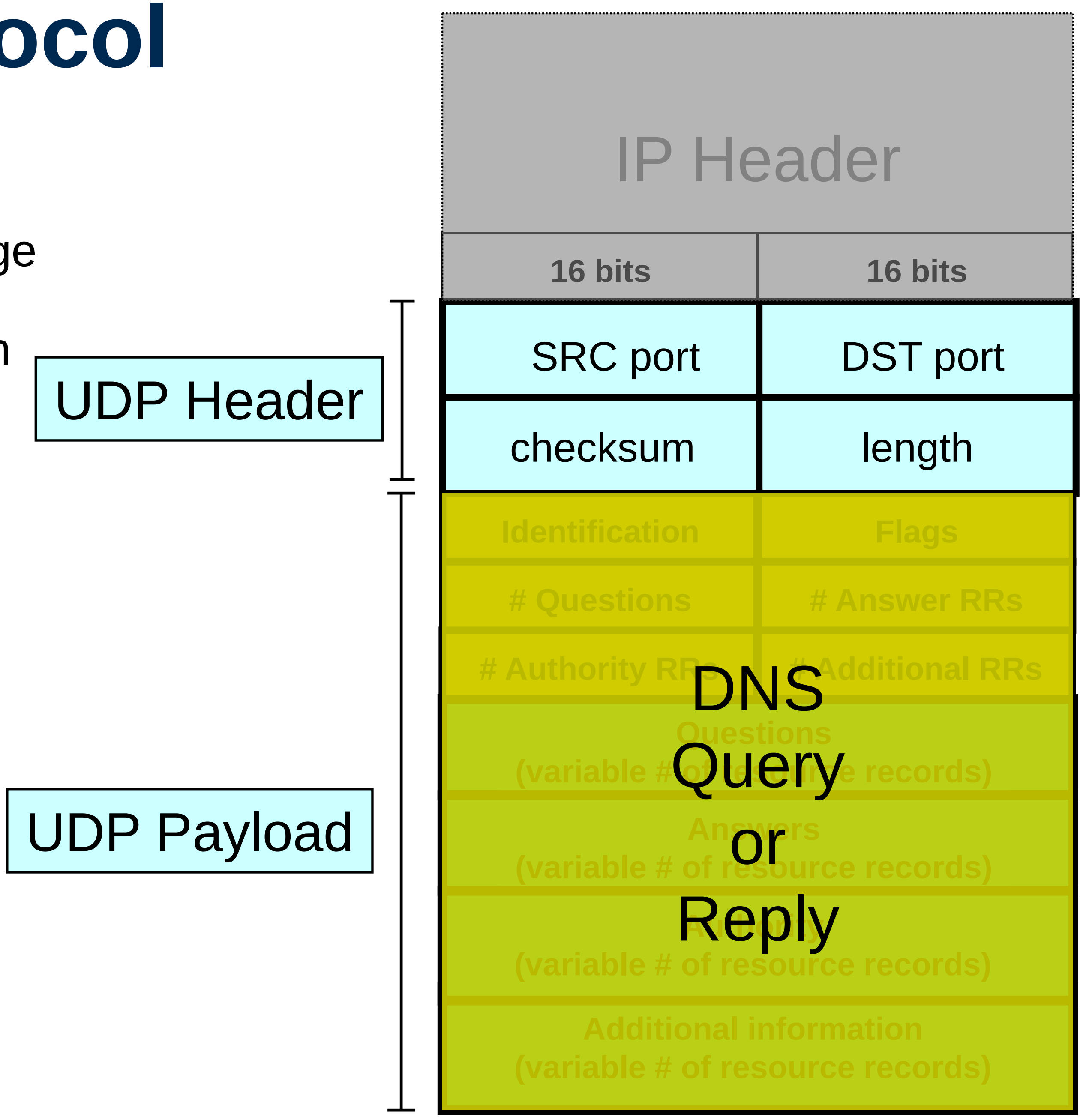
IPv6 NS

DNS Protocol

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

Primarily uses UDP

Frequently, both clients and servers use port 53

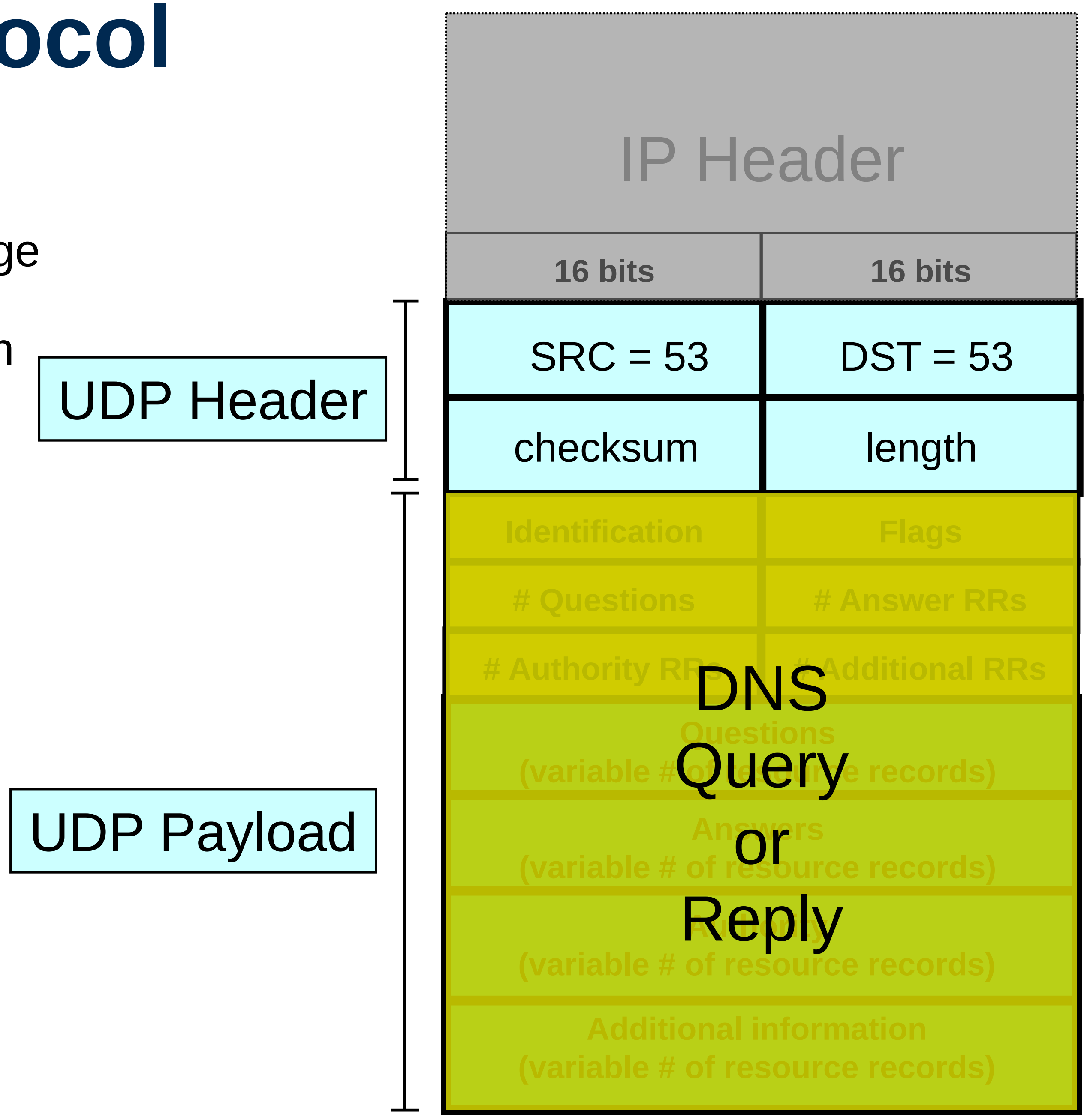


DNS Protocol

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

Primarily uses UDP

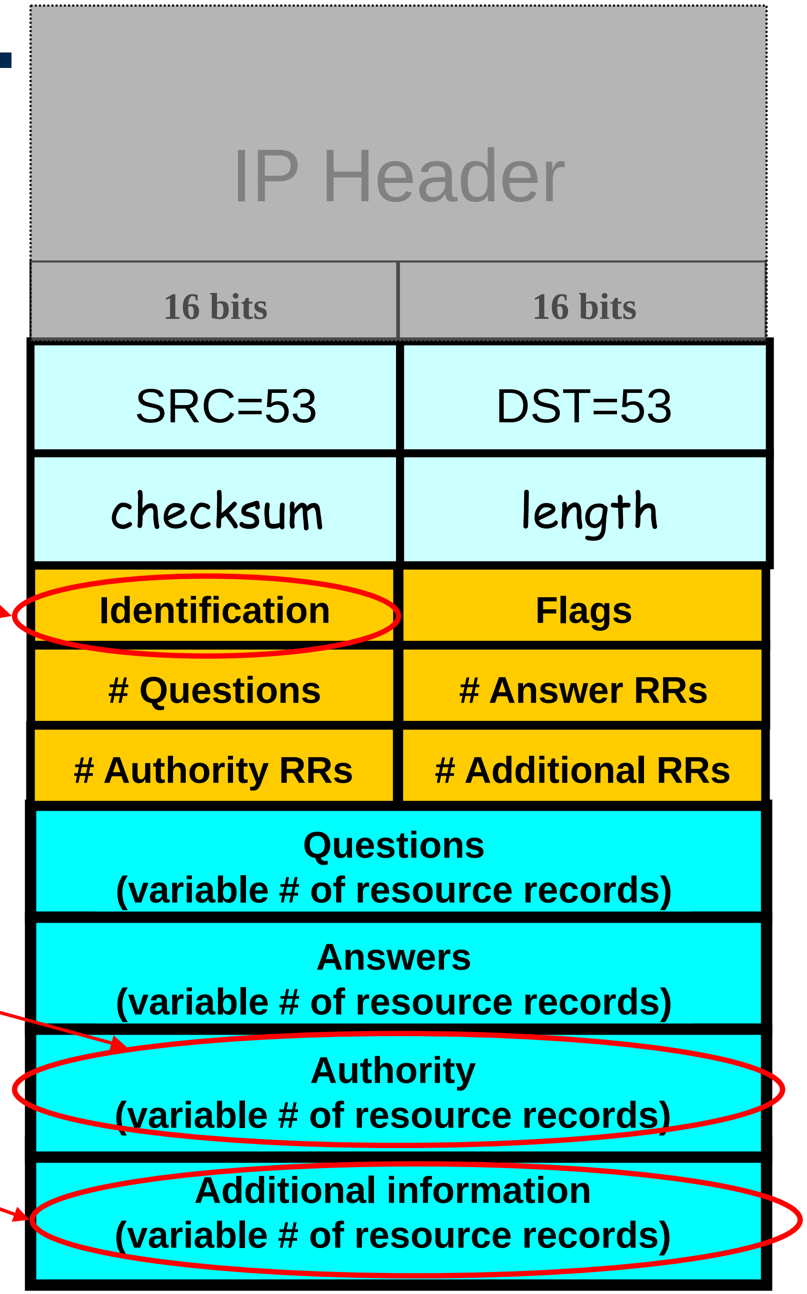
Frequently, both clients and servers use port 53



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.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400   IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

;; QUESTION SECTION:
;eecs.berkeley.edu.            IN

;; ANSWER SECTION:
eecs.berkeley.edu.      86400  IN

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400  IN
eecs.berkeley.edu.      86400  IN
eecs.berkeley.edu.      86400  IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400  IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400  IN      A       169.229.60.153
cgl.UCSF.edu.           86400  IN      A       169.230.27.20
adns1.berkeley.edu.     172800 IN      A       128.32.136.3
adns1.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800 IN      A       128.32.136.14
adns2.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::e
```

What if the **`eecs.berkeley.edu`** server is untrustworthy? Could its operator steal, say, all of our web surfing to another web server?

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

;; QUESTION SECTION:
;eecs.berkeley.edu.            IN

;; ANSWER SECTION:
eecs.berkeley.edu.      86400  IN

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400  IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.CS.berkeley.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     86400  IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400  IN      A       169.229.60.153
cgl.UCSF.edu.           86400  IN      A       169.230.27.20
adns1.berkeley.edu.     172800 IN      A       128.32.136.3
adns1.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800 IN      A       128.32.136.14
adns2.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::e
```

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

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

;; QUESTION SECTION:
;eecs.berkeley.edu.             IN

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400   IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
mit.edu.                30      IN      NS      www.mit.edu.

;; ADDITIONAL SECTION:
www.mit.edu.            30      IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

What could happen if the **eecs.berkeley.edu** server returns the following to us instead?

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      8640

;; AUTHORITY SECTION:
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      8640
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
mit.edu.                30      IN      NS      www.mit.edu.

;; ADDITIONAL SECTION:
www.mit.edu.            30      IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

Client will cache `www.mit.edu` mapping to an IP address under Berkeley's control. (It could have been any IP address we want.)

dig eecs.berkeley.edu

```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400  IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.      86400  IN      NS      cgl.UCSF.edu.
eecs.berkeley.edu.      86400  IN      NS      ns.eecs.berkeley.edu.
eecs.berke                                      berkeley.edu.
eecs.berke                                      berkeley.edu.
mit.edu.                                        .edu.

;; ADDITIONAL SECTION:
www.mit.edu.            30     IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400  IN      A       169.229.60.153
cgl.UCSF.edu.           86400  IN      A       169.230.27.20
adns1.berkeley.edu.     172800 IN      A       128.32.136.3
adns1.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800 IN      A       128.32.136.14
adns2.berkeley.edu.     3600   IN      AAAA    2607:f140:ffff:fffe::e
```

Mapping disappears after 30 seconds. We could make it persist for weeks, or disappear even quicker.

dig eecs.berkeley.edu

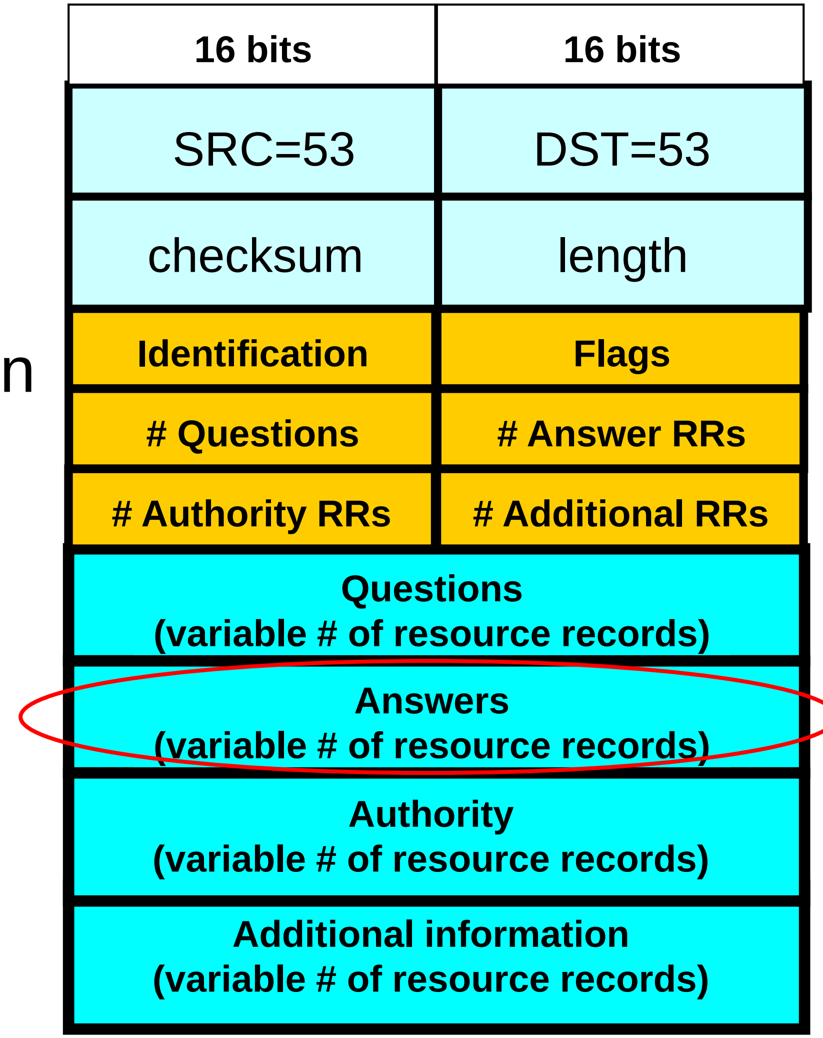
```
; <<>> DiG 9.8.4-P1-RedHat-9.8.4-3.P1.fc16 <<>> eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

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

;; ANSWER SECTION:
eecs.berkeley.edu.      86400   IN      A       128.32.244.172

;; AUTHORITY SECTION:
eecs.berkeley.edu.                                    u.
eecs.berkeley.edu.      86400   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      86400   IN      NS      adns2.berkeley.edu.
mit.edu.                30      IN      NS      www.mit.edu.

;; ADDITIONAL SECTION:
www.mit.edu.            30      IN      A       169.229.60.61
ns.eecs.berkeley.edu.   86400   IN      A       169.229.60.153
cgl.UCSF.edu.           86400   IN      A       169.230.27.20
adns1.berkeley.edu.     172800  IN      A       128.32.136.3
adns1.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     172800  IN      A       128.32.136.14
adns2.berkeley.edu.     3600    IN      AAAA    2607:f140:ffff:fffe::e
```

Cache poisoning!

Eavesdropping

- An eavesdropper can see our query and 16 bit transaction identifier
- Race to send a spoofed response
- Only partially mitigated

Blind spoofing



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

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

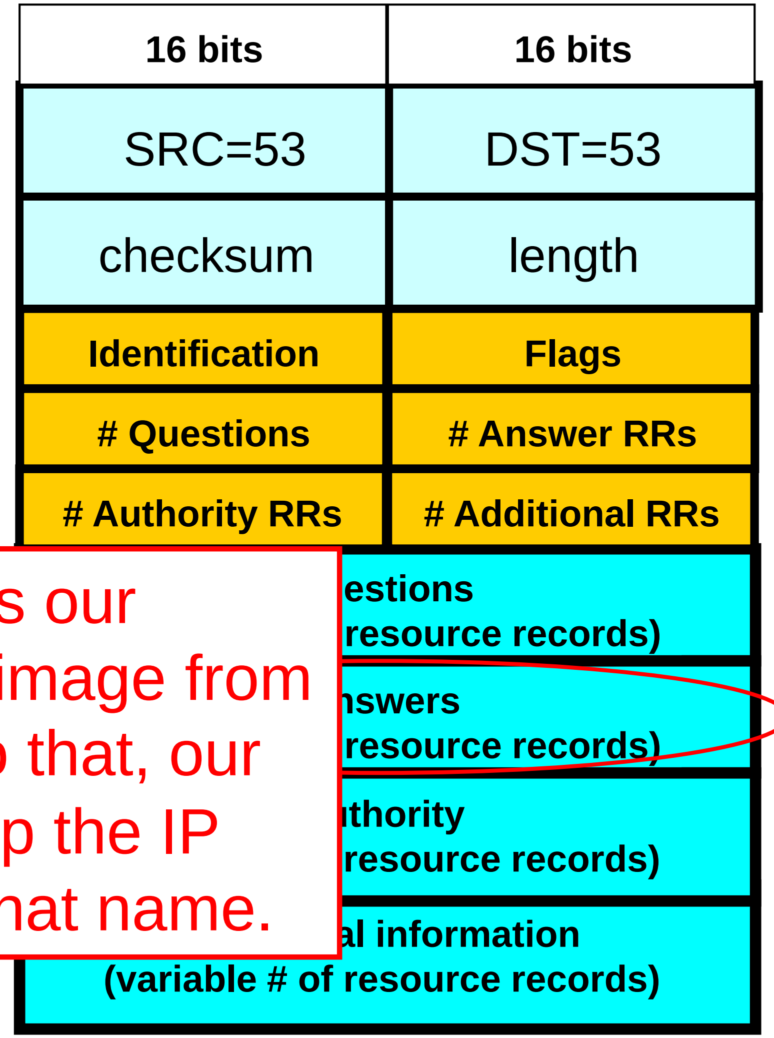
Suppose, e.g., we visit a web page under its control:

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

Blind spoofing

- If we look up `mail.google.com`; how can an **off-path** attacker feed us a bogus answer before the legitir
- How (even mail.

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.



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

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

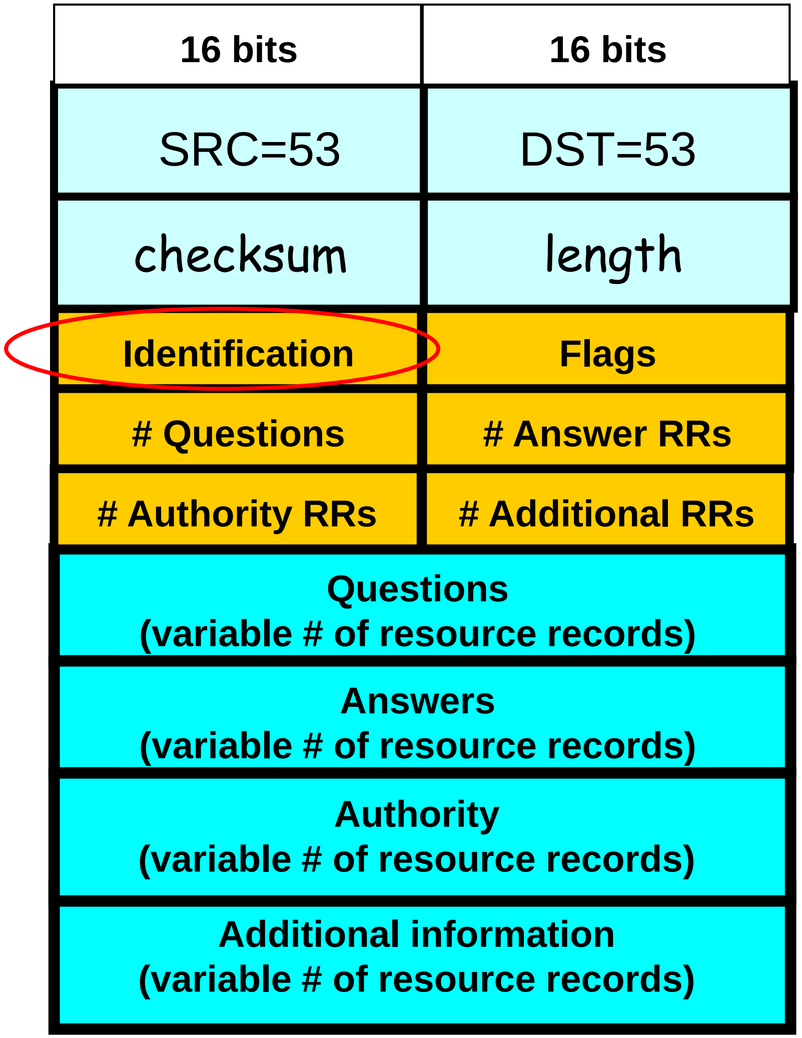
Blind spoofing

Fix?

Once attacker knows we are 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

Blind spoofing

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)	

DNS Blind Spoofing (Kaminsky)

- 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

Flooding with responses

- Suppose attacker can generate 50 forged replies for each random query
- Odds are 1/65536 but repetition wins the day
- If repeated using automated tools can take over in ~10 seconds

Kaminsky Blind Spoofing

For each lookup of `randomk.google.com`, attacker spoofs many 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 attacker wins the race, not only has it poisoned `mail.google.com` …

Kaminsky Blind Spoofing

For each lookup of `randomk.google.com`, attacker spoofs many 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 attacker wins the race, not only has it 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

Kaminsky Blind Spoofing

For each lookup of `randomk.google.com`, attacker spoofs many 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 attacker wins the race, not only has it 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.)

Total entropy: 16 bits

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

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

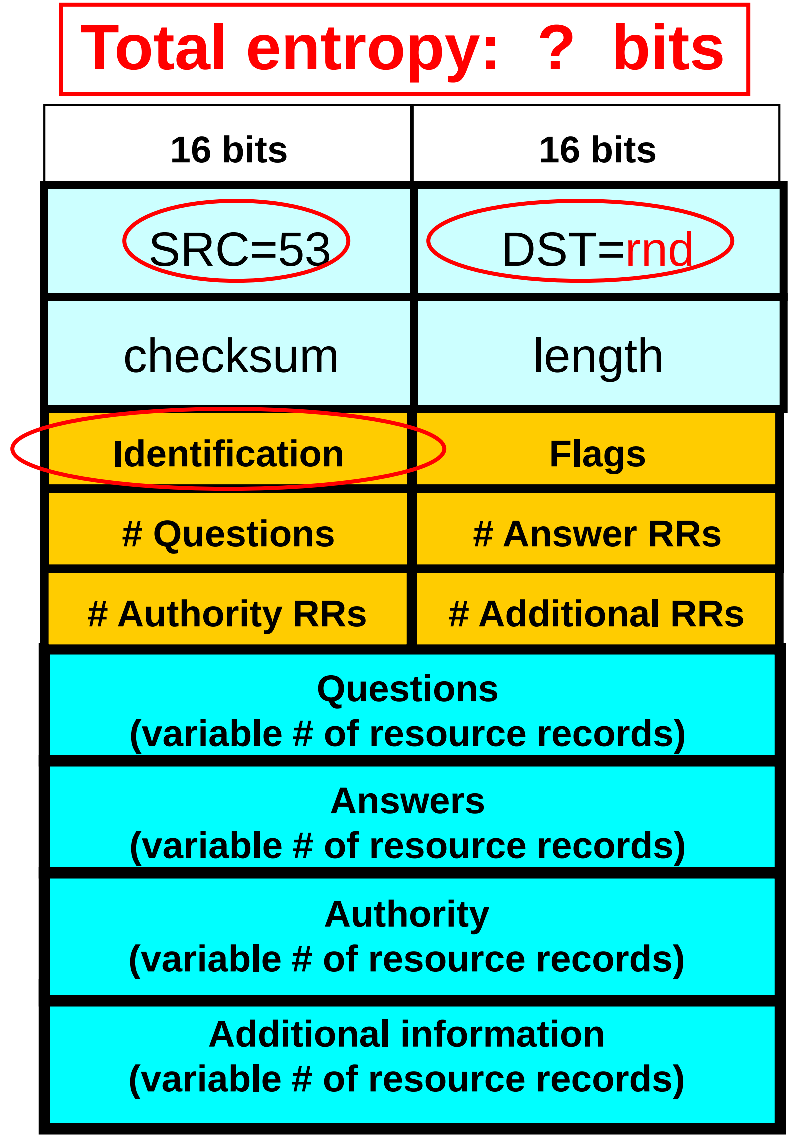
On a request, DST port = 53. SRC port usually also 53 - but not fundamental, just convenient.

Total entropy: 16 bits

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

"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.

Total entropy: 32 bits

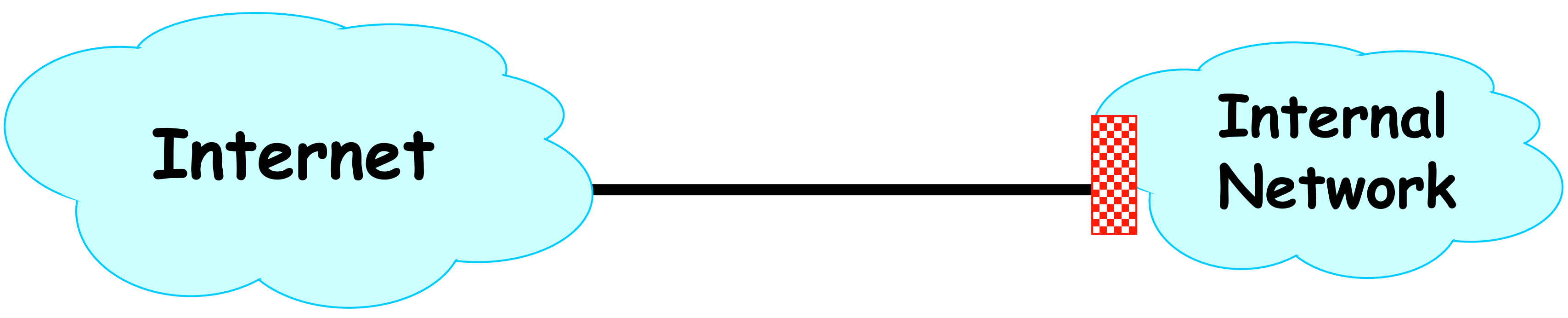
16 bits	16 bits
SRC=53	DST=rnd
checksum	length
Identification	**Flags**
# Questions	**# Answer RRs**
# Authority RRs	**# Additional RRs**
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable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

Firewalls

- Harden set of systems against external attack
- More network services → greater risk
  - Larger attack surface
- Can turn off unnecessary services
  - Requires knowledge of all services running
  - Sometimes trusted users require access
- Scaling issues
  - Hundreds/thousands of systems
  - Many different operating systems, hardware, users

Taming Management Complexity

- Possibly more scalable defense: Reduce risk by blocking in the network outsiders from having unwanted access your network services
  - Interpose a firewall the traffic to/from the outside must traverse
  - Chokepoint can cover thousands of hosts
    - Where in everyday experience do we see such chokepoints?



Selecting a Security Policy

- Firewall enforces an (access control) policy:
  - Who is allowed to talk to whom, accessing what service?
- Distinguish between inbound & outbound connections
  - Inbound: attempts by external users to connect to services on internal machines
  - Outbound: internal users to external services
  - Why? Because fits with a common threat model. There are thousands of internal users (and we've vetted them). There are billions of outsiders.
- Conceptually simple access control policy:
  - Permit inside users to connect to any service
  - External users restricted:
    - Permit connections to services meant to be externally visible
    - Deny connections to services not meant for external access

Default policies

- Default allow
  - Begin by permitting external access to services
  - Turn off as problems recognized
- Default deny
  - Begin by denying external access to services
  - Turn on access on case-by-case basis
- Generally we use default deny
  - Flexibility vs conservative design
  - Flaws in default deny are noticed more quickly (less painfully)

Stateful Packet Filter

- Stateful packet filter is a router that checks each packet against security rules and decides to forward or drop it
  - Firewall keeps track of all connections (inbound/outbound)
  - Each rule specifies which connections are allowed/denied (*access control policy*)
  - A packet is forwarded if it is part of an allowed connection

Example rule

```
allow tcp connection 4.5.5.4:* -> 3.1.1.2:80
```

- Permits TCP connection that is
  - Initiated by host 4.5.5.4
  - Connecting to port 80 of host 3.1.1.2
- Permits any packet (*) associated with connection
- Firewall keeps table of allowed active connections
  - Checks traffic against table

Example rule

```
allow tcp connection *:*/in -> 3.1.1.2:80/out
```

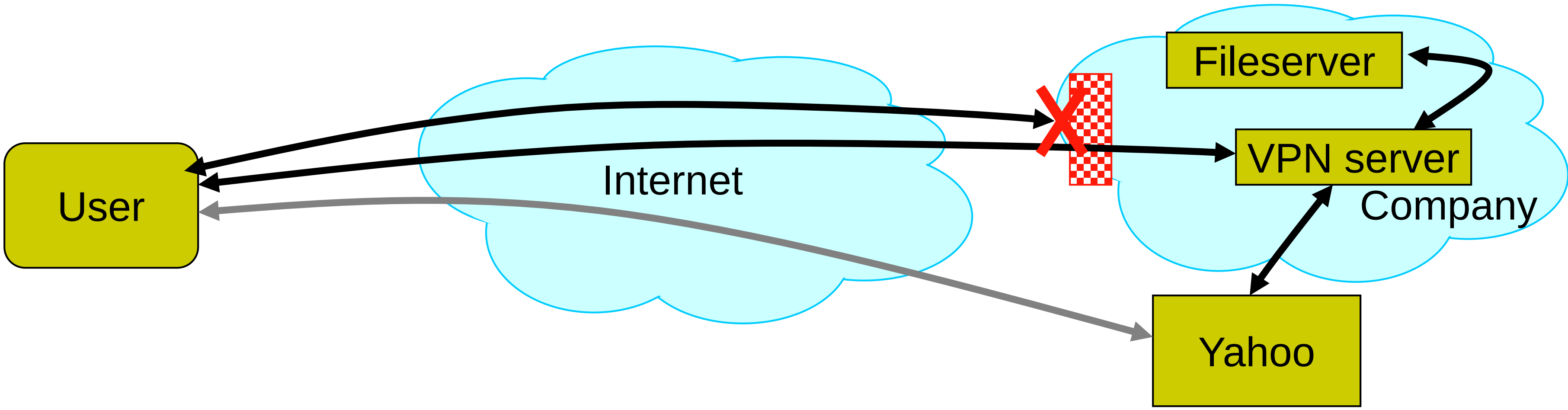
- Permits TCP connection that is
  - Initiated by any internal host (`*:*`)
  - Connecting to port 80 of 3.1.1.2 on external network
- Permits any packet (`*`) associated with connection
- `/in` indicates network interface

Example ruleset

```
allow tcp connection *:*/in -> *:*/out
allow tcp connection *:*/out -> 1.2.2.3:80/in
```

- Permits all outbound TCP connections
  - Those initiated by internal hosts
- Permits inbound TCP connection to web server (port 80) at IP address 1.2.2.3

Secure External Access to Inside Machines



- Often need to provide secure remote access to a network protected by a firewall
    - Remote access, telecommuting, branch offices, …
- Create secure channel (Virtual Private Network, or VPN) to tunnel traffic from outside host/network to inside network
    - Provides Authentication, Confidentiality, Integrity
    - However, also raises perimeter issues

(Try it yourself at http://www.net.berkeley.edu/vpn/)

Firewall Advantages

- Central control – easy administration and update
  - Single point of control: update one config to change security policies
  - Potentially allows rapid response
- Easy to deploy – transparent to end users
  - Easy incremental/total deployment to protect 1000's
- Addresses an important problem
  - Security vulnerabilities in network services are rampant
  - Easier to use firewall than to directly secure code …

Firewall Disadvantages

- Functionality loss – less connectivity, less risk
  - May reduce network's usefulness
  - Some applications don't work with firewalls
    - Two peer-to-peer users behind different firewalls
- The malicious insider problem
  - Assume insiders are trusted
    - Malicious insider (or anyone gaining control of internal machine) can wreak havoc
- Firewalls establish a security perimeter
  - Like Eskimo Pies: "hard crunchy exterior, soft creamy center"
  - Threat from travelers with laptops, cell phones, …