Admin

- Assignment 0 is out!
What happens when you type a URL and press enter?
Domain Name System (DNS)
Client

stanford.edu?

171.67.215.200

DNS Server
How does the "DNS server" work?
Client

DNS Recursive Resolver
Client \(\text{stanford.edu?}\) DNS Recursive Resolver
DNS

Client

stanford.edu?

DNS Recursive Resolver

Root Nameserver
Client \( \rightarrow \) DNS Recursive Resolver \( \rightarrow \) Root Nameserver

- Client queries for `stanford.edu`.
- Recursive Resolver queries `stanford.edu`.
- Recursive Resolver sends the query to the Root Nameserver.
Client ➔ DNS Recursive Resolver ➔ Root Nameserver

stanford.edu?

See ".edu" NS

".edu" Nameserver
DNS

Client

stanford.edu?

DNS Recursive Resolver

stanford.edu?

Root Nameserver

See ".edu" NS

".edu" Nameserver

See "stanford.edu" NS
What happens when you type a URL and press enter?

1. **Client** asks **DNS Recursive Resolver** to lookup a hostname (**stanford.edu**).

2. **DNS Recursive Resolver** sends DNS query to **Root Nameserver**
   - **Root Nameserver** responds with IP address of **TLD Nameserver** (".edu" Nameserver)

3. **DNS Recursive Resolver** sends DNS query to **TLD Nameserver**
   - **TLD Nameserver** responds with IP address of **Domain Nameserver** ("stanford.edu" Nameserver)

4. **DNS Recursive Resolver** sends DNS query to **Domain Nameserver**
   - **Domain Nameserver** is authoritative, so replies with server IP address.

5. **DNS Recursive Resolver** finally responds to **Client**, sending server IP address (171.67.215.200)
DNS + HTTP

stanford.edu?

DNS Recursive Resolver

Client

Server

1.2.3.4
DNS + HTTP

Client

stanford.edu?

1.2.3.4

DNS Recursive Resolver

Server

1.2.3.4
DNS + HTTP

DNS Recursive Resolver

stanford.edu?

1.2.3.4

HTTP Request

1.2.3.4
DNS + HTTP

Client → DNS Recursive Resolver

stanford.edu?
1.2.3.4

DNS Recursive Resolver → Server

HTTP Request
HTTP Response

Client → Server

1.2.3.4
Attacks on DNS
DNS hijacking

- Attacker changes target DNS record to point to attacker IP address
  - Causes all site visitors to be directed to attacker's web server
- Motivation
  - Phishing
  - Revenue through ads, cryptocurrency mining, etc.
- How do they do it?
DNS hijacking

Client

Hijacked DNS Resolver

Malicious Server
9.9.9.9

Server
1.2.3.4
DNS hijacking

Client → Hijacked DNS Resolver

stanford.edu?

Malicious Server
9.9.9.9

Server
1.2.3.4

Feross Aboukhadjahe
DNS hijacking

Client ➔ Hijacked DNS Resolver ➔ Malicious Server

stanford.edu?
9.9.9.9

9.9.9.9
1.2.3.4
DNS hijacking

Client → Hijacked DNS Resolver
stanford.edu?
9.9.9.9

Hijacked DNS Resolver → Malicious Server
HTTP Request

9.9.9.9

Server
1.2.3.4
DNS hijacking

Client → Hijacked DNS Resolver
stanford.edu?
9.9.9.9

HTTP Request

Hijacked DNS Resolver → Malicious Server
9.9.9.9

HTTP Response

Malicious Server → Server
1.2.3.4
DNS hijacking vectors

- Hijacked recursive DNS resolver (shown previously)
- Hijacked DNS nameserver
- Compromised user account at DNS provider
- Malware changes user's local DNS settings
- Hijacked router
86% of Education Industry Experienced DNS Attack in Past Year

The education industry also has the lowest adoption of network security policy management automation at only 8%, according to a new report.
DNS privacy

- Queries are in plaintext
- ISPs have been known to sell this data
- **Pro tip:** Consider switching your DNS settings to 1.1.1.1 or another provider with a good privacy policy
What’s next in making Encrypted DNS-over-HTTPS the Default

Selena Deckelmann | September 6, 2019

In 2017, Mozilla began working on the DNS-over-HTTPS (DoH) protocol, and since June 2018 we’ve been running experiments in Firefox to implement DoH on all HTTPS connections. As of this writing, we have millions of users on the network, and we have been learning a lot along the way.
What happens when you type a URL and press enter?
HTTP

Client → Request → Server
Demo: Make an HTTP request
Demo: Make an HTTP request

curl https://twitter.com

curl https://twitter.com > twitter.html

open twitter.html
HTTP request

GET / HTTP/1.1
Host: twitter.com
User-Agent: Mozilla/5.0 ...
GET / HTTP/1.1

Method	Path	Protocol Version
HTTP response

HTTP/1.1 200 OK
Content-Length: 9001
Content-Type: text/html; charset=UTF-8
Date: Tue, 24 Sep 2019 20:30:00 GMT

<!DOCTYPE html ...
HTTP/1.1 200 OK

Protocol Version Status Code Status Message
HTTP

- **Client-server model** - Client asks server for resource, server replies
- **Simple** - Human-readable text protocol
- **Extensible** - Just add HTTP headers
- **Transport protocol agnostic** - Only requirement is reliability
- **Stateless** - Two requests have no relation to each other
HTTP is stateless?

- Obviously, we interact with "stateful" servers all the time
- "Stateless" means the HTTP protocol itself does not store state
- If state is desired, is implemented as a layer on top of HTTP
HTTP Status Codes

- **1xx** - Informational ("Hold on")
- **2xx** - Success ("Here you go")
- **3xx** - Redirection ("Go away")
- **4xx** - Client error ("You messed up")
- **5xx** - Server error ("I messed up")
HTTP Success Codes

- **200 OK** - Request succeeded
- **206 Partial Content** - Request for specific byte range succeeded
Range Request

GET /video.mp4 HTTP/1.1
Range: bytes=1000-1499

Response

HTTP/1.1 206 Partial Content
Content-Range: bytes 1000-1499/1000000
HTTP Redirection Codes

- **301 Moved Permanently** - Resource has a new permanent URL
- **302 Found** - Resource temporarily resides at a different URL
- **304 Not Modified** - Resource has not been modified since last cached
HTTP Client Error Codes

- **400 Bad Request** - Malformed request
- **401 Unauthorized** - Resource is protected, need to authorize
- **403 Forbidden** - Resource is protected, denying access
- **404 Not Found** - Ya'll know this one
HTTP Server Error Codes

- **500 Internal Server Error** - Generic server error
- **502 Bad Gateway** - Server is a proxy; backend server is unreachable
- **503 Service Unavailable** - Server is overloaded or down for maintenance
- **504 Gateway Timeout** - Server is a proxy; backend server responded too slowly
HTTP with a proxy server
HTTP with a proxy server

Client → Proxy → Server
HTTP with a proxy server
HTTP with a proxy server

Client -> Proxy
Request --> Request
Response -->
Proxy -> Server
HTTP with a proxy server
HTTP request

GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 ...
Host: example.com
Header Name
Header Value
HTTP headers

- Let the client and the server pass additional information with an HTTP request or response
- Essentially a map of key-value pairs
- Allow experimental extensions to HTTP without requiring protocol changes
Useful HTTP request headers

- **Host** - The domain name of the server (e.g. `example.com`)
- **User-Agent** - The name of your browser and operating system
- **Referer** - The webpage which led you to this page (misspelled)
- **Cookie** - The cookie server gave you earlier; keeps you logged in
- **Range** - Specifies a subset of bytes to fetch
Useful HTTP request headers (pt 2)

- **Cache-Control** - Specifies if you want a cached response or not
- **If-Modified-Since** - Only send resource if it changed recently
- **Connection** - Control TCP socket (e.g. `keep-alive` or `close`)
- **Accept** - Which type of content we want (e.g. `text/html`)
- **Accept-Encoding** - Encoding algorithms we understand (e.g. `gzip`)
- **Accept-Language** - What language we want (e.g. `es`)
Demo: Make an HTTP request with headers
Demo: Make an HTTP request with headers

curl https://twitter.com --header "Accept-Language: es" --silent | grep JavaScript

curl https://twitter.com --header "Accept-Language: ar" --silent | grep JavaScript
Demo: User-Agent Examples
HTTP response

HTTP/1.1 200 OK
Content-Length: 9001
Content-Type: text/html; charset=UTF-8
Date: Tue, 24 Sep 2019 20:30:00 GMT

<!DOCTYPE html ...
Useful HTTP response headers

- **Date** - When response was sent
- **Last-Modified** - When content was last modified
- **Cache-Control** - Specifies whether to cache response or not
- **Expires** - Discard response from cache after this date
- **Set-Cookie** - Set a cookie on the client
- **Vary** - List of headers which affect response; used by cache
Vary on user language

HTTP/1.1 200 OK
Cache-Control: public, max-age=31536000
Vary: Accept-Language
Useful HTTP response headers (pt 2)

- **Location** - URL to redirect the client to (used with 3xx responses)
- **Connection** - Control TCP socket (e.g. `keep-alive` or `close`)
- **Content-Type** - Type of content in response (e.g. `text/html`)
- **Content-Encoding** - Encoding of the response (e.g. `gzip`)
- **Content-Language** - Language of the response (e.g. `ar`)
- **Content-Length** - Length of the response in bytes
Demo: Implement an HTTP client

- Not magic!
- Steps:
  - Open a TCP socket
  - Send HTTP request text over the socket
  - Read the HTTP response text from the socket
Im[plem]ent an HTTP client

```javascript
const net = require('net')

const socket = net.createConnection({
    host: 'example.com',
    port: 80
})

const request = `GET / HTTP/1.1
Host: example.com

.slice(1)

socket.write(request)
socket.pipe(process.stdout)
```

79  Feross Aboukhadjeh
Implement an HTTP client (take 2)

```javascript
const dns = require('dns')
const net = require('net')

dns.lookup('example.com', (err, address) => {
  if (err) throw err

  const socket = net.createConnection({
    host: address,
    port: 80
  })

  const request =`
GET / HTTP/1.1
Host: example.com

`.slice(1)

  socket.write(request)
  socket.pipe(process.stdout)
})
```

80 Feross Aboukhadjihe
What happens when you type a URL and press enter?

1. Perform a **DNS lookup** on the hostname *(example.com)* to get an IP address *(1.2.3.4)*

2. Open a **TCP socket** to *1.2.3.4* on port *80* (the HTTP port)

3. Send an **HTTP request** that includes the desired path *(/)*

4. Read the **HTTP response** from the socket

5. Parse the HTML into the DOM

6. Render the page based on the DOM

7. Repeat until all external resources are loaded:
   - If there are pending external resources, make HTTP requests for these (run steps 1-4)
   - Render the resources into the page
DNS Recursive Resolver

Client