

Lecture 25:

Network Security

CSE 123: Computer Networks

Stefan Savage



UCSDCSE

About the final

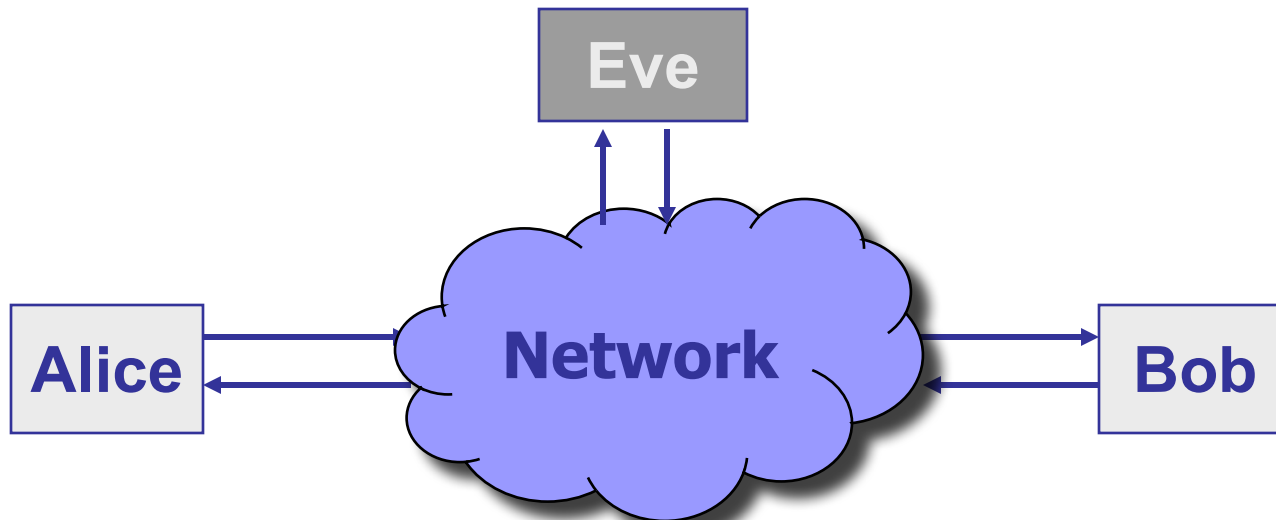
- Similar in style to midterm
 - ◆ Some combination of easy questions, short answer and more in-depth questions
 - ◆ Sample final from 2010 on the Web site
- Closed book/notes, but one note sheet is fine
- Will focus more on second half of class
 - ◆ Up until link-state lecture, material will probably only appear in T/F or short-answer questions
 - ◆ Link-state and on is fair game for more in-depth questions (esp TCP)

Basic TCP/IP Security Issues

- No Authentication/Authorization
 - ◆ Anyone can send to any port on any host; port scanning, Denial of Service (DoS), worms
- No Attribution
 - ◆ Nothing enforces correctness of IP address; IP spoofing
- Network packets not private
 - ◆ Intermediate networks not necessarily trusted; packet sniffing
- TCP/IP state can be easy to guess
 - ◆ TCP connection spoofing, blind port scanning

1. Packet Sniffing

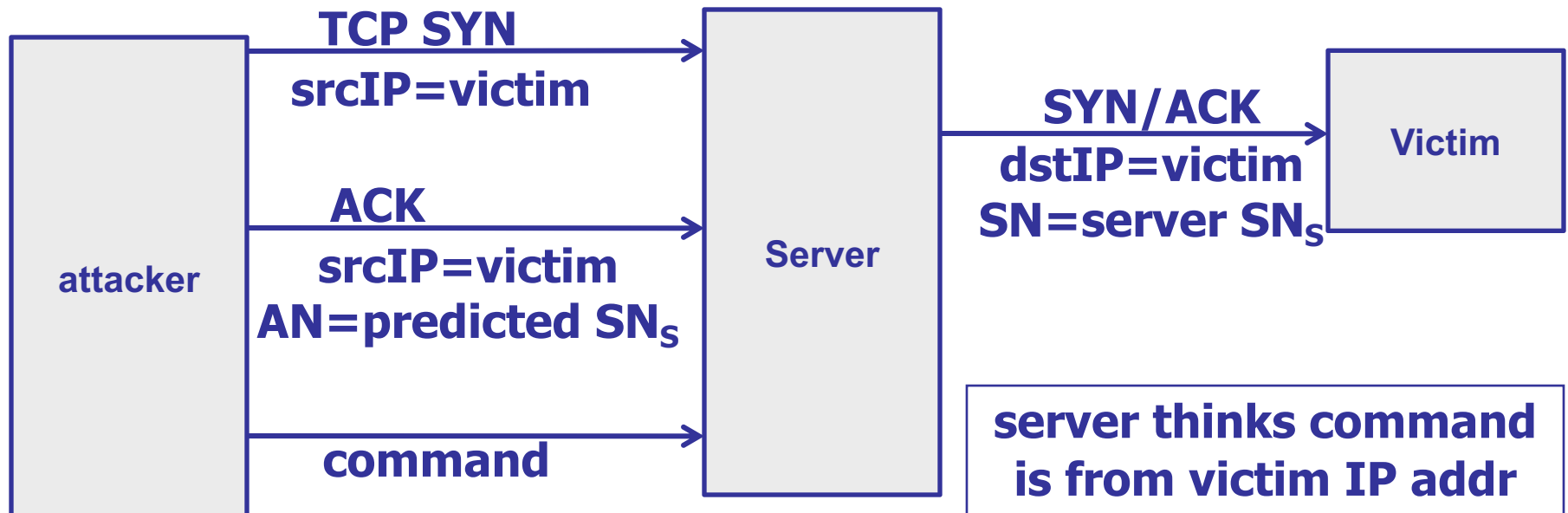
- Promiscuous NIC reads all packets
 - ◆ Read all unencrypted data (e.g., “wireshark”)
 - ◆ ftp, telnet (and POP, IMAP) send passwords in clear!



Prevention: Encryption

2. TCP Connection Spoofing

- Why random initial sequence numbers? (SN_C , SN_S)
- Suppose init. sequence numbers are predictable
 - ♦ Attacker can create TCP session on behalf of forged source IP
 - » Breaks IP-based authentication (e.g. SPF, /etc/hosts)



Example Denial-of-Service (DoS) vulnerability [Watson'04]

- Suppose attacker can guess seq. number for an existing connection:
 - ◆ Attacker can send Reset packet to close connection. Results in DoS.
 - ◆ Naively, success prob. is $1/2^{32}$ (32-bit seq. #'s).
 - ◆ Most systems allow for a large window of acceptable seq. #'s
 - » Much higher success probability.
- Attack is most effective against long lived connections (expensive to set up again; BGP)

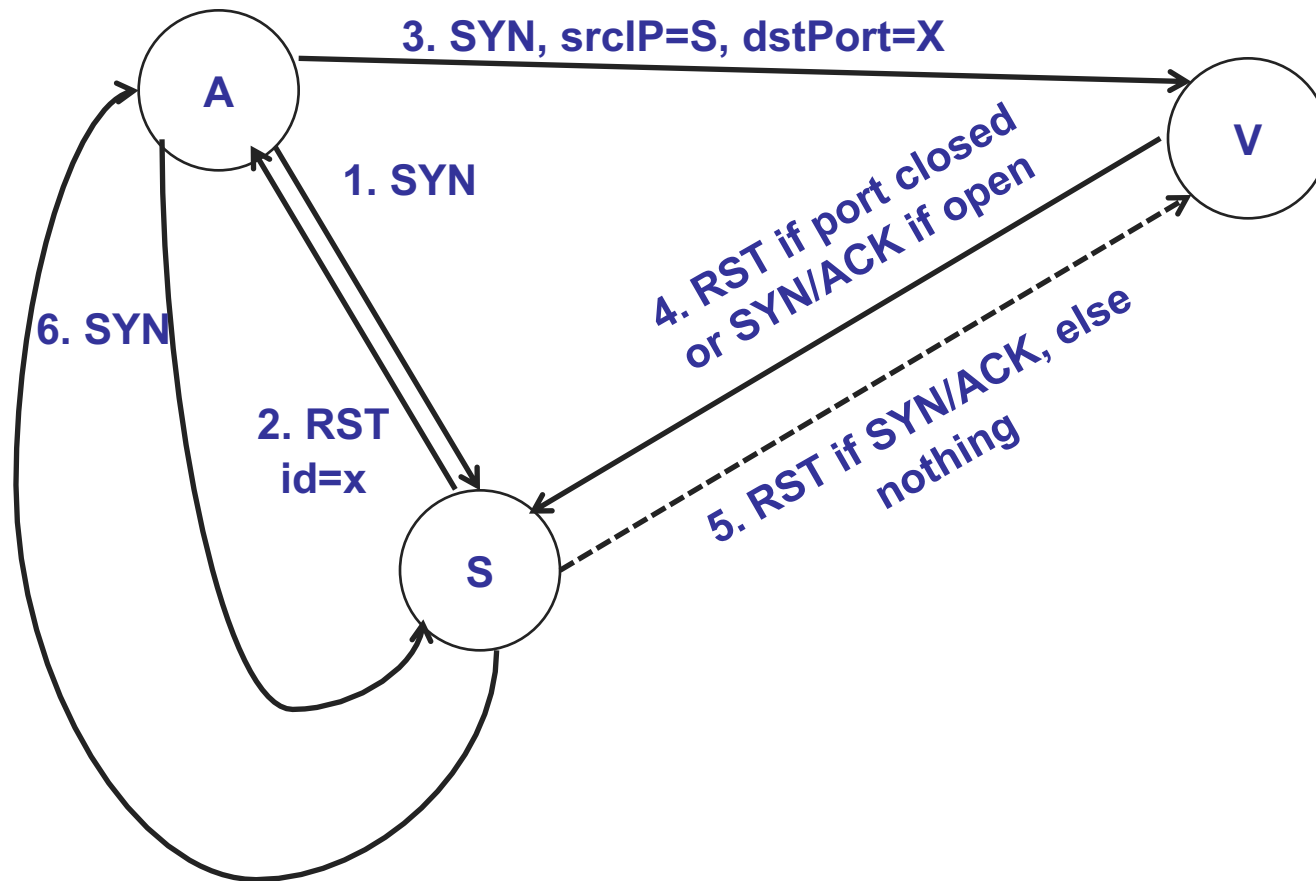
Random initial TCP SNs

- Unpredictable SNs prevent basic packet injection
 - ◆ ... but attacker can inject packets after eavesdropping to obtain current SN
- Most TCP stacks now generate random SNs
 - ◆ Random generator should be unpredictable
 - ◆ GPR'06: Linux RNG for generating SNs is predictable
 - » Attacker repeatedly connects to server
 - » Obtains sequence of SNs
 - » Can predict next SN
 - » Attacker can now do TCP spoofing (create TCP session with forged source IP)

Blind port scanning

- Similar issue: identification field
 - ◆ Hosts typically increment by one after each packet to ensure id field is unique (recall: for fragmentation)
 - ◆ If you receive a pkt from host A at time t_1 with id =10, and another packet at time t_2 with id=12, you can infer... that host A sent another packet somewhere

Blind port scanning



7. Response from S.... What is value of id?
x+1 or x+2?

Protocol Vulnerabilities

- Network protocols are frequently designed assuming all actors are benign
- What if one participant in a communication doesn't obey the protocol?
- Simple example: routing
 - ◆ Nothing prevents UCSD from claiming to "own" 128.95/16 (University of Washington)
 - ◆ This happens, both on purpose and by accident
- Other examples
 - ◆ TCP congestion control
 - ◆ DNS poisoning

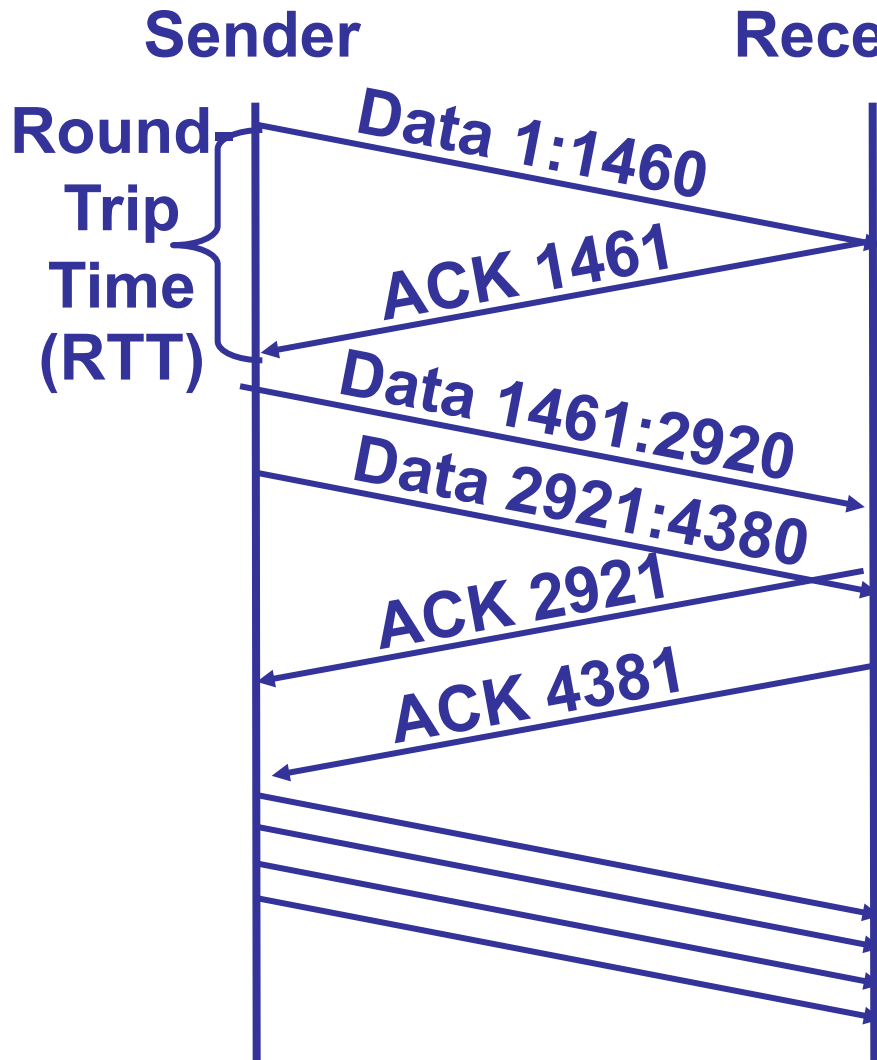
How TCP congestion control works

- Sender maintains “congestion window”
 - ◆ Limit on amount of outstanding data
 - ◆ Grows when data is successfully delivered
 - ◆ Shrinks when data is lost
- Receiver sends ACKs in response to data
 - ◆ ACKs tell sender that data has been received
 - ◆ Indicate the next data item expected
- Works if everyone plays fair
 - ◆ Sender could ignore protocol and send faster
 - ◆ What about receiver? (e.g., Web browser)

Sources of vulnerability

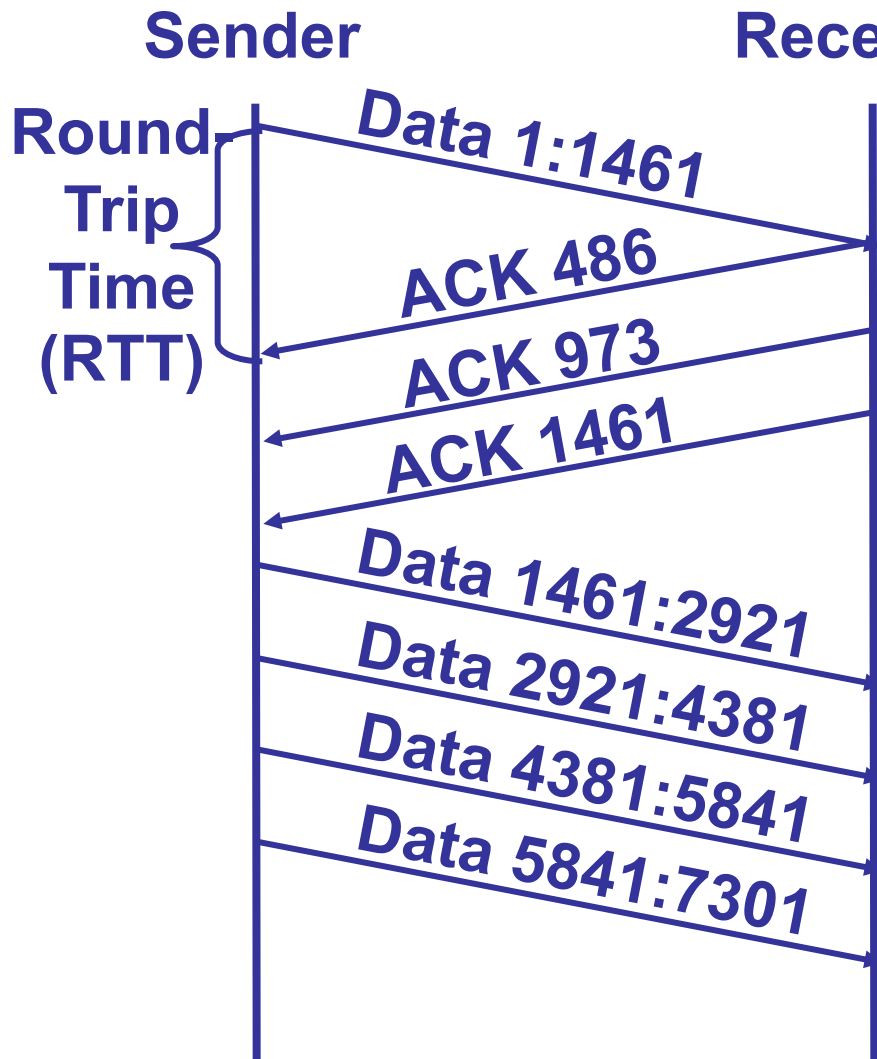
- ACKs *mean* things that they don't *prove*
 - ◆ I was sent in response to a data packet
 - ◆ That data packet has been received
 - ◆ I have received all the data up to X-1
 - ◆ I have (still) not yet received data X
- Sender assumes things that aren't necessarily true
 - ◆ At most one ACK generated per data packet
 - ◆ Every ACK acknowledges a full-sized packet

What's supposed to happen



- Rule: grow window by one full-sized packet for each valid ACK received
- Congestion window doubles each round trip time

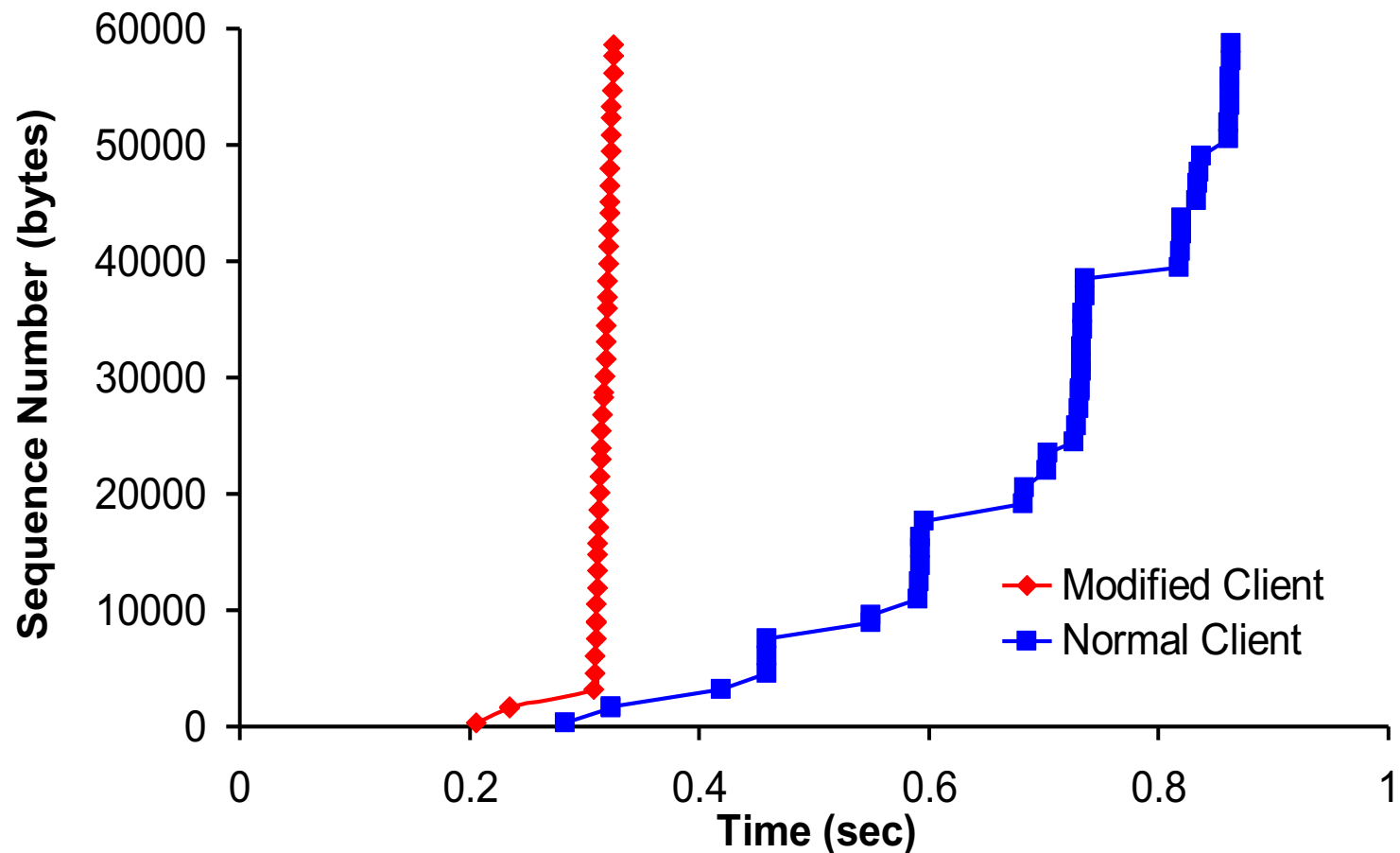
Example of breaking the rules



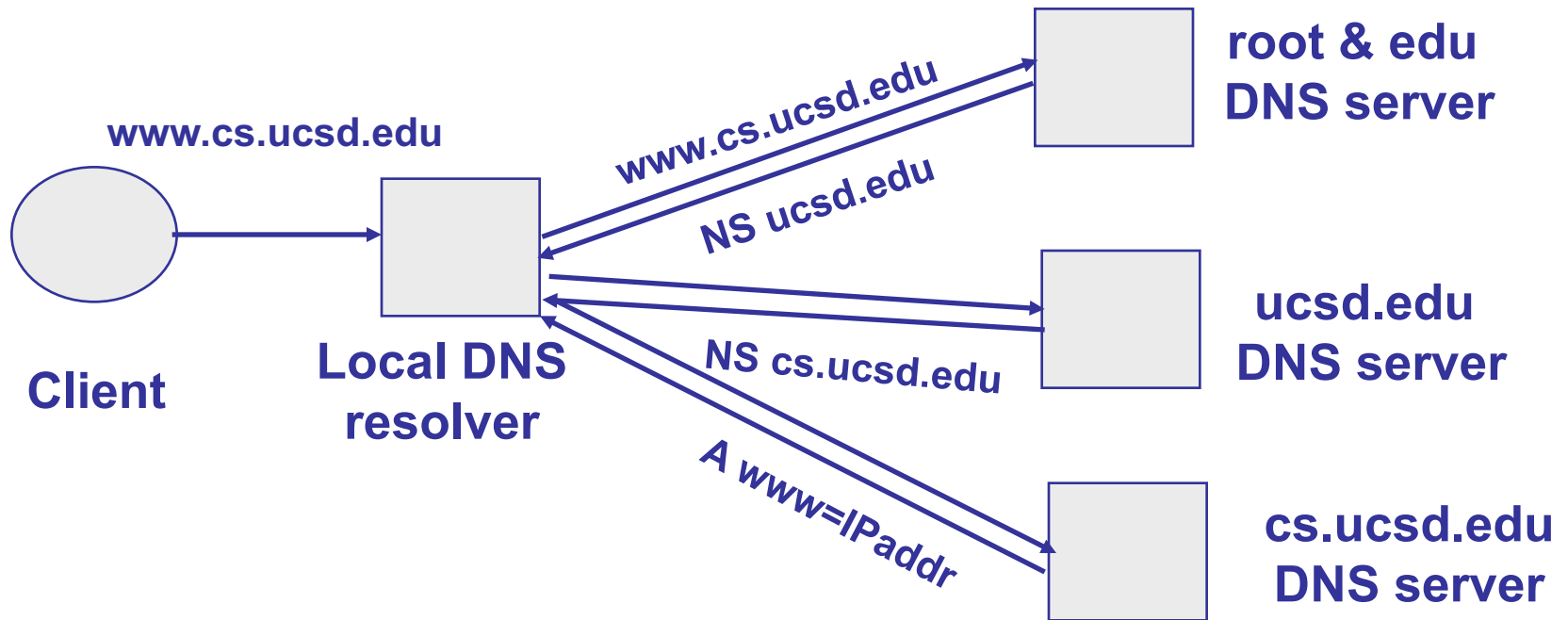
- Rule: grow window by one full-sized packet for each valid ACK received
- Send M ACKs for one pkt
- Growth factor proportional to M

Why my Web browser is faster than yours

Page fetch from CNN.com



Recall how DNS lookup works



DNS record types (partial list):

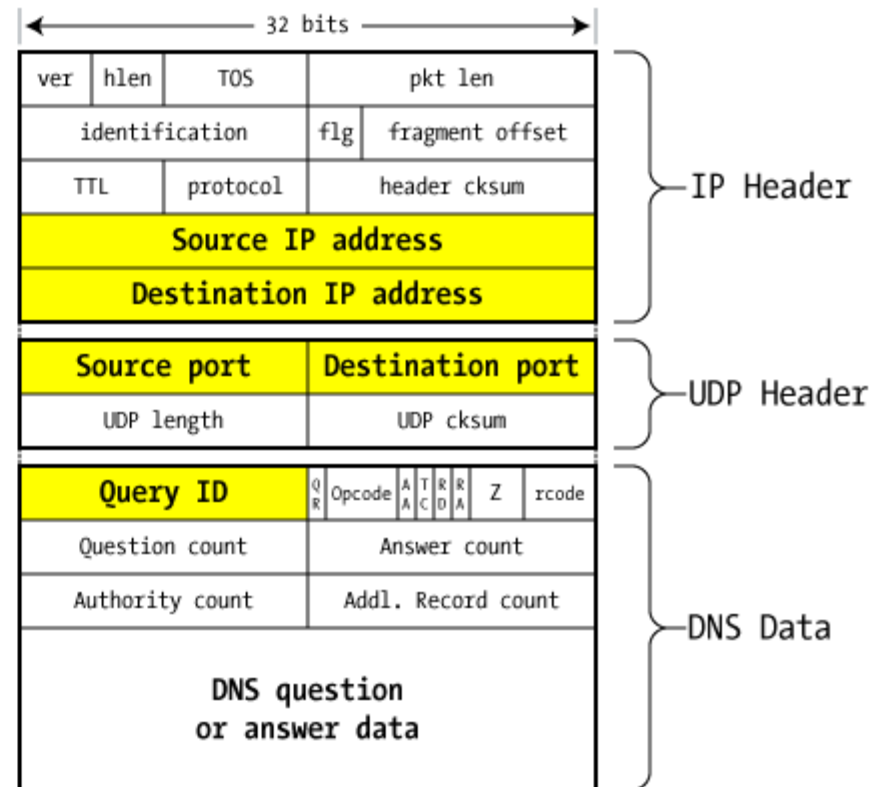
- **NS:** name server (points to other server)
- **A:** address record (contains IP address)
- **MX:** address in charge of handling email

But... what if some one lies?

- Lots of systems and lots of people trust that DNS resolutions are correct
- What if you asked for www.cs.ucsd.edu but I gave you the IP address for www.cs.berkeley.edu instead?
- DNS responses are cached
 - ◆ Lifetime (TTL) of data controlled by responder
 - ◆ A lie could live a long time...
- But how to lie?

DNS Packet

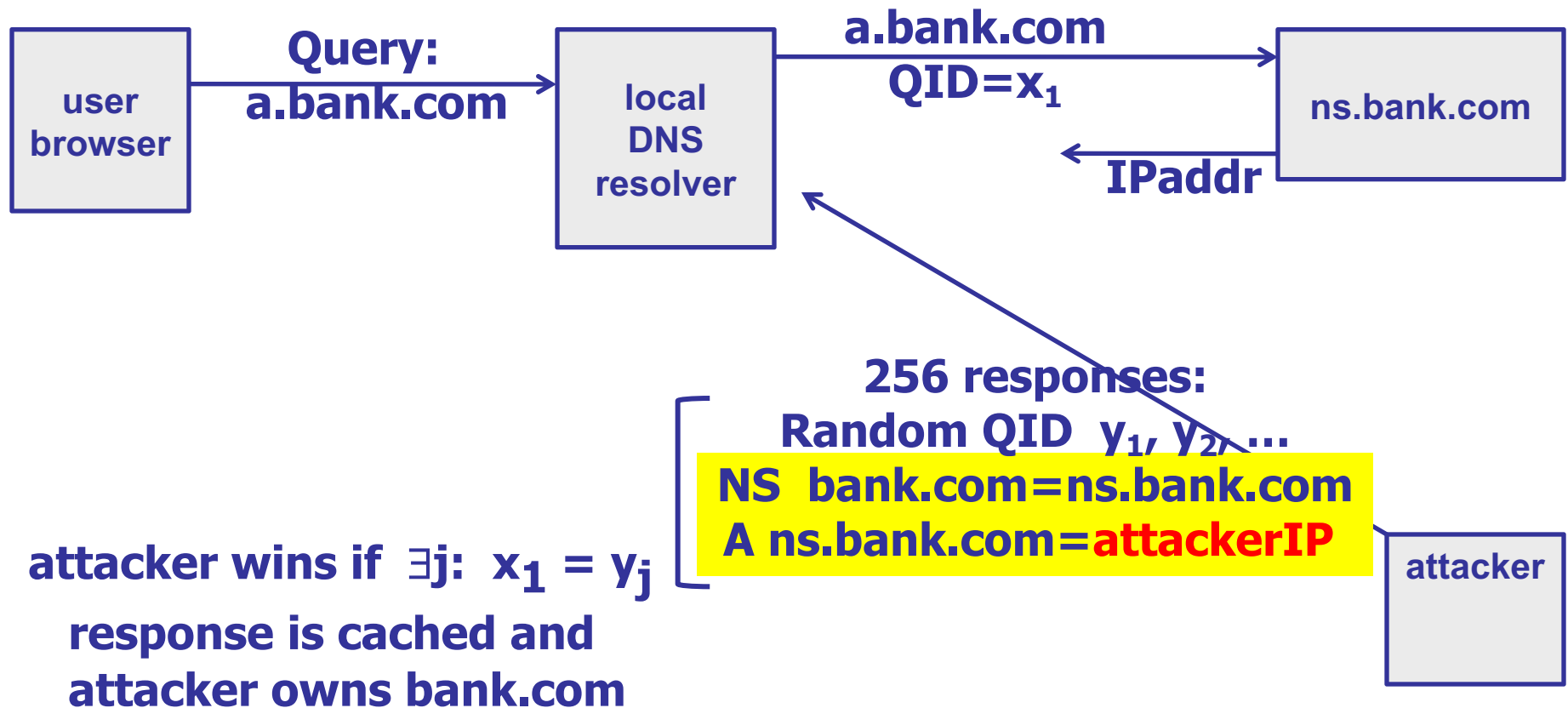
- Query ID:
 - ◆ 16 bit random value
 - ◆ Links response to query
- Response QueryID must match request QueryID or response is rejected
- If you are a “man-in-the-middle” then easy to spoof but tough to do “blind”



(from Steve Friedl)

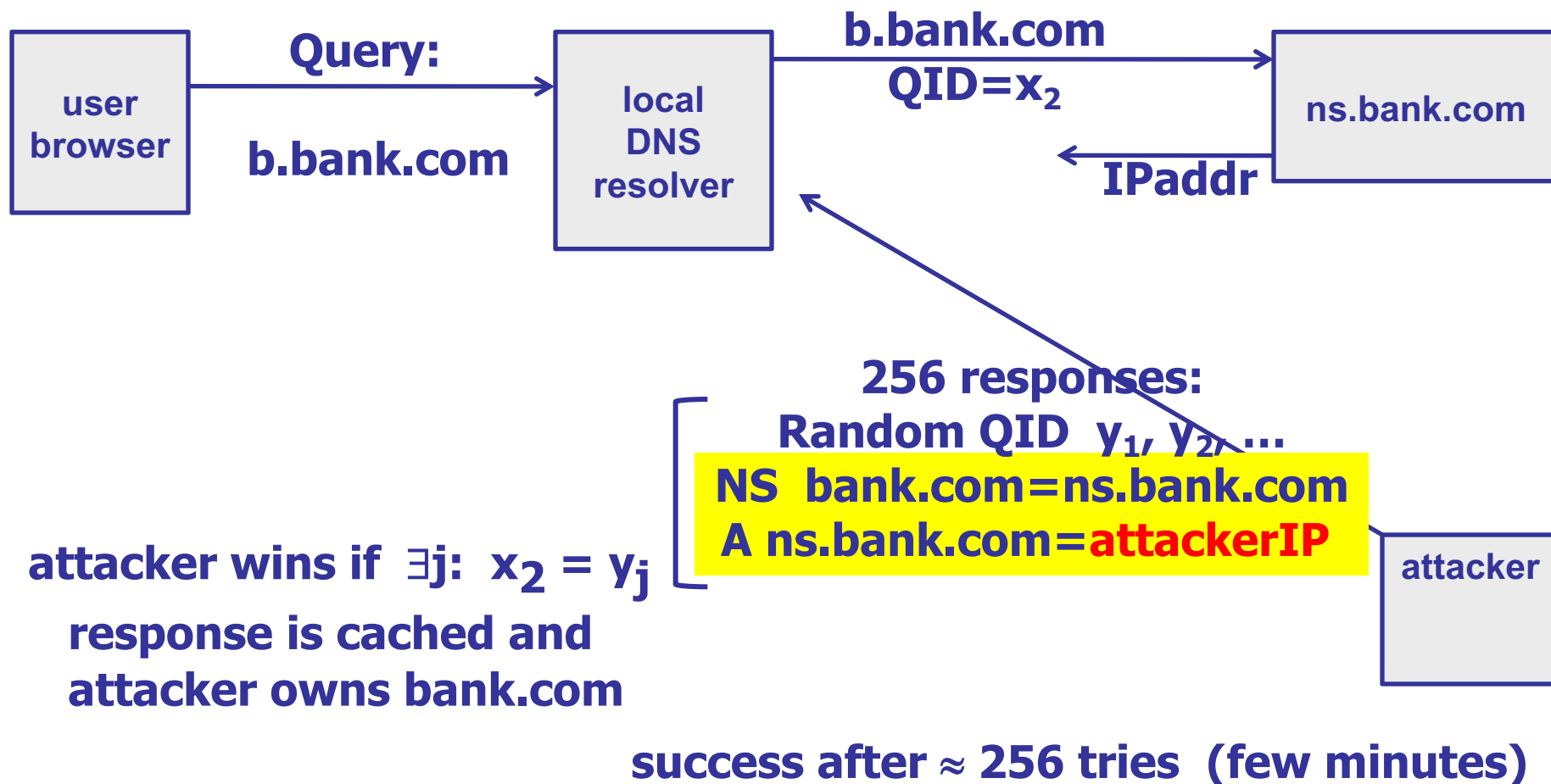
DNS cache poisoning (a la Kaminsky'08)

- Victim machine visits attacker's web site, downloads Javascript



If at first you don't succeed ...

- Victim machine visits attacker's web site, downloads Javascript



Defenses

- Increase Query ID size. How? Some proposals
 - ◆ Randomize src port, additional 11 bits
Now attack takes several hours
 - ◆ Ask every DNS query twice:
 - » Attacker has to guess QueryID correctly twice (32 bits)
 - » Not clear DNS system can handle load
- DNSSec
 - ◆ Authenticated version of DNS that has digital signatures authenticating each response
 - ◆ Not widely deployed

Network Perimeter Defense

- Idea: network defenses on “outside” of organization (e.g., between org and Internet)
 - ◆ Pros/Cons?
- Typical elements
 - ◆ Firewalls
 - ◆ Network Address Translation
 - ◆ Network Intrusion Detection/Prevention Systems(NIDS/NIPS)
(and other kinds of network content analysis)

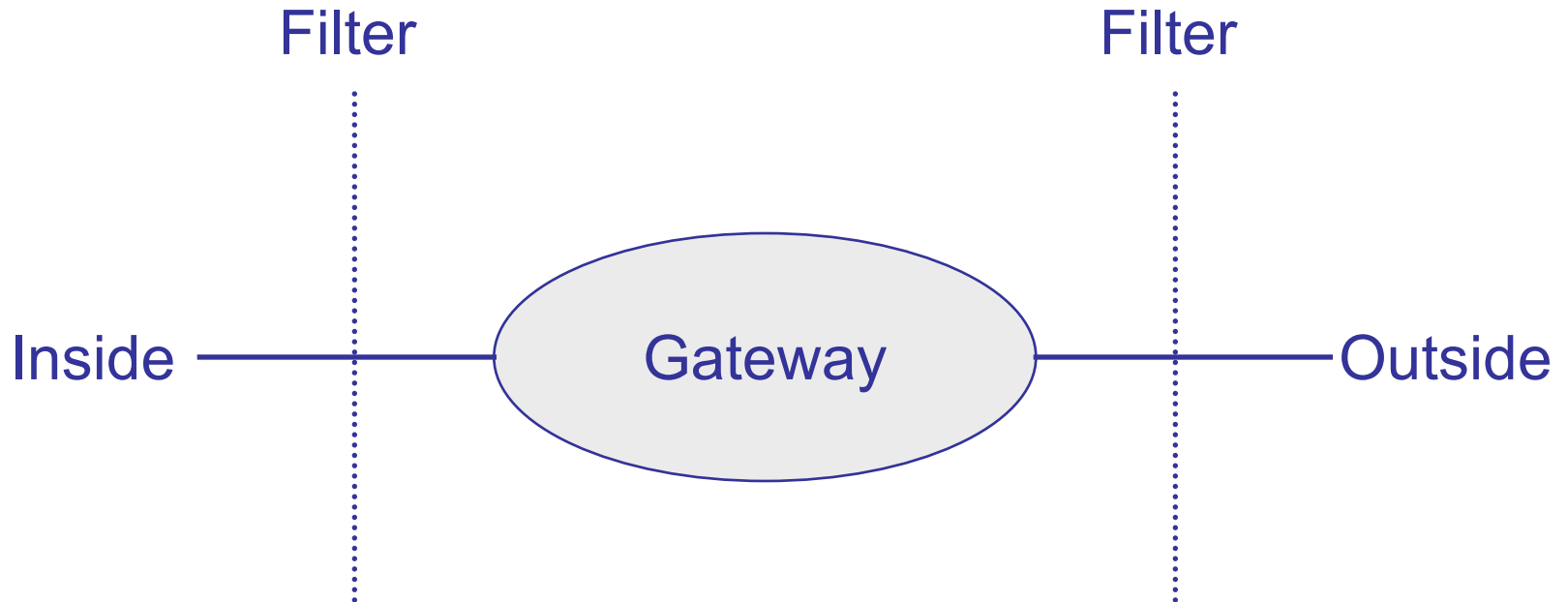
Firewalls

- Problem: Protecting or isolating one part of the network from other parts
 - ◆ In particular: protect your network from global Internet
- Need to filter or otherwise limit network traffic
 - ◆ How to configure this information?
- Questions:
 - ◆ What information do you use to filter?
 - ◆ Where do you do the filtering?

Kinds of Firewalls

- Personal firewalls
 - ◆ Run at the end hosts
 - ◆ e.g. Norton, Windows, etc.
 - ◆ Benefit: has more application/user specific information
- Network firewalls
 - ◆ Intercept and evaluate communications from many hosts
- Filter Based
 - ◆ Operates by filtering based on packet headers
- Proxy based
 - ◆ Operates at the level of the application
 - ◆ e.g. HTTP web proxy

Network Firewalls



- Filters protect against “bad” communications.
- Protect services offered internally from outside access.
- Provide outside services to hosts located inside.

Packet Filtering Firewalls

- Packet filtering firewalls can take advantage of the following information from network and transport layer headers:
 - ◆ Source
 - ◆ Destination
 - ◆ Source Port
 - ◆ Destination Port
 - ◆ Flags (e.g. ACK)
- Some firewalls keep state about open TCP connections
 - ◆ Allows conditional filtering rules of the form “if internal machine has established the TCP connection, permit inbound reply packets”

Ports

- Ports are used to distinguish applications and services on a machine.
- Low numbered ports are often reserved for server listening.
- High numbered ports are often assigned for client requests.
- Port 7 (UDP,TCP): echo server
- Port 13 (UDP,TCP): daytime
- Port 20 (TCP): FTP data
- Port 21 (TCP): FTP control
- Port 22 (TCP): ssh
- Port 25 (TCP): SMTP (Mail)
- Port 80 (TCP): HTTP
- Port 123 (UDP): NTP
- Port 143 (TCP): IMAP
- Port 2049 (UDP): NFS

Filter Example

Action	ourhost	port	theirhost	port	comment
block	*	*	BAD	*	untrusted host
allow	mailhst	25	*	*	allow our SMTP port

Apply rules from top to bottom with assumed *default* entry:

Action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

Problematic entry intended to allow outbound connections to SMTP from inside:

Action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connect to their SMTP

This allows all connections from port 25, but an outside machine can run *anything* on its port 25!

Filter Example Continued

Permit *outgoing* calls to port 25.

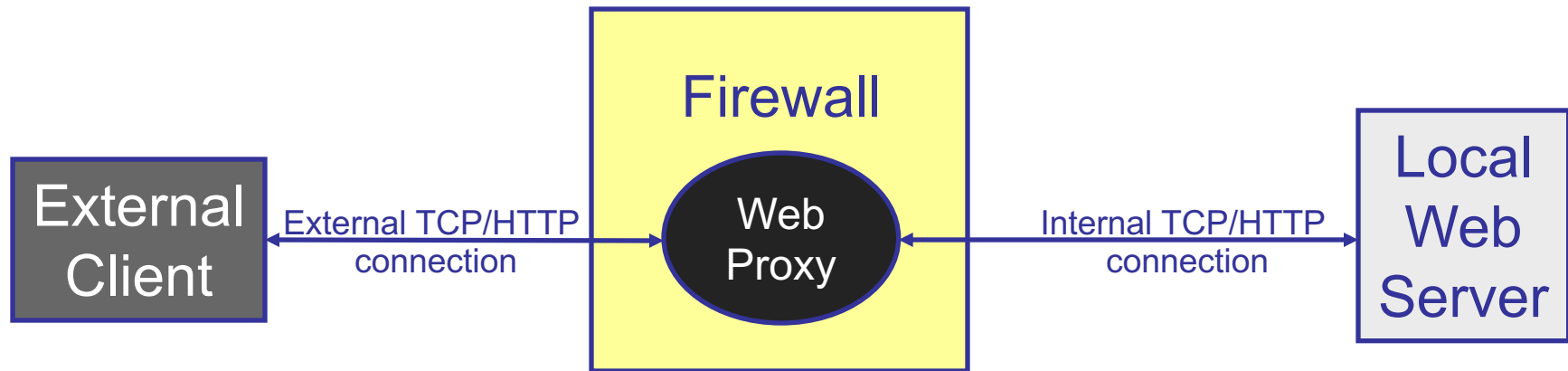
Action	src	port	dest	port	flags	comment
allow	132.239.6.*	*	*	25	*	their SMTP
allow	*	25	*	*	ACK	their replies

This filter doesn't protect against IP address spoofing.
The bad hosts can "pretend" to be one of the hosts with
addresses 132.239.6.* .

Principles for Firewall Configuration

- Least Privilege:
 - ◆ Turn off everything that is unnecessary
(e.g. Web Servers should disable port 25 [SMTP])
- Failsafe Defaults:
 - ◆ By default should reject
 - ◆ (Note that this can cause usability problems...)
- Egress Filtering:
 - ◆ Filter outgoing packets too!
 - ◆ You know the valid IP addresses for machines internal to the network, so drop those that aren't valid.
 - ◆ This can help prevent DoS attacks in the Internet.

Proxy-based Firewalls



- Proxy acts like *both* a client and a server.
- Able to filter using application-level info
 - ◆ For example, permit some URLs to be visible outside and prevent others from being visible.
- Proxies can provide other services too
 - ◆ Caching, load balancing, etc.
 - ◆ Key escrow (e.g., reverse proxies for ssh, SSL)

Firewalls Pro/Con

- Benefits
 - ◆ Reduced “attack surface” against external attackers
 - ◆ Filter out lots of “noise” in network traffic (helps focus attention)
 - ◆ Reduced liability (common practice)
- Costs
 - ◆ Actually cost: both hardware and administration
 - ◆ Bottleneck and single point of failure on network
 - ◆ False sense of security
 - » Limited language (addresses, ports); doesn't help with worms/viruses, ssh exploits, cross-site scripting, etc
 - » Inside vs outside model is fragile (once an internal host is compromised firewall does no good); What about wireless laptops?

Network content analysis

- Lots of devices want to look at network traffic for security
 - ◆ Network Intrusion detection/prevention Systems (NIDS/NIPS)
 - » Try to find signatures of attacks or malware
 - ◆ Spam filters
 - » Try to detect unwanted e-mail
 - ◆ Data leakage
 - » Try to prevent sensitive information from leaving company
 - ◆ Traffic differentiation
 - » Filter or slow down p2p traffic (e.g. Comcast)

Challenges

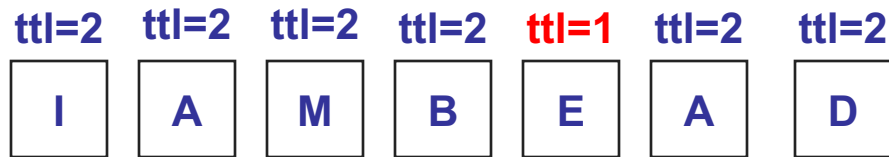
- Expensive to look into each packet
 - ◆ 10Gbps -> ~1M packets per second... ns' per byte
- Network vantage point is imperfect
 - ◆ What does a packet *mean*?
 - ◆ What if a session is encrypted?
 - ◆ Network evasion?

Network evasion

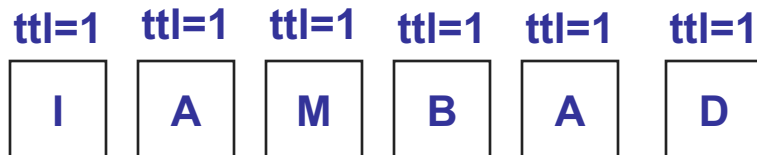
- Typically network intrusion detection systems are deployed like firewalls (between internal network and Internet)
- Key assumption is that NIDS sees the same traffic as destination host
- Not quite true...
 - ◆ Lots of ways to evade a NIDS by exploiting ambiguity

TTL evasion

- Suppose destination is 2 hops after NIDS box
- This is what NIDS sees (each letter is one pkt)

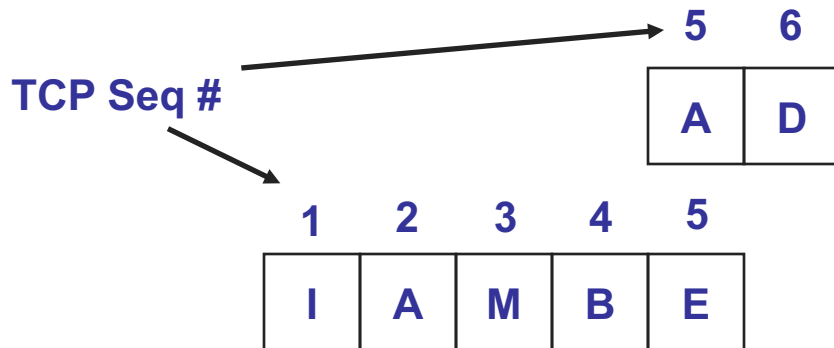


- This is destination sees



Sequence # evasion

- Suppose attacker sends two packets



- What does destination see?
 - ◆ IAMBED? IAMBAD? IAMBBD? Depends on host
- **Lots** of other evasion techniques...

Solution

- Protocol normalization
 - ◆ NIDS rewrites packets to remove all ambiguity
 - ◆ E.g.
 - » all packets have ttl rewritten to reach any internal destination
 - » IDS tracks each flow and does not allow overlapping packets
- Can be very tricky to get right and expensive
 - ◆ Potentially must buffer large amounts of data
 - ◆ What if you get seq #2 through #100, but not seq #1?
 - ◆ Tradeoff: when to drop data vs when to buffer

Summary

- Core protocols not designed for security
 - ◆ Eavesdropping, Packet injection, Route stealing, DNS poisoning
 - ◆ Patched over time to prevent basic attacks
(e.g. random TCP SN)
- More secure variants exist (limited deployment)
 - IP -> IPsec
 - DNS -> DNSsec
 - BGP -> SBGP
- Still need to be careful about protocol semantics
- Perimeter defenses can shrink the “attack surface” exposed by network, but can’t stop everything this way

That's all folks...

Thanks and we'll see you at the final

Good luck!