



*The International  
Conference for High  
Performance Computing,  
Networking, Storage  
and Analysis*

**SC12**

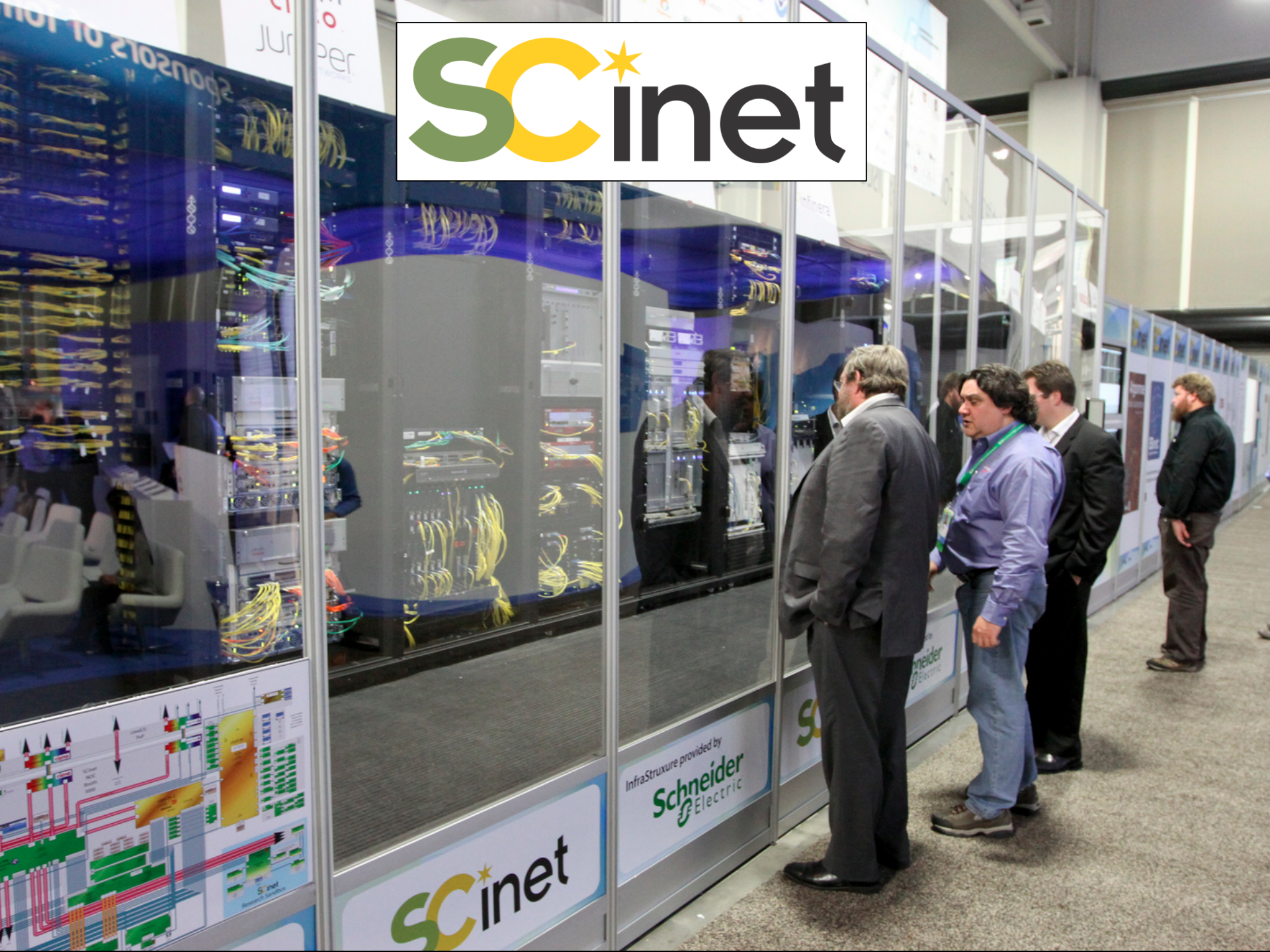
**Salt Lake City, Utah**

# SC12

- 9500 attendees
- 350+ corporate exhibitors in 150,000 ft<sup>2</sup> of space
- 7 days of tutorials, workshops, technical papers, presentations, etc...




# SCinet




InfraStruxure provided by  
**Schneider Electric**

# SCinet



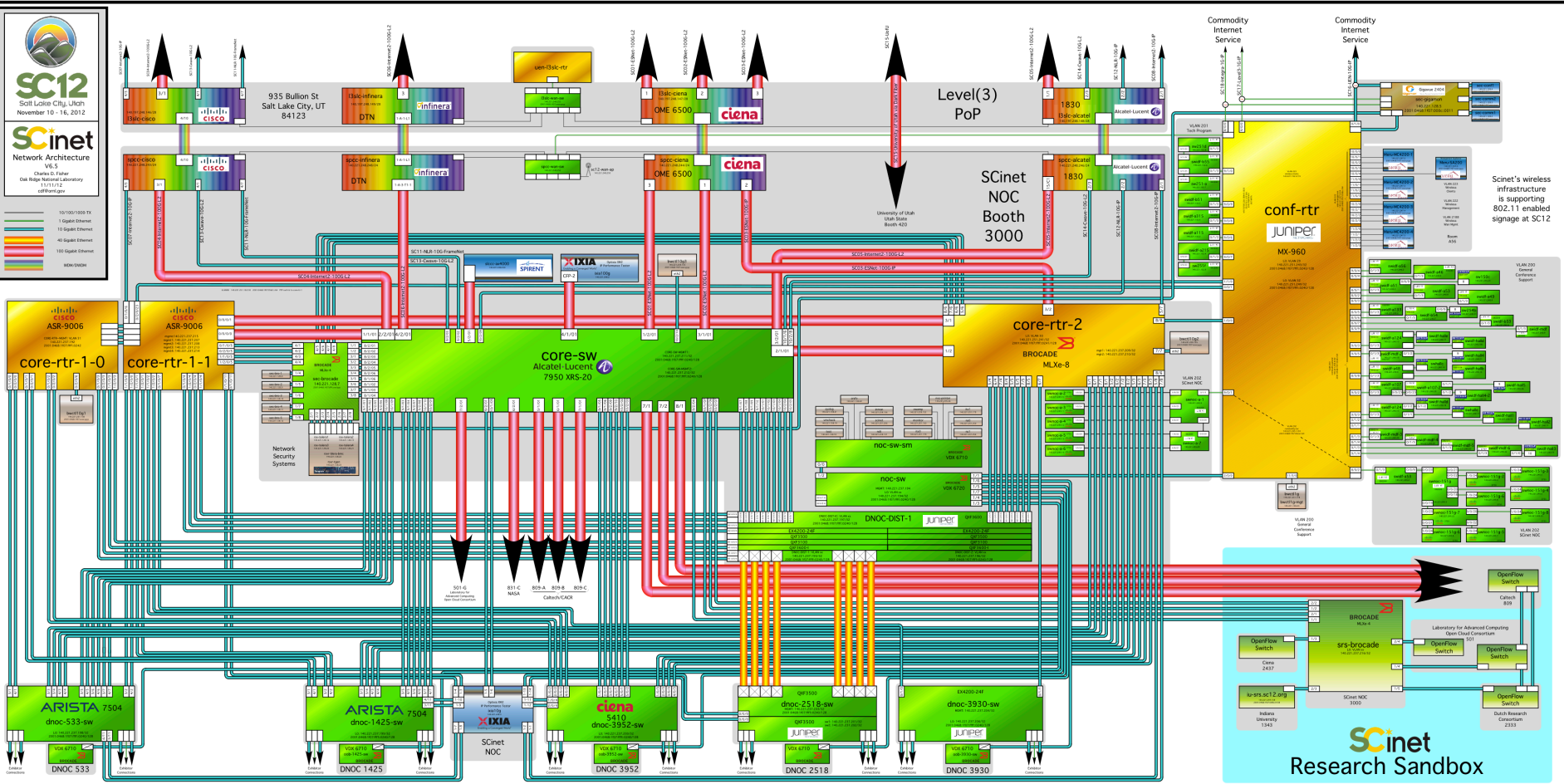
**SC12**  
Salt Lake City, Utah  
November 10 - 16, 2012



**Scinet**  
Network Architecture  
VLS

Quality D. Fisher  
Old Ridge Network Laboratory  
51131712  
scinet@scinet.org

- 100/100/1000 T3
- 10 Gbps Ethernet
- 100 Gbps Ethernet
- 40 Gbps Ethernet
- 100 Gbps Ethernet
- 802.11ac



100 Gbit/s links on conference floor to Internet

ELEC / COMP 177 – Fall 2012

# Computer Networking

## → Security and Firewalls

Some slides from Kurose and Ross, *Computer Networking*, 5<sup>th</sup> Edition  
Some slides from CS244a @ Stanford

# Schedule

- **Homework #6 - Presentation on security/privacy**
  - **Topic selection** – Due NOW
  - **Slides** – Due Monday, Nov 26<sup>th</sup>
  - **Present!** – Tuesday, Nov 27<sup>th</sup> (and Thursday)
- **Project #3** – Due Tuesday, Dec 4<sup>th</sup>
  - Questions?
- **Lab Practical Exams** – See individual signup

# Homework #6 Grading

## 50% TECHNICAL CONTENT

- Accuracy?
- Depth?
- Timeliness (material from last 3 years)

This will improve if you practice your talk before class!

Use a timer: **8-9 minutes!**

## 50% PRESENTATION

- Coherent organization?
- Effectively covered topic? (including an introduction)
- Visual presentation (slides)
  - Graphics?
  - Spelling?
- Oral presentation
  - Grammar?
  - Delivery (pacing, volume, eye contact)

# Network Security



# Security

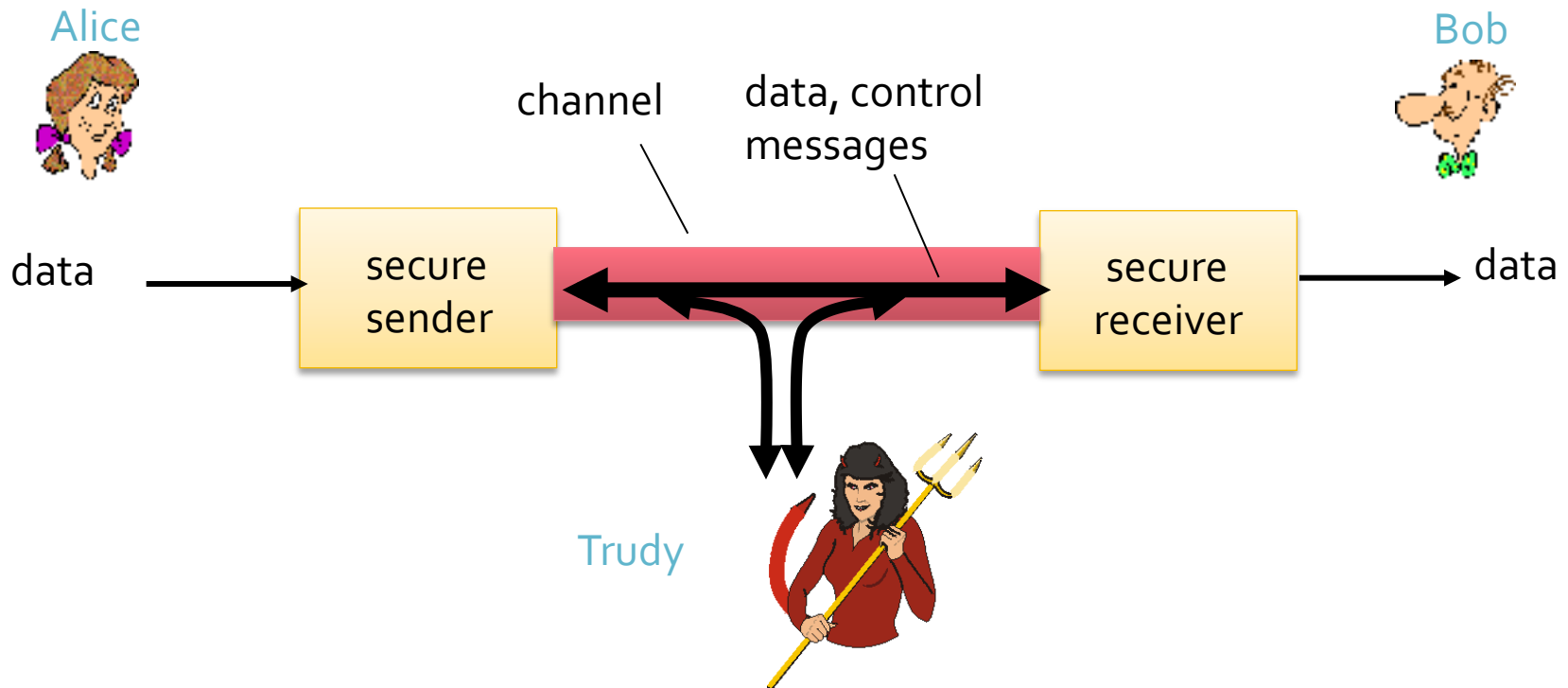
- Large, active field with many participants
  - Academic researchers
  - Engineers in industry
  - Bad guys
- **Impossible, completely impossible to cover it in one day!**
  - Think of today as a random sampling...
- **Take COMP 178 Network Security in Spring**

# What is Network Security?

- **Confidentiality:** only sender and intended receiver should “understand” message contents
  - Sender encrypts message
  - Receiver decrypts message
- **Authentication:** sender and receiver each want to confirm identity of each other
- **Message integrity:** sender and receiver each want to ensure message not altered (in transit, or afterwards) without detection
- **Access and availability:** services must be accessible and available to users

# Friends and Enemies: Alice, Bob, Trudy

- Well-known names in network security world
  - From a 1978 encryption paper
- Bob and Alice want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



# Who might Bob, Alice be?

- Real-life Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- On-line banking client/server
- DNS servers
- Routers exchanging routing table updates
- Peers on a P2P network
- ... and many other possibilities!

# There are Bad Guys (and Girls) Out There!

- Q: What can a “bad guy” do?
- A: A lot!
  - **Eavesdrop**: intercept messages
  - Actively **insert messages** into connection
  - **Impersonate**: fake (spoof) source address in packet (or any field in packet)
  - **Hijack**: “take over” ongoing connection by removing sender or receiver, inserting himself in place
  - **Denial of service**: prevent service from being used by others (e.g., by overloading resources)

# Security Impacts

# Buffer Overflow Vulnerability

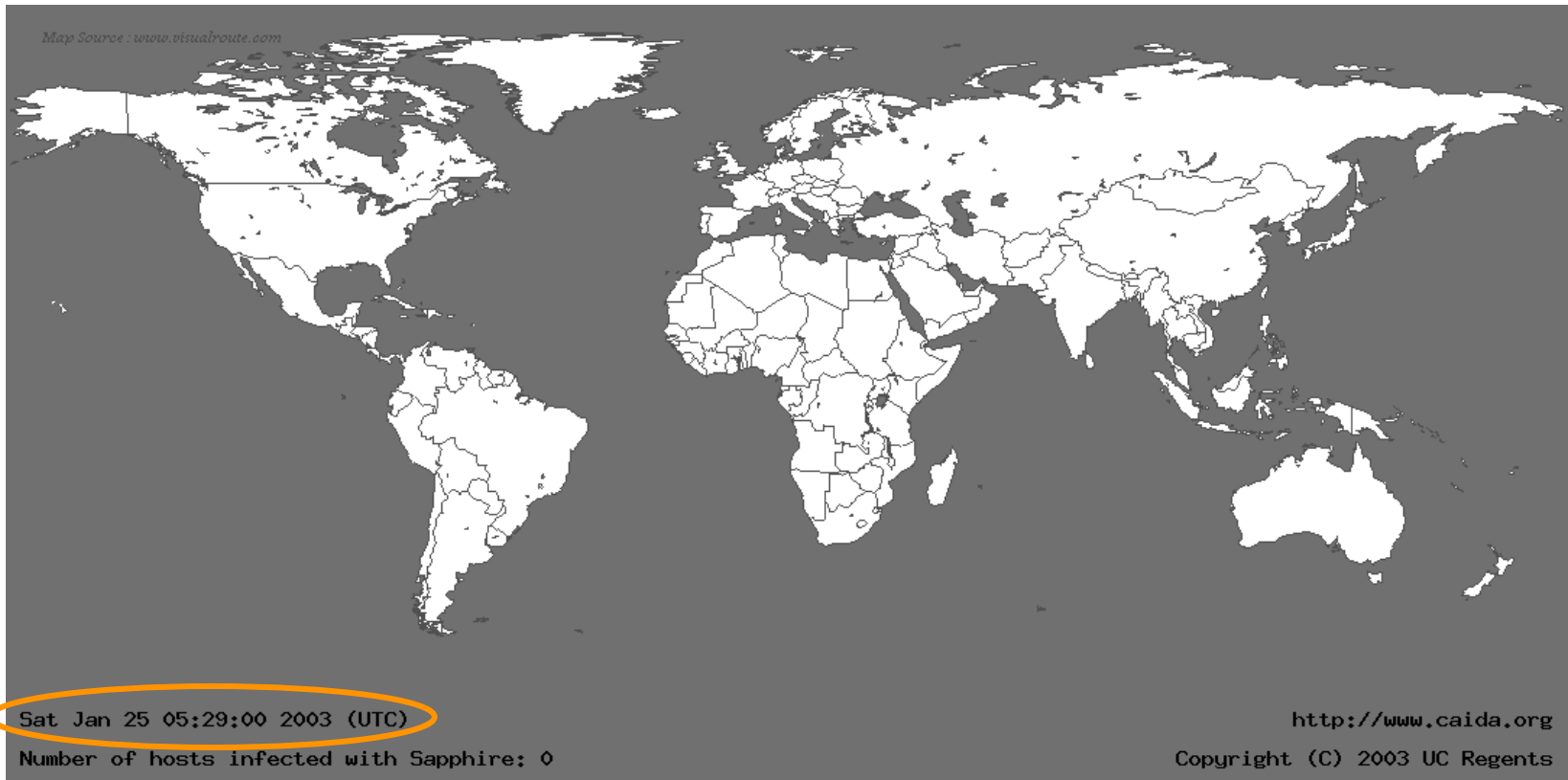
- What is a buffer overflow?
  - `char buf1[8]="";`  
`char buf2[8]="";`  
`strcat(buf1, "excessive");`
- End up overwriting two characters beyond buf1!
- What is beyond my buffer in memory?
  - Other variables and data? (probably buf2)
  - The stack? (further out)
  - **The return address to jump to after my function finishes?**
- If app is running as administrator, attacker now has full access!
- Today's example: Buffer overflow in popular Microsoft products
  - SQL Server 2000 (business)
  - SQL Server Desktop Engine (home!)
  - Incoming (**untrusted**) data from the network overflows a buffer

# Slammer Worm

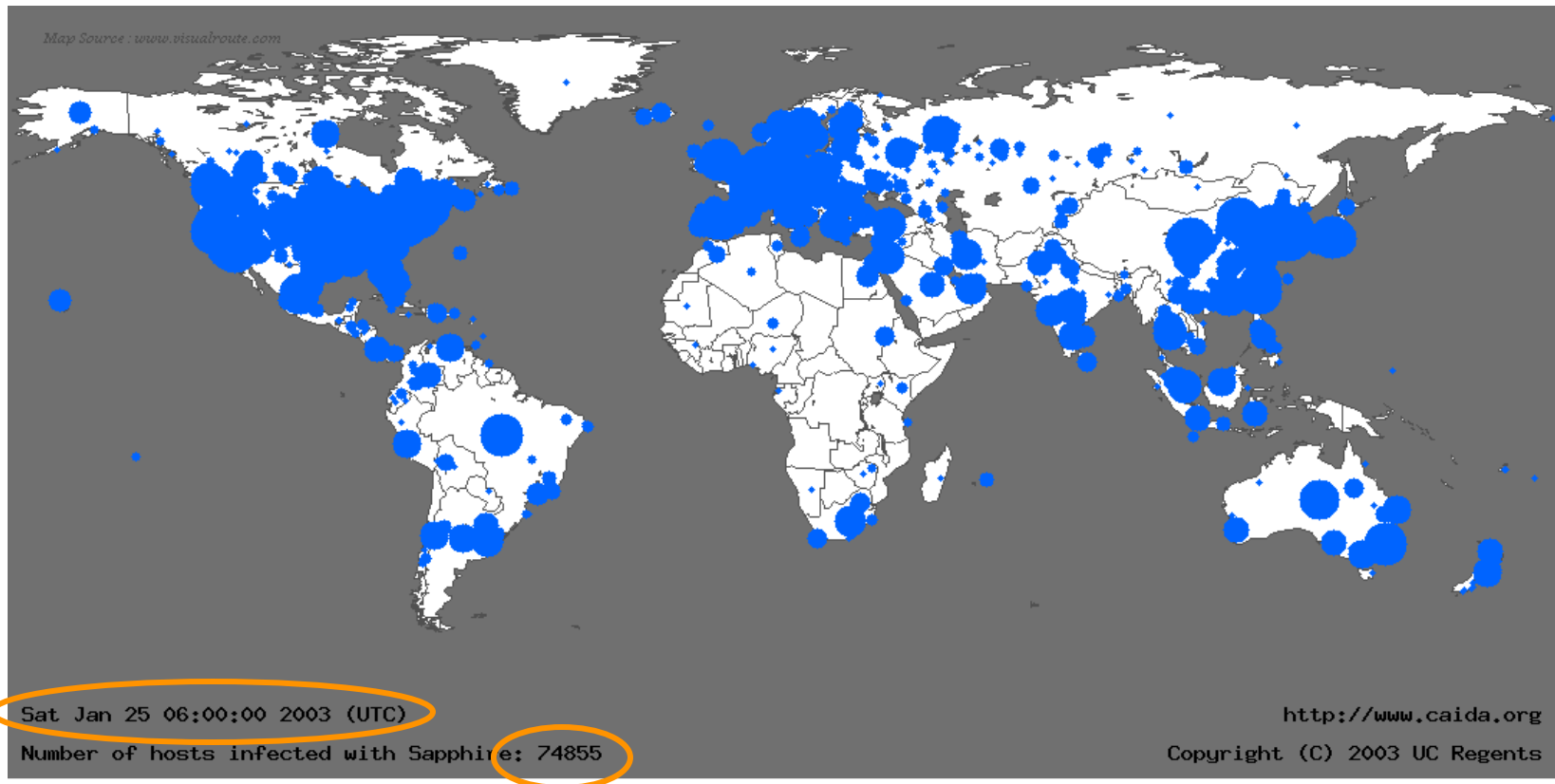
- Worm = Spreads without human intervention
- Exploits buffer overflow to run its own program
  - Generate a random IP address
  - Try to connect and exploit vulnerability on that host



# Life Just Before Slammer



# Life Just After Slammer



# Vulnerability Exploited – Rapid Spread

- Slammer exploited **connectionless UDP** service, rather than connection-oriented TCP.
  - Entire worm fit in a single packet! (376 bytes)
  - When scanning, worm could “fire and forget” – no handshake
  - Stateless!
- Worm infected 75,000+ hosts in 10 minutes (despite broken random number generator).
  - At its peak, infected population doubled every 8.5 seconds
- Progress limited by Internet bandwidth
  - 55 million scans/sec
  - Hindered as links became overloaded

# Why Security?

- First victim at 12:15am
- By 12:45, transcontinental links starting to fail
  - The **traffic** from Slammer overwhelmed legitimate applications – denial of service attack
- 300,000 access points downed in Portugal
- All cell and Internet in Korea failed (27 million people)
- 5 root name servers were knocked offline
- 911 didn't respond (Seattle)
- Continental Airlines ticketing system offline
  - Newark airport hub shut down

# Recovery

- Filter UDP packets with destination port = 1434
  - Got lucky here - obscure port!
    - Almost all traffic was worm related – minimal collateral damage
  - A common port (TCP 80 for HTTP or UDP 53 for DNS) would have knocked legitimate applications offline
- Look for machines that are trying to scan, and clean them
- Human-based recovery took far longer than initial infection

# Security vs Internet Design

# Internet Design Fundamentals

1. Packet-based (statistical multiplexing)
2. Routing is hop-by-hop and destination-based
3. Global addressing: IP addresses
4. Simple to join (as infrastructure)
5. Smart end hosts (end-to-end argument)
6. Hierarchical naming service

# Internet Design vs. Security

- 1. Packet-based (statistical multiplexing)**
  - Simple design
  - How to keep someone from hogging resources?
  - Difficult to put a bound on resource usage (no notion of flow, and source addresses can't be trusted)
  - Community is allergic to per-flow state
2. Routing is hop-by-hop and destination-based
3. Global addressing: IP addresses
4. Simple to join (as infrastructure)
5. Smart end hosts (end-to-end argument)
6. Hierarchical naming service



# Internet Design vs. Security

1. Packet-based (statistical multiplexing)
2. **Routing is hop-by-hop and destination-based**
  - Don't know where packets are coming from
    - Source address can be spoofed
  - Fixes are available but not widely deployed
3. Global addressing: IP addresses
4. Simple to join (as infrastructure)
5. Smart end hosts (end-to-end argument)
6. Hierarchical naming service

# Internet Design vs. Security

1. Packet-based (statistical multiplexing)
2. Routing is hop-by-hop and destination-based
3. **Global addressing (IP addresses)**
  - Everyone can talk to everyone – democracy!
  - Even people who don't necessarily want to be talked to (“every psychopath is your next door neighbor” – Dan Geer)
4. Simple to join (as infrastructure)
5. Smart end hosts (end-to-end argument)
6. Hierarchical naming service

# Internet Design vs. Security

1. Packet Based (statistical multiplexing)
2. Routing is hop-by-hop, destination-based
3. Global Addressing (IP addresses)
4. **Simple to join (as infrastructure)**
  - Untrusted infrastructure
    - Easy to grow organically, but...
  - My router has to trust what your router says
  - Misbehaving routers can violate data integrity and privacy
5. Smart end-hosts (end-to-end argument)
6. Hierarchical Naming Service

# Internet Design vs. Security

1. Packet-based (statistical multiplexing)
2. Routing is hop-by-hop, destination-based
3. Global addressing (IP addresses)
4. Simple to join (as infrastructure)
5. **Smart end-hosts (end-to-end argument)**
  - Decouple hosts and infrastructure = innovation at the edge!
  - **Giving power to least trusted actors**
    - Assume end hosts are “good”
    - How to guarantee good behavior?
  - How to protect complex functionality at end-points?
6. Hierarchical naming service

# Internet Design vs. Security

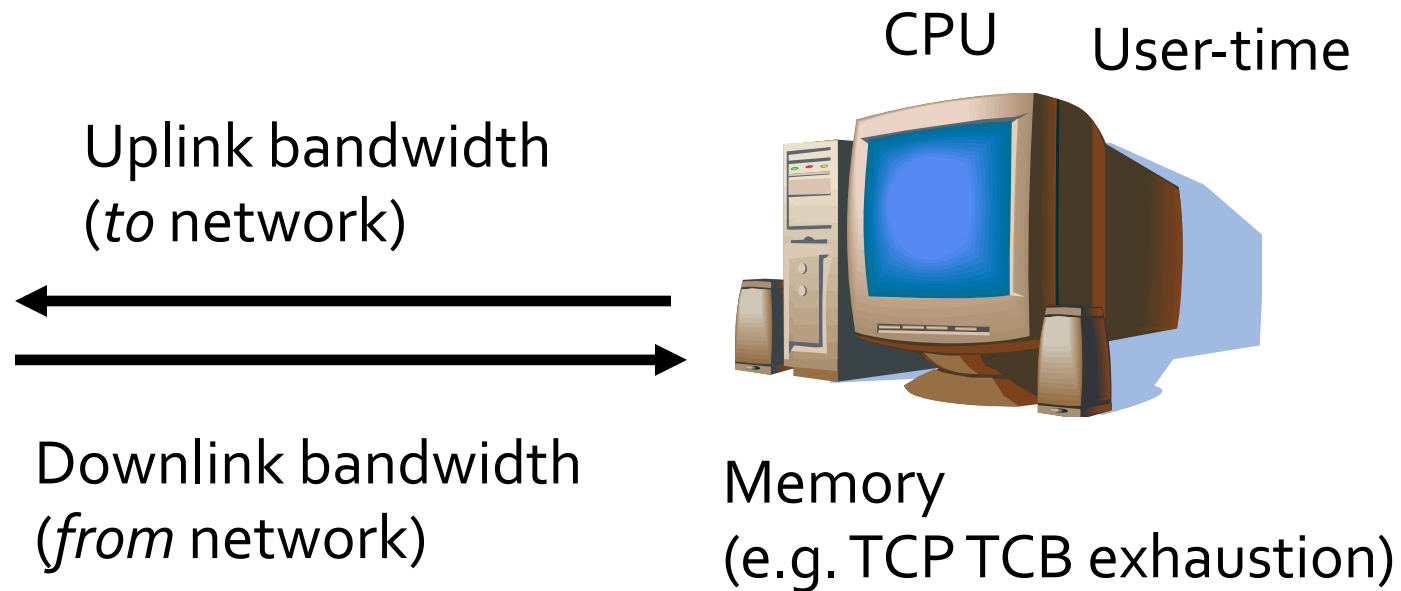
1. Packet-based (statistical multiplexing)
2. Routing is hop-by-hop, destination-based
3. Global addressing (IP addresses)
4. Simple to join (as infrastructure)
5. Smart end hosts (end-to-end argument)
6. **Hierarchical naming service**
  - Lots of caching along the way
  - Need protection/trust at each point or response to name request can be modified

# Attacks

# Some Network Attack Classes

- Focus today on **network-based** attacks
  - Virus, worms spreading/filtering mechanisms, break-ins etc. are beyond scope
- **Denial of Service (DOS)**
  - Limit availability of a network resource to one or more legitimate users
- **Reconnaissance**
  - Discover vulnerabilities/resources; then attack!
- **Masquerade as someone else**
  - Gain unauthorized access to network components (host, router, network, etc...)
  - Then, Denial of Service!

# DoS: Via Resource Exhaustion



Many different resources can be exhausted!

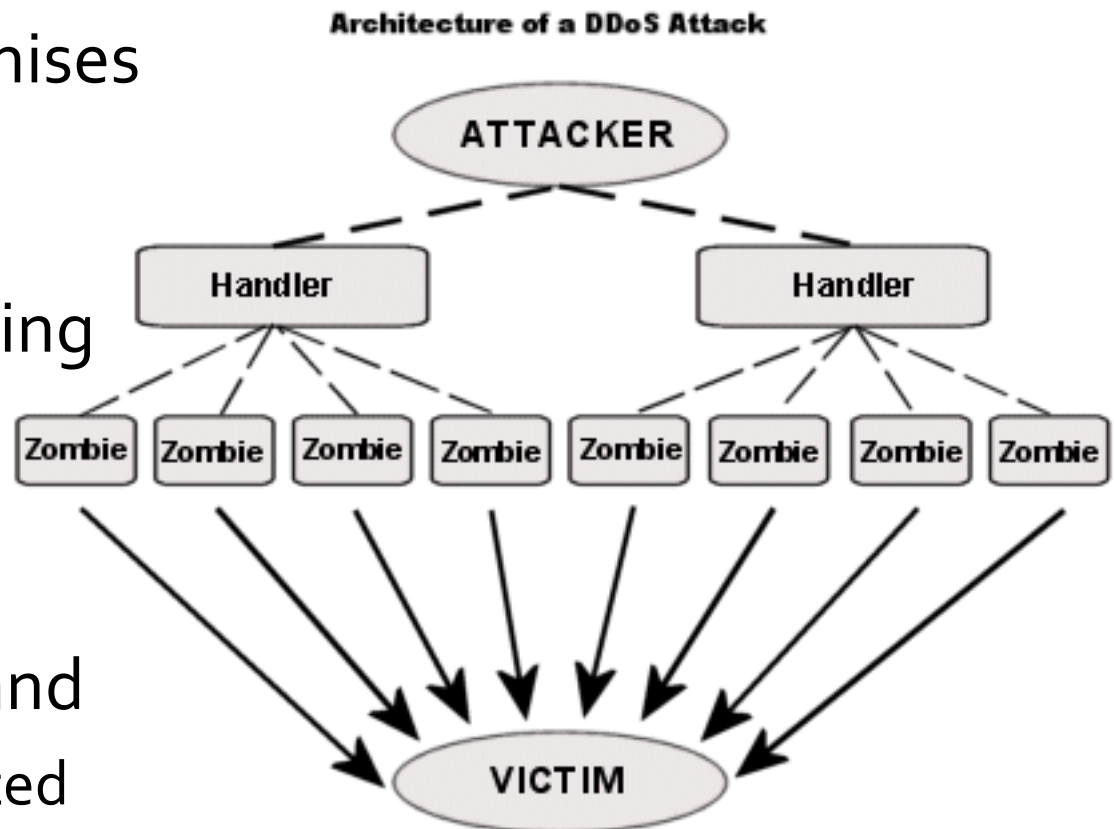


# DoS: Via Resource Exhaustion

- Resource – Uplink bandwidth
  - Saturate uplink bandwidth using legitimate requests (e.g. download large image)
  - Solution? Use a CDN (Akamai)
  - Solution? Admission control at the server (not a network problem!)
- Resource – CPU time
  - Similar to above
- Resource – Victim Memory
  - TCP connections require state, can try to exhaust
  - E.g. SYN Flood

# Distributed DoS (DDoS)

- **Better attack**
- Attacker compromises multiple hosts
- Installs malicious program to do bidding (bots)
- Bots flood (or otherwise attack) victims on command
  - Attack is coordinated



# Distributed DoS (DDoS)

- Bot-networks of 1-10 million hosts have been seen in the wild
  - Difficult to provide an accurate count
    - Estimates vary by an order or magnitude or more!
  - Aggregate bandwidth > 20Gbps (probably more)
- With a botnet, you can flood the victim with perfectly legitimate-looking requests (i.e. download random pages at Amazon.com)

# Resource – Downlink Saturated

- Assume attacker generates enough traffic to saturate downlink bandwidth
- What can the server do?
  - Nothing?
- What can the network do?
  - Ideally want network to drop bad packets
  - How to tell if a packet is part of a legitimate flow? (requires per flow state?)
  - Even harder, how to tell if a SYN packet is part of a legitimate request?

# Botnets and Spam



# DoS Attack – TCP Reset

- TCP connections closed via reset packet
- Unique TCP flow
  - Source IP, source port
  - Destination IP, destination port
  - Sequence number in connection
- **An attacker can guess all of these and close your connection!**
  - Destination IP – target victim
  - Destination port – well known service (HTTP, BGP)
  - Source IP - client being “spoofed”
  - Source port – guess in range 1025 through 49,152
  - Sequence number – have to guess in 32-bit range
    - Most systems allow for a large window (16k) of acceptable seq. #'s
    - Only have to land a packet in the window – now  $2^{32} / 16k$  guesses
  - A team of bots can easily send this many guesses quickly
- Attack is most effective against long lived connections, e.g. BGP
  - Connection lost = route “flaps”
  - MD5 checksums in BGP allow routers to ignore bogus resets

# Reconnaissance Attacks

- To attack a victim, first discover available resources
  - What OS do they run? What apps are running? What is the network topology?
- Many commonly used reconnaissance techniques
  - Port scanning
  - Host/application fingerprinting
    - Example: SYN fingerprinting to identify OS
  - Traceroute
  - DNS (e.g. reverse DNS scanning)
  - SNMP
- These are meant for use by admins to diagnose network problems!
  - Trade-off between the ability to diagnose a network and reveal security sensitive information

# Port Scanning with nmap

- We used nmap in the firewall lab
  - How does it determine if a port is open, closed, or “filtered”?
  - Differences between TCP and UDP?



# TCP Port Scanning with nmap

- **Option 1:** Try to open a socket and close it immediately (3-way handshake) – “heavyweight”
- **Option 2:** Partial connection – “lightweight”
  - Scanner sends SYN
  - Target sends SYN-ACK
    - Scanner sends RST (aborts connection establishment)
- Other scan types exist but are more limited
  - Only work with specific operating systems...

# UDP Port Scanning with nmap

- Harder – **UDP is connectionless**
- Hosts *should* send back ICMP port-unreachable message if UDP message received on closed port
  - Option 1 – nmap infers “port open” from lack of ICMP failure message
  - **What if UDP ports are firewalled? False positive!**
- Option 2 – Send out application-specific UDP messages (i.e. a DNS message to port 53)
  - Drawbacks – Not generic, time-consuming to implement

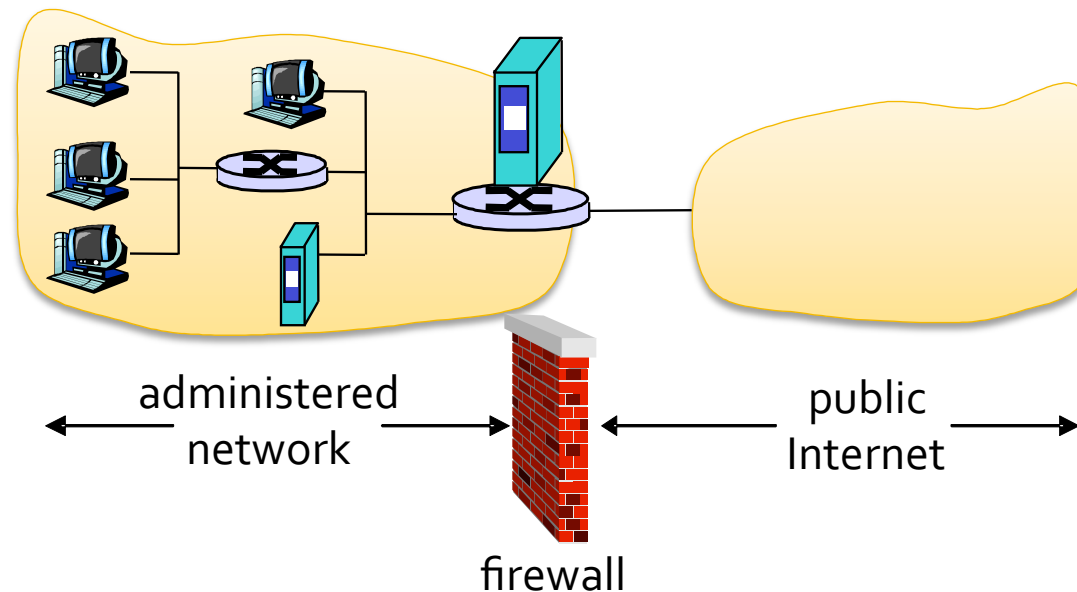
# Defenses

# Encryption

- A subject for another day (or **class**)...
- We can encrypt many things
  - DNS records (DNSsec)
  - IP packets (IPsec)
  - Application-layer communication (HTTPS)

# Firewalls

- Isolates organization's internal net from larger Internet
- Allows some packets to pass but blocks others

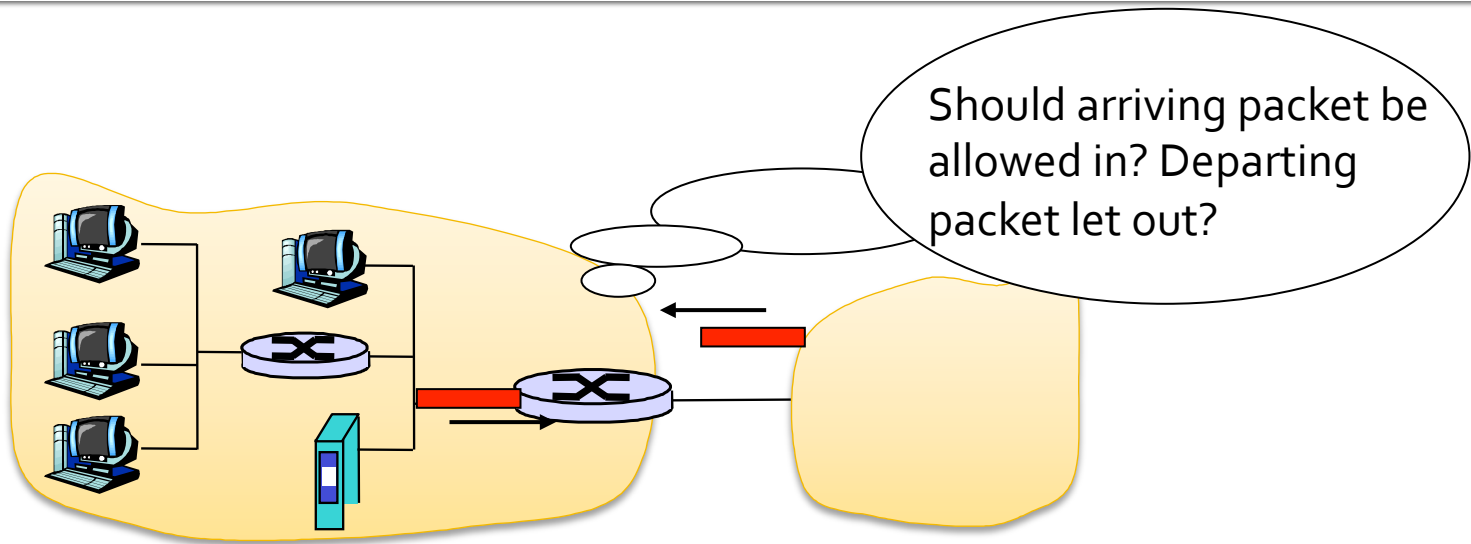


# Firewalls Motivations

- Allow only authorized access to inside network (set of authenticated users/hosts)
  - Prevent denial of service attacks
    - SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections
  - Prevent illegal modification/access of internal data
    - e.g., attacker replaces CIA’s homepage with something else
  - Types of firewalls:
    - Stateless packet filters
    - Stateful packet filters
- Can run either in the network (e.g. at router) or on the host itself



# Stateless Packet Filtering



- Internal network connected to Internet via **router firewall**
- Router filters packet-by-packet
- **Decision to forward/drop packet** based on:
  - Source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

# Stateless Packet Filtering Examples

- Example 1: Block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23
  - All incoming, outgoing UDP flows and telnet connections are blocked.
- Example 2: Block inbound TCP segments with ACK=0
  - Prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside



# Stateless Packet Filtering: more examples

<u>Policy</u>	<u>Firewall Setting</u>
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack	Drop all ICMP packets going to a "broadcast" address (eg 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

# Access Control Lists

- Table of rules, applied top to bottom to incoming packets: (action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

# Stateful Packet Filtering

- **Stateless** packet filter: heavy handed tool
  - admits packets that “make no sense,” e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- **Stateful** packet filter: track status of every TCP connection
  - Track connection setup (SYN), teardown (FIN)
    - Can determine whether incoming, outgoing packets “makes sense”
  - Timeout inactive connections at firewall: no longer admit packets

# Limitations of Firewalls

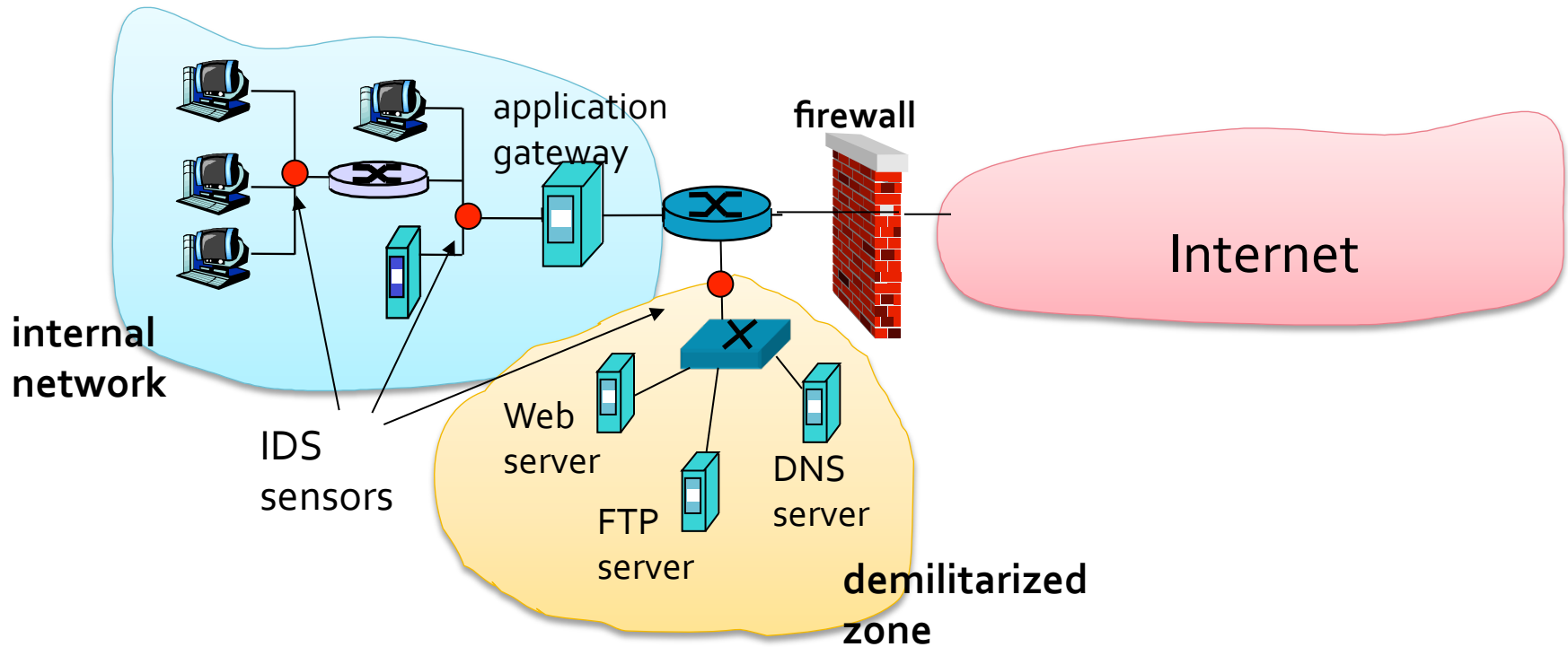
- **IP spoofing:** router can't know if data "really" comes from claimed source
- Tradeoffs:
  - Degree of communication with outside world
  - Level of security
- Many highly protected sites still suffer from attacks
  - How to distinguish between legitimate and attack requests?

# Intrusion Detection Systems

- Packet filtering:
  - Operates on TCP/IP headers only
  - No correlation check among sessions
- **IDS: intrusion detection system**
  - **Deep packet inspection:** look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
  - Examine correlation among multiple packets
    - Port scanning
    - Network mapping
    - DoS attack
  - **Why is this hard / expensive?**

# Intrusion Detection Systems

- Multiple IDSs
  - Different types of checking at different locations



# Summary

- Internet not designed for security
- Many, many attacks
  - Defense is very difficult
  - Attackers are smart; Broken network aids them!
- Retrofitting solutions often break original design principles
  - Some of these solutions work, some of the time
  - Some make the network inflexible, brittle
- Time to go back to the drawing board?