

# CS244a: An Introduction to Computer Networks

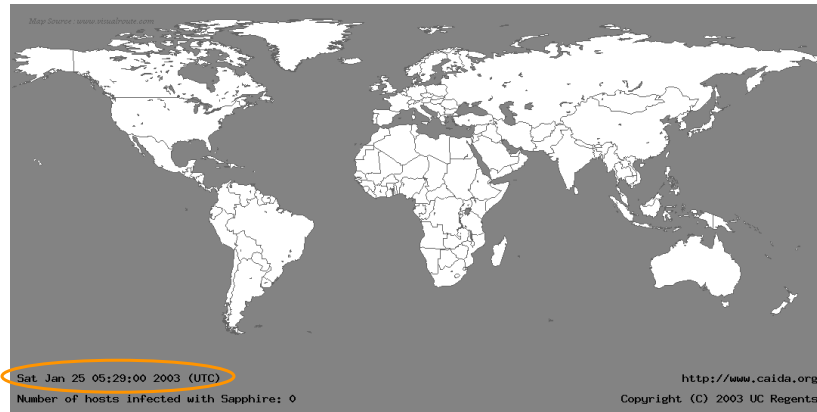
## Security

---



## Announcements (?)

## Life Just Before Slammer

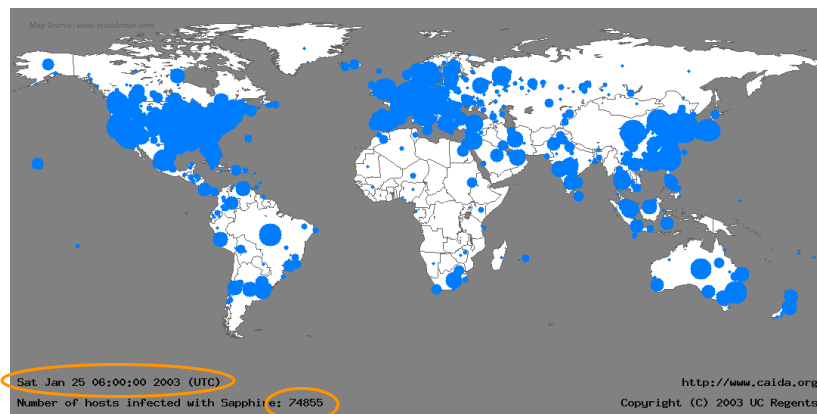


Winter 2008

CS244a

3

## Life Just After Slammer



Winter 2008

CS244a

4

## A Lesson in Economy

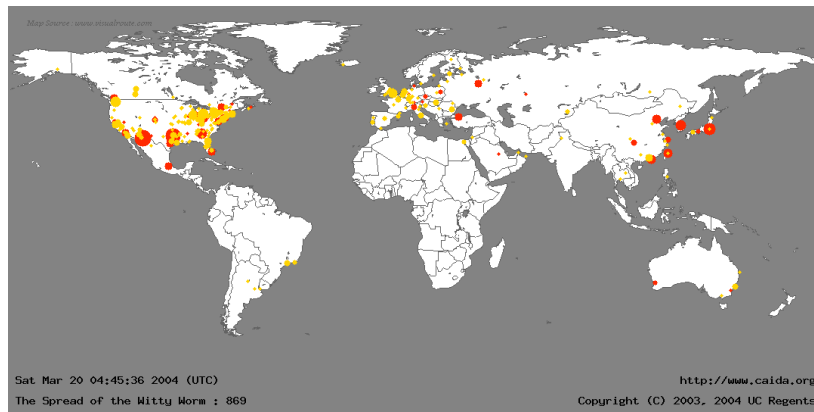
- ❖ Slammer exploited connectionless UDP service, rather than connection-oriented TCP.
- ❖ *Entire worm* fit in a single packet! (376 bytes)
  - ⇒ When scanning, worm could “fire and forget”.

*Stateless!*
- ❖ Worm infected 75,000+ hosts in 10 minutes (despite broken random number generator).
  - ❖ At its peak, **doubled every 8.5 seconds**
- ❖ Progress limited by the Internet’s *carrying capacity* (= 55 million scans/sec)

## Why Security?

- ❖ First victim at 12:15am
- ❖ By 12:45, transcontinental links starting to fail
- ❖ 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)
- ❖ Flights canceled

# Witty Worm



Winter 2008

CS244a

7

# Today

- ➔ ❖ Network Security Goals
- ❖ Security vs. Internet Design
- ❖ Attacks
- ❖ Defenses
- ❖ Worms

Winter 2008

CS244a

8

## Network Security Goals

- ❖ **Availability**  
(everyone can reach all network resources all the time)
- ❖ **Protection**  
(protect users from interactions they don't want)
- ❖ **Authenticity**  
(know who you are speaking with)
- ❖ **Data Integrity**  
(protect data en-route)
- ❖ **Privacy**

## Network Security Goals

- ❖ **Availability**  
(everyone can reach all network resources all the time)
- ❖ **Protection**  
(protect users from interactions they don't want)
- ❖ **Authenticity**  
(know who you are speaking with)
- ❖ **Data Integrity**  
(protect data en-route)
- ❖ **Privacy**

# Today

- ❖ Network Security Goals
- ➔ ❖ Security vs. Internet Design
- ❖ Attacks
- ❖ Defenses
- ❖ Worms

# Internet Design

- ❖ Destination routing
- ❖ Packet based (statistical multiplexing)
- ❖ Global addressing (IP addresses)
- ❖ Simple to join (as infrastructure)
- ❖ Power in end hosts (end-to-end arg)
- ❖ "Ad hoc" naming system

## Internet Design vs. Security

- ❖ Destination routing
    - Keeps forwarding tables small
    - Simple to maintain forwarding tables
    - How do we know where packets are coming from?
      - Probably simple fix to spoofing, why isn't it in place?
  - ❖ Packet based (statistical multiplexing)
  - ❖ Global addressing (IP addresses)
  - ❖ Simple to join (as infrastructure)
  - ❖ Power in end hosts (end-to-end arg)
  - ❖ "Ad hoc" naming system
- 

## Internet Design vs. Security

- ❖ Destination Routing
  - ❖ Packet Based (statistical multiplexing)
    - Simple + Efficient
    - Difficult resource bound per-communication
      - How to keep someone from hogging?  
(remember, we can't rely on source addresses)
  - ❖ Global Addressing (IP addresses)
  - ❖ Simple to join (as infrastructure)
  - ❖ Power in End Hosts (end-to-end arg)
  - ❖ "Ad hoc" naming system
-

## Internet Design vs. Security

- ❖ Destination routing
- ❖ Packet based (statistical multiplexing)
- ❖ **Global Addressing (IP addresses)**
  - Very democratic
  - **Even people who don't necessarily want to be talked to**  
("every psychopath is your next door neighbor" - Dan Geer)
- ❖ Simple to join (as infrastructure)
- ❖ Power in end hosts (end-to-end arg)
- ❖ "Ad hoc" naming system

## Internet Design vs. Security

- ❖ Destination routing
- ❖ Packet based (statistical multiplexing)
- ❖ Global addressing (IP addresses)
- ❖ **Simple to join (as infrastructure)**
  - Very democratic
  - **Misbehaving routers can do very bad things**
  - **No model of trust between routers**
- ❖ Power in End Hosts (end-to-end arg)
- ❖ "Ad hoc" naming system



## Internet Design vs. Security

- ❖ Destination routing
- ❖ Packet based (statistical multiplexing)
- ❖ Global addressing (IP addresses)
- ❖ Simple to join (as infrastructure)
- ❖ **Power in end-hosts (end-to-end arg)**
  - Decouple hosts and infrastructure = innovation at the edge!
  - **Giving power to least trusted actors**
    - How to guarantee good behavior?
- ❖ "Ad hoc" naming system

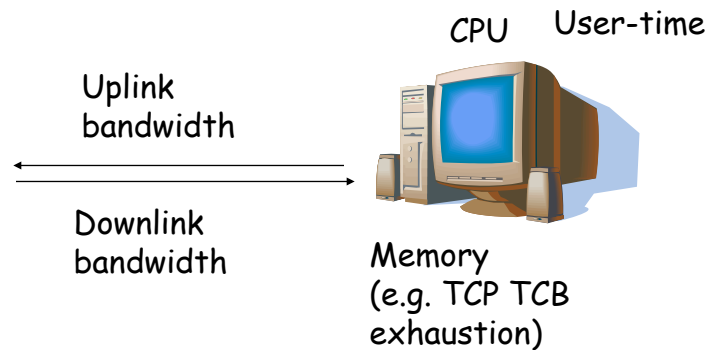
## Internet Design vs. Security

- ❖ Packet Based (statistical multiplexing)
- ❖ Destination Routing
- ❖ Global Addressing (IP addresses)
- ❖ Simple to join (as infrastructure)
- ❖ Power in End Hosts (end-to-end arg)
- ❖ **"Ad hoc" naming system**
  - Seems to work OK
  - **Fate sharing w/ hierarchical system**
  - **Off route = more trusted elements**

# Today

- ❖ Network Security Goals
- ❖ Security vs. Internet Design
- ➔ ❖ "Attacks" (how attacks leverage these weaknesses in practice)
  - ❖ Denial of service
  - ❖ Indirection
  - ❖ Reconnaissance
- ❖ Defenses
- ❖ Worms

## DoS: Via Resource Exhaustion



## DoS: Via Resource Exhaustion

### ❖ Uplink bandwidth

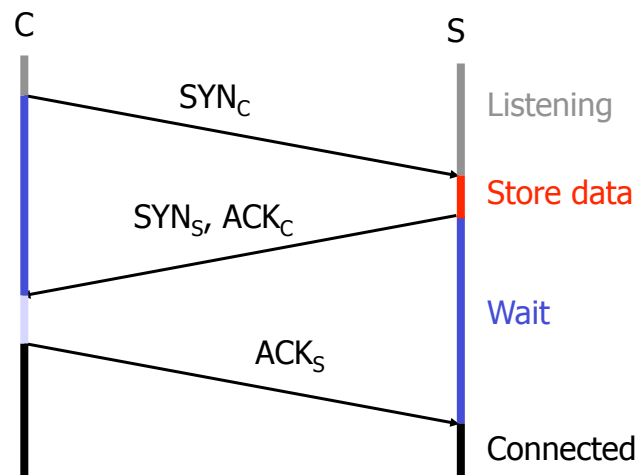
- Saturate uplink bandwidth using legitimate requests (e.g. download large image)
- Solution: use a CDN (Akamai)
- Solution: admission control at the server (not a network problem ??)

### ❖ CPU time similar to above

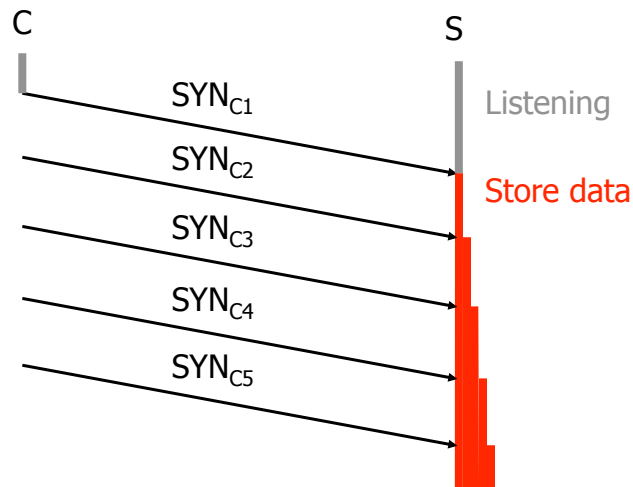
### ❖ Victim Memory

- TCP connections require state, can try to exhaust
- E.g. SYN Flood (next few slides)

## TCP Handshake



## Example: SYN Flooding



Winter 2008

CS244a

23

## Protection against SYN Attacks

### ❖ SYN Cookies

[Bernstein, Schenk]

- ❖ Client sends SYN
- ❖ Server responds to Client with SYN-ACK cookie
  - $sqn = f(\text{src addr}, \text{src port}, \text{dest addr}, \text{dest port}, \text{rand})$
  - Server does not save state
- ❖ Honest client responds with ACK(sqn)
- ❖ Server checks response
  - If matches SYN-ACK, establishes connection
- ❖ Drop Random TCB in SYN\_RCVD state (likely to be attackers)

Winter 2008

CS244a

24

## Distributed DoS (DDoS)

- ❖ Attacker compromises multiple hosts
- ❖ Installs malicious program to do her bidding (bots)
- ❖ Bots flood (or otherwise attack) victims on command; Attack is coordinated
- ❖ Bot-networks of 80k to 100k have been seen in the wild
  - ❖ Aggregate bandwidth > 20Gbps (probably more)
- ❖ E.g. blue security

## What About Downlink? (flooding)

- ❖ Assume attacker generates enough traffic to saturate downlink bandwidth
- ❖ What can the server do?
- ❖ 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?
- ❖ Is the phone network immune to such attacks?

## DoS Aplenty

- ❖ Attacker guesses TCP seq. number for an existing connection:
  - ❖ Attacker can send Reset packet to close connection. Results in DoS.
  - ❖ Most systems allow for a large window of acceptable seq. #'s
  - ❖ Only have to land a packet in
  - ❖ Attack is most effective against long lived connections, e.g. BGP.

- ❖ Congestion control DoS attack

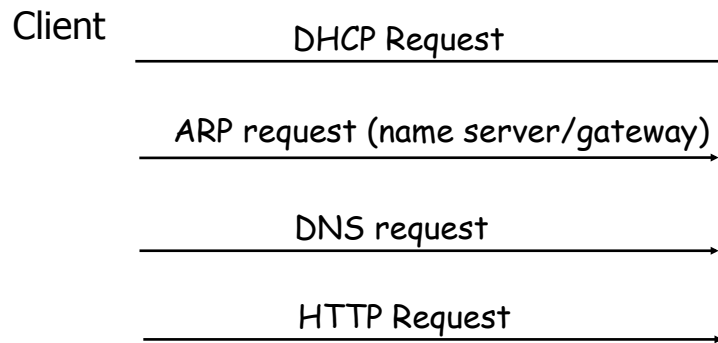


- ❖ Generate TCP flow to force target to repeatedly enter retransmission timeout state
- ❖ Difficult to detect because packet rate is low

## Indirection Attacks

- ❖ Rely on connecting to "end-points" to get content/access services
- ❖ Unfortunately network end-points (e.g. ips, DNS names) are loosely bound
- ❖ Long history of problems

## Example, fetching a web page



## DNS Vulnerability

- ❖ Users/hosts typically trust the host-address mapping provided by DNS

## Bellovin/Mockapetris Attack

- ❖ Trust relationships use symbolic addresses
  - ❖ `/etc/hosts.equiv` contains `friend.stanford.edu`
- ❖ Requests come with numeric source address
  - ❖ Use reverse DNS to find symbolic name
  - ❖ Decide access based on `/etc/hosts.equiv`, ...
- ❖ Attack
  - ❖ Spoof reverse DNS to make host trust attacker

## Reverse DNS

- ❖ Given numeric IP address, find symbolic addr
- ❖ To find 222.33.44.3,
  - ❖ Query `44.33.222.in-addr.arpa`
  - ❖ Get list of symbolic addresses, e.g.,

1	IN	PTR	server.small.com
2	IN	PTR	boss.small.com
3	IN	PTR	ws1.small.com
4	IN	PTR	ws2.small.com



## Attack

- ❖ Gain control of DNS service for evil.org
- ❖ Select target machine in good.net
- ❖ Find trust relationships
  - ❖ SNMP, finger can help find active sessions, etc.
  - ❖ Example: target trusts host1.good.net
- ❖ Connect
  - ❖ Attempt rlogin from coyote.evil.org
  - ❖ Target contacts reverse DNS server with IP addr
  - ❖ Use modified reverse DNS to say "addr belongs to host1.good.net"
  - ❖ Target allows rlogin

## DNS rebinding attacks

- ❖ Java/Javascript implements "same domain" policy for outbound connections
- ❖ "BAD" DNS server can return arbitrary IP for connection

!!EXAMPLE HERE!!

## TCP Connection Spoofing

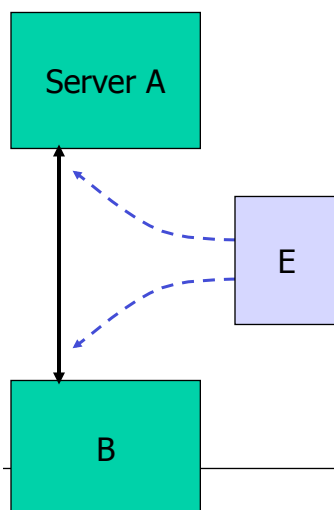
- ❖ Each TCP connection has an associated state
  - ❖ Client IP and port number; same for server
  - ❖ Sequence numbers for client, server flows
- ❖ Problem
  - ❖ Easy to guess state
    - Port numbers are standard
    - Sequence numbers (use to be) chosen in predictable way

Winter 2008

CS244a

35

## IP Spoofing Attack



- ❖ A, B trusted connection
  - ❖ Send packets with predictable seq numbers
- ❖ E impersonates B to A
  - ❖ Opens connection to A to get initial seq number
  - ❖ SYN-floods B's queue
  - ❖ Sends packets to A that resemble B's transmission
  - ❖ E cannot receive, but may execute commands on A
- ❖ Other ways to spoof source IP?

CS244a

36


## Reconnaissance/Misc

- ❖ To attack a victim, first discover available resources
- ❖ Many commonly used reconnaissance techniques
  - ❖ Port scanning
  - ❖ Host/application fingerprinting
  - ❖ Traceroute
  - ❖ DNS (reverse DNS scanning, Zone transfer)
  - ❖ 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

## Anecdotes ...

- ❖ Large bot networks exist that scan the Internet daily looking for vulnerable hosts (we've identified at least 16,000 participating hosts)
- ❖ SSH trolling very popular (90% of all entries in host logs)
- ❖ Old worms still endemic on Internet (e.g. Code Red)
  - ❖ Seem to come and go in mass
  - ❖ Surreptitious scanning effort?

## Today

- ❖ Network Security Goals
- ❖ Security vs. Internet Design
- ❖ Attacks
-  ❖ Defenses
- ❖ Worms

## Firewalls

- ❖ Keep out unwanted traffic
- ❖ Can be done in the network (e.g. network perimeter) or at the host
- ❖ Many mechanisms
  - Packet filters
  - Stateful packet filters
  - Proxies

## Packet Filters

- ❖ Make a decision to drop a packet based on packet header
  - Protocol type
  - Transport ports
  - Source/Dest IP address
  - Etc.
- ❖ Usually done on router at perimeter of network
- ❖ And on virtually all end-hosts today
  - Windows XP sp2
  - Linux "iptables"

## Packet Filters: Problem

- ❖ Assume firewall rule  
(allow from port 53 and port 80)
- ❖ Easy for an attacker to send packets  
from port 53 or 80
- ❖ Further attacker can forge source
- ❖ **Not very effective for stopping packets  
from unwanted senders**

## Stateful Packet Filter

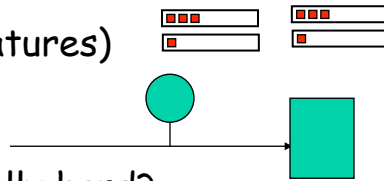
- ❖ Idea: Only allow traffic initiated by client
  - For each flow request (e.g. SYN or DNS req) keep a little state
  - Ensure packets received from Internet belong to an existing flow
  - To be effective must keep around sequence numbers per flow
- ❖ Very common, used in all NAT boxes today
  - Stateful NATs downside: failure → all connection state is lost!

## Proxies

- ❖ Want to look “deeper” into packets
  - Application type
  - Content
- ❖ Can do by reconstructing TCP flows and “peering” in, however this is really hard (digression next slide)

## Passive Reconstruction of TCP Stream

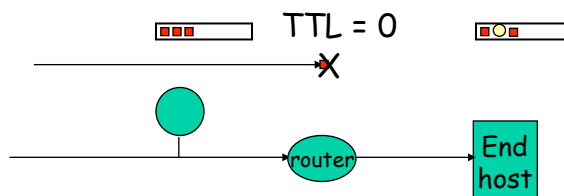
- ❖ Use passive network element to reconstruct TCP streams
- ❖ "Peer" into stream to find harmful payload (e.g. virus signatures)



- ❖ Why is this really hard?

## Reconstructing Streams

- ❖ Must know the client's view of data
  - Have to know if packet reaches destination (may not if TTL is too short)
  - Have to know how end-host manages overlapping TCP sequence numbers
  - Have to know how end-host manages overlapping fragments



## Proxies

- ❖ Full TCP termination in the network
- ❖ Often done transparently (e.g. HTTP proxies)
- ❖ Allows access to objects passed over network (e.g. files, streams etc.)
- ❖ Does not have same problems as stream reconstruction
- ❖ Plus can do lots of other fun things (e.g. content caching)

## Proxy Discussion

- ❖ Proxies duplicate per-flow state held by clients
- ❖ How does this break end-to-end semantics of TCP?
  - E.g. what if proxy crashes right after reading from client? (lost data!)
- ❖ How to fix?
  - Lots of work in this area



## Final Comments

- ❖ 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?  
(see Nick for details 😊 )