

# Lecture 13 - Network Security

CSE497b - Spring 2007

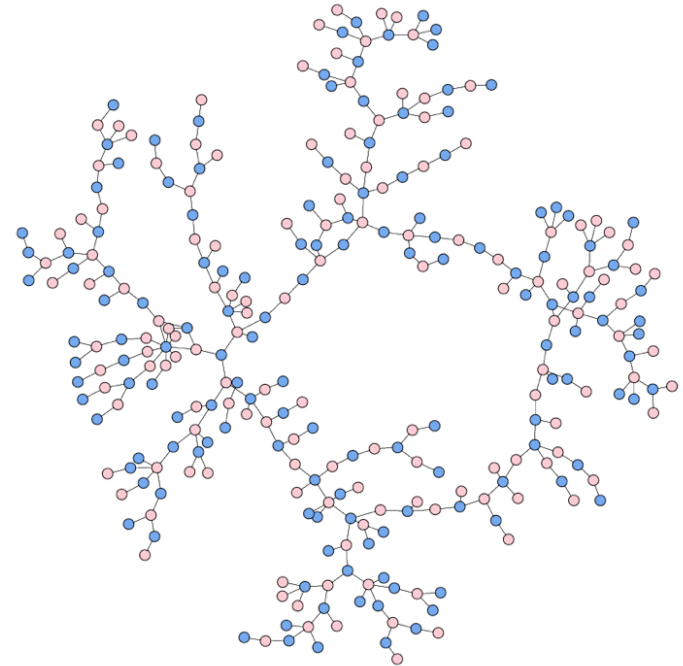
Introduction Computer and Network Security

Professor Jaeger

[www.cse.psu.edu/~tjaeger/cse497b-s07/](http://www.cse.psu.edu/~tjaeger/cse497b-s07/)

# Exploiting the network ...

- The Internet is extremely vulnerable to attack
  - it is a huge open system ...
  - which adheres to the *end-to-end* principle
    - smart end-points, dumb network



- Can you think of any *large-scale attacks* that would be enabled by this setup?

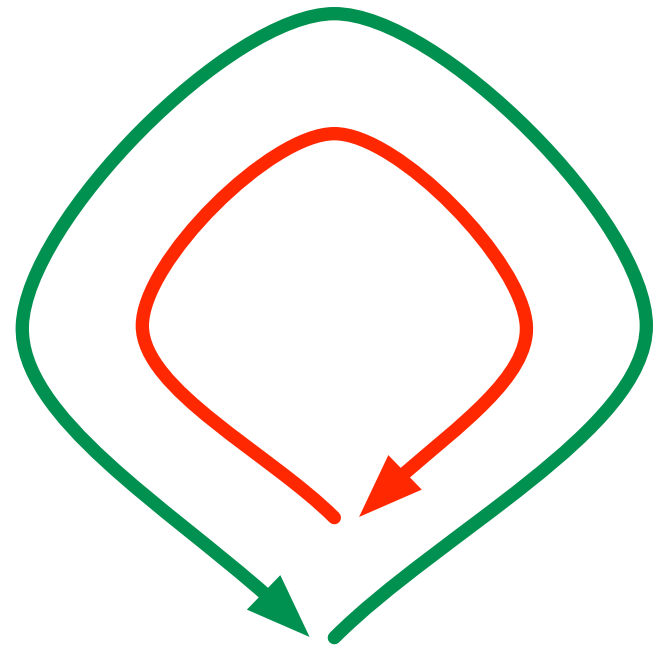
# Malware

- *Malware* - software that exhibits malicious behavior (typically manifest on user system)
  - *virus* - self-replicating code, typically transferring by shared media, filesystems, email, etc.
  - *worm* - self propagating program that travels over the network
- The behaviors are as wide ranging as imagination
  - *backdoor* - hidden entry point into system that allows quick access to elevated privileges
  - *rootkit* - system replacement that hides adversary behavior
  - *key logger* - program that monitors, records, and potentially transmits keyboard input to adversary
  - *trojan* - malicious software disguised as legitimate program

# Worms

- A worm is a self-propagating program.
- As relevant to this discussion
  1. Exploits some vulnerability on a target host ...
  2. (often) embeds itself into a host ...
  3. Searches for other vulnerable hosts ...
  4. Goto (1)

- Q: Why do we care?



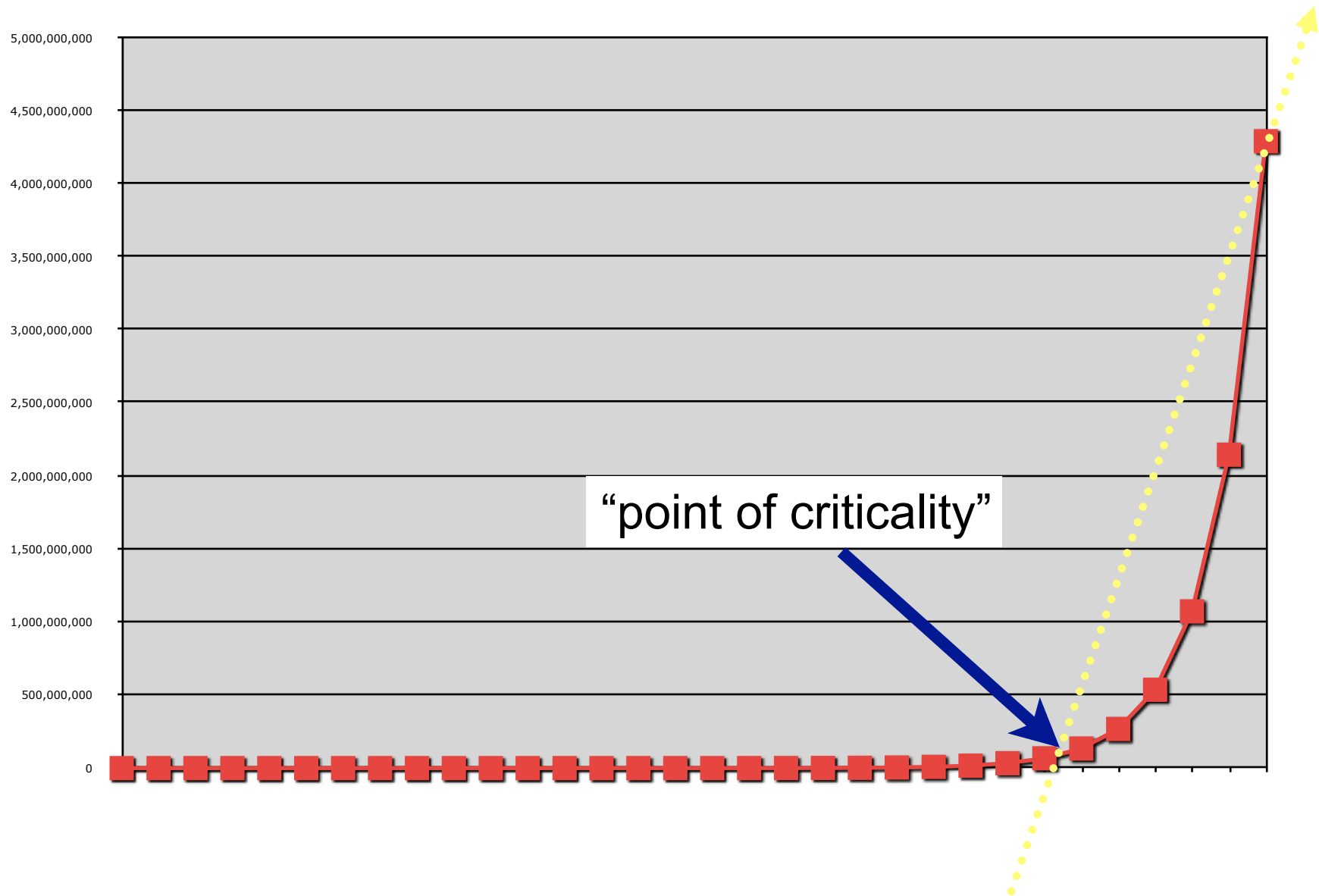
# The Danger

- What makes worms so dangerous is that infection grows at an exponential rate
  - A simple model:
    - $s$  (search) is the time it takes to find vulnerable host
    - $i$  (infect) is the time it takes to infect a host
  - Assume that  $t=0$  is the *worm outbreak*, the number of hosts at  $t=j$  is

$$2^{(j/(s+i))}$$

- For example, if  $(s+i = 1)$ , what is it at time  $t=32$ ?

# The result



# The Morris Worm

- Robert Morris, a 23 year old doctoral student from Cornell
  - Wrote a small (99 line) program
  - November 3rd, 1988
  - Simply disabled the Internet
- How it did it
  - Reads /etc/password, they tries the obvious choices and dictionary, /usr/dict words
  - Used local /etc/hosts.equiv, .rhosts, .forward to identify hosts that are related
    - Tries cracked passwords at related hosts (if necessary)
    - Uses whatever services are available to compromise other hosts
  - Scanned local interfaces for network information
  - Covered its tracks (set its own process name to sh,

# Code Red

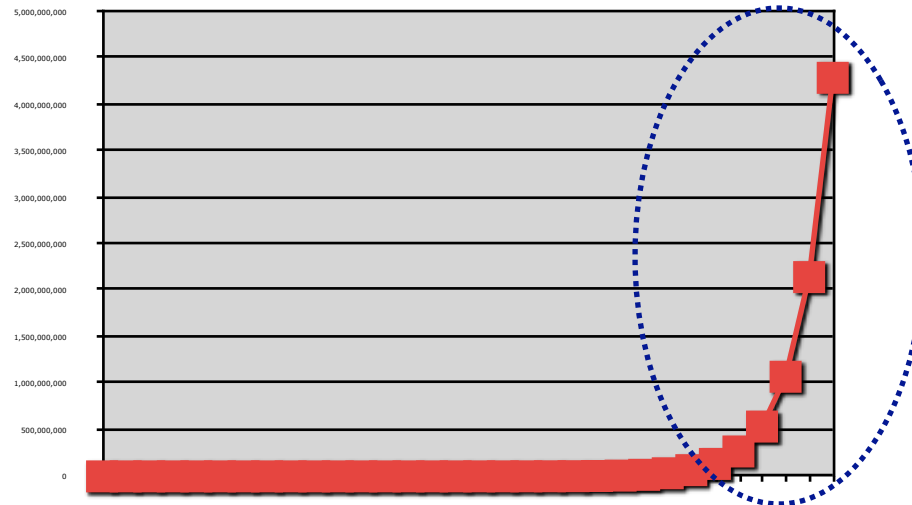
- Anatomy of a worm: Maiffret (good reading)
- Exploited a Microsoft IIS web-server vulnerability
  - A vanilla buffer overflow (allows adversary to run code)
  - Scans for vulnerabilities over random IP addresses
  - Sometimes would deface the served website
- July 16th, 2001 - outbreak
  - CRv1- contained bad randomness (fixed IPs searched)
  - CRv2 - fixed the randomness,
    - added DDOS of [www.whitehouse.gov](http://www.whitehouse.gov)
    - Turned itself off and on (on 1st and 16th of month)
  - August 4 - Code Red II
    - Different code base, same exploit
    - Added local scanning (biased randomness to local IPs)

# Worms and infection

- The effectiveness of a worm is determined by how good it is at identifying vulnerable machines
  - Morris used local information at the host
  - Code Red used what?
- Multi-vector worms use lots of ways to infect
  - E.g., network, DFS partitions, email, drive by downloads ...
  - Another worm, Nimda did this
- Lots of scanning strategies
  - Signpost scanning (using local information, e.g., Morris)
  - Random IP - good, but waste a lot of time scanning dark or unreachable addresses (e.g., Code Red)
  - Local scanning - biased randomness
  - Permutation scanning - instance is given part of IP space

# Other scanning strategies

- Hit-list scanning
  - Setup - use “low and slow” scanning to determine which hosts are vulnerable (i.e., create a *hit list*)
  - Start the worm, passing the list of vulnerable hosts, reduce/device the list at each host
  - Gets past the slow start part, gets right into the exponential
  - Essentially removes the window to stop worm



# Other scanning strategies

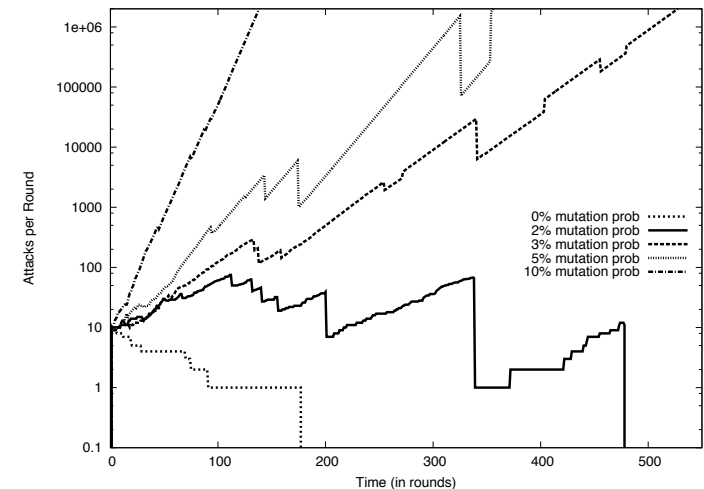
- The doomsday worm: a flash worm
  - Create a hit list of **all** vulnerable hosts
    - Staniford et al. argue this is feasible
    - Would contain a 48MB list
  - Do the infect and split approach
  - Use a zero-day vulnerability

• **Result: saturate the Internet is less than 30 seconds!**

# Parasitic worm ...

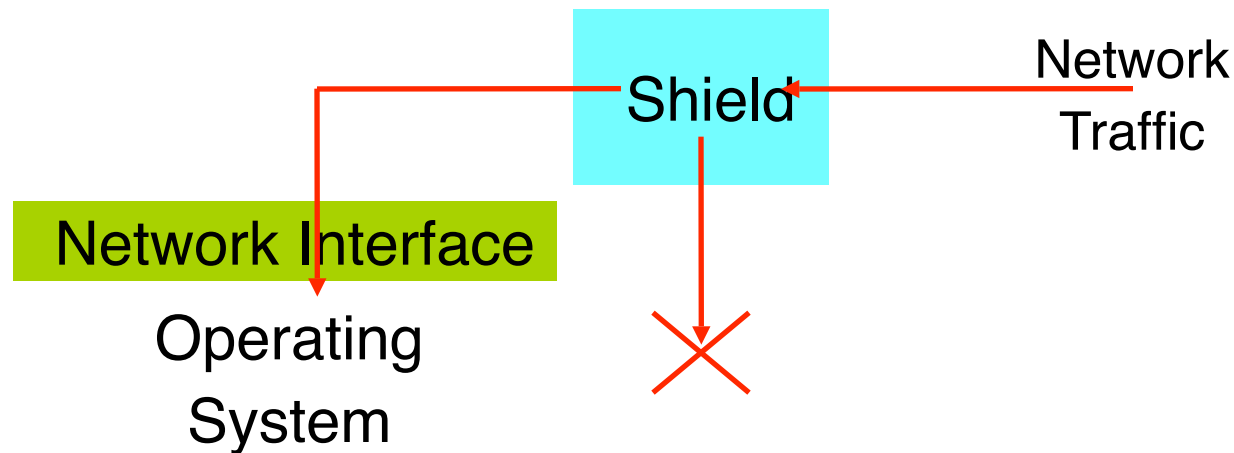
- Insight: most worm mitigation strategies are based on the detection of attack as it occurs
- What if a program was smart enough to find new vulnerabilities on its own?
  - It could periodically change its infection strategy (*mutate*)
  - Then forget old attack vectors
  - Each mutation requires new detection mechanisms

**Result:** basically unstoppable, “point of criticality” reached only after a few mutations [Butler ‘05]



# Worms: Defense Strategies

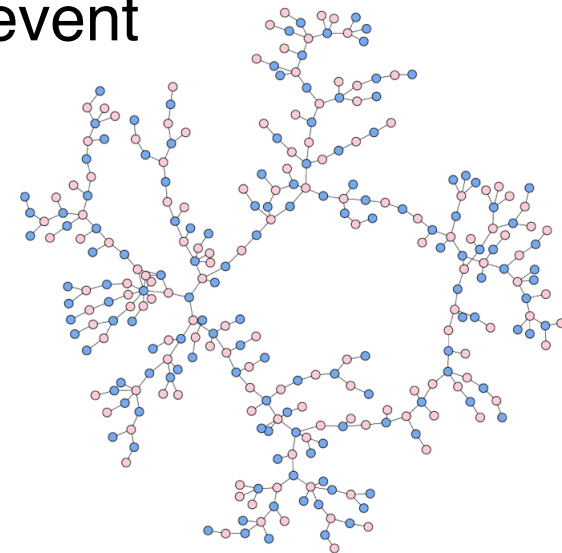
- (Auto) patch your systems: most, if not all, large worm outbreaks have exploited known vulnerabilities (with patches)
- Heterogeneity: use more than one vendor for your networks
- Shield (Ross): provides filtering for known vulnerabilities, such that they are protected immediately (analog to virus scanning)



- Filtering: look for unnecessary or unusual communication patterns, then drop them on the floor
  - This is the dominant method, getting sophisticated (Arbor Networks)

# Advanced Methods

- Quarantine - how do stop it once it is out?
  - *Internet Quarantine: Requirements for Containing Self-Propagating Code*. David Moore, Colleen Shannon, Geoffrey M. Voelker, Stefan Savage
- Assume you have a LAN/WAN environment
  - We have already talked about how to prevent
  - Q1: How do you recognize a worm?
  - Q2: How do you stop a worm?
- Much work in this area ...
  - number of new addresses contacted
  - number of incomplete IP handshakes
  - number of connections to new local hosts (CQI?)



# Denial of Service

- Intentional prevention of access to valued resource
  - CPU, memory, disk (system resources)
  - DNS, print queues, NIS (services)
  - Web server, database, media server (applications)
- This is an attack on *availability* (*fidelity*)
- **Note:** launching DOS attacks is easy
- **Note:** preventing DOS attacks is hard
  - Mitigate the path most frequently traveled

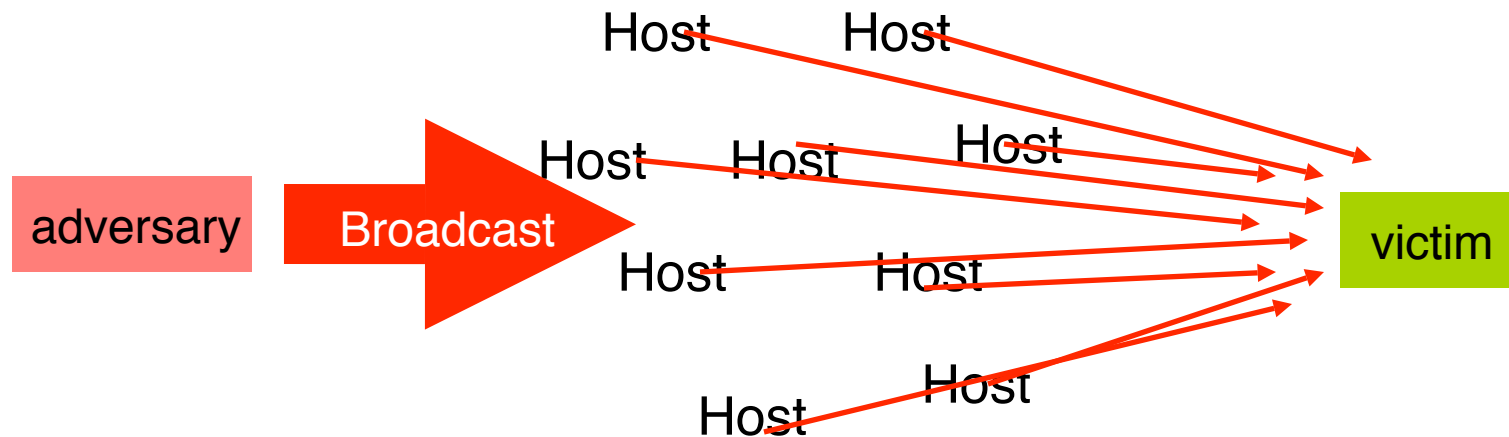
# D/DOS (generalized by Mirkovic)

- Send a stream of packets/requests/whatever ...
  - many PINGS, HTML requests, ...
- Send a few malformed packets
  - causing failures or expensive error handling
  - low-rate packet dropping (TCP congestion control)
  - “ping of death”
- Abuse legitimate access
  - Compromise service/host
  - Use its legitimate access rights to consume the rights for domain (e.g., local network)
  - E.g., someone runs a recursive file operation on root of NFS partition



# SMURF Attacks

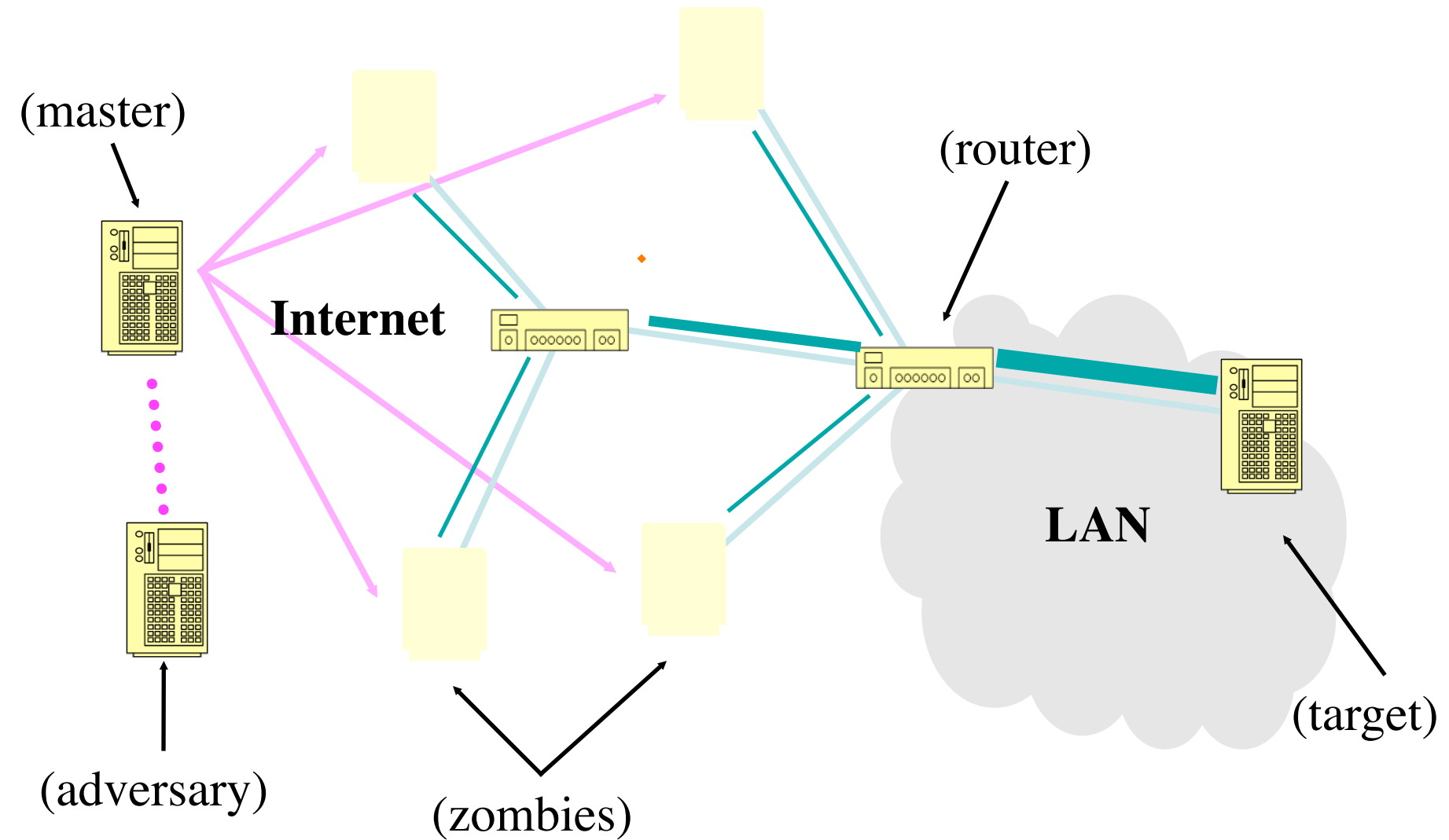
- This is one of the deadliest and simplest of the DOS attacks (called a *naturally amplified* attack)
  - Send a large number of PING packets on the broadcast IP addresses (e.g., 192.168.27.254)
  - Set the source packet IP address to be your victim
  - All hosts will reflexively respond to the ping at your victim
  - ... and it will be crushed under the load.
  - Fraggle: UDP based SMURF



# Distributed denial of service

- **DDOS**: Network oriented attacks aimed at preventing access to network, host or service
  - Saturate the target's network with traffic
  - Consume all network resources (e.g., SYN)
  - Overload a service with requests
    - Use “expensive” requests (e.g., “sign this data”)
  - Can be extremely costly (e.g., Amazon)
- Result: service/host/network is unavailable
- Frequently distributed via other attack
- **Note**: IP is often hidden (spoofed)

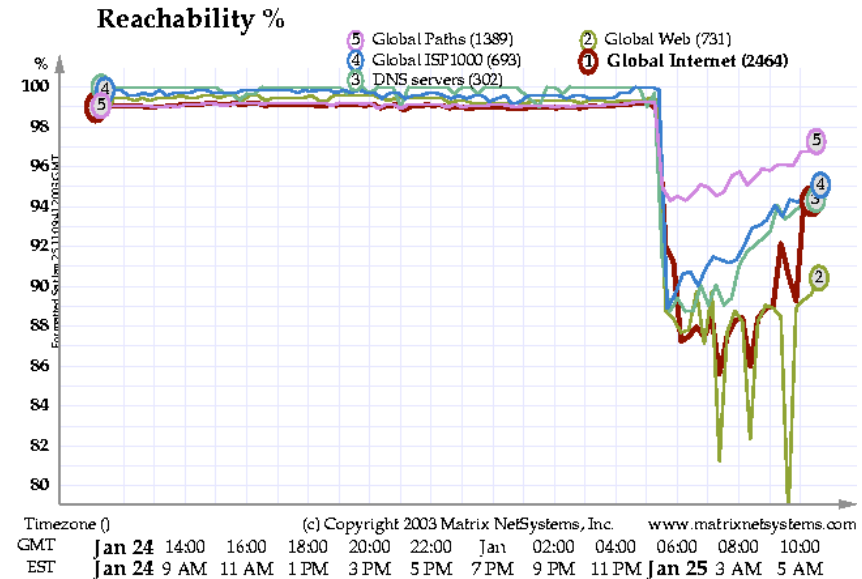
# The canonical DDOS attack



# Why DDOS

- What would motivate someone DDOS?

- An axe to grind ...
- Curiosity (script kiddies) ...
- Blackmail
- Information warfare ...



- Internet is an open system ...

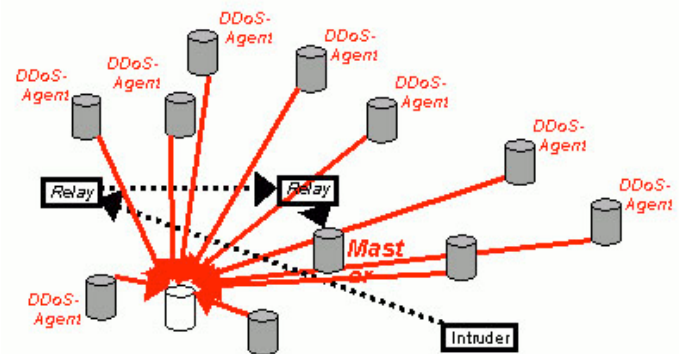
- Packets not authenticated, probably can't be
  - Would not solve the problem just move it (firewall)
- Too many end-points can be remote controlled

# Why is DDOS possible? (cont.)

- Interdependence - services dependent on each other
  - E.g., Web depends on TCP and DNS, which depends on routing and congestion control, ...
- Limited resources (or rather *resource imbalances*)
  - Many times it takes few resources on the client side to consume lots of resources on the server side
  - E.g., SYN packets consume lots of internal resources
- You tell me .. (as said by Mirkovic et al.)
  - Intelligence and resources not co-located
  - No accountability
  - Control is distributed

# DDoS Mitigation

- Better Host Security
  - Limit availability of zombies, not feasible
  - Prevent compromise, viruses, ...
- Quality of Service Guarantees (QoS)
  - Pre- or dynamically allocate bandwidth
  - E.g., diffserv, RSVP
  - Helps where such things are available ...
- Protocols
  - *traceback* (reconstruct path based on “marked” packets)
  - *pushback* (back-pressure)
- Content replication
  - E.g., CDS, for static content
- Ingress/Egress Filtering
  - Helps spoofed sources, little else



# DDOS Reality

- None of the “protocol oriented” solutions have really seen any adoption
  - too many untrusting, ill-informed, mutually suspicious parties must play together well (*hint*: human nature)
  - solution have many remaining challenges
- Real Solution
  - Large ISP police their ingress/egress points very carefully
  - Watch for DDOS attacks and filter appropriately
    - e.g., BGP (routing) tricks, blacklisting, whitelisting
  - Products in existing that coordinate view from many points in the network to identify upswings in
  - Interestingly, this is the same way they deal with *worms* ...