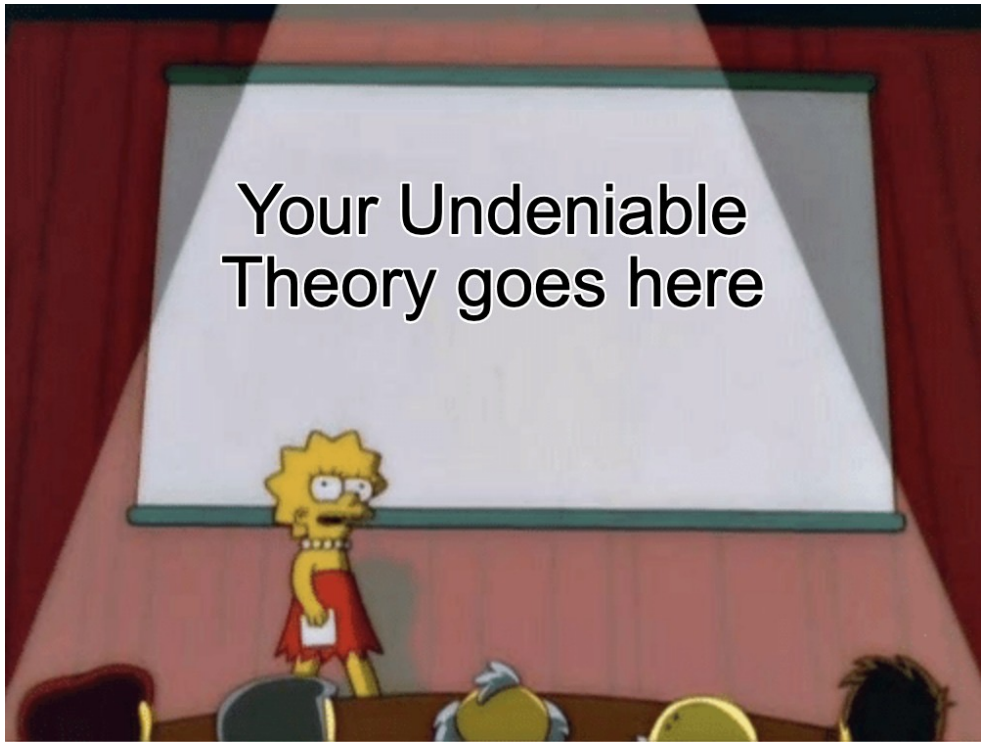
CSCI 445:
Mobile Application Security

Prof. Adwait Nadkarni

Derived from slides by William Enck

Project Presentations

- *On April 30th*
- These are "status" presentations of *7-15 mins duration (depending on # groups)*
  - RQs
  - Analysis you are doing
  - Findings (optional)
  - Anticipated Results and Findings
- 1 – 5 bonus credits
- *Let me know by April 18th, i.e., Thursday*




Worms

Worms

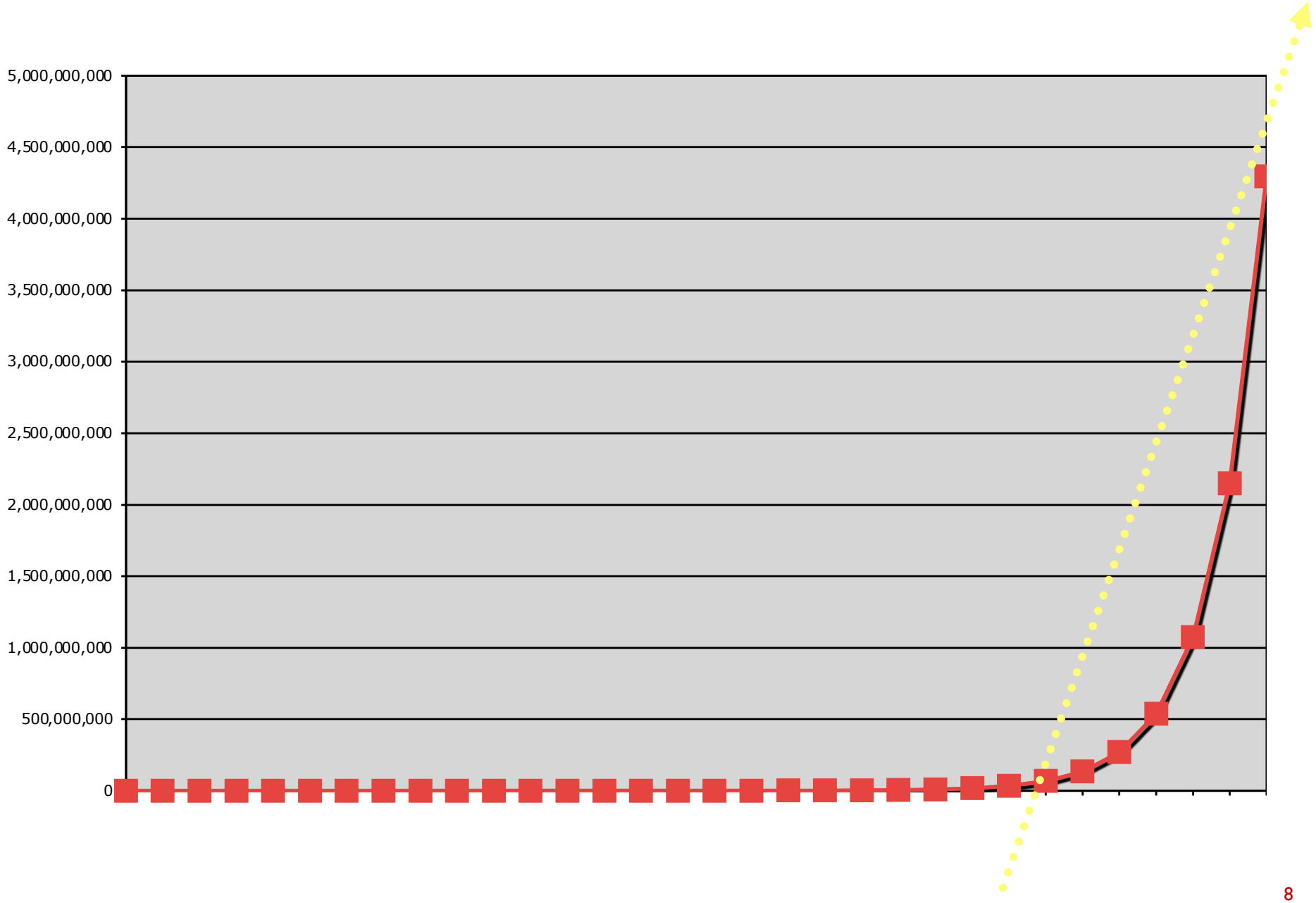
- A worm is a self-propagating program that:
  1. Exploits some vulnerability on a target host
  2. (often) embeds itself into a host …
  3. Searches for other vulnerable hosts …
  4. Goto step 1

The Danger

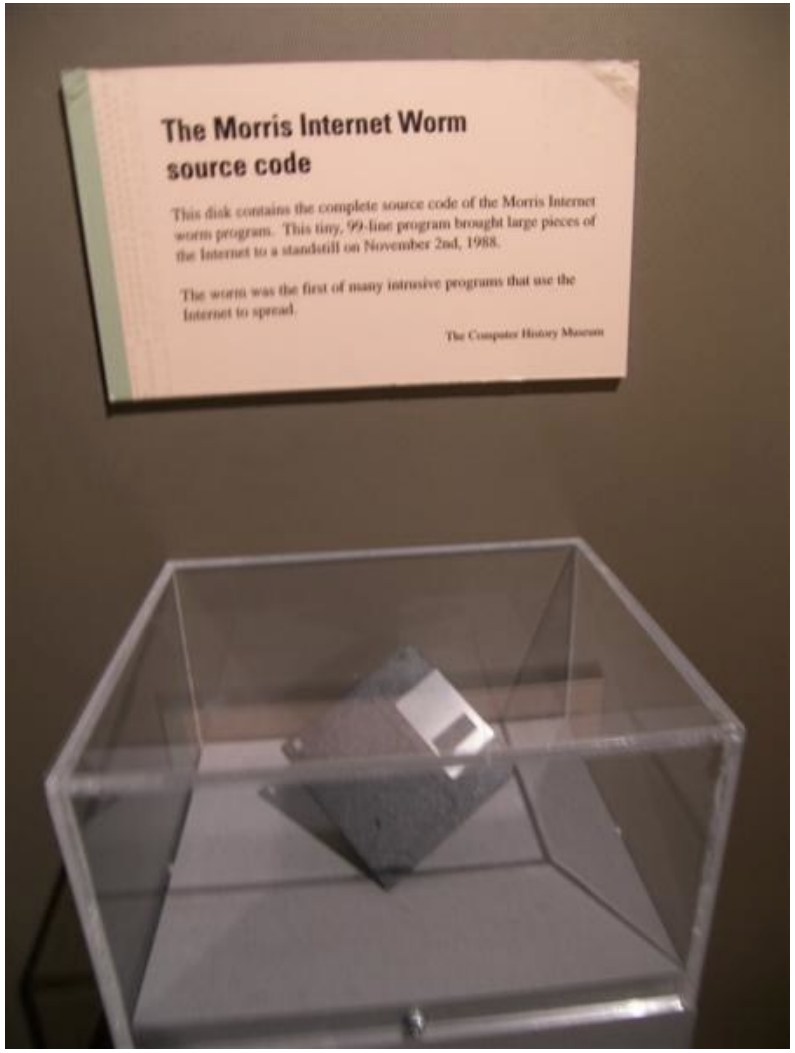
- What makes worms so dangerous is that infection grows at an exponential rate
  - A simple model:
    - *s* (search) is the time it takes to find vulnerable host
    - *i* (infect) is the time is take to infect a host
  - Assume that t=0 is the *worm outbreak*, the number of hosts at t=j is

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

The result


5,000,000,000
4,500,000,000
4,000,000,000
3,500,000,000
3,000,000,000
2,500,000,000
2,000,000,000
1,500,000,000
1,000,000,000
500,000,000
0

The Morris Worm



Robert Morris

- 1988: Graduate student at Cornell University
- Son of Robert Morris, chief scientist at National Computer Security Center (division of NSA)
- Now a professor at MIT

November 2nd, 1988

- 6pm: someone ran a program at a computer at MIT
- The program collected host, network, and user info...
- ... and then spread to other machines running Sun 3, VAX, and some BSD variants
- ... rinse and repeat

November 2nd, 1988

- Computers became multiply infected
- Systems became overloaded with processes
- Swap space became exhausted, and machines failed
- Wednesday night: UC Berkeley captures copy of program

- 5AM Thursday: UC Berkeley builds *sendmail* patch to stop spread of worm
- Difficult to spread knowledge of fix
  - Not coincidentally, the Internet was running slow
- *Estimated at* around 6,000 machines (~10% of Internet) infected at cost of $10M-$100M

Morris Worm: Attack Vectors

- rsh: terminal client with network (IP)-based authentication
- fingerd: used *gets* call without bounds checking
- sendmail: DEBUG mode allows remote user to run commands
  - lots of sendmail daemons running in DEBUG mode (on by default)

Morris Worm: Propagation

- Worm would ask host if it was infected
  - If answer was no, worm would infect
  - If answer was yes, worm would infect with some small probability (to thwart trivial countermeasure)
- But... bug allowed worm to spread much faster than anticipated, infecting the same machines multiple times
- Lesson: Always thoroughly debug your worms.

Code Red - 2001

- Exploited a Microsoft IIS web-server buffer overflow
  - Scans for vulnerabilities over random IP addresses
  - Sometimes would deface the compromised website
- Initial outbreak on July 16th, 2001
  - version 1: contained bad randomness (fixed IPs searched)
  - version 2: fixed the randomness,
    - added DDoS of www.whitehouse.gov
    - Turned itself off and on (on 1st and 19th of month, attack 20-27th, dormant 28-31st)
- August 4 - Code Red II
  - Different code base, same exploit
  - Added local scanning (biased randomness to local IPs)
  - Killed itself in October of 2001

Stuxnet

- First reported June 2010
- Exploited **unknown vulnerabilities**
  - Not one zero-day
  - Not two zero-days
  - Not three zero-days
  - But four zero-days!
    - print spooler bug
    - handful of escalation-of-privilege vulnerabilities

Stuxnet

- Spread through infected USB drives
  - bypasses "***air gaps***"
- Worm actively targeted SCADA systems (i.e., industrial control systems)
  - looked for WINCC or PCS 7 SCADA management system
    - attempted 0-day exploit
    - also tried using default passwords
  - apparently, specifically targeted Iran's nuclear architecture

Stuxnet

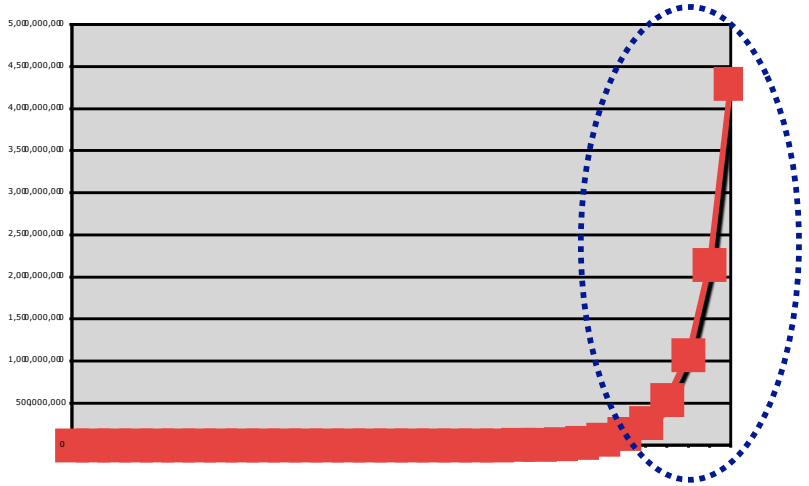
- Once SCADA system compromised, worm attempts to reprogram Programmable Logic Controllers (PLCs)
- Forensics aggravated by lack of logging in SCADA systems

Worms and infection

- **The effectiveness of a worm is determined by how good it is at identifying vulnerable machines**
- Multi-vector worms use lots of ways to infect: e.g., network, email, drive by downloads, etc.
- Example scanning strategies:
  - **Random IP:** select random IPs; wastes a lot of time scanning "dark" or unreachable addresses (e.g., Code Red)
  - **Signpost scanning:** use info on local host to find new targets (e.g., Morris)
  - **Local scanning:** biased randomness
  - **Permutation scanning:** "hitlist" based on shared pseudorandom sequence; when victim is already infected, infected node chooses new random position within sequence

Other scanning strategies

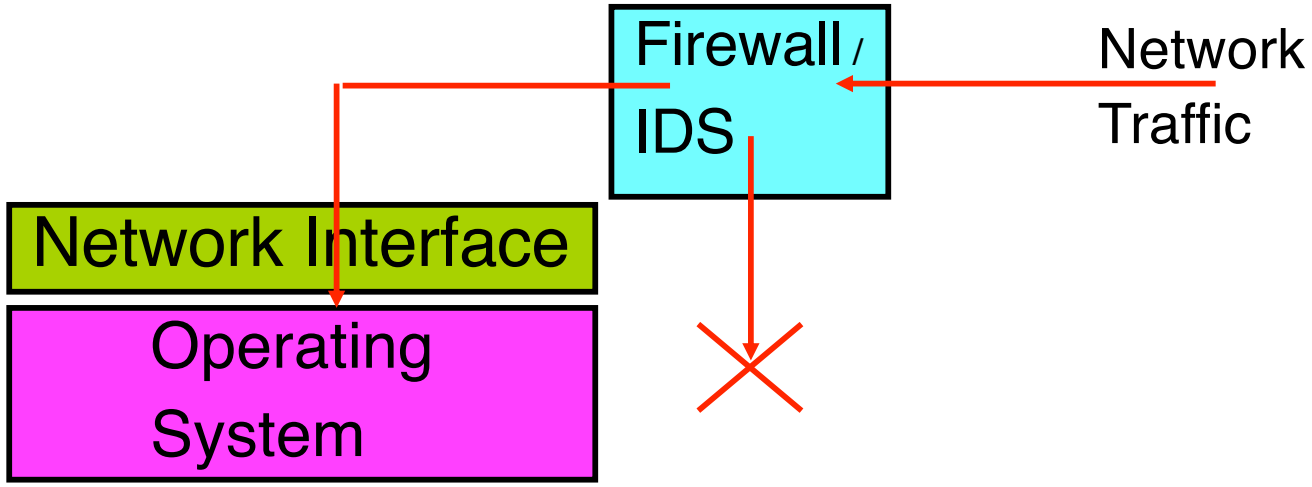
- The doomsday worm: a flash worm
  - Create a hit list of all vulnerable hosts
    - Staniford et al. argue this is feasible
    - Would contain a 48MB list
  - Do the infect and split approach
  - Use a zero-day exploit



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

Worms: Defense Strategies

- (Auto) **patch** your systems: most large worm outbreaks have exploited known vulnerabilities (Stuxnet is an exception)
- **Heterogeneity**: use more than one vendor for your networks
- **IDS**: 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

Denial-of-Service (DoS)

Denial-of-Service (DoS)

- 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**
- Launching DoS attacks is easy
- Preventing DoS attacks is very hard

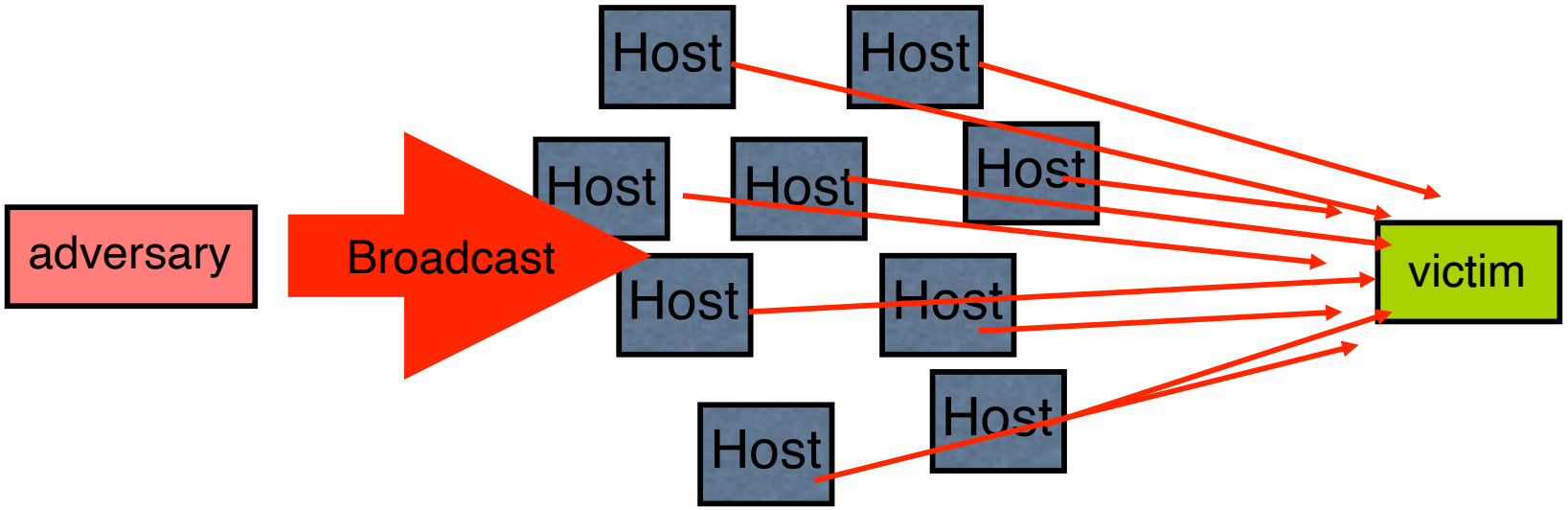
Canonical DoS - Request Flood

- Overwhelm some resource with requests
- e.g., web-server, phone system
- Most effective when processing request is expensive

Smurf Attacks

Example: SMURF Attacks

- Simple DoS attack:
  - Send a large number PING packets to a network's 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.
  - This is an **amplification attack** and a **reflection attack**



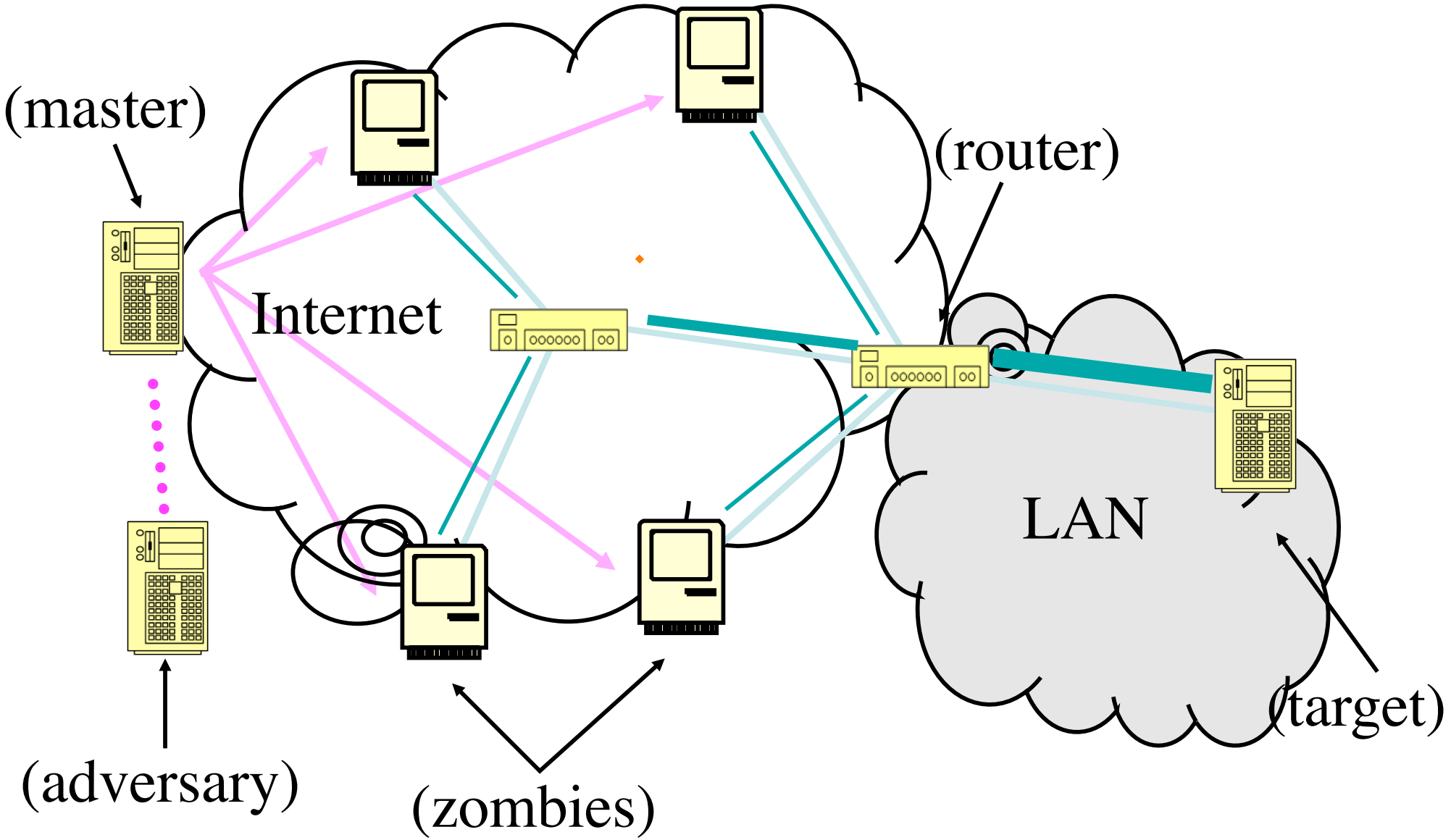
Distributed Denial-of-service (DDoS)

- 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 flooding)
  - Overload a service with requests
    - Use "expensive" requests (e.g., "sign this data")
  - Can be extremely costly
- Result: service/host/network is unavailable
- Criminals sometimes use DDoS for racketeering
- Note: IP addresses of perpetrators are often hidden (spoofed)

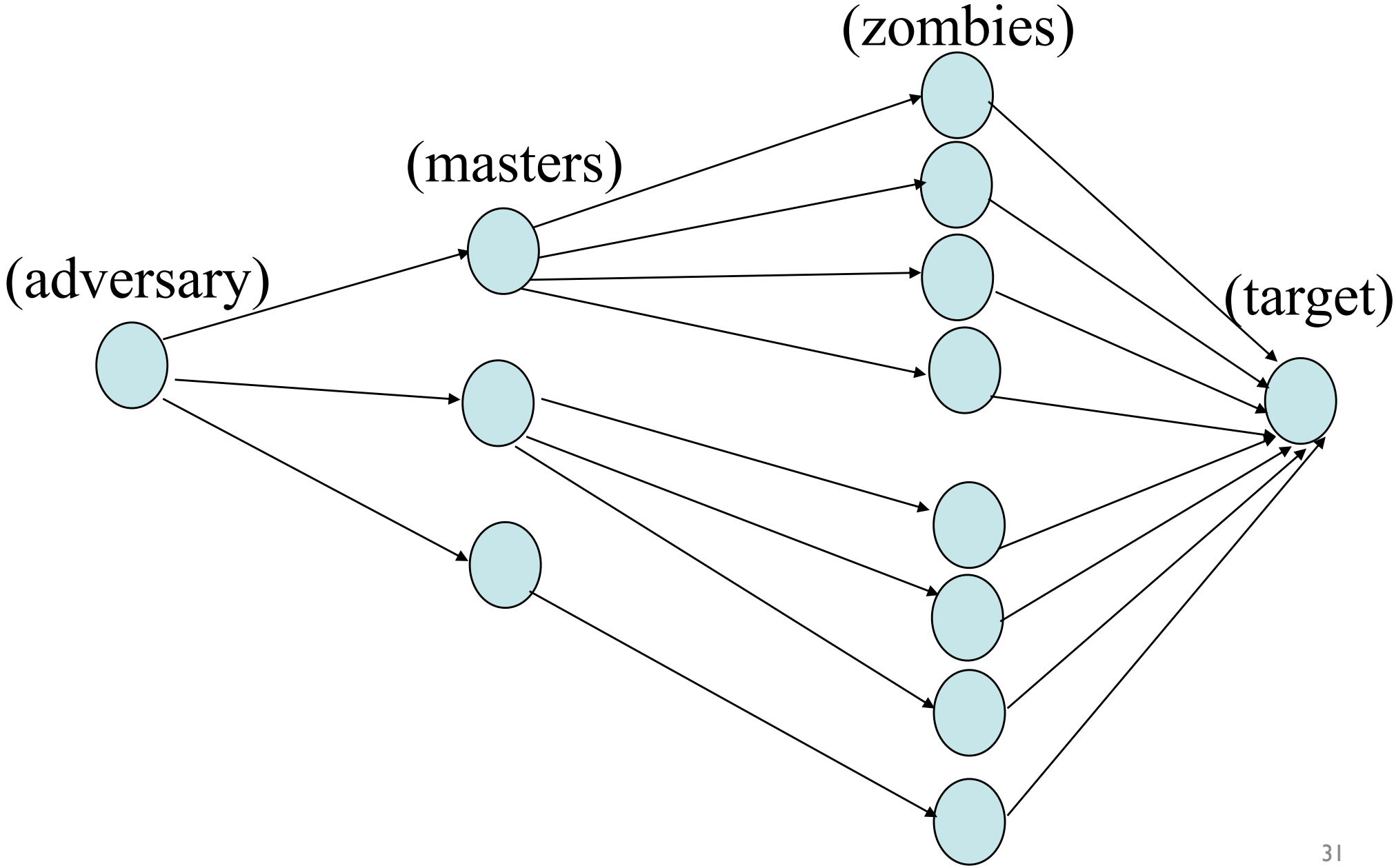
(D)DoS Techniques 101 (Don't do these.)

- Send a stream of legitimate 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)

The canonical DDoS attack



Adversary Network



Why DDoS?

- Motivations:
  - An axe to grind
  - Curiosity (script kiddies)
  - Blackmail / racketeering
  - Information warfare
  - Distraction

Q: An easy fix?

- How do you solve distributed denial of service?

Simple DDoS Mitigation

- **Ingress/Egress Filtering**: Helps spoofed sources, not much else
- Better Security
  - Limit availability of zombies (not feasible)
  - Prevent compromise and viruses (maybe in wonderful magic land where it rains chocolate and doughnuts)
- Quality of Service Guarantees (QoS)
  - Pre- or dynamically allocated bandwidth (e.g., diffserv)
  - Helps where such things are available
- Content replication
  - E.g,. CDS
  - Useful for static content

The End