CS155: Computer and Network Security

Programming Project 3 – Spring 2005

Shayan Guha
sguha05@stanford.edu
(with many slides “borrowed” from Matt Rubens)

Project Overview

1) Use standard network monitoring tools to examine different networking protocols
2) Use a packet capture library to automatically detects and records FTP transfers
3) Write a program to perform an injection attack on the RLOGIN protocol
4) Write a simple intrusion detection system to identify SYN floods and port scans

Goals of the assignment

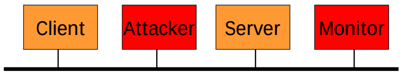
- Get some hands on networking experience
- Learn how secure different protocols are
- Learn about common attacks on clear-text protocols
- See what goes into building a basic network intrusion detection system
- DON’T end up in jail
  - Never test your code outside of the boxes environment!

Setup

- You are given four cow images corresponding to three separate machines on the network
  - Client, server, and attacker, monitor
- There are a number of users on the client sending network requests to services on the server
- First, the attacker (you!) is trying to perform different attacks (the first 3 parts) on the client and server
- Later, you as the monitor are trying to detect SYN floods and port scans from the attacker

Setup (2)

- All four boxes are located on the same Ethernet hub
- Ethernet is a broadcast medium
  - Every machine sees every packet, regardless of address!
    - Normally, packets not intended for a host are discarded by the network card
    - But in promiscuous mode all packets are available!



Setup (3)

- To start up the boxes for the first 3 parts, follow these steps
  - xterm –e ./string &
    - Make sure to use the copy of *string* included with the cow images!
      - Otherwise the attacker will not be to see the network traffic.
  - xterm –e [open|closed]box clientcow 10.64.64.64 &
  - xterm –e [open|closed]box servercow 10.64.64.65 &
  - xterm –e [open|closed]box attackcow 10.64.64.66 &
- You must use these exact IP addresses!

Setup (4)

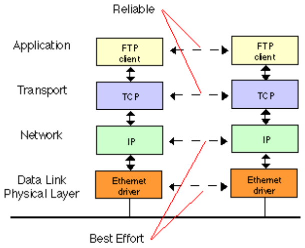
- You are NOT given an account on the client and server machines
  - If you're good you might get one soon!
    - Once you have a password, you can remotely shutdown the client and server with
      - ssh [username]@[ipaddr] /sbin/halt
        - We installed halt as setuid-root (bad idea in general!)
  - But until then, you won't be able to do a clean shutdown on clientcow and servercow
    - So keep a backup of the original images to avoid fscking

Quick TCP/IP Review

TCP/IP Overview

- On this assignment, we are only dealing with protocols that run over TCP/IP (except for ICMP echo requests in part 4)
- We assume a basic knowledge on the level of packets and ports
  - If you're not that comfortable with this, stop by office hours

Relevant Network Layers



From http://www.erg.abdn.ac.uk/users/gorry/course/images/ftp-tcp-enet.gif

Cliffs Notes Version

- Each TCP packet that you see is actually a TCP packet wrapped inside of an IP packet wrapped inside of an Ethernet packet.

Ethernet Header
IP Header
TCP Header
Application Data

TCP Flags

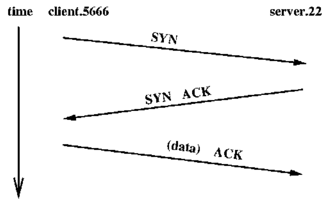
- Synchronize flag [SYN]
  - Used to initiate a TCP connection
- Acknowledgement flag [ACK]
  - Used to confirm received data
- Finish flag [FIN]
  - Used to shut down the connection

TCP Flags (2)

- Push flag [PSH]
  - Do not buffer data on receiver side – send directly to application level
- Urgent flag [URG]
  - Used to signify data with a higher priority than the other traffic
    - I.e Ctrl+C interrupt during an FTP transfer
- Reset flag [RST]
  - Tells receiver to tear down connection immediately

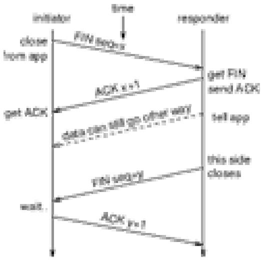
Connection setup

- “Three-way handshake”



From http://www.cs.colorado.edu/~tor/sadocs/tcpip/3way.png

Connection termination



- Either side can initiate termination
  - Note that the first FIN packet may still contain data!

From http://homepages.feis.herts.ac.uk/~cs2_sn2/sn2-img62.png

The actual assignment (finally!)

Phase 1: Sniffing

- Goal: observe network traffic, learn about different protocols
  - Also: gain access to client and server machines in order to make Phases 2 and 3 easier!
- Installed tools (must be run as root):
  - Tcpdump
    - Old faithful, just gives raw packet info
  - Tethereal
    - Like tcpdump, but with more smarts about protocols
  - Tcpflow
    - Focuses on the payload of the packets
    - Great for examining application level data (i.e passwords)!

Tcpdump options

- All three network monitoring tools take similar command line options
  - Can filter packets by address, port, protocol, length, TCP flags, etc.
    - Make sure to read the tcpdump manpage closely!
  - For your submission, we want you to list the options that you used to isolate the packets containing username/password information.

Phase 2: File Eavesdropping

- Manual packet sniffing is an interesting exercise, but programmatically capturing packets is much more powerful
- In this part of the assignment, you will write a program to reconstruct a sniffed FTP file transfer

Libpcap

- Libpcap is a packet capture library written in C
  - It allows you to write code to automate packet sniffing attacks.
- The library is fairly simple to use
  - Pseudocode:

```
while (true) {
   packet = pcap_next();
   // do something with the packet
}
```

- We give you starter code in /home/user/pp3/sniff.c on the attackcow image.

What to do

- Figure out which packets correspond to an FTP file transfer
- Detect when a transfer starts and create a local file to store the data
- Extract data from packets and write them to the file
- Figure out when the transfer completes, close the file, and exit the program

What to do (2)

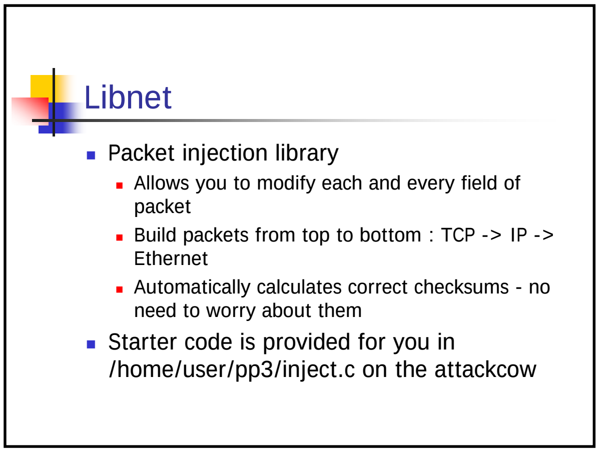
- The hard part is figuring out how to parse the various layers of headers.
  - You can find the header definitions at:
    - Ethernet: /usr/include/net/ethernet.h
    - IP: /usr/include/netinet/ip.h
    - TCP: /usr/include/netinet/tcp.h
- You'll also need to figure out how FTP data transfers work
  - Using the techniques you learned in Phase 1 might be more productive than poring over protocol docs

Phase 3: Packet Injection

- RLOGIN- allows remote login session
  - Very similar to Telnet
- Does not ask for password if the client machine is mentioned in /etc/hosts.equiv or ~/.rhosts
  - (big convenience…. even bigger vulnerability)
- After authentication- the rest of the traffic is in the clear!
- Uses one TCP channel for communication

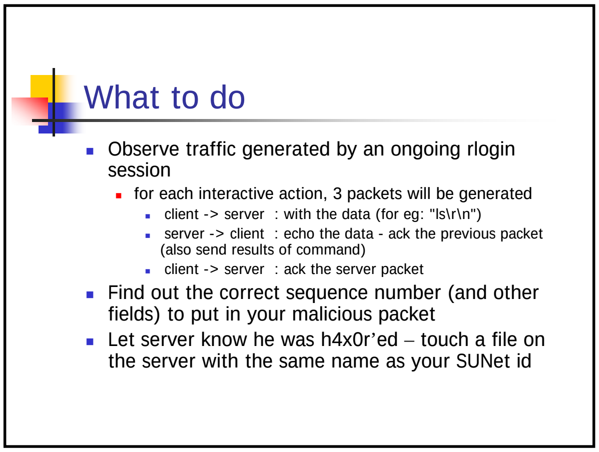
Attacks

- Can spoof an entire TCP connection
  - If the spoofed sender is present in /etc/hosts.equiv or ~/.rhosts, server won't ask for password
- Already established session can be hijacked by spurious injections (what you will do)
  - You can run any command on the server with the permissions of the client
    - i.e. /sbin/halt (if halt is setuid-root), rm –rf, etc.



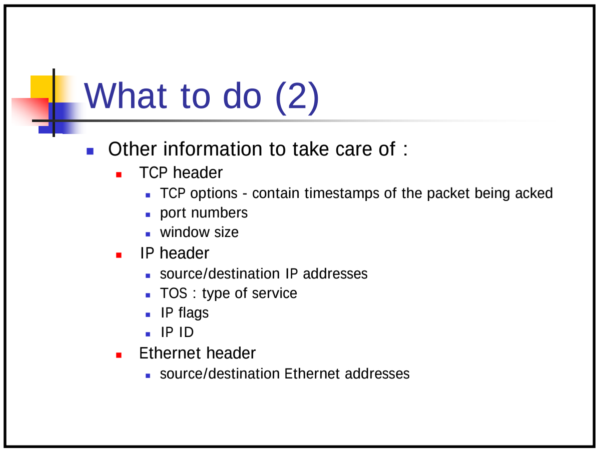
Libnet

- Packet injection library
  - Allows you to modify each and every field of packet
  - Build packets from top to bottom : TCP -> IP -> Ethernet
  - Automatically calculates correct checksums - no need to worry about them
- Starter code is provided for you in /home/user/pp3/inject.c on the attackcow



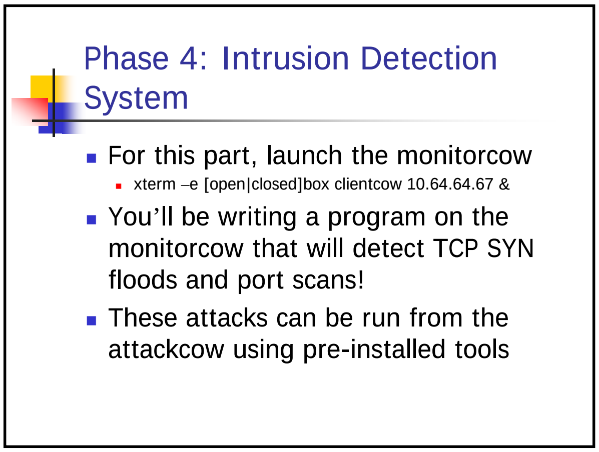
What to do

- Observe traffic generated by an ongoing rlogin session
  - for each interactive action, 3 packets will be generated
    - client -> server : with the data (for eg: "ls\r\n")
    - server -> client : echo the data - ack the previous packet (also send results of command)
    - client -> server : ack the server packet
- Find out the correct sequence number (and other fields) to put in your malicious packet
- Let server know he was h4x0r'ed – touch a file on the server with the same name as your SUNet id



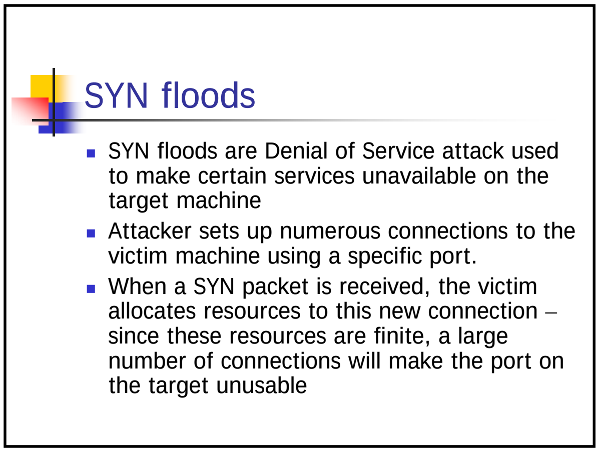
What to do (2)

- Other information to take care of :
  - TCP header
    - TCP options - contain timestamps of the packet being acked
    - port numbers
    - window size
  - IP header
    - source/destination IP addresses
    - TOS : type of service
    - IP flags
    - IP ID
  - Ethernet header
    - source/destination Ethernet addresses



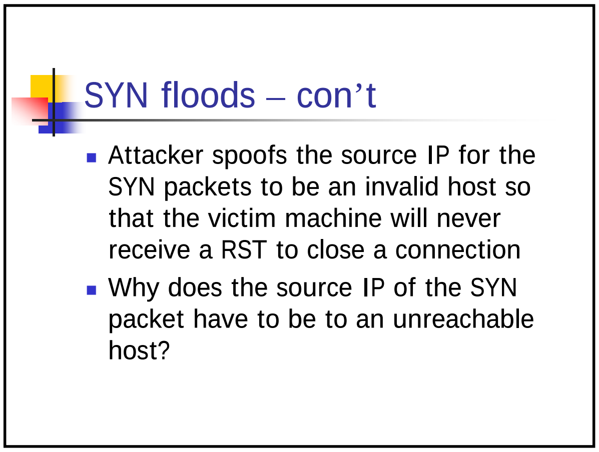
Phase 4: Intrusion Detection System

- For this part, launch the monitorcow
  - xterm –e [open|closed]box clientcow 10.64.64.67 &
- You'll be writing a program on the monitorcow that will detect TCP SYN floods and port scans!
- These attacks can be run from the attackcow using pre-installed tools



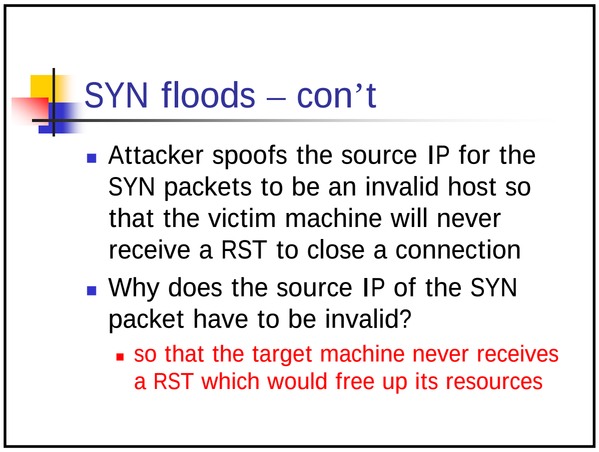
SYN floods

- SYN floods are Denial of Service attack used to make certain services unavailable on the target machine
- Attacker sets up numerous connections to the victim machine using a specific port.
- When a SYN packet is received, the victim allocates resources to this new connection – since these resources are finite, a large number of connections will make the port on the target unusable



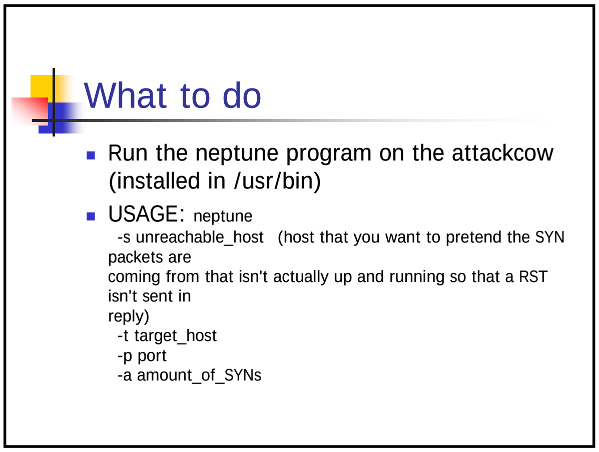
SYN floods – con't

- Attacker spoofs the source IP for the SYN packets to be an invalid host so that the victim machine will never receive a RST to close a connection
- Why does the source IP of the SYN packet have to be to an unreachable host?



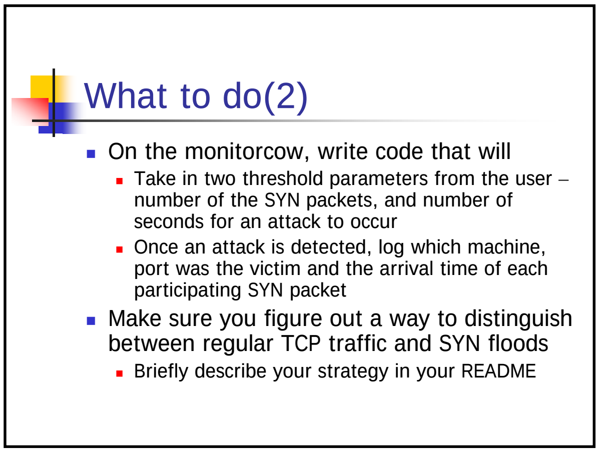
SYN floods – con't

- Attacker spoofs the source IP for the SYN packets to be an invalid host so that the victim machine will never receive a RST to close a connection
- Why does the source IP of the SYN packet have to be invalid?
  - so that the target machine never receives a RST which would free up its resources



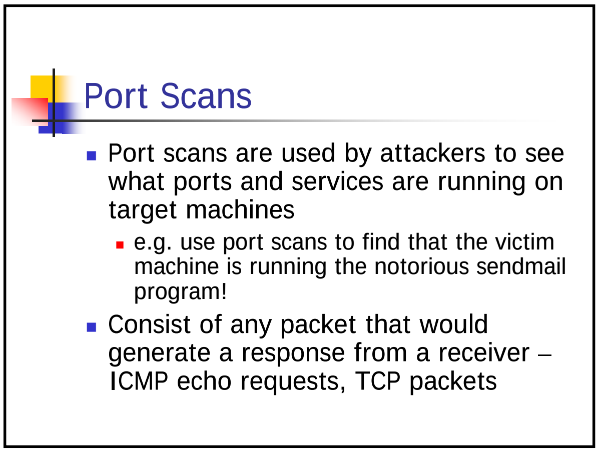
What to do

- Run the neptune program on the attackcow (installed in /usr/bin)
- USAGE: neptune
  -s unreachable_host (host that you want to pretend the SYN packets are
  coming from that isn't actually up and running so that a RST isn't sent in
  reply)
  -t target_host
  -p port
  -a amount_of_SYNs



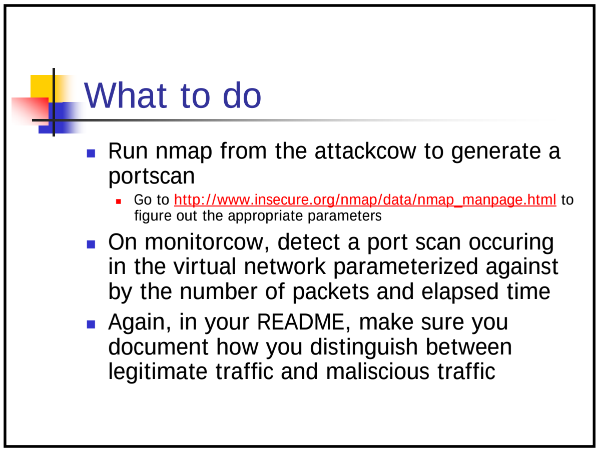
What to do(2)

- On the monitorcow, write code that will
  - Take in two threshold parameters from the user – number of the SYN packets, and number of seconds for an attack to occur
  - Once an attack is detected, log which machine, port was the victim and the arrival time of each participating SYN packet
- Make sure you figure out a way to distinguish between regular TCP traffic and SYN floods
  - Briefly describe your strategy in your README



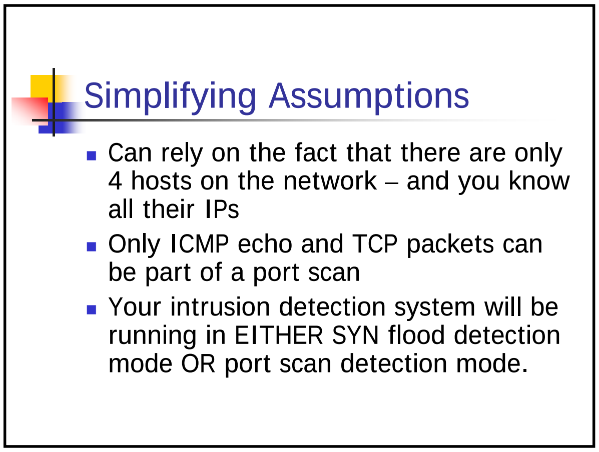
Port Scans

- Port scans are used by attackers to see what ports and services are running on target machines
  - e.g. use port scans to find that the victim machine is running the notorious sendmail program!
- Consist of any packet that would generate a response from a receiver – ICMP echo requests, TCP packets



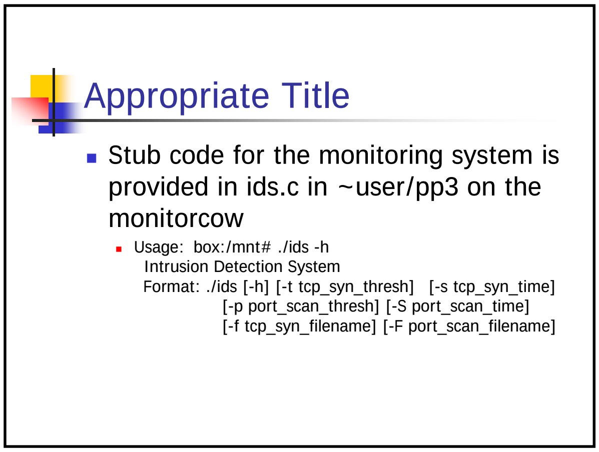
What to do

- Run nmap from the attackcow to generate a portscan
  - Go to http://www.insecure.org/nmap/data/nmap_manpage.html to figure out the appropriate parameters
- On monitorcow, detect a port scan occuring in the virtual network parameterized against by the number of packets and elapsed time
- Again, in your README, make sure you document how you distinguish between legitimate traffic and maliscious traffic



Simplifying Assumptions

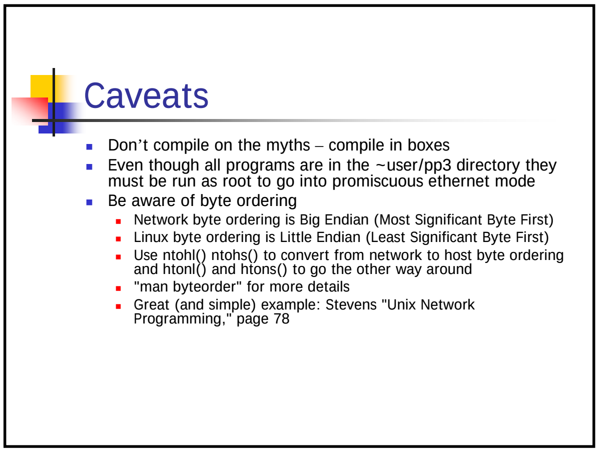
- Can rely on the fact that there are only 4 hosts on the network – and you know all their IPs
- Only ICMP echo and TCP packets can be part of a port scan
- Your intrusion detection system will be running in EITHER SYN flood detection mode OR port scan detection mode.



Appropriate Title

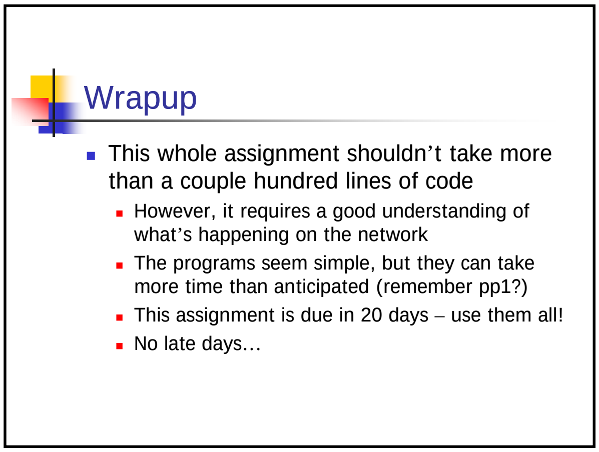
- Stub code for the monitoring system is provided in ids.c in ~user/pp3 on the monitorcow
  - Usage:

```
box:/mnt# ./ids -h
Intrusion Detection System
Format: ./ids [-h] [-t tcp_syn_thresh]   [-s tcp_syn_time]
              [-p port_scan_thresh] [-S port_scan_time]
              [-f tcp_syn_filename] [-F port_scan_filename]
```



Caveats

- Don't compile on the myths – compile in boxes
- Even though all programs are in the ~user/pp3 directory they must be run as root to go into promiscuous ethernet mode
- Be aware of byte ordering
  - Network byte ordering is Big Endian (Most Significant Byte First)
  - Linux byte ordering is Little Endian (Least Significant Byte First)
  - Use ntohl() ntohs() to convert from network to host byte ordering and htonl() and htons() to go the other way around
  - "man byteorder" for more details
  - Great (and simple) example: Stevens "Unix Network Programming," page 78



Wrapup

- This whole assignment shouldn't take more than a couple hundred lines of code
  - However, it requires a good understanding of what's happening on the network
  - The programs seem simple, but they can take more time than anticipated (remember pp1?)
  - This assignment is due in 20 days – use them all!
  - No late days…