CS 3204
Operating Systems

Lecture 38

Godmar Back

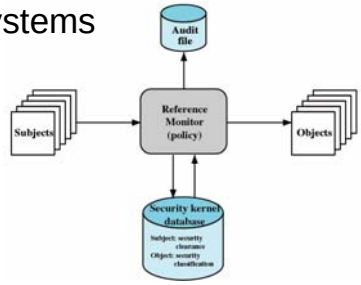
Announcements

- Project 4 due Wed, May 3, 11:59pm
  - Coming up rapidly, should have finished buffer cache implementation & testing and have designed inode format; should be working on extensible files (if you follow suggested implementation order)
- Skim 16.1-16.5 (security)
- Read chapter 13 (networking)

Security & Protection

Trusted Systems

- MLS-Multilevel Security
  - Unclassified
  - Confidential
  - Secret
  - Top Secret
- No read up
- No write down
  - *-property



Properties:
- Complete mediation (mandatory access control on every access)
- Isolated/tamper-proof reference monitor
- Verification (the hardest)

Some Attacks

- Abuse of valid privilege
  - Admin decides to delete your mp3s
- Denial of service attack
  - Run this loop on your P4:
    - while (1) { mkdir("x"); chdir("x"); }
- Sniffing/Listening attack
- Trojan Horse
- Worm or virus

Simple Stack Overflow Example

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

int
main()
{
    char buf[8];
    printf("Enter your name: ");
    gets(buf);
    printf("Your name is: %s\n", buf);
}

void
remove_all_files()
{
    printf("remove all files called!\n");
}
```

```
> nm ./stackattack | grep remove_all_file
080483e4 T remove_all_files
> od -h badinput
0000000 83e4 0804 83e4 0804 83e4 0804 83e4 0804
*
0000120
> ./stackattack < badinput
Enter your name: Your name is: ä
remove all files called!
```

- In practice, attacker usually sends position-independent code along that exec()'s a shell

Some Countermeasures

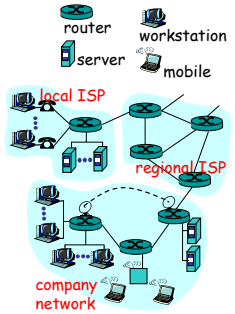
- Logging:
  - Keep an audit log of all actions performed
  - Protect log (from theft & forgery)
- Principle of least privilege
  - "need-to-know" basis
  - hard to implement: how can you be sure the program will still work? How can you be sure you've given just enough privileges and not more?
  - example: Linux SE
- Verification & Proofs
  - Problem of verifying the specification vs. implementation

Networking

(Most slides from Kurose/Ross: Computer Networking – A Top Approach Featuring The Internet)

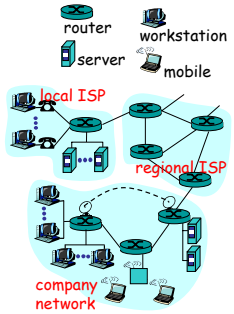
The Internet: "nuts and bolts" view

- millions of connected computing devices: *hosts = end systems*
- running *network apps*
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = ***bandwidth***
- *routers:* forward packets



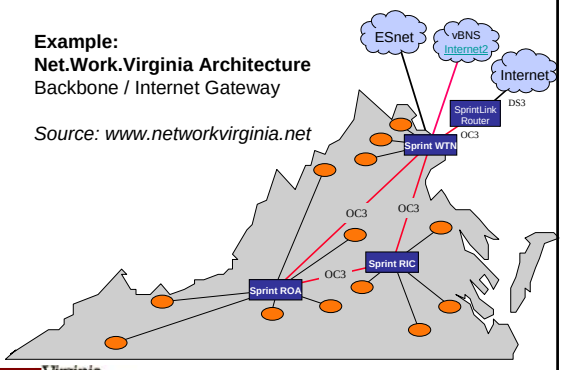
The Internet: "nuts and bolts" view

- *protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, FTP, PPP
- *Internet:* "network of networks"
  - loosely hierarchical
  - public Internet versus private intranets
  - Internet vs internet
- Internet standards
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



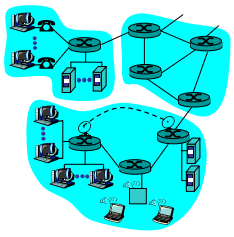
Example:
Net.Work.Virginia Architecture
Backbone / Internet Gateway

Source: www.networkvirginia.net



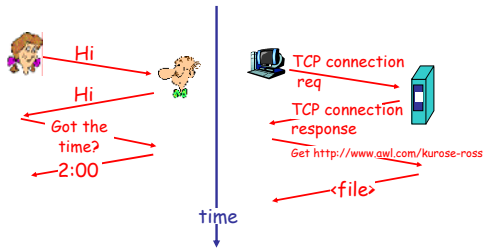
The Internet: a service view

- communication *infrastructure* enables distributed applications:
  - Web, email, games, e-commerce, file sharing
- communication services provided to apps:
  - Connectionless unreliable
  - Connection-oriented reliable



What's a protocol?

a human protocol and a computer network protocol:



What's a protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when msgs received, or other events

network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt