Chapter 13:
Operating Systems and Security

UNIX is basically a simple operating system,
but you have to be a genius to understand the simplicity.
— Dennis Ritchie

And it is a mark of prudence never to trust wholly
in those things which have once deceived us.
Rene Descartes

OS and Security

- OSs are large, complex programs
  - Many bugs in any such program
  - We have seen that bugs can be security threats
- Here we are concerned with security provided by OS
  - Not concerned with threat of bad OS software
- Concerned with OS as security ***enforcer***
- In this section we only scratch the surface

OS Security Challenges

- Modern OS is **multi-user** and **multi-tasking**
- OS must deal with
  - Memory
  - I/O devices (disk, printer, etc.)
  - Programs, threads
  - Network issues
  - Data, etc.
- OS must protect processes from other processes and users from other users
  - Whether accidental or malicious

OS Security Functions

- *Memory protection*
  - *Protect memory from users/processes*
- *File protection*
  - *Protect user and system resources*
- *Authentication*
  - *Determines and enforce authentication results*
- *Authorization*
  - *Determine and enforces access control*

Memory Protection

- *Fundamental problem*
  - *How to keep users/processes separate?*
- *Separation*
  - *Physical separation separate devices*
  - *Temporal separation one at a time*
  - *Logical separation sandboxing, etc.*
  - *Cryptographic separation make information unintelligible to outsider*
  - *Or any combination of the above*

Memory Protection

- **Fence** users cannot cross a specified address
  - Static fence fixed size OS
  - Dynamic fence fence register
- Base/bounds register lower and upper address limit
- Assumes contiguous space

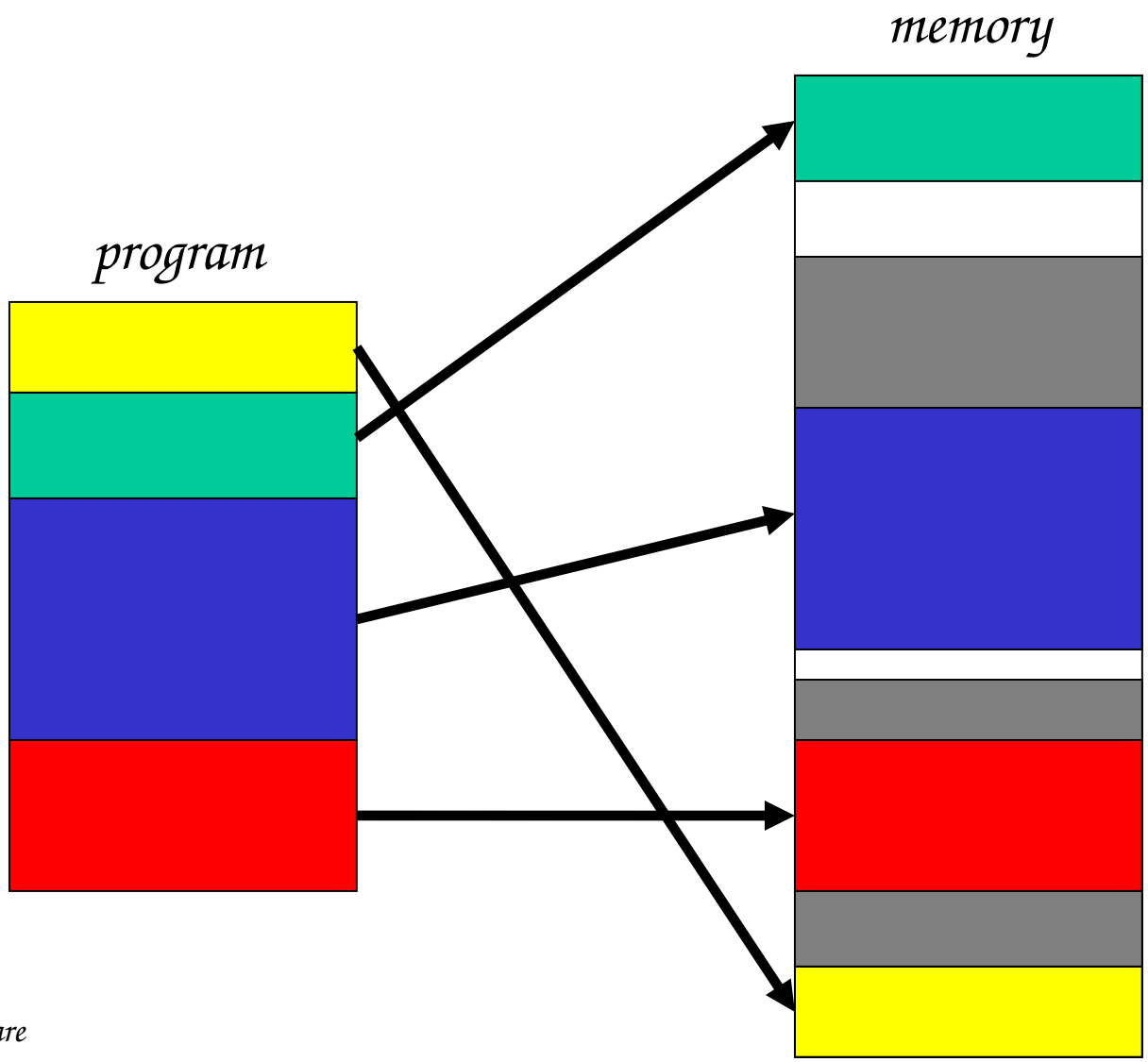
Memory Protection

- *Tagging specify protection of each address*
  - + *Extremely fine-grained protection*
  - - *High overhead can be reduced by tagging sections instead of individual addresses*
  - - *Compatibility*
- *More common is segmentation and/or paging*
  - *Protection is not as flexible*
  - *But much more efficient*

Segmentation

- Divide memory into logical units, such as
  - Single procedure
  - Data in one array, etc.
- Can enforce different access restrictions on different segments
- Any segment can be placed in any memory location (if location is large enough)
- OS keeps track of actual locations

Segmentation



Segmentation

- *OS can place segments anywhere*
- *OS keeps track of segment locations as* <segment,offset>
- *Segments can be moved in memory*
- *Segments can move out of memory*
- *All address references go thru OS*

Segmentation Advantages

- Every address reference can be checked
  - Possible to achieve **complete mediation**
- Different protection can be applied to different segments
- Users can share access to segments
- Specific users can be restricted to specific segments

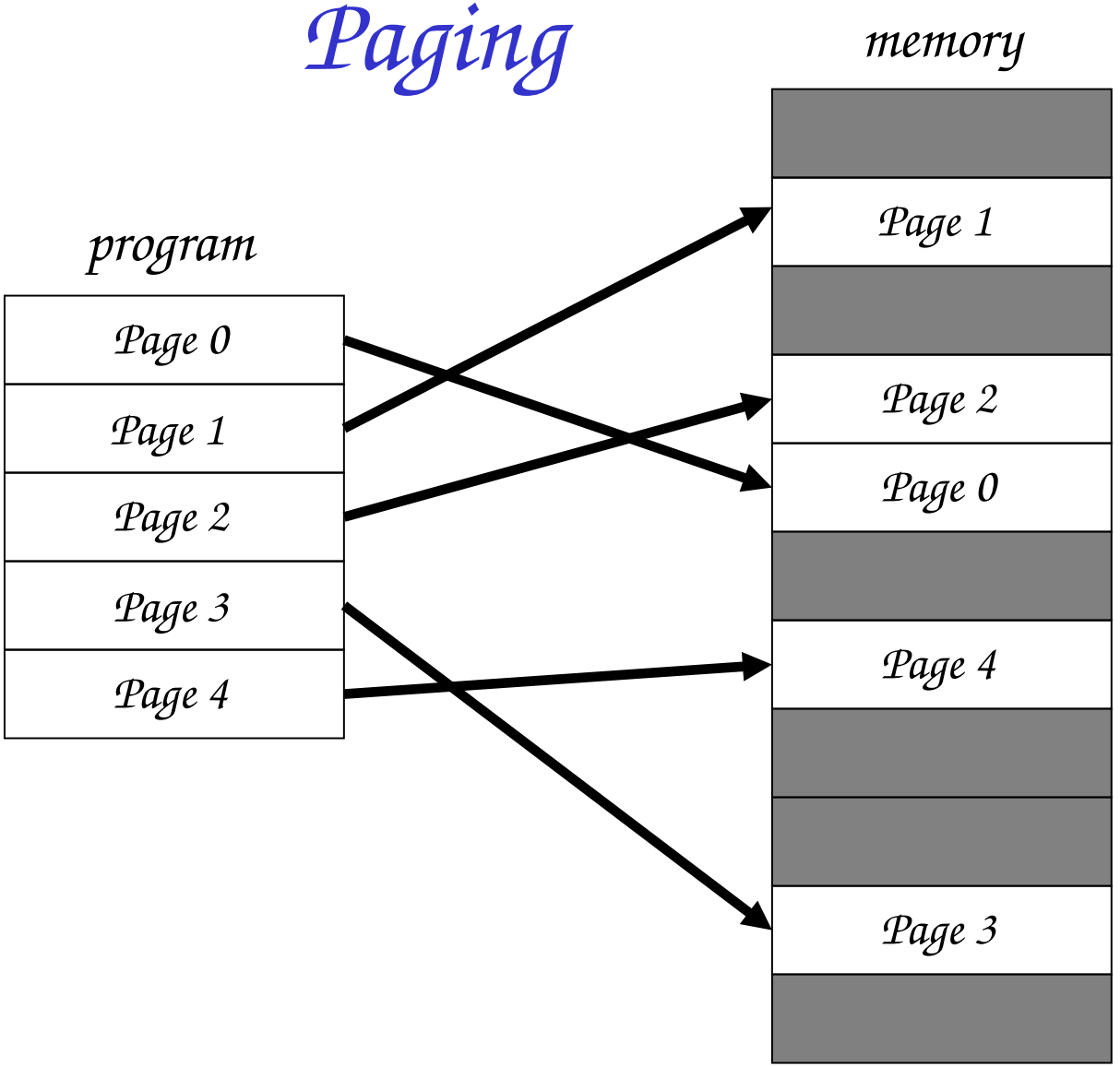
Segmentation Disadvantages

- *How to reference* <segment,offset> *?*
  - *OS must know* segment ***size*** *to verify access is within* segment
  - *But some segments can grow during execution (for example, dynamic memory allocation)*
  - *OS must keep track of* ***variable*** *segment sizes*
- *Memory fragmentation is also a problem*
  - *Compacting memory changes tables*
- *A lot of work for the OS*
- *More complex* ▲ *more chance for mistakes*

Paging

- *Like segmentation, but fixed-size segments*
- *Access via* <page,offset>
- *Plusses and minuses*
  - \+ *Avoids fragmentation, improved efficiency*
  - \+ *OS need not keep track of variable segment sizes*
  - \- *No logical unity to pages*
  - \- *What protection to apply to a given page?*

Paging


memory
program
Page 0
Page 1
Page 2
Page 3
Page 4
Page 1
Page 2
Page 0
Page 4
Page 3

Other OS Security Functions

- OS must enforce access control
- Authentication
  - Passwords, biometrics
  - Single sign-on, etc.
- Authorization
  - ACL
  - Capabilities
- These topics discussed previously
- OS is an attractive target for attack!

Trusted Operating System

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*
- *Judge based on specified policy*
- *Security depends on trust relationships*

- *Note: Some authors use different terminology!*

Trusted 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

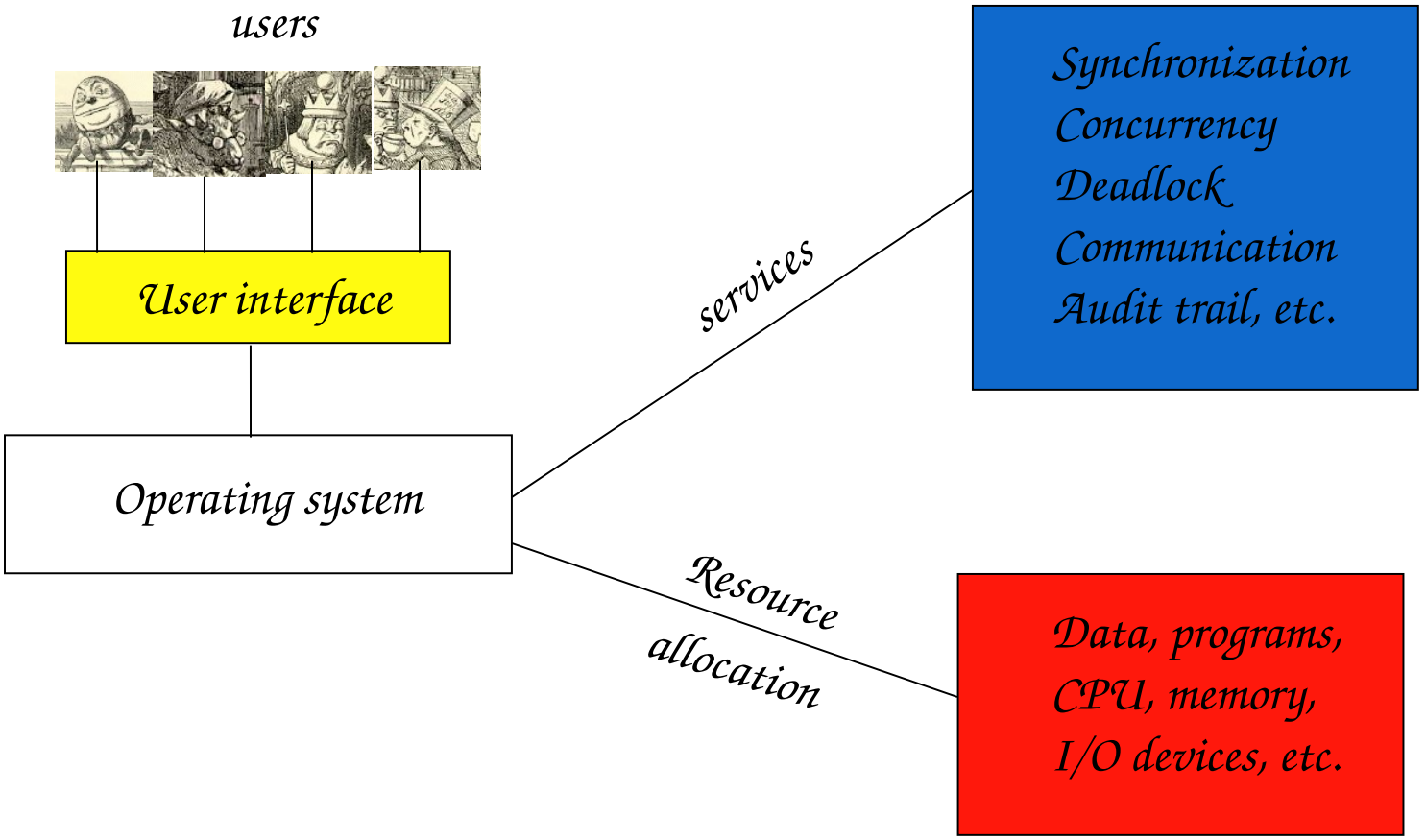
General Security Principles

- 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

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*

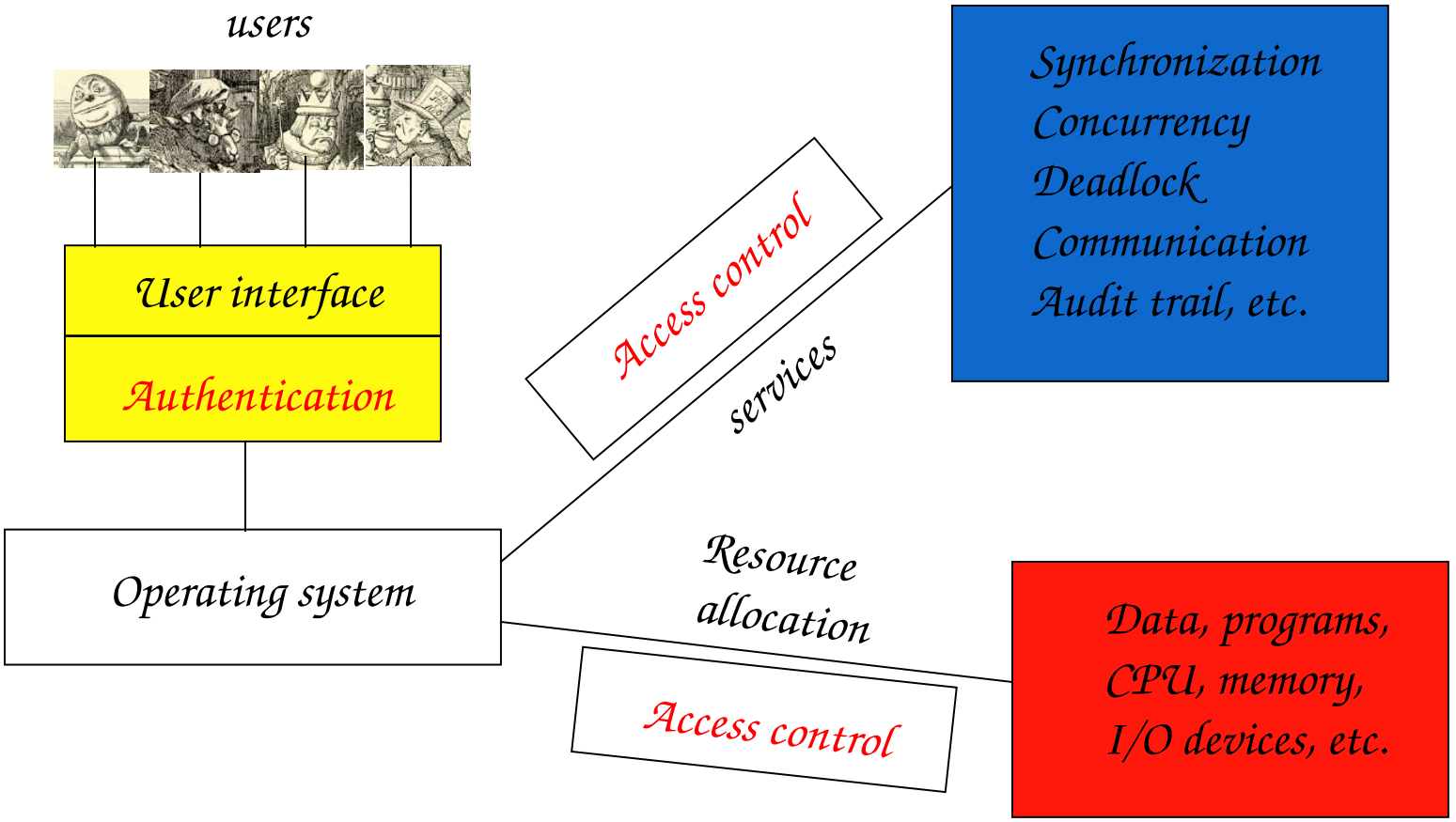
OS Services


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



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

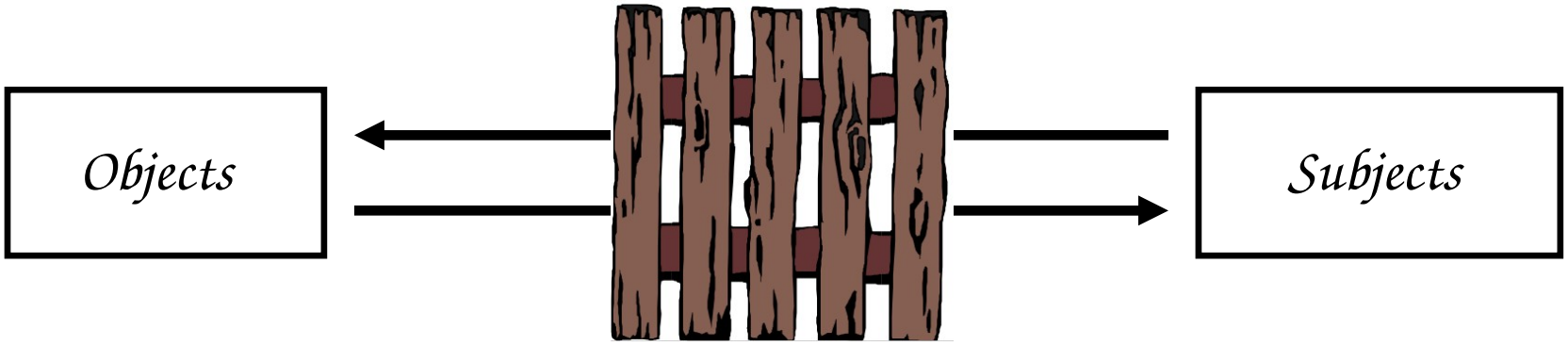
- ***Kernel*** *is the lowest-level part of the OS*
- *Kernel is responsible for*
  - *Synchronization*
  - *Inter-process communication*
  - *Message passing*
  - *Interrupt handling*
- *The* ***security kernel*** *is the part of the kernel that deals with security*
- *Security kernel contained within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



Reference monitor

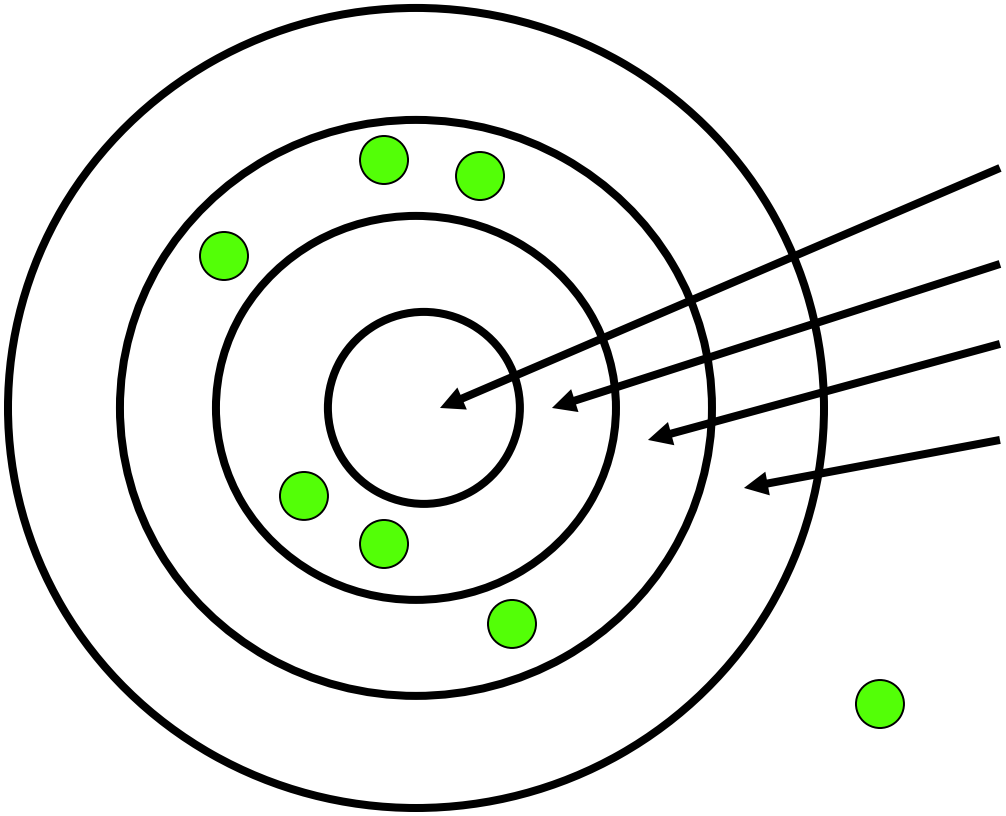
Trusted Computing Base

- **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

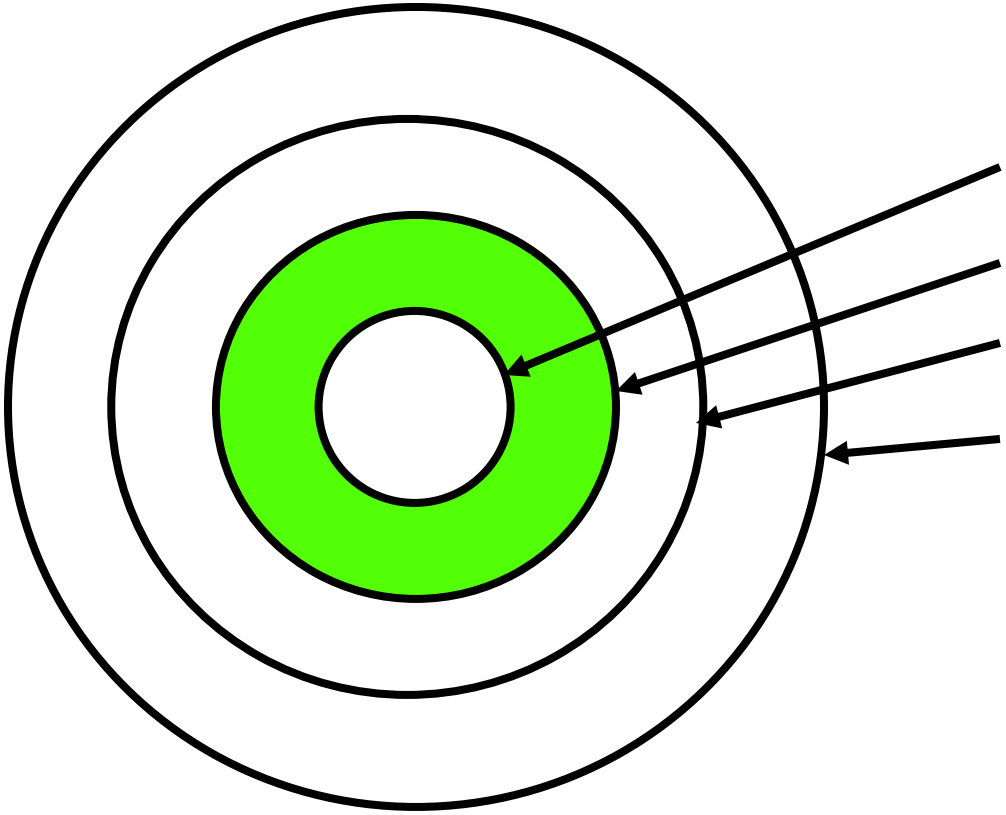


Hardware
OS kernel
Operating system
User space

Security critical activities

Problem: No clear security **layer**

Better TCB Design

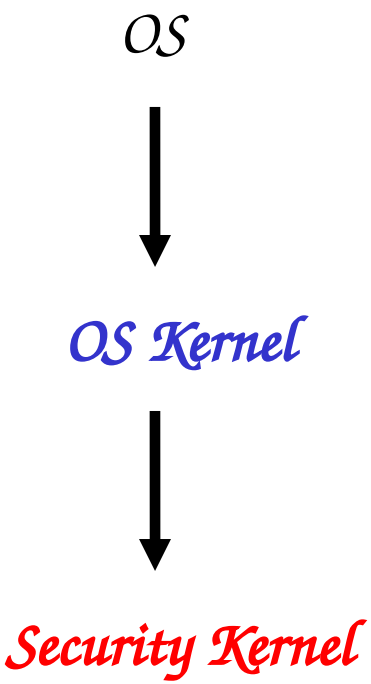


Hardware
Security kernel
Operating system
User space

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



NGSCB

Next Generation Secure Computing Base

- *NGSCB* pronounced "n-scub" (the G is silent)
- Was supposed to be part of Vista OS
  - Vista was once known as *Longhorn…*

- *TCG* (Trusted Computing Group)
  - Led by Intel, TCG makes special hardware
- NGSCB is the part of Windows that will interface with TCG hardware
- TCG/NGSCB formerly TCPA/Palladium
  - Why the name changes?

NGSCB

- The original motivation for TCPA/Palladium was digital rights management (DRM)
- Today, TCG/NGSCB is promoted as general security-enhancing technology
  - DRM just one of many potential applications
- Depending on who you ask, TCG/NGSCB is
  - Trusted computing
  - Treacherous computing

Motivation for TCG/NGSCB

- ***Closed systems:*** Game consoles, etc.
  - Good at protecting secrets (tamper resistant)
  - Good at forcing people to pay for software
  - Limited flexibility
- ***Open systems:*** PCs
  - Incredible flexibility
  - Poor at protecting secrets
  - Very poor at defending their own software
- TCG: closed system security on open platform
- "virtual set-top box inside your PC" Rivest

TCG/NGSCB

- TCG provides tamper-resistant hardware
  - Secure place to store cryptographic key
  - Key secure from a user with admin privileges!
- TCG hardware is in addition to ordinary hardware, not in place of it
- PC has two OSs regular OS and special ***trusted*** OS to deal with TCG hardware
- NGSCB is Microsoft's trusted OS

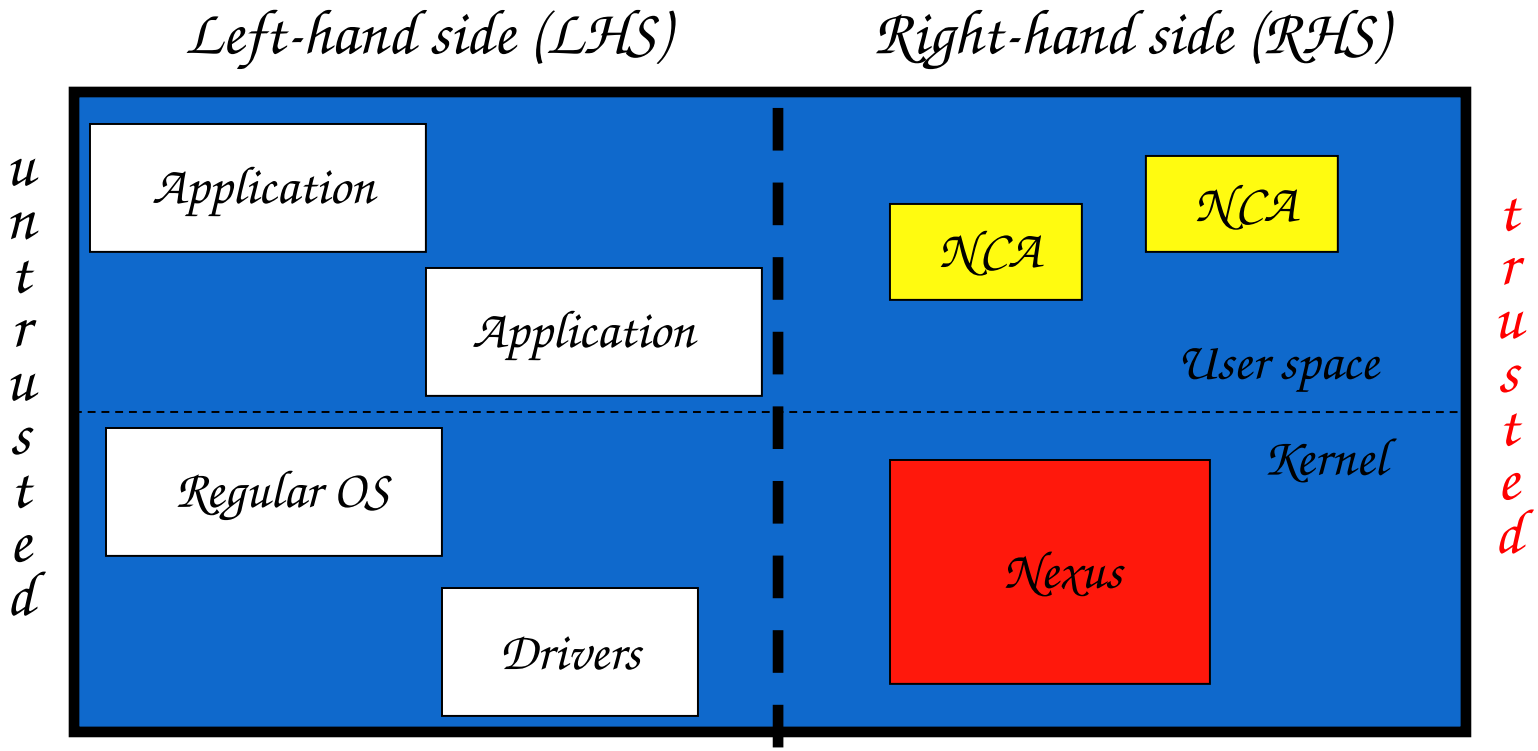
NGSCB Design Goals

- Provide ***high assurance***
  - High confidence that system behaves correctly
  - Correct behavior even if system is under attack
- Provide ***authenticated operation***
  - Authenticate "things" (software, devices, etc.)
- Protection against hardware tampering is concern of TCG, not NGSCB

NGSCB Disclaimer

- Specific details are sketchy
- Based on available info, Microsoft may not have resolved all of the details
  - Maybe un-resolvable?
- What follows: author's best guesses
- This should all become much clearer in the not-too-distant future
  - At least I thought so a couple of years ago…

NGSCB Architecture



- *Nexus* is the Trusted Computing Base in NGSCB
- The *NCA* (Nexus Computing Agents) talk to Nexus and LHS

NGSCB

- NGSCB has 4 "feature groups"
  1. **Strong process isolation**
     - Processes do not interfere with each other
  2. **Sealed storage**
     - Data protected (tamper resistant hardware)
  3. **Secure path**
     - Data to and from I/O protected
  4. **Attestation**
     - "Things" securely authenticated
     - Allows TCB to be extended via NCAs
- All are aimed at malicious code
- 4. also provides (secure) extensibility

NGSCB Process Isolation

- ***Curtained memory***
- *Process isolation and the OS*
  - *Protect trusted OS (Nexus) from untrusted OS*
  - *Isolate trusted OS from untrusted stuff*
- *Process isolation and NCAs*
  - *NCAs isolated from software they do not trust*
- *Trust determined by users, to an extent…*
  - *User **can** disable a trusted NCA*
  - *User **cannot** enable an untrusted NCA*

NGSCB Sealed Storage

- Sealed storage contains **secret** data
  - If **code X** wants access to secret, a hash of X must be verified (integrity check of X)
  - Implemented via symmetric key cryptography
- Confidentiality of secret is protected since only accessed by trusted software
- Integrity of secret is assured since it's in sealed storage

NGSCB Secure Path

- Secure path for input
  - From keyboard to Nexus
  - From mouse to Nexus
  - From any input device to Nexus
- Secure path for output
  - From Nexus to the screen
- Uses crypto (digital signatures)

NGSCB Attestation (1)

- Secure authentication of ***things***
  - Authenticate devices, services, code, etc.
  - Separate from user authentication
- Public key cryptography used
  - Certified key pair required
  - Private key not user-accessible
  - Sign and send result to remote system
- ***TCB extended via attestation of NCAs***
  - This is a major feature!

NGSCB Attestation (2)

- Public key used for attestation
  - However, public key reveals the user identity
  - Using public keys, anonymity would be lost
- Trusted third party (TTP) can be used
  - TTP verifies signature
  - Then TTP vouches for signature
  - Anonymity preserved (except to TTP)
- Support for zero knowledge proofs
  - Verify knowledge of a secret without revealing it
  - Anonymity "preserved unconditionally"

NGSCB Compelling Apps (1)

- Type your Word document in Windows
  - I.e., the untrusted LHS
- Move document to trusted RHS
- Read document carefully
- Digitally sign the document
- Assured that "what you see is what you sign"
  - Practically impossible to get this on your PC

NGSCB Compelling Apps (2)

- Digital Rights Management (DRM)
- Many DRM problems solved by NGSCB
- **Protect secret** sealed storage
  - Impossible without something like NGSCB
- **Scraping data** secure path
  - Cannot prevent without something like NGSCB
- **Positively ID users**
  - Higher assurance with NGSCB

NGSCB According to MS

- All of Windows works on untrusted LHS
- User is in charge of…
  - Which Nexus(es) will run on system
  - Which NCAs will run on system
  - Which NCAs allowed to identify system, etc.
- No external process enables Nexus or NCA
- Nexus can't block, delete, censor data
  - NCA ***does***, but NCAs ***authorized*** by user
- Nexus is open source

NGSCB Critics

- ***Many*** critics we consider two
- Ross Anderson
  - Perhaps the most influential critic
  - Also one of the harshest critics
- Clark Thomborson
  - Lesser-known critic
  - Criticism strikes at heart of NGSCB

Anderson's NGSCB Criticism (1)

- Digital object controlled by its creator, not user of machine where it resides: Why?
  - Creator can specify the NCA
  - If user does not accept NCA, access is denied
  - Aside: This is critical for, say, MLS applications
- If Microsoft Word encrypts all documents with key only available to Microsoft products
  - Then difficult to stop using Microsoft products

Anderson's NGSCB Criticism (2)

- Files from a compromised machine could be blacklisted to, e.g., prevent music piracy
- Suppose everyone at SJSU uses same pirated copy of Microsoft Word
  - If you stop this copy from working on all NGSCB machines, SJSU users will not use NGSCB
  - Instead, make all NGSCB machines refuse to open documents created with this copy of Word…
  - …so SJSU user can't share docs with NGSCB user…

Anderson's NGSCB Criticism (3)

- Going off the deep end…
  - "The Soviet Union tried to register and control all typewriters. NGSCB attempts to register and control all computers."
  - "In 2010 President Clinton may have two red buttons on her desk one that sends missiles to China and another that turns off all of the PCs in China…"

Thomborson's NGSCB Criticism

- NGSCB acts like a ***security guard***
- By passive observation, NGSCB "security guard" can see sensitive info
- Former student worked as security guard at apartment complex
  - By passive observations…
  - …he learned about people who lived there

Thomborson's NGSCB Criticism

- Can NGSCB spy on you?
- According to Microsoft
  - Nexus software is public
  - NCAs can be debugged (for development)
  - NGSCB is strictly "opt in"
- Loophole?
  - Release version of NCA can't be debugged ***and*** debug and release versions differ

NGSCB Bottom Line (1)

- NGCSB: ***trusted OS*** on an open platform
- Without something similar, PC may lose out
  - Particularly in entertainment-related areas
  - Copyright holders will not trust PC
  - Already lost? (iPod, Kindle, iPad, etc., etc.)
- With NGSCB, will users lose some control of their PCs?
- But NGSCB users must choose to "opt in"
  - If user does not opt in, what has been lost?

NGSCB Bottom Line (2)

- NGSCB is a ***trusted system***
- ***Only trusted system can break security***
  - By definition, an untrusted system is not trusted with security critical tasks
  - Also by definition, a trusted system is trusted with security critical tasks
  - If untrusted system is compromised, security is not at risk
  - If a trusted system is compromised (or simply malfunctions), security is at risk

Software Summary

- Software flaws
  - Buffer overflow
  - Race conditions
  - Incomplete mediation
- Malware
  - Viruses, worms, etc.
- Other software-based attacks

Software Summary

- Software Reverse Engineering (SRE)
- Digital Rights Management (DRM)
- Secure software development
  - Penetrate and patch
  - Open vs closed source
  - Testing

Software Summary

- Operating systems and security
  - How does OS enforce security?
- Trusted OS design principles
- Microsoft's NGSCB
  - A trusted OS for DRM

Course Summary

- Crypto
  - Symmetric key, public key, hash functions, cryptanalysis
- Access Control
  - Authentication, authorization
- Protocols
  - Simple auth., SSL, IPSec, Kerberos, GSM
- Software
  - Flaws, malware, SRE, Software development, trusted OS