

CE 815 - Secure Software Systems

Secure Architecture

Mehdi Kharrazi

Department of Computer Engineering
Sharif University of Technology

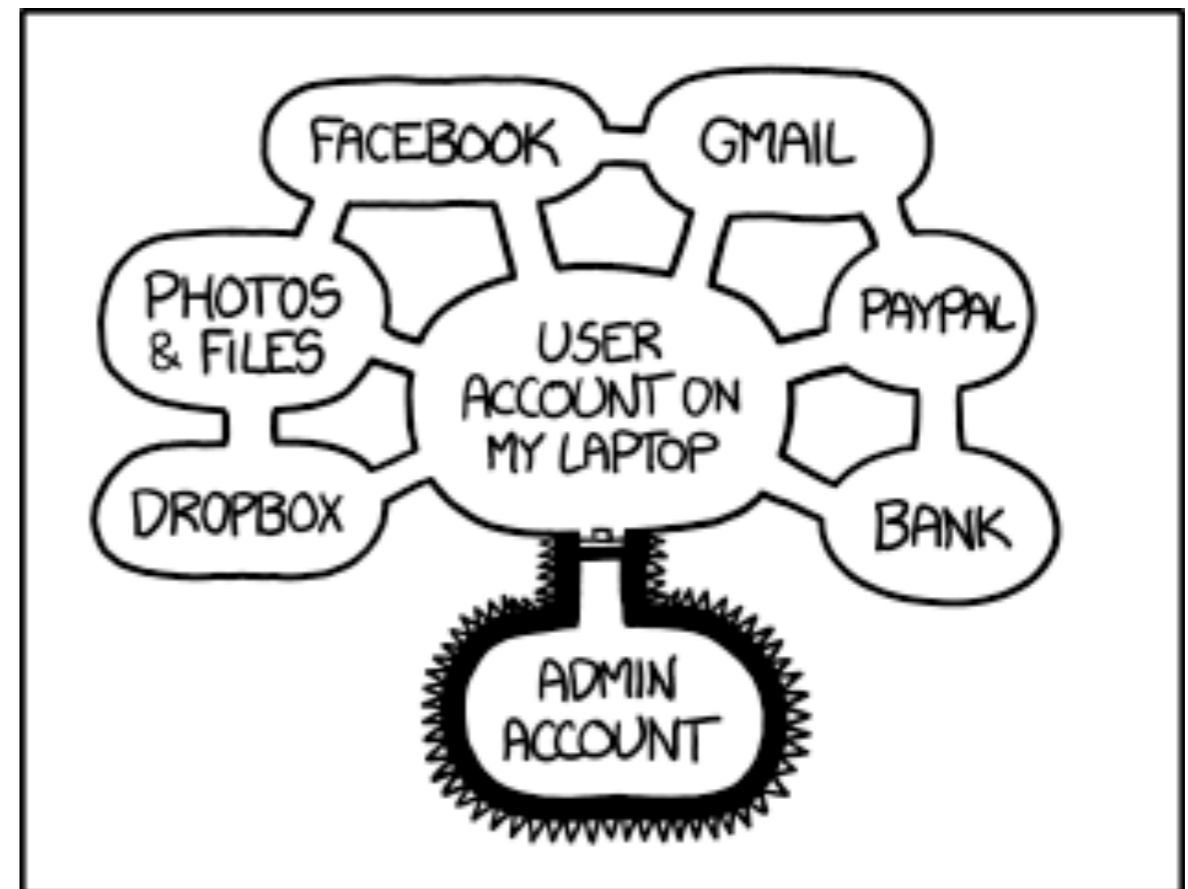


Acknowledgments: Some of the slides are fully or partially obtained from other sources. A reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.



Secure Architecture

- How to come up with a secure architecture?
- What design principals should be followed?
- What are the available mechanisms?
- How do you trust the code getting executed?



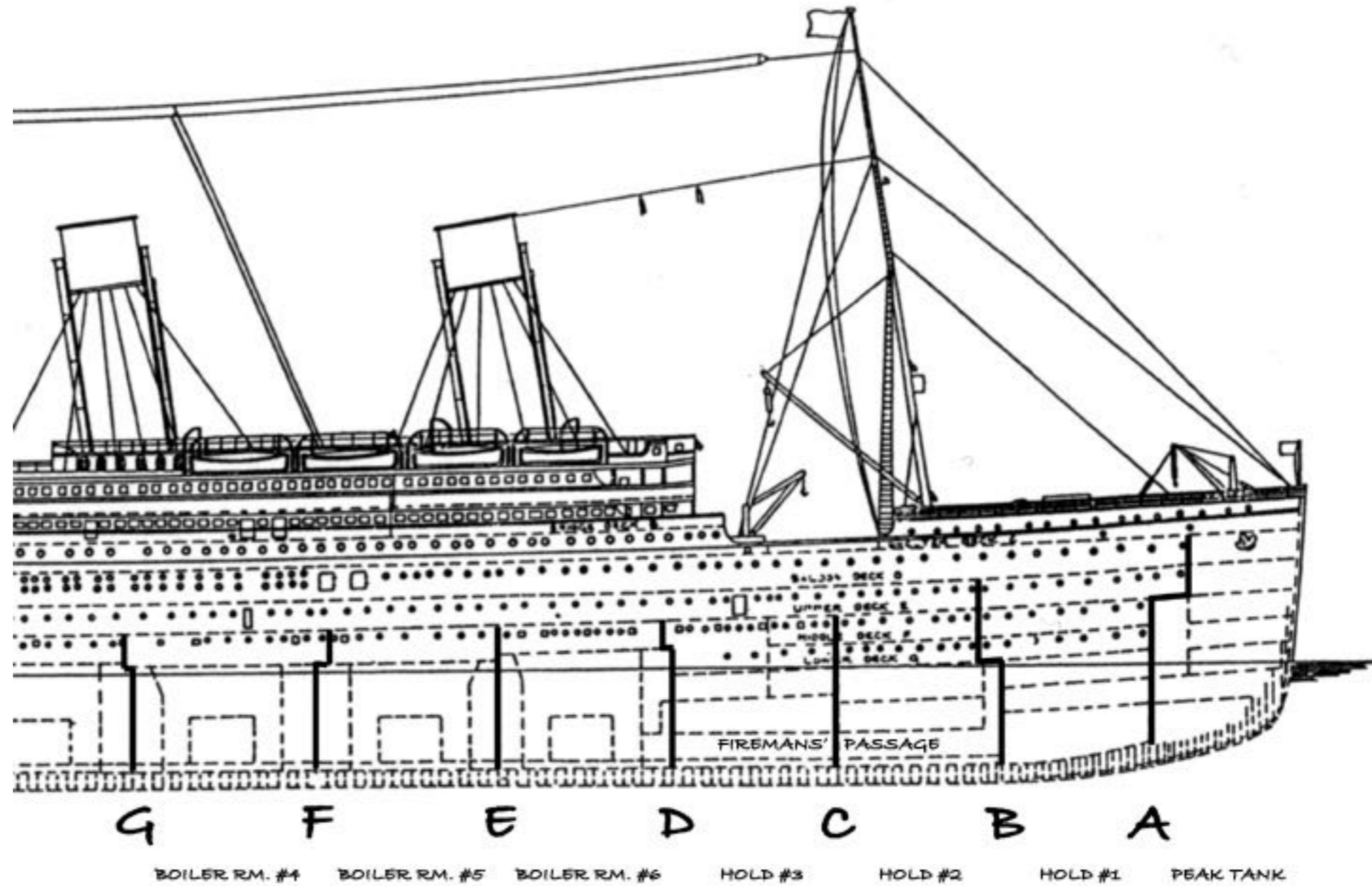
IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS, BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION. xkcd.com



Building a Secure system



okieboat.com



Bulkheads & Compartments in the Bow Section

imsa.edu



Isolation

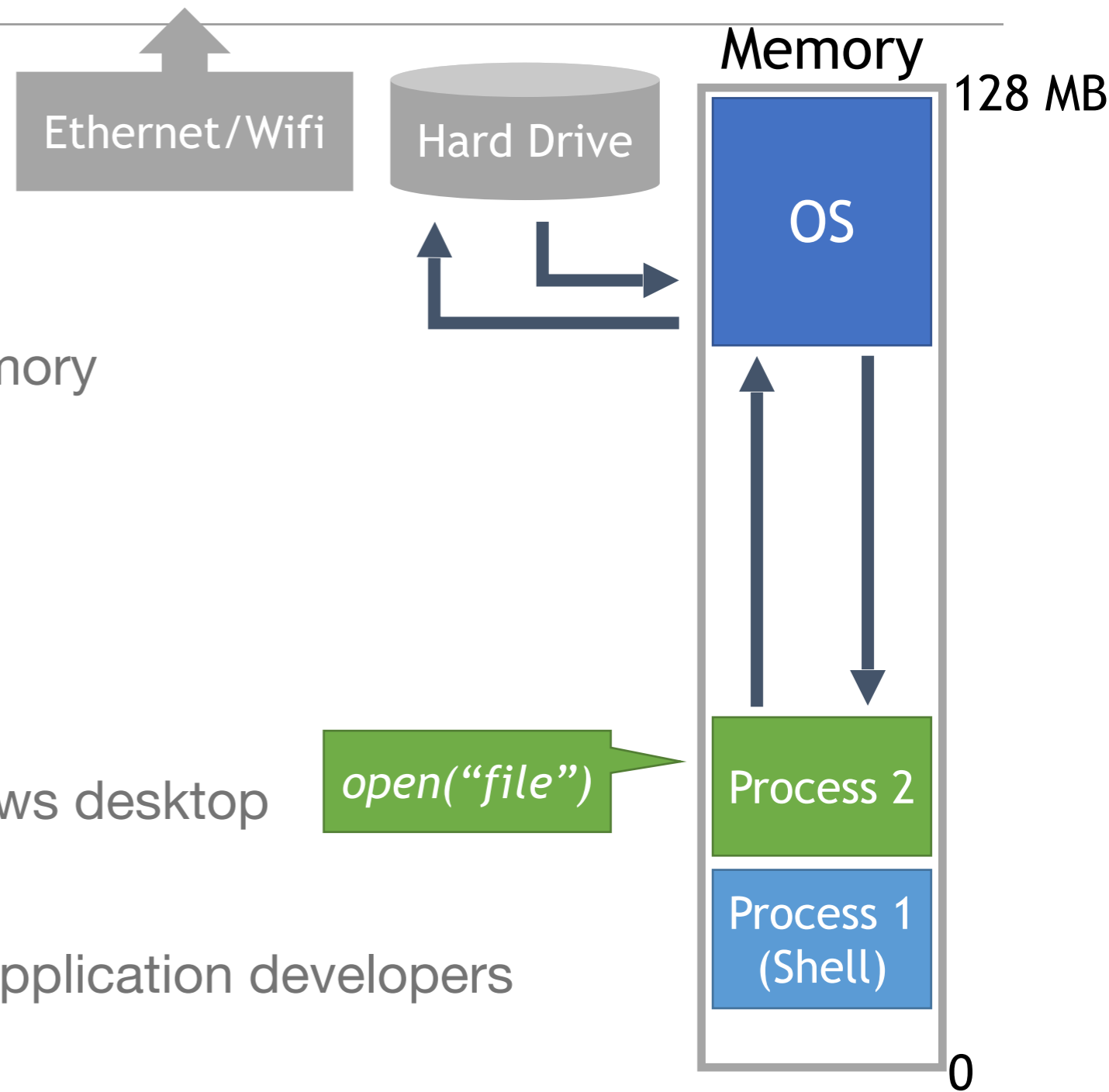






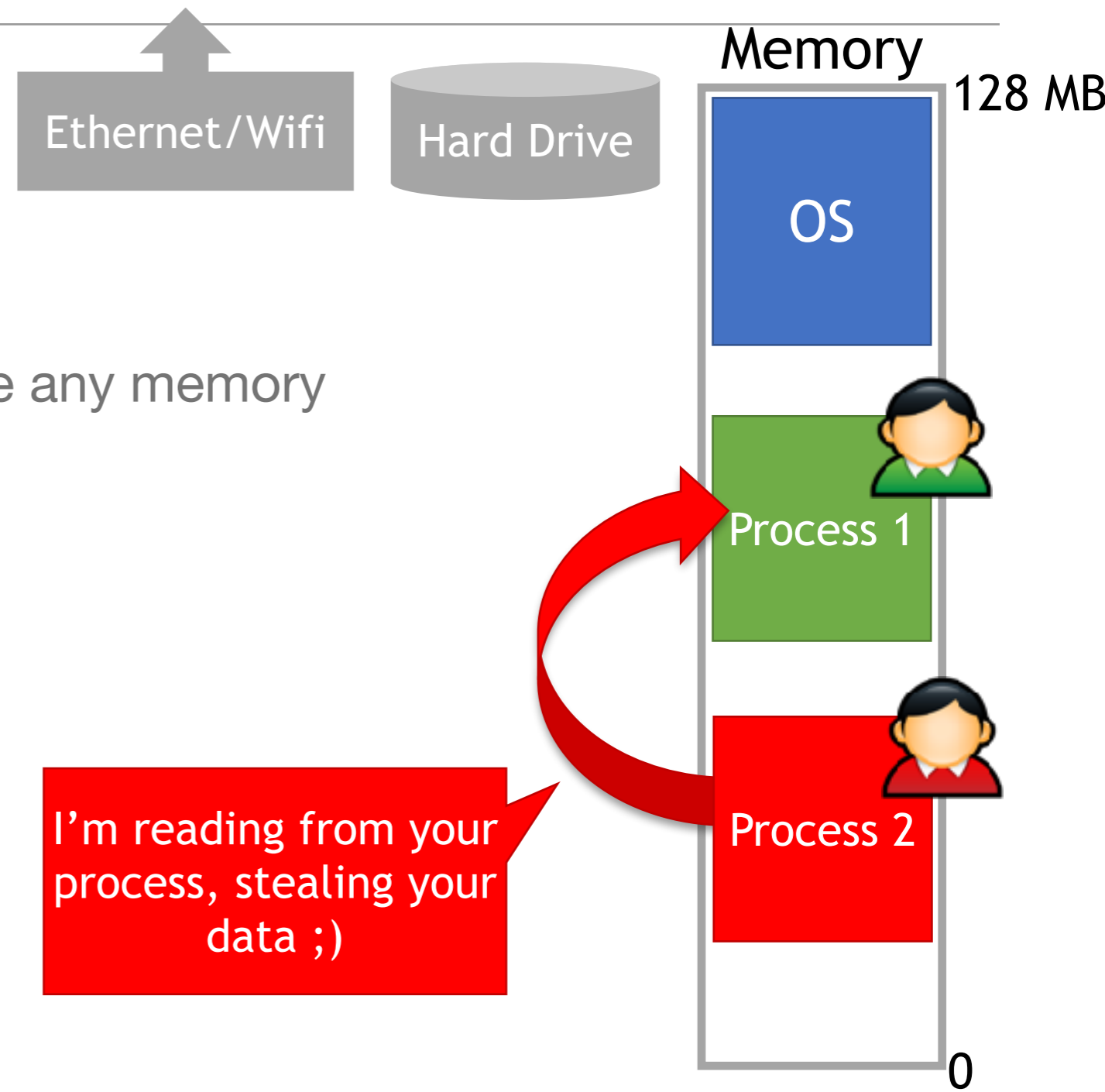
System Model

- On bootup, the Operating System (OS) loads itself into memory
 - DOS or Windows 3.1
 - Typically places itself in high memory
- What is the role of the OS?
 - Allow the user to run processes
 - Often comes with a shell
 - Text shell like bash
 - Graphical shell like the Windows desktop
 - Provides APIs to access devices
 - Offered as a convenience to application developers





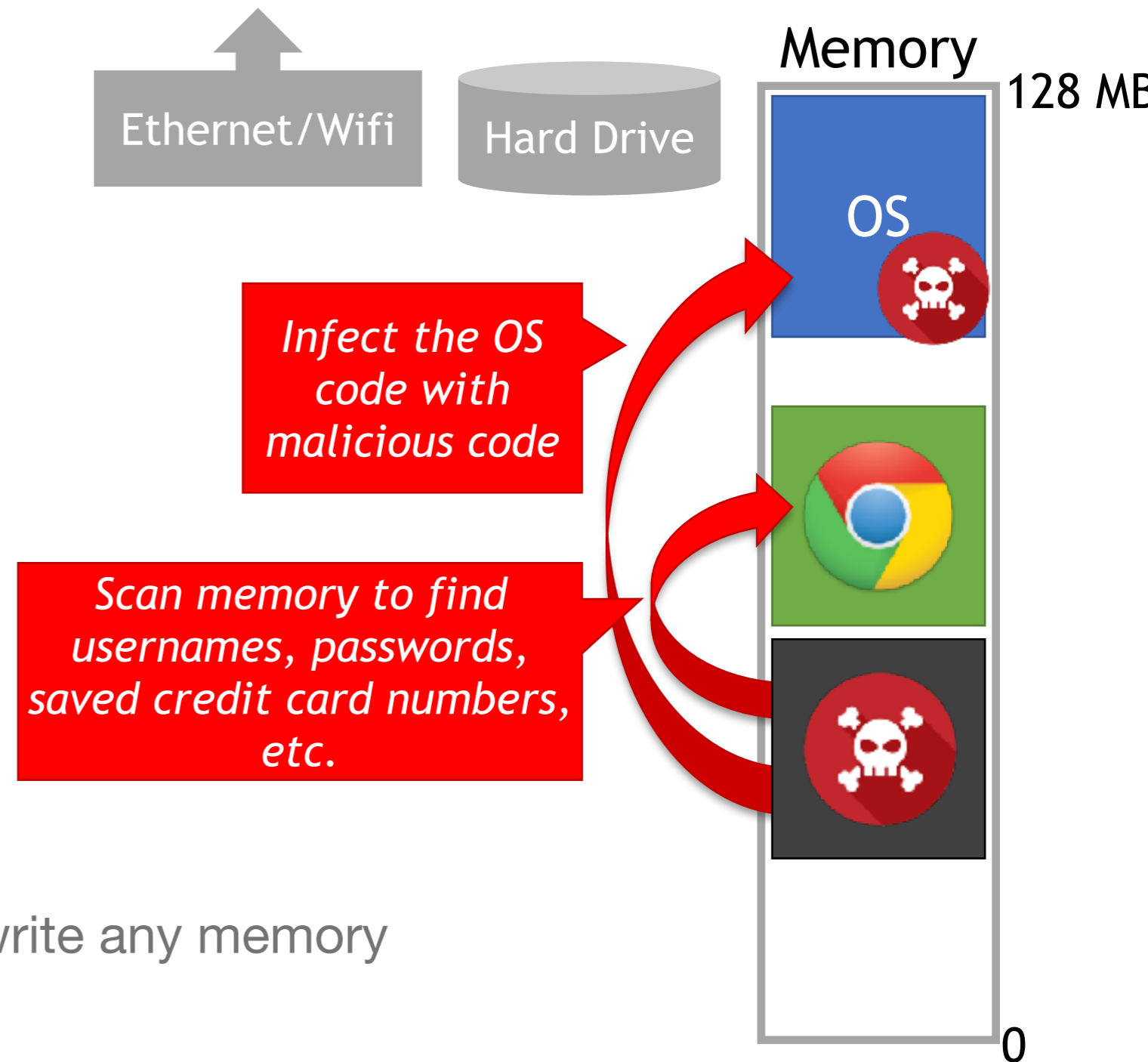
Memory Unsafety



- Problem: any process can read/write any memory



Memory Unsafety



- Problem: any process can read/write any memory



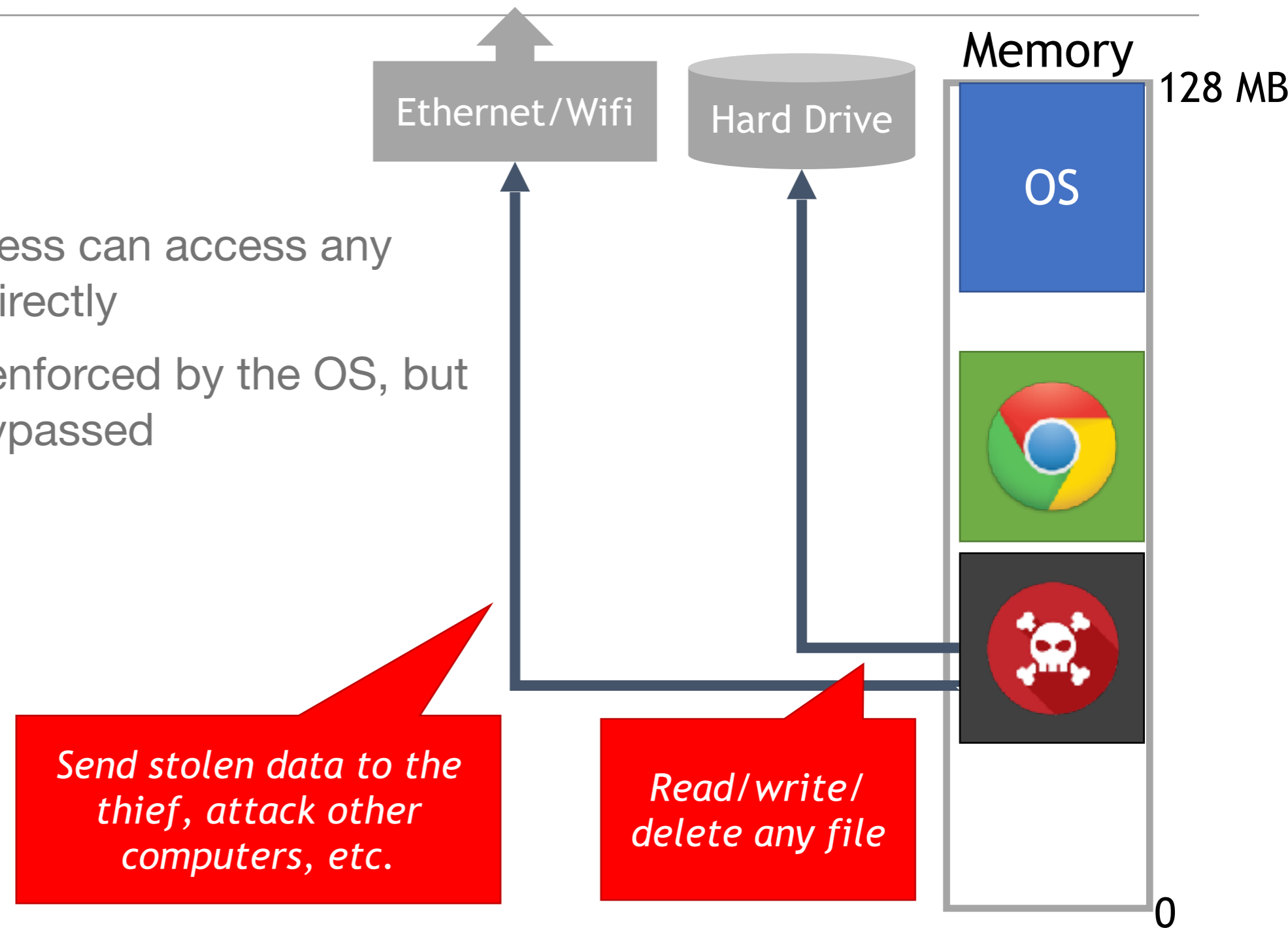
Device Unsafety

- Problem: any process can access any hardware device directly
- Access control is enforced by the OS, but OS APIs can be bypassed



Device Unsafety

- Problem: any process can access any hardware device directly
- Access control is enforced by the OS, but OS APIs can be bypassed





Older system issues

- Old systems did not protect memory or devices
 - Any process could access any memory
 - Any process could access any device
- Problems
 - No way to enforce access controls on users or devices
 - Processes can steal from or destroy each other
 - Processes can modify or destroy the OS
- On old computers, systems security was literally impossible



Hardware support for isolation



Towards Modern Architecture

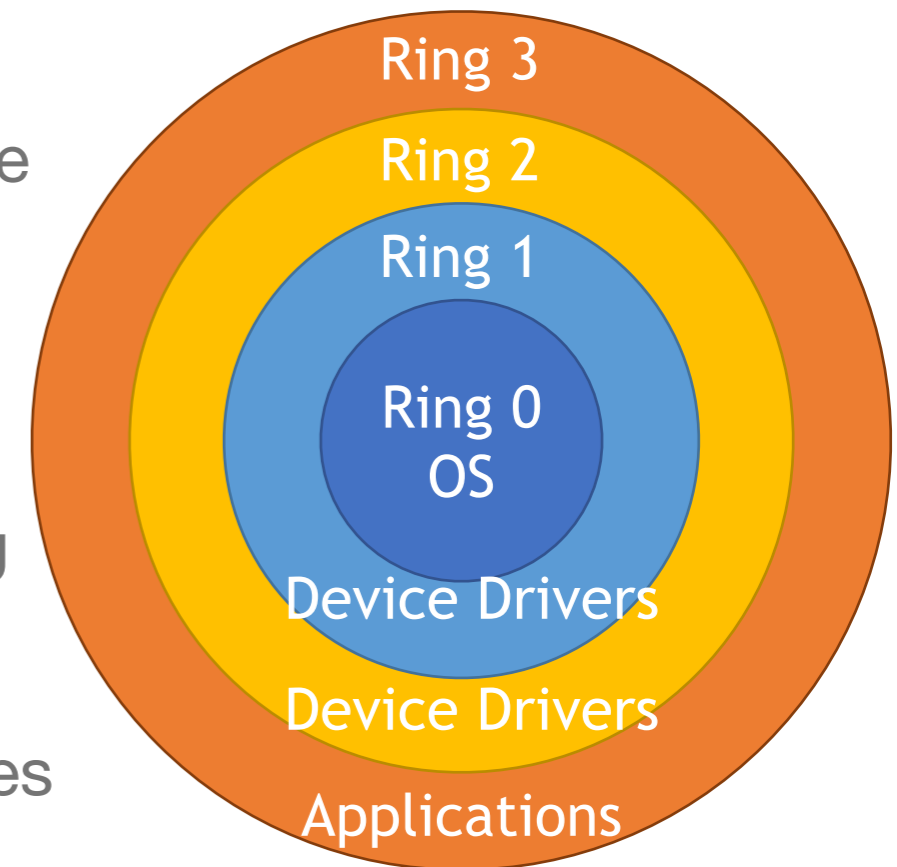
- To achieve systems security, we need process isolation
 - Processes cannot read/write memory arbitrarily
 - Processes cannot access devices directly
- How do we achieve this?
- Hardware support for isolation
 - Protected mode execution (a.k.a. process rings)
 - Virtual memory





Protected Mode

- Most modern CPUs support protected mode
- x86 CPUs support three rings with different privileges
 - Ring 0: Operating System
 - Code in this ring may directly access any device
 - Ring 1, 2: device drivers
 - Code in these rings may directly access some devices
 - May not change the protection level of the CPU
 - Ring 3: userland
 - Code in this ring may not directly access devices
 - All device access must be via OS APIs
 - May not change the protection level of the CPU
- Most OSes only use rings 0 and 3





System Boot Sequence

- On startup, the CPU starts in 16-bit real mode
 - Protected mode is disabled
 - Any process can access any device
- BIOS executes, finds and loads the OS
- OS switches CPU to 32-bit protected mode
 - OS code is now running in Ring 0
 - OS decides what Ring to place other processes in
- Shell gets executed, user may run programs
 - User processes are placed in Ring 3



Restriction on Privileged Instructions

- What CPU instructions are restricted in protected mode?
 - Any instruction that modifies the CR0 register
 - Controls whether protected mode is enabled
 - Any instruction that modifies the CR3 register
 - Controls the virtual memory configuration
 - hlt – Halts the CPU
 - sti/cli – enable and disable interrupts
 - in/out – directly access hardware devices
- If a Ring 3 process tries any of these things, it immediately crashes



Changing Modes

- Applications often need to access the OS APIs
 - Writing files
 - Displaying things on the screen
 - Receiving data from the network
 - etc...
- But the OS is Ring 0, and processes are Ring 3
- How do processes get access to the OS?
 - Invoke OS APIs with special assembly instructions
 - Interrupt: `int 0x80`
 - System call: `sysenter` or `syscall`
 - `int/sysenter/syscall` cause a mode transfer from Ring 3 to Ring 0



Mode Transfer

- Application executes trap (int) instruction
 - EIP, CS, and EFLAGS get pushed onto the stack
 - Mode switches from ring 3 to ring 0
- Save the state of the current process
 - Push EAX, EBX, ..., etc. onto the stack
- Locate and execute the correct syscall handler
- Restore the state of process
 - Pop EAX, EBX, ... etc.
- Place the return value in EAX
- Use iret to return to the process
 - Switches back to the original mode (typically 3)





Virtual Memory Implementation

- Each process has its own virtual memory space
 - Each process has a page table that maps its virtual space into physical space
 - CPU translates virtual address to physical addresses on-the-fly
- OS creates the page table for each process
 - Installing page tables in the CPU is a protected, Ring 0 instruction
 - Processes cannot modify their page tables
- What happens if a process tries to read/write memory outside its page table?
 - Segmentation Fault or Page Fault
 - Process crashes
 - In other words, no way to escape virtual memory



Security Policy in General



Security Policy

- Defines a security perimeter

Because you can't
secure everything



www.forcecontracting.co.uk



Security Policy

- Defines a security perimeter
- Standards codify the what should be done
- Guidelines explain how it will be done



How do you create a policy?

- Option #1 Risk Assessment:
 - Identify assets and their value
 - Identify the threats
 - Calculate the risks
 - Conduct a Cost-Benefit Analysis
- Option #2: Adopt “Best Practices.”



Threat Modeling

- Threat modeling is the process of systematically identifying the threats faced by a system
- Identify things of value that you want to protect
- Enumerate the attack surfaces
- Hypothesize attackers and map them to
 - Things of value they want from (1)
 - Their ability to target vulnerable surfaces from (2)
- Survey mitigations
- Balance costs versus risks





Identify Things of Value

- Saved passwords
- Monetizable credentials (webmail, social networks)
- Access to bank accounts, paypal, venmo, credit cards, or other financial services
- Pics, messages, address book, browsing/search history (for blackmail)
- Sensitive business documents
- Access to sensors (camera, mic, GPS) or network traffic (for surveillance)
- The device itself
 - Steal it and sell it
 - Use the CPU and network for other criminal activity





Enumerate Attack Surfaces

- Intercept and compromise the handset in transit
- Backdoor the OS
- Steal the device and use it
- Direct connection via USB
- Close proximity radios (Bluetooth, NFC)
- Social engineering, e.g. trick the user into installing malicious app(s)
- Exploit vulnerabilities in the OS or apps (e.g. email clients, web browsers)
- Passive eavesdropping on the network
- Active network attacks (e.g. man-in-the-middle, SMS of death)



Techniques For Drafting Policies

- Assign a specific “owner” to everything that is to be protected.
- Be positive
- Be realistic in your expectations
- Concentrate on education and prevention



Remember, Risk Cannot Be Eliminated

- You can purchase a UPS...
 - But the power failure may outlast the batteries
 - But the UPS may fail
 - But the cleaning crew may unplug it
 - But the UPS may crash due to a software error.



Spaf's first principle of security administration:

“If you have responsibility for security, but have no authority to set rules or punish violators, your own role in the organization is to take the blame when something big goes wrong.”



Security Principles



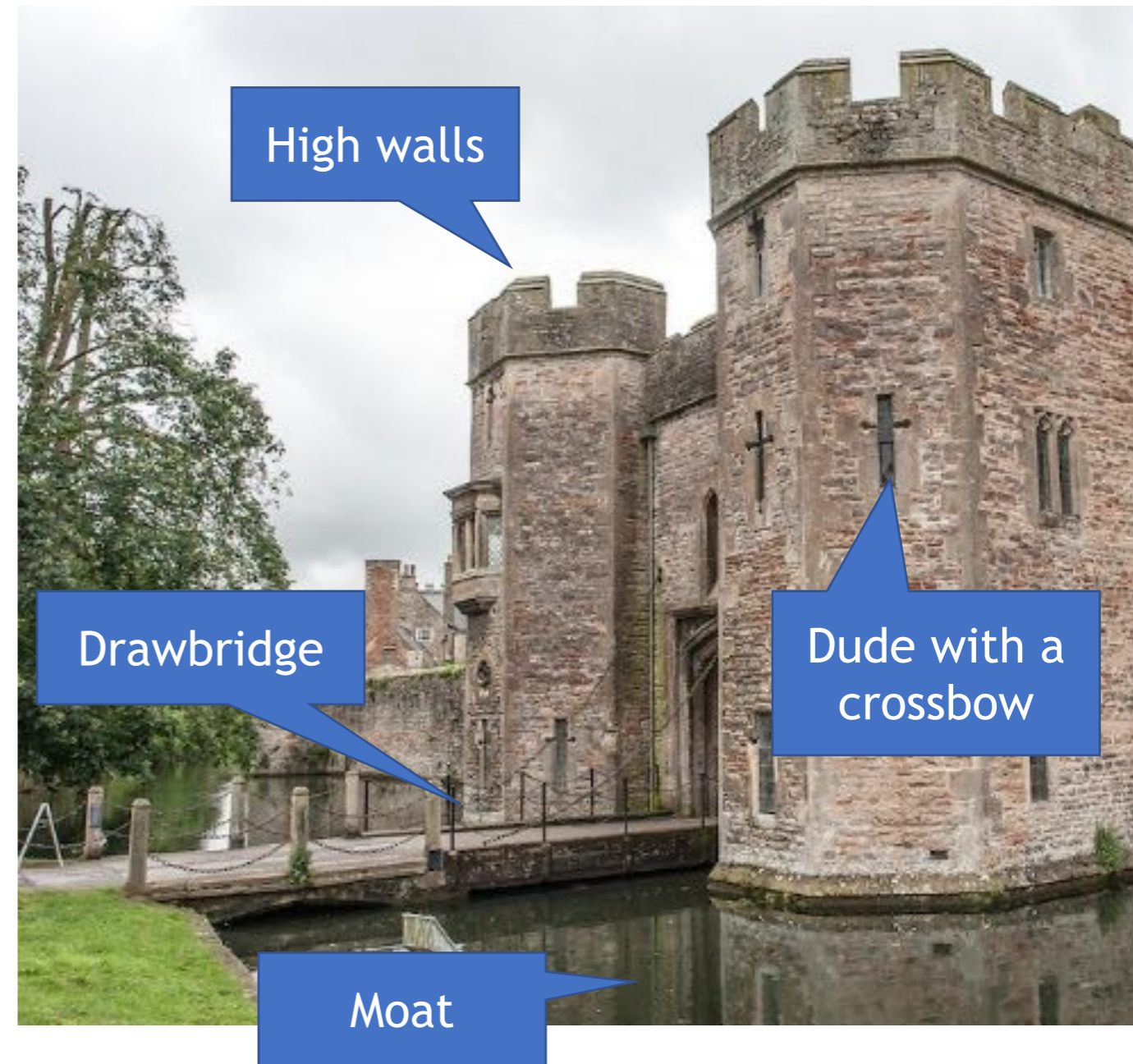
Security Principles

- Designing secure systems (and breaking them) remains an art
- Security principles help bridge the gap between art and science
 - Developed by Saltzer and Schroeder
 - “The Protection of Information in Computer Systems”, 1975



Defense in Depth

- Don't depend on a single protection mechanism, since they are apt to fail
- Even very simple or formally verified defenses fail
- Layering defenses increases the difficulty for attackers
- Defenses should be complementary!





Defence in Depth Example

- Problem: Bank.
 - How to secure the money?
- Solution: Defence in depth.
 - Guards inside bank.
 - Closed-circuit cameras monitor activity.
 - Tellers do not have access to vault.
 - Vault has multiple defences:
 - Time-release.
 - Walls and lock complexity.
 - Multiple compartments.



Example

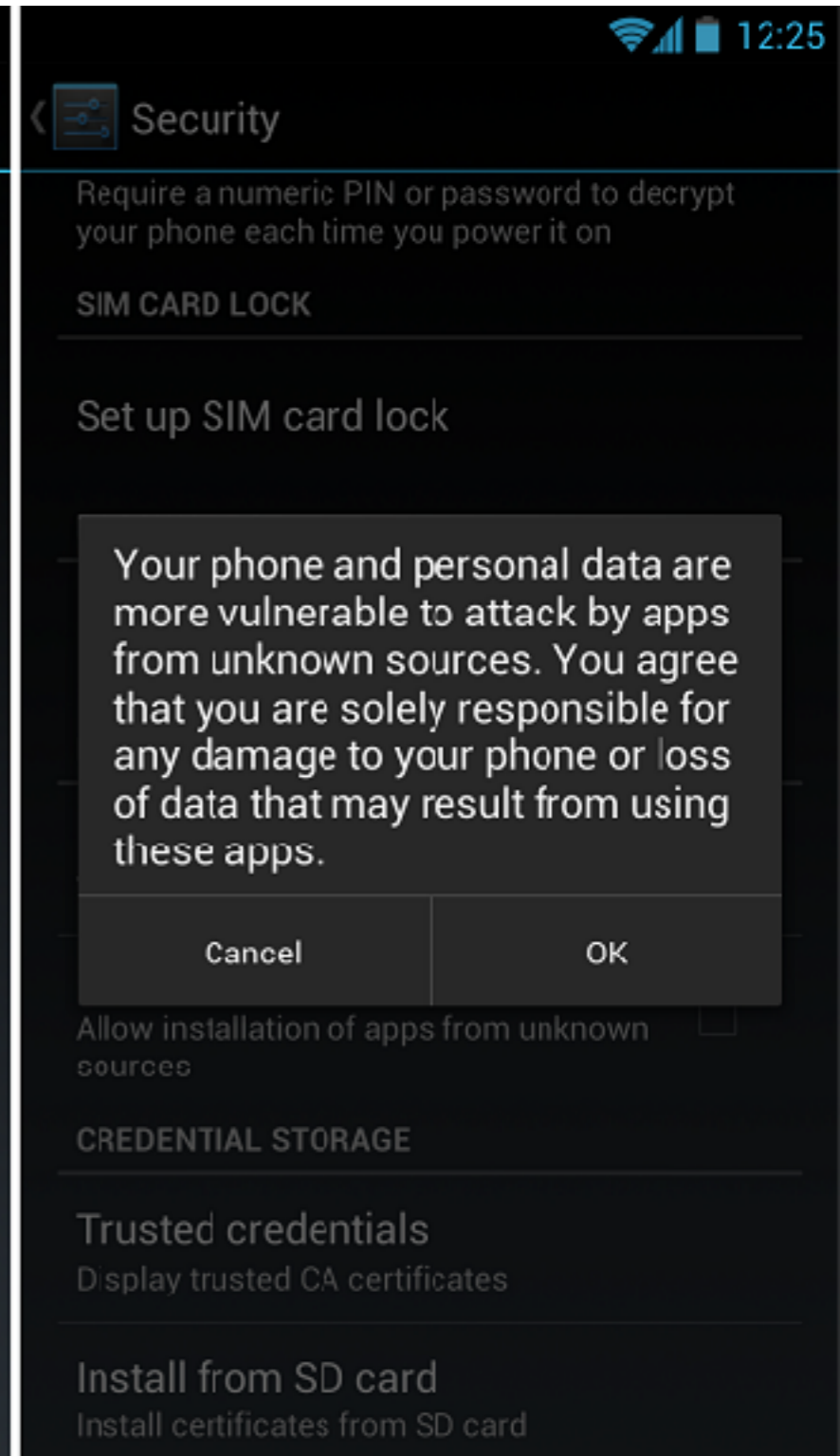
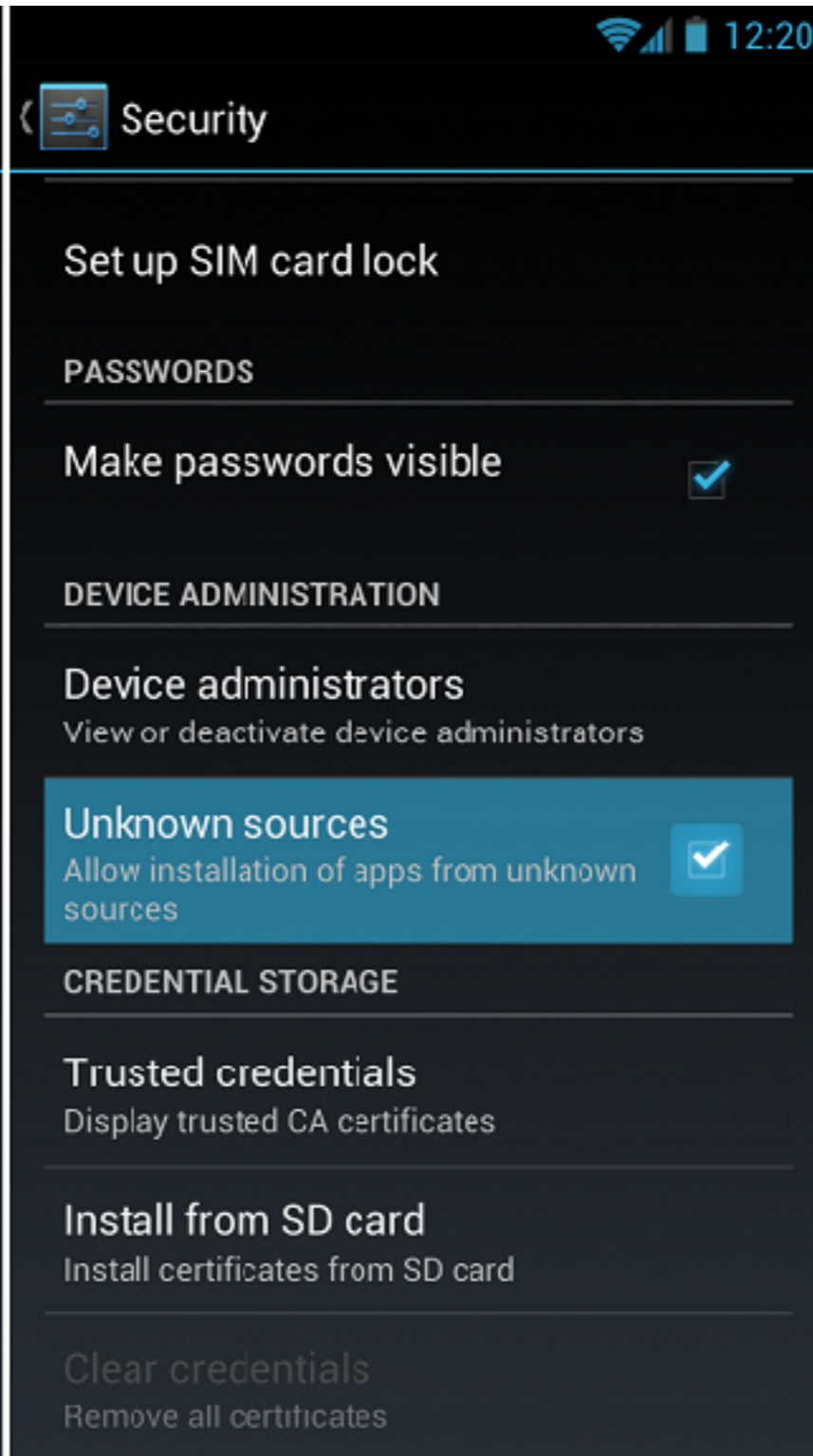
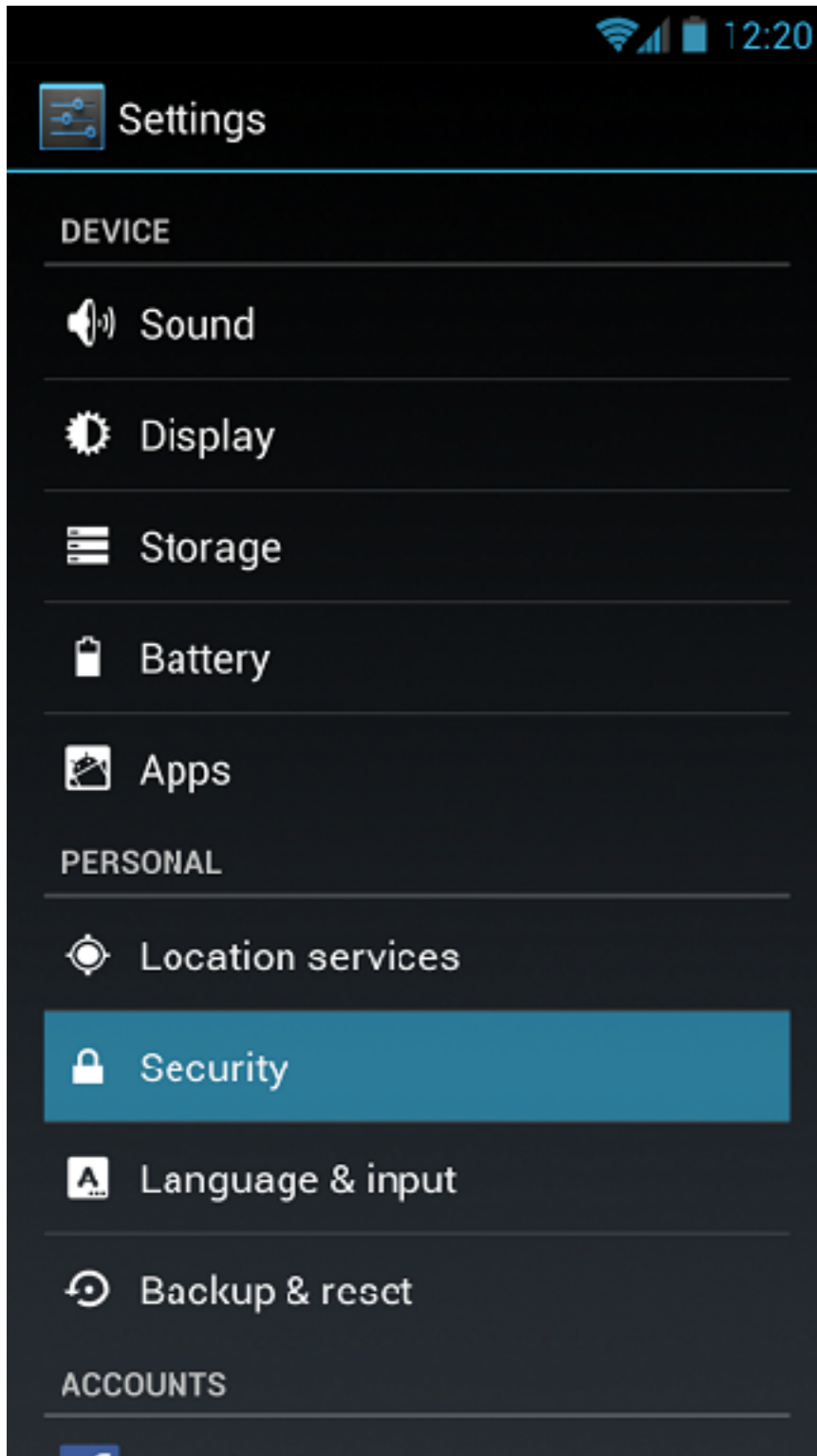
- Built-in security features of Windows 10
 - Secure boot: cryptographically verified bootup process
 - Bitlocker full-drive encryption
 - Kernel protections, e.g. Address Space Layout Randomization (ASLR)
 - Cryptographic signing for device drivers
 - User authentication
 - User Account Control: permission check for privileged operations
 - Anti-virus and anti-malware
 - Firewall
 - Automated patching
 - System logs



Fail-safe Defaults

- The absence of explicit permission is equivalent to no permission
- Systems should be secure "out-of-the-box"
 - Most users stick with defaults
 - Users should "opt-in" to less-secure configurations
- Examples. By default...
 - New user accounts do not have admin or root privileges
 - New apps cannot access sensitive devices
 - Passwords must be >8 characters long
 - Etc.







Separation of Privilege

- Privilege, or authority, should only be distributed to subjects that require it
- Some components of a system should be less privileged than others
 - Not every subject needs the ability to do everything
 - Not every subject is deserving of full trust
- Examples:
 - Two signatures required for a check
 - Two authorized personnel required to fire a nuclear missile



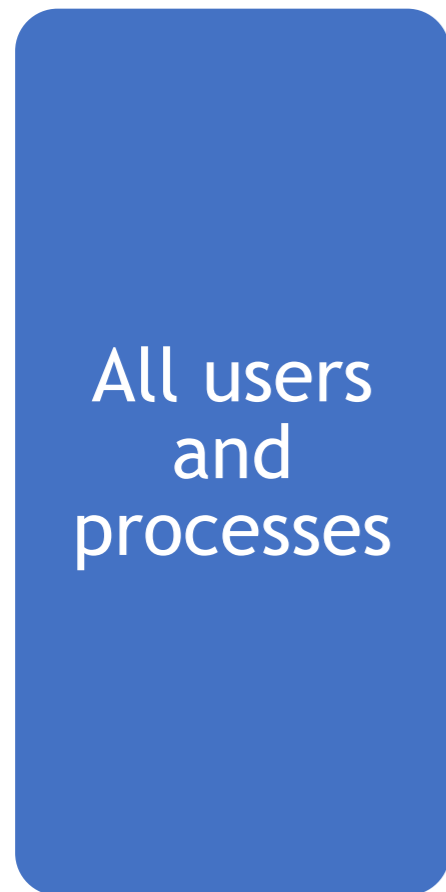
Least Privilege

- Subjects should possess only that authority that is required to operate successfully
- Closely related to separation of privilege
- Not only should privilege be separated, but subjects should have the least amount necessary to perform a task

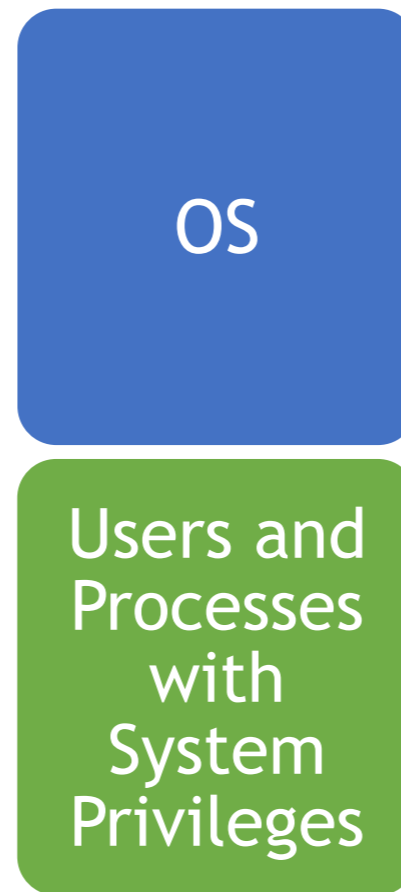


Privilege Over Time

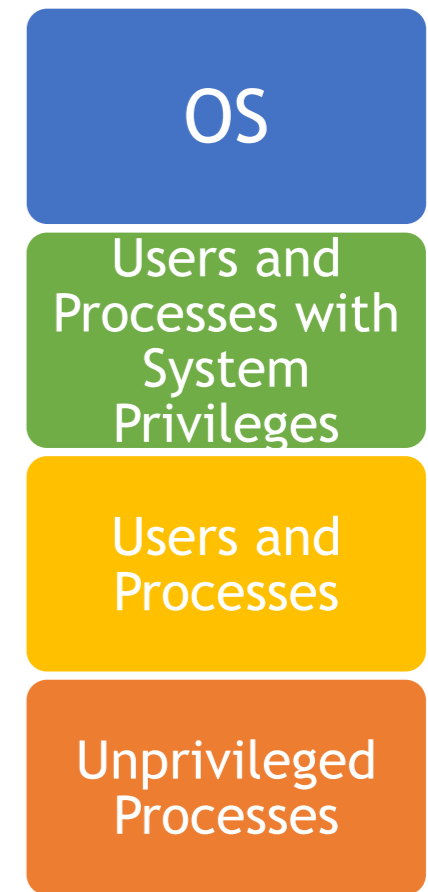
DOS, Windows 3.1



Win 95 and 98



Win NT, XP, 7, 8, 10
Linux, BSD, OSX





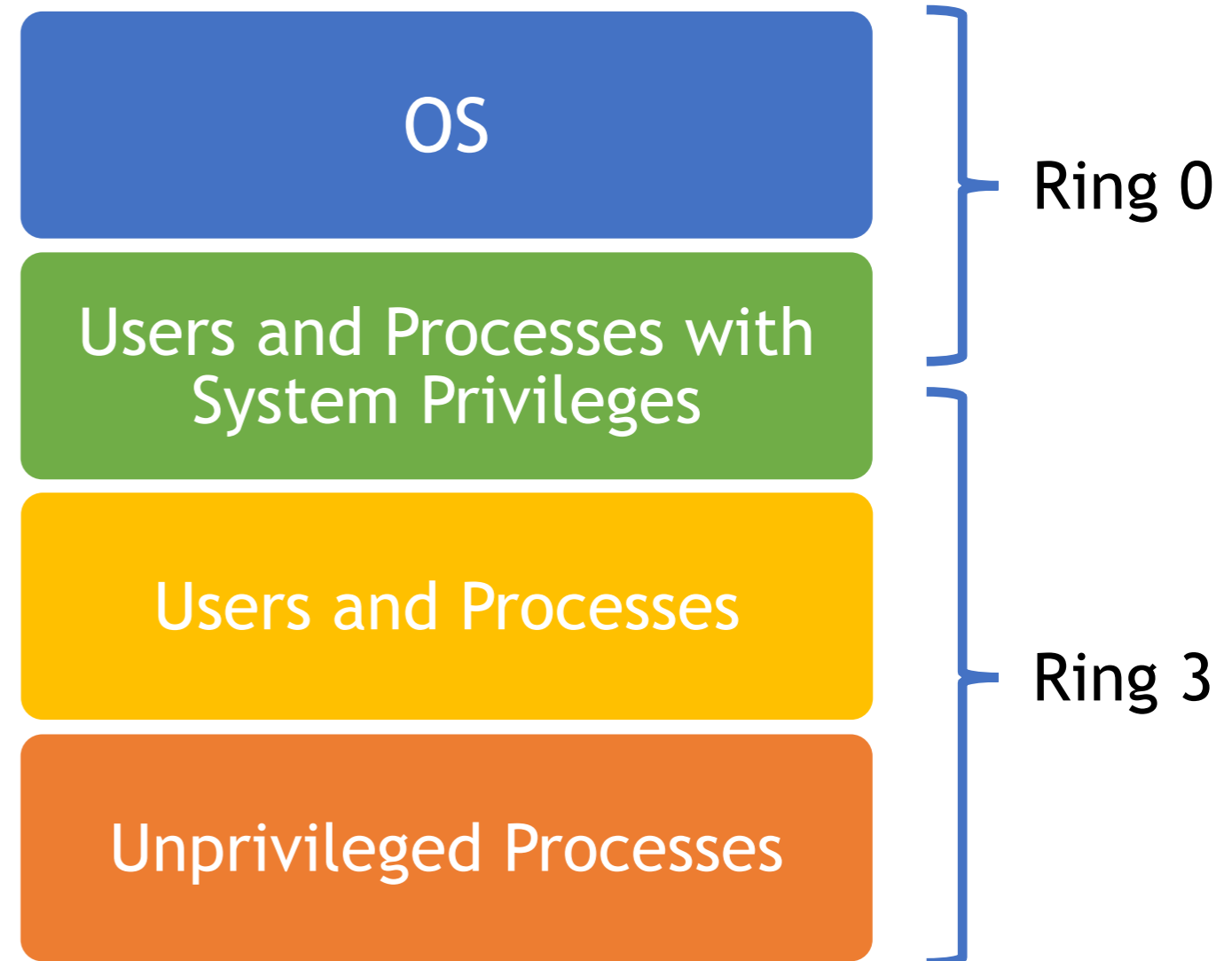
Privilege Hierarchy

Device drivers, kernel modules, etc.

sudo, “administrator” accounts, OS services

Everything that is isolated and subject to access control

chroot jails, containers, low-integrity processes





Example: Chrome Multiprocess Architecture

Chrome is split across many processes

“Core” process has user-level privileges

- May read/write files
- May access the network
- May render to screen

Each tab, extension, and plugin has its own process

- Parse HTML, CSS, JS
- Execute JS
- **Large attack surface!**
- Thus, have no privileges
- All I/O requests are sent to the core process

Task	Memory	CPU	Network	Process ID
Browser	110,180K	1	0	3988
App: Inbox	369,052K	1	0	804
App: Google Calendar	83,548K	0	0	700
Tab: Google Keep	91,052K	0	0	7092
Tab: CS 4740/6740	1,104K	0	0	6596
Tab: Multi-process Architecture - The Chromium Projects	1,484K	0	0	4684
Tab: Chromium Blog: Multi-process Architecture	10,128K	0	0	4036
Tab: Process isolation - Wikipedia, the free encyclopedia	1,080K	0	0	3060
Tab: Multi-Processes in Browsers: Chrome, Internet Explorer, Firefox and WebKit - Softpedia	8,058K	0	0	2992
Extension: uMatrix	27,696K	0	0	2324
Extension: Google Now	7,510K	0	0	2008
Extension: Churnalism	3,576K	0	0	2884
Extension: Mailvelope	1,058K	0	0	3028
Extension: uBlock	36,708K	0	0	6056
Extension: Bookmark Manager	10,320K	0	0	3956
GPU Process	35,348K	0	N/A	5412
Plug-in: Shockwave Flash	12,192K	0	0	5584
Tab: JavaScript APIs - Google Chrome	36,136K	0	0	6360



Compromise Recording

- Concede that attacks will occur, but record the fact
- Auditing approach to security
 - Detection and recovery
- "Tamper-evident" vs. "Tamper-proof"





Logging

- Log everything
- Better yet, use remote logging
 - Ensures that attacker with local access cannot erase logs
- Logs are useless if they aren't monitored
- Advanced approaches
 - Intrusion Detection Systems (IDS)
 - Anomaly detection
 - Machine learning-based approaches

```
root@kali:~# cat /var/log/apache2/access.log
121.54.29.89 [25/Nov/2011:13:16:28 +0100] "GET /wp/wp-content/themes/lightbox/images/submit_btn.png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=8109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.27 Safari/532.0 FVE-IGR"
121.54.29.89 - - [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/themes/lightbox/images/sidebar_bg.png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=8109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.27 Safari/532.0 FVE-IGR"
121.54.29.89 - - [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/themes/lightbox/images/content_bottom.png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=8109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.27 Safari/532.0 FVE-IGR"
121.54.29.89 [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/plugins/quick-contact-forms/css/images/bg_input.png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=8109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.27 Safari/532.0 FVE-IGR"
121.54.29.89 [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/plugins/quick-contact-forms/css/images/bg_input.png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=8109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.27 Safari/532.0 FVE-IGR"
121.54.29.89 - - [25/Nov/2011:13:16:30 +0100] "GET /wp/wp-content/plugins/quick-contact-forms/css/images/bg_input.png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=8109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; AppleWebKit/532.0 (KHTML, like Gecko) Chrome/3.0.195.27 Safari/532.0 FVE-IGR"
66.249.72.162 - - [25/Nov/2011:13:16:47 +0100] "GET /wp/?p=1224 HTTP/1.1" 200 10917 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
90.191.29.80 [25/Nov/2011:13:17:05 +0100] "GET /wp/?feed=rss2 HTTP/1.1" 304 163 "-" "RSSOWL/2.1.2.201108131738 (Windows; U; en)"
66.228.54.164 [25/Nov/2011:13:19:25 +0100] "GET /wp/?feed=rss2 HTTP/1.1" 304 163 "-" "Python httplib2/5Rev5"
90.191.29.80 [25/Nov/2011:13:22:00 +0100] "GET /wp/?feed=rss2 HTTP/1.1" 304 163 "-" "RSSOWL/2.1.2.201108131738 (Windows; U; en)"
66.249.66.9 - - [25/Nov/2011:13:24:37 +0100] "GET /wp/?tag=windows-8 HTTP/1.1" 200 13187 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
66.249.66.9 - - [25/Nov/2011:13:25:20 +0100] "GET /wp/?page_id=1199 HTTP/1.1" 200 12661 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
90.191.29.80 - - [25/Nov/2011:13:27:05 +0100] "GET /wp/?feed=rss2 HTTP/1.1" 304 163 "-" "RSSOWL/2.1.2.201108131738 (Windows; U; en)"
root@kali:~#
```



Work Factor

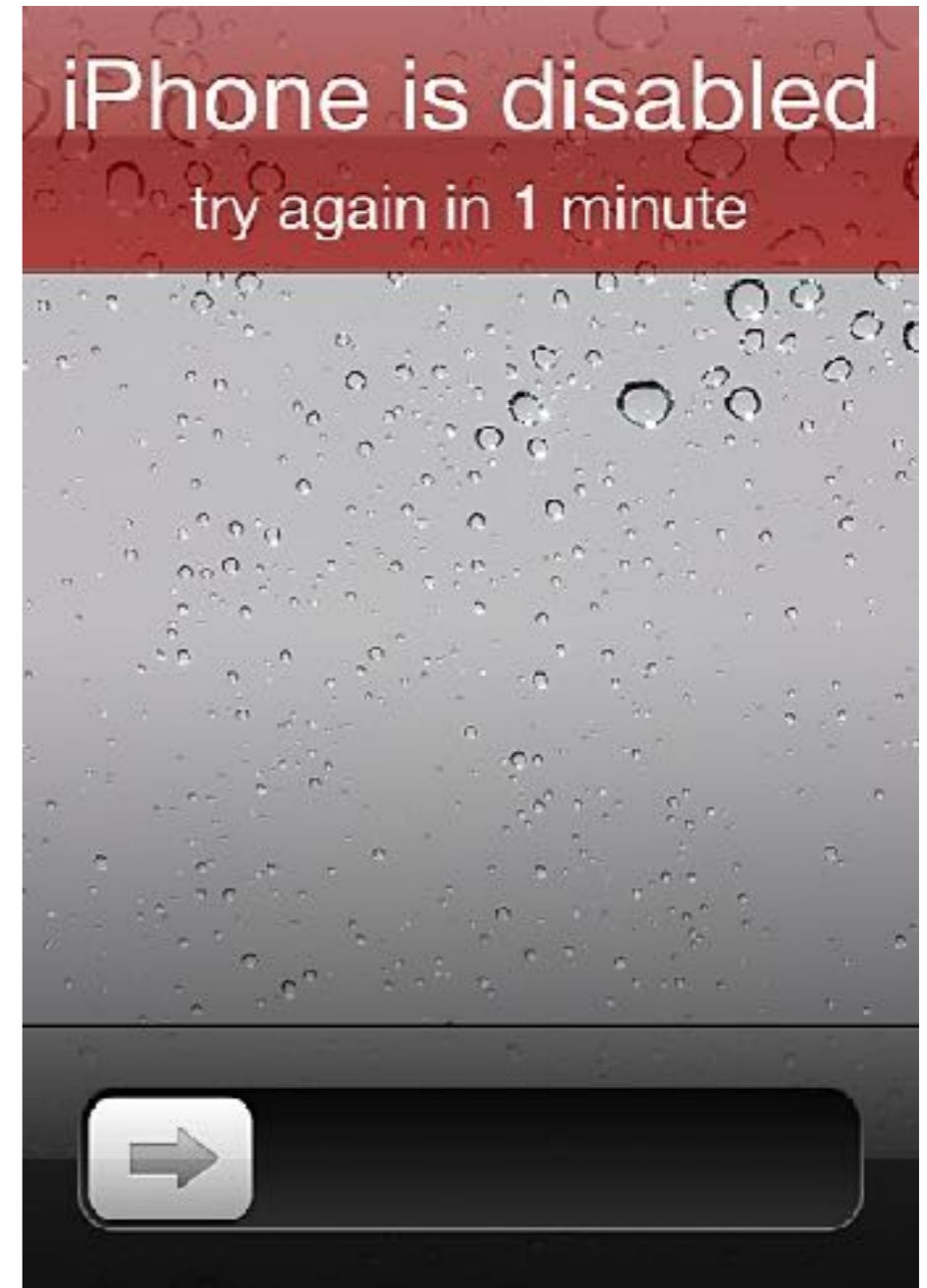
- Increase the difficulty of mounting attacks
- Sometimes utilizes non-determinism
 - e.g. increasing entropy used in ASLR
- Sometimes utilizes time
 - Increase the lengths of keys
 - Wait times after failed password attempts





Authentication Rate Limiting

- Short delay after each failed authentication attempt
 - Delays may increase as the consecutive failed attempts increase
- Does not prevent password cracking attempts, but slows them down





Open Design

- Kerckhoff's Principle: A cryptosystem should be secure even if everything about the system, except the key, is public knowledge
- Generalization: A system should be secure even if the adversary knows everything about its design
 - Design does not include runtime parameters like secret keys
- Contrast with “security through obscurity”



Open Design Example:

- Problem: MPAA wants control over DVDs.
 - Region coding, unskippable commercials.
- Solution: CSS (Content Scrambling System)
 - CSS algorithm kept secret.
 - DVD Players need player key to decrypt disk key on DVD to decrypt movie for playing.
 - Encryption uses 40-bit keys.
 - People w/o keys can copy but not play DVDs.
- What happened next?
 - CSS algorithm reverse engineered.
 - Weakness in algorithm allows disk key to be recovered in an attack of complexity 2^{25} , which takes only a few seconds.

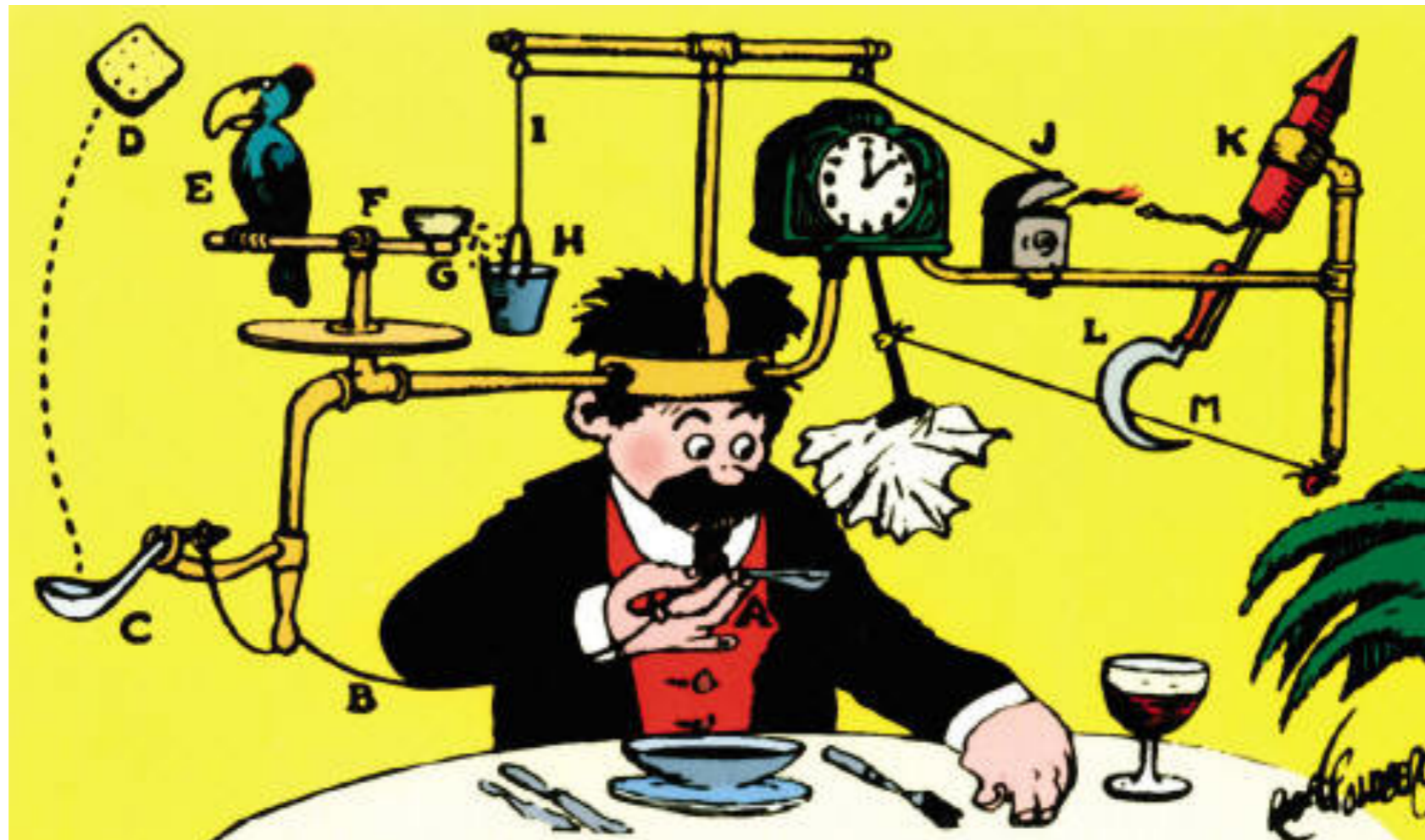


Closed Source

- Security through obscurity.
- Assumes code in binary can't be read
 - what about disassemblers?
 - what about decompilers?
 - what about debuggers?
 - what about strings, lsof, truss, /proc?
- Reverse engineering.

Economy of Mechanism

Would you depend on a defense system designed like this?





Economy of Mechanism

- Simplicity of design implies a smaller attack surface
- Correctness of protection mechanisms is critical
 - "Who watches the watcher?"
 - We need to be able to trust our security mechanisms
 - (Or, at least quantify their efficacy)
- Essentially the KISS principle
 - Keep it simple, stupid



Example

- Existing operating systems are monolithic
 - Kernel contains all critical functionality
 - Process and memory management, file systems, network stack, etc...
- Micro-kernel OS
 - Kernel only contains critical functionality
 - Direct access to hardware resources
 - Process and memory management
 - Small attack surface
 - All other functionality runs in separate processes
 - File systems, network stack, device drivers
- Examples
 - 1) GNU Hurd 2) seL4 – formally verified!



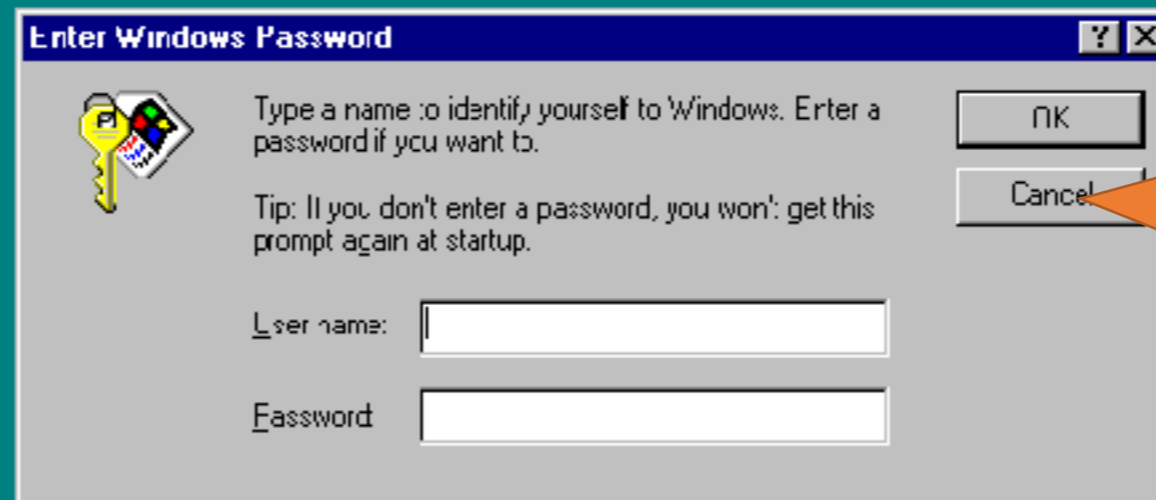
Complete Mediation



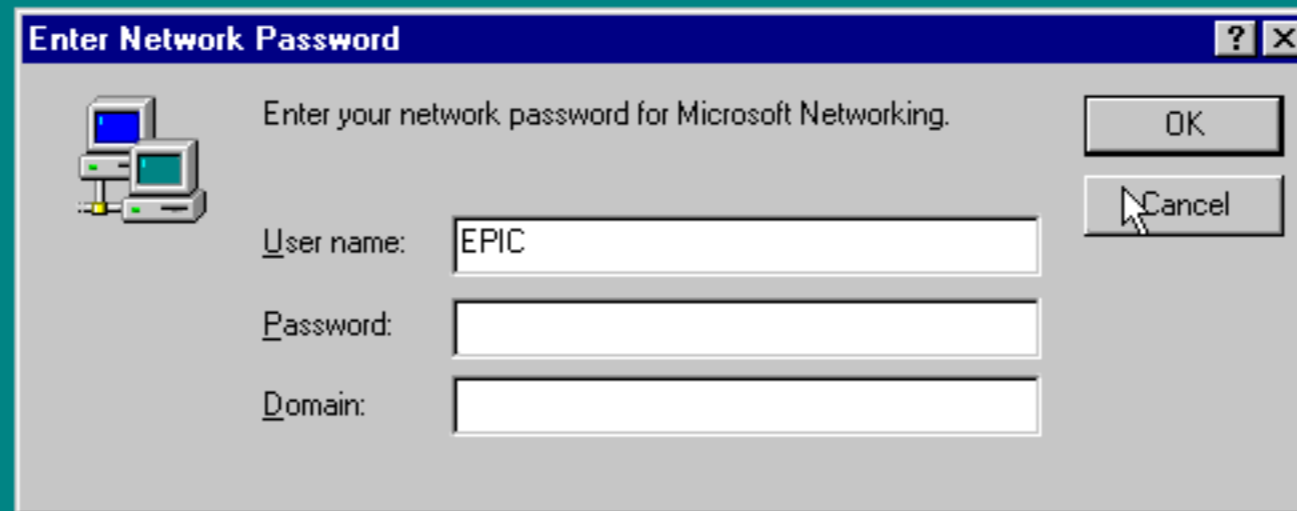


Complete Mediation

- Every access to every object must be checked for authorization
- Incomplete mediation implies that a path exists to bypass a security mechanism
- In other words, isolation is incomplete



By default, user could click Cancel to bypass the password check :(



Forgotten your password ?
No problem



Acknowledgments/References

- [Wilson'19] CS 2550 - Foundations of Cybersecurity, Christo Wilson, Northeastern University, Spring 2019
- [Garfinkel'04] CSCI E-170: Computer Security, Usability & Privacy, Simson L. Garfinkel, MIT, 2004
- [Walden'12] CSC 666 -- Secure Software Engineering, James Walden, Northern Kentucky University, Fall 2012
- [Boneh'15] CS 155, Computer Security, Dan Boneh, Stanford University, 2015
- [Steflik'13] CS-328 Internet and Mobile Programming, Dick Steflik, Binghamton University, Fall 2013
- [Toshev'16] Security architecture of the Java platform, Martin Toshev, Voxxed Days Luxembourg, 2016