2550 Intro to cybersecurity

L19: systems

abhi shelat

Thanks Christo & Steve Myers for slides!

OPERATION - SE ATTACKS

- Systems security: FAILURES of IMPLEMENTATION
FAILURES of DESIGN.

ABSTRACTION - crypto

Threat Model

Principles

Intro to System Architecture

Hardware Support for Isolation

Examples

Threat modeling

① Identify assets to protect.

② Enumerating the attack surface.

③ Define the adversary (power) (goals)

④ Survey & choose mitigations.

⑤ Balancing cost versus risks

Threat modeling

Threat modeling is the process of systematically identifying the threats faced by a system

Threat modeling

Threat modeling is the process of systematically identifying the threats faced by a system

1. Identify assets to protect

Threat modeling

Threat modeling is the process of systematically identifying the threats faced by a system

1. Identify assets to protect
2. Enumerate the attack surfaces

Threat modeling

Threat modeling is the process of systematically identifying the threats faced by a system

1. Identify assets to protect
2. Enumerate the attack surfaces
3. Define adversary’s power and goals
   - Adversary’s goal: assets they want from (1)
   - Power: ability to target vulnerable surfaces from (2)

Threat modeling

Threat modeling is the process of systematically identifying the threats faced by a system

1. Identify assets to protect
2. Enumerate the attack surfaces
3. Define adversary's power and goals
   - Adversary's goal: assets they want from (1)
   - Power: ability to target vulnerable surfaces from (2)
4. Survey mitigations

Threat modeling

Threat modeling is the process of systematically identifying the threats faced by a system

1. Identify assets to protect
2. Enumerate the attack surfaces
3. Define adversary's power and goals
   - Adversary's goal: assets they want from (1)
   - Power: ability to target vulnerable surfaces from (2)
4. Survey mitigations
5. Balance costs versus risks

Identify Assets of Value

- passwords important assets
- credentials in general (SSN, email address, birthday, account names on social media)
- contacts, addresses.
- pictures, private medical data
- credit card info,
- 2FA tokens (physical)
- tax docs

→ webcam & microphone & sensors like gyroscope.

→ Private Information (location data, fitbit data,)

Identify Assets of Value

Saved passwords

Identify Assets of Value

Saved passwords

Monetizable credentials (webmail, social networks)

Identify Assets of Value

- Saved passwords
- Monetizable credentials (webmail, social networks)
- Access to bank accounts, paypal, venmo, credit cards, or other financial services

Identify Assets 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)

Identify Assets 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

Identify Assets 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)

Identify Assets 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

② Enumerate Attack Surfaces

generic
- Device IO ports (USB, power, microphone), WIFI, Bluetooth
- Laptops in general (easily stolen)
- web service (network port on which your service runs)
- Network itself. (Ethernet)
→ Operating system (backdoor??)
- Human (social engineering attacks)

⟹ (context-specific attack surfaces)

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Passive eavesdropping on the network

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Passive eavesdropping on the network

Active network attacks (e.g. man-in-the-middle, SMS of death)

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Passive eavesdropping on the network

Active network attacks (e.g. man-in-the-middle, SMS of death)

Backdoor access to the OS

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Passive eavesdropping on the network

Active network attacks (e.g. man-in-the-middle, SMS of death)

Backdoor access to the OS

Exploit vulnerabilities in the apps (e.g. email clients, web browsers)

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Passive eavesdropping on the network

Active network attacks (e.g. man-in-the-middle, SMS of death)

Backdoor access to the OS

Exploit vulnerabilities in the apps (e.g. email clients, web browsers)

Social engineering, e.g. trick the user into installing malicious app(s)

Enumerate Attack Surfaces

Intercept and compromise the handset in transit

Steal the device and use it

Direct connection via USB

Close proximity radios (Bluetooth, NFC)

Passive eavesdropping on the network

Active network attacks (e.g. man-in-the-middle, SMS of death)

Backdoor access to the OS

Exploit vulnerabilities in the apps (e.g. email clients, web browsers)

Social engineering, e.g. trick the user into installing malicious app(s)

Cybercrime

③ Adversary Define goals & power.

Activity.

- High-level goal: $$$ profit $$$

→ Goal: running an arbitrary process on your computer

- Ransomware
- Botnets · attacker creating their own "cloud service"
- Spyware · browser history
- Adware → $$
- mining attacks

- Power: social attacks, USB, zero-day vulnerability
  "click on a link" ←
  "run a program"

Cybercrime

High-level goal: $$$ profit $$$

Immediate goal: running a process on a victim's computer

- Ransomware
- Botnet
- Spyware
- Adware

Cybercrime

High-level goal: $$$ profit $$$

Immediate goal: running a process on a victim's computer

- Ransomware
- Botnet
- Spyware
- Adware

How to do this?

- Infected storage media (e.g. USB keys)
- Malicious attachments or downloads
- Exploits targeting the OS or common apps
- Guess or crack passwords for remote desktop, etc.

Mitigations & their costs (tools for security)

Authentication

- Physical and remote access is restricted

- Access control
  - → DAC
  - → MAC
- Firewalls, Intrusion detection systems
- Malware- antivirus scanners
- Pwd managers
- Secure/Remote Logging

Mitigations & their costs

Authentication

- Physical and remote access is restricted

Mitigations & their costs

Authentication

- Physical and remote access is restricted

Mitigations & their costs

Authentication

- Physical and remote access is restricted

Mitigations & their costs

Authentication

- Physical and remote access is restricted

Access control

- Processes cannot read/write any file
- Users may not read/write each other's files arbitrarily
- Modifying the OS and installing software requires elevated privileges

Firewall

- Unsolicited communications from the internet are blocked
- Only authorized processes may send/receive messages from the internet

Anti-virus

- All files are scanned to identify and quarantine known malicious code

Logging

- All changes to the system are recorded
- Sensitive applications may also log their activity in the secure system log

Question: how do you build these mitigations?

In other words, how do you build secure systems?

How do you reduce their costs?

Threat Model

Principles

Intro to System Architecture

Hardware Support for Isolation

Examples

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

Security Principles/Heuristics

Principles

Defense-in-depth

Open Design → Kerchoff's

Least Privilege

Separation of Privilege

Heuristics

Compromise Recording/Logging

Work Factor

Secure Defaults

Simplicity

Complete Mediation

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!

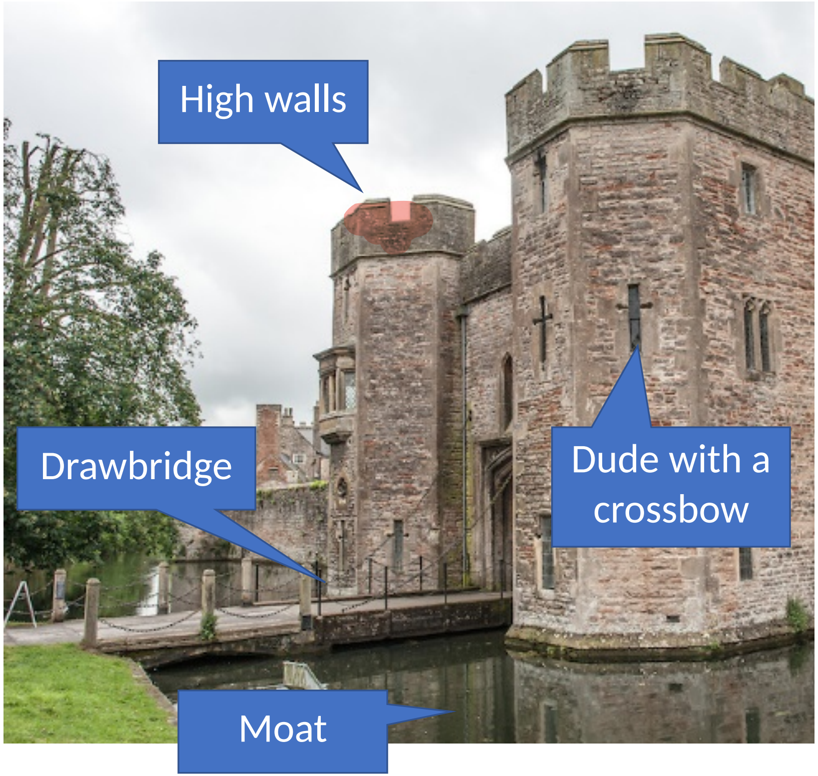
Defense in Depth

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!



Example

Built-in security features of Modern OS

- Secure boot: cryptographically verified bootup process
- full-drive encryption
- Kernel protections, e.g. Address Space Layout Randomization (ASLR) NX DEP
- Cryptographic signing for device drivers
- User authentication
- User Account Control: permission check for privileged operations
- Firewall
- Automated patching
- System logs

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”

Security by Default

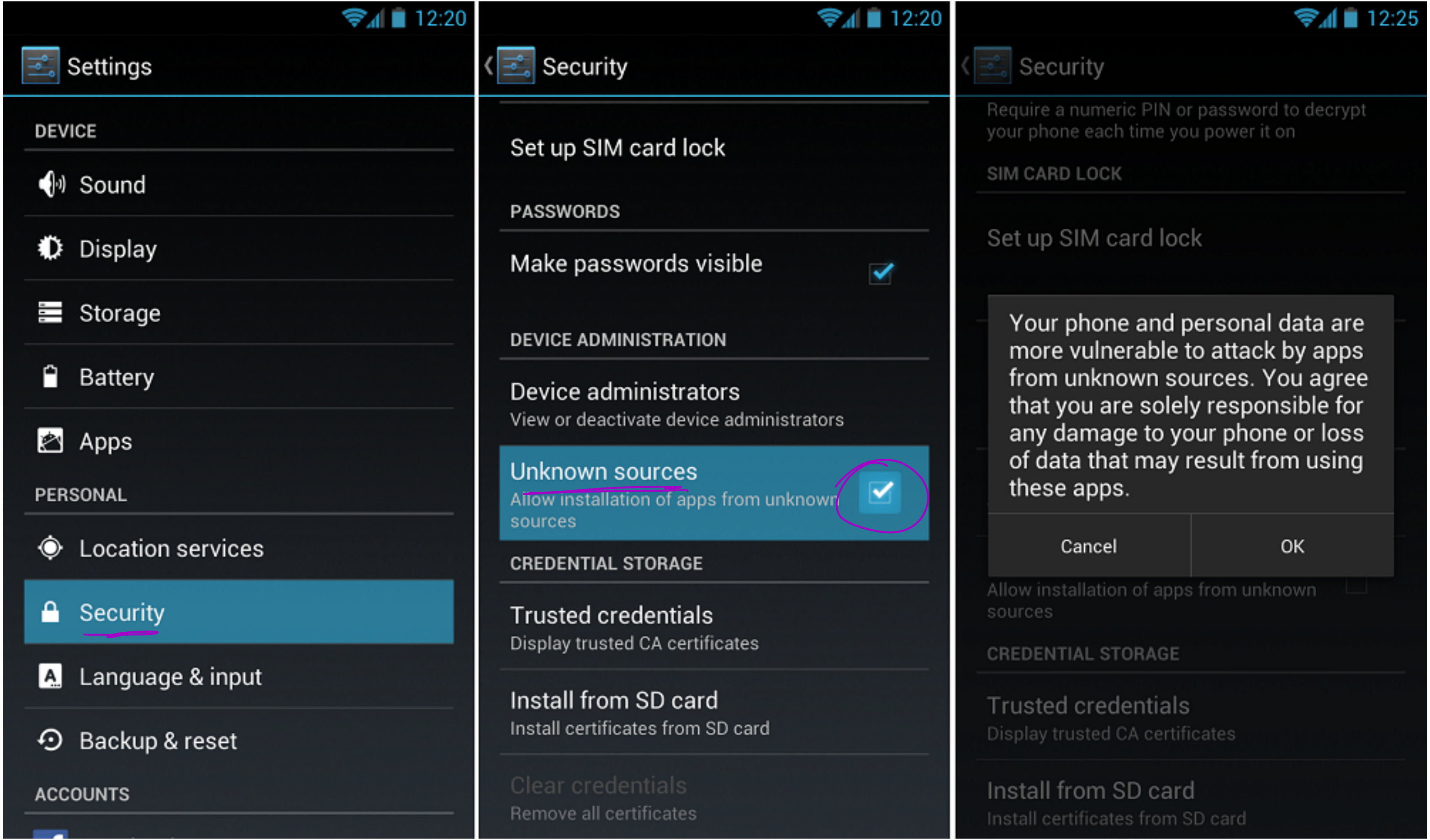
The absence of explicit permission is equivalent to no permission

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.


12:20
Settings
DEVICE
Sound
Display
Storage
Battery
Apps
PERSONAL
Location services
Security
Language & input
Backup & reset
ACCOUNTS
12:20
Security
Set up SIM card lock
PASSWORDS
Make passwords visible
DEVICE ADMINISTRATION
Device administrators
View or deactivate device administrators
Unknown sources
Allow installation of apps from unknown sources
CREDENTIAL STORAGE
Trusted credentials
Display trusted CA certificates
Install from SD card
Install certificates from SD card
Clear credentials
Remove all certificates
12:25
Security
Require a numeric PIN or password to decrypt your phone each time you power it on
SIM CARD LOCK
Set up SIM card lock
Your phone and personal data are more vulnerable to attack by apps from unknown sources. You agree that you are solely responsible for any damage to your phone or loss of data that may result from using these apps.
Cancel
OK
Allow installation of apps from unknown sources
CREDENTIAL STORAGE
Trusted credentials
Display trusted CA certificates
Install from SD card
Install certificates from SD card

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

DESKTOP

- one subject.
- not enforced usually on most laptops

MOBILE system

- every app runs as a separate user

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

Docker

/root

chroot

Privilege Over Time

DOS, Windows 3.1

All users and processes

Privilege Over Time

DOS, Windows 3.1

All users and processes

Win 95 and 98

OS

Users and Processes with System Privileges

Privilege Over Time

SE Linux (MAC)

UNIX

Linux '91

DOS, Windows 3.1

All users and processes

Win 95 and 98

OS

Users and Processes with System Privileges

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

OS

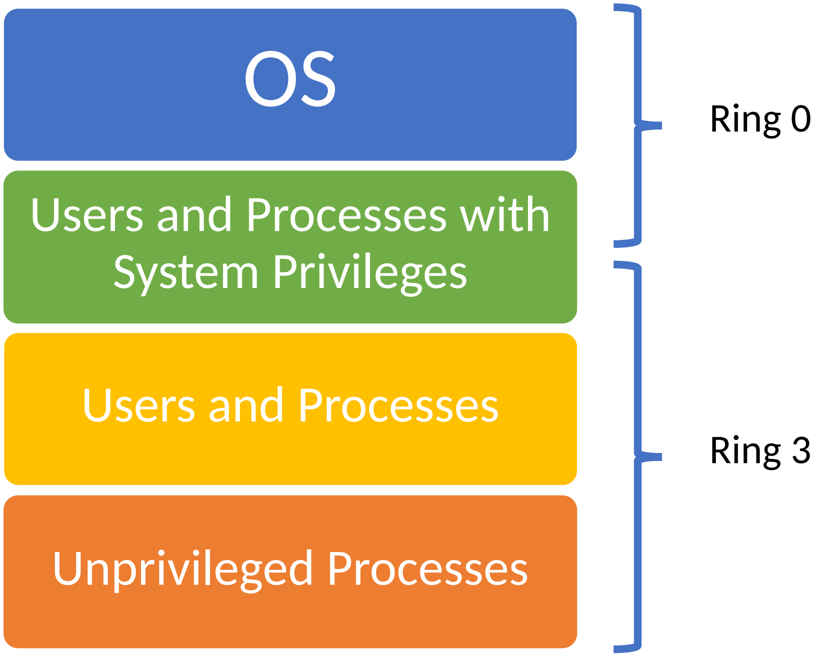
Users and Processes with System Privileges

Users and Processes

Unprivileged Processes

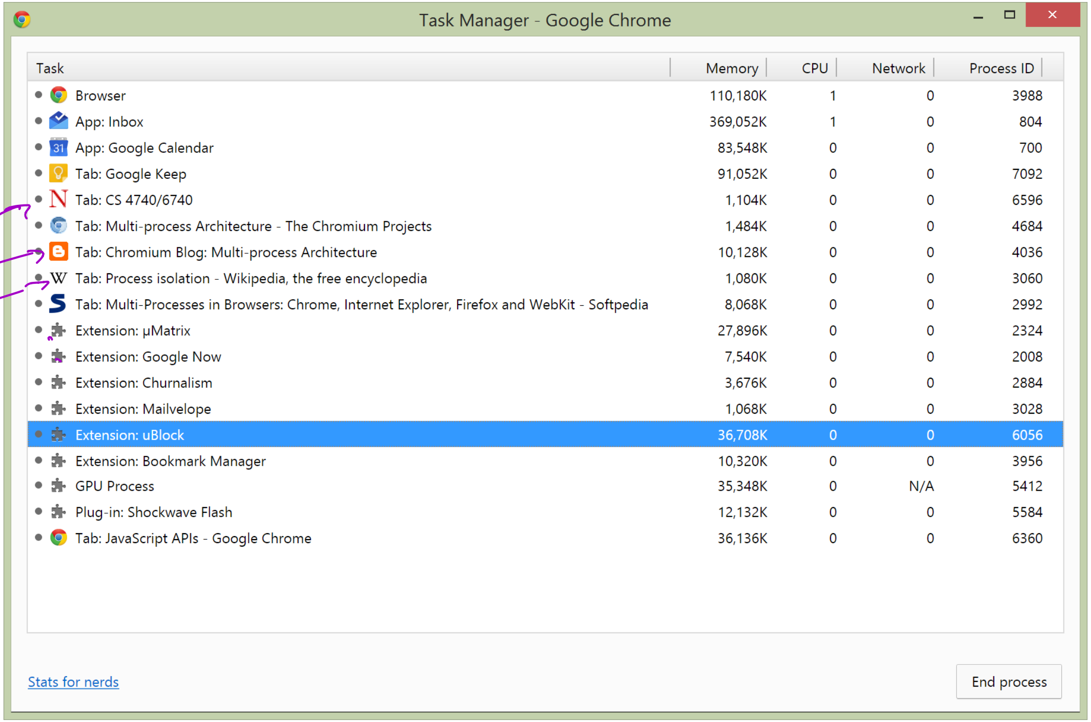
Privilege Hierarchy

- 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


Task Manager - Google Chrome

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	8,068K	0	0	2992
Extension: µMatrix	27,896K	0	0	2324
Extension: Google Now	7,540K	0	0	2008
Extension: Churnalism	3,676K	0	0	2884
Extension: Mailvelope	1,068K	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	12,132K	0	0	5584
Tab: JavaScript APIs - Google Chrome	36,136K	0	0	6360

Stats for nerds

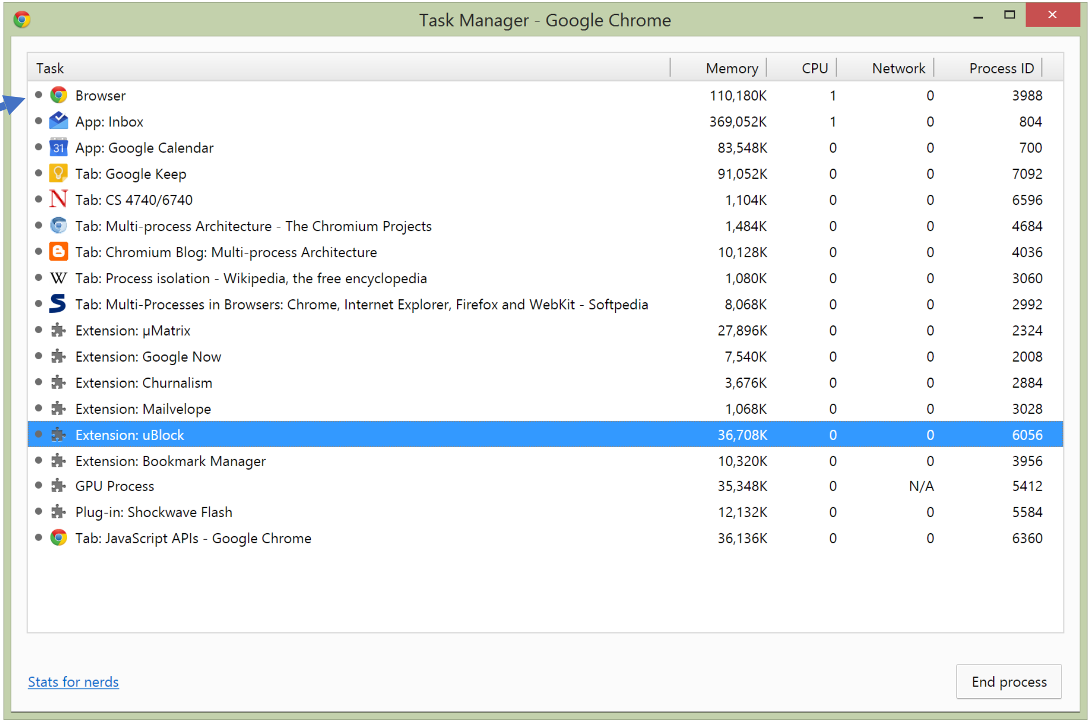
End process

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



Task Manager - Google Chrome

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	8,068K	0	0	2992
Extension: µMatrix	27,896K	0	0	2324
Extension: Google Now	7,540K	0	0	2008
Extension: Churnalism	3,676K	0	0	2884
Extension: Mailvelope	1,068K	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	12,132K	0	0	5584
Tab: JavaScript APIs - Google Chrome	36,136K	0	0	6360

Stats for nerds

End process

Example: Chrome Multiprocess Architecture

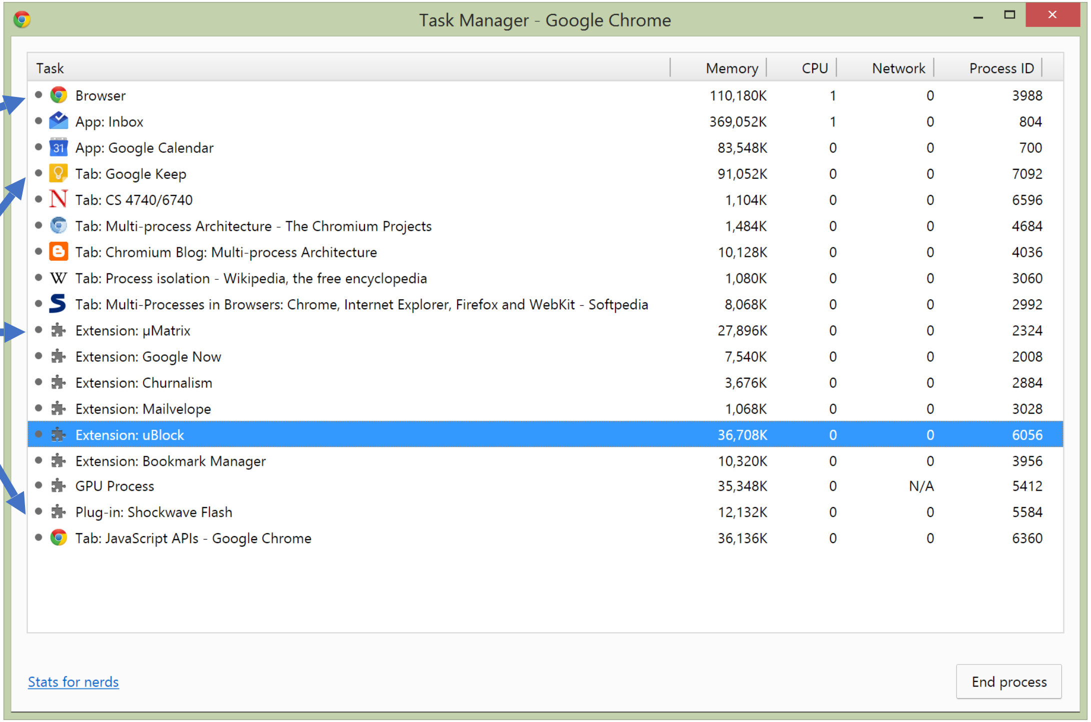
Chrome is split across many processes

“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



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	8,068K	0	0	2992
Extension: µMatrix	27,896K	0	0	2324
Extension: Google Now	7,540K	0	0	2008
Extension: Churnalism	3,676K	0	0	2884
Extension: Mailvelope	1,068K	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	12,132K	0	0	5584
Tab: JavaScript APIs - Google Chrome	36,136K	0	0	6360

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 Manager - Google Chrome

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	8,068K	0	0	2992
Extension: μMatrix	27,896K	0	0	2324
Extension: Google Now	7,540K	0	0	2008
Extension: Churnalism	3,676K	0	0	2884
Extension: Mailvelope	1,068K	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	12,132K	0	0	5584
Tab: JavaScript APIs - Google Chrome	36,136K	0	0	6360

Stats for nerds

End process

Example: Chrome Multiprocess Architecture

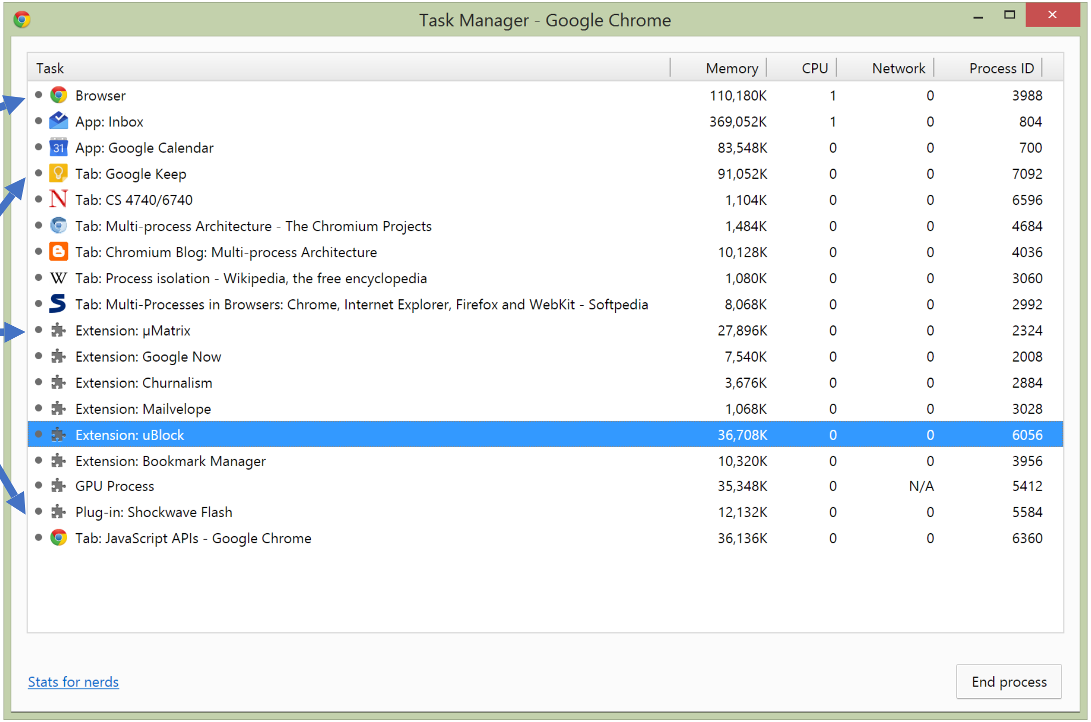
Chrome is split across many processes

“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 Manager - Google Chrome

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	8,068K	0	0	2992
Extension: µMatrix	27,896K	0	0	2324
Extension: Google Now	7,540K	0	0	2008
Extension: Churnalism	3,676K	0	0	2884
Extension: Mailvelope	1,068K	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	12,132K	0	0	5584
Tab: JavaScript APIs - Google Chrome	36,136K	0	0	6360

Stats for nerds

End process

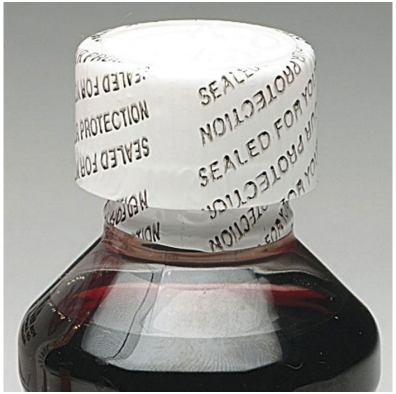
Compromise Recording

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

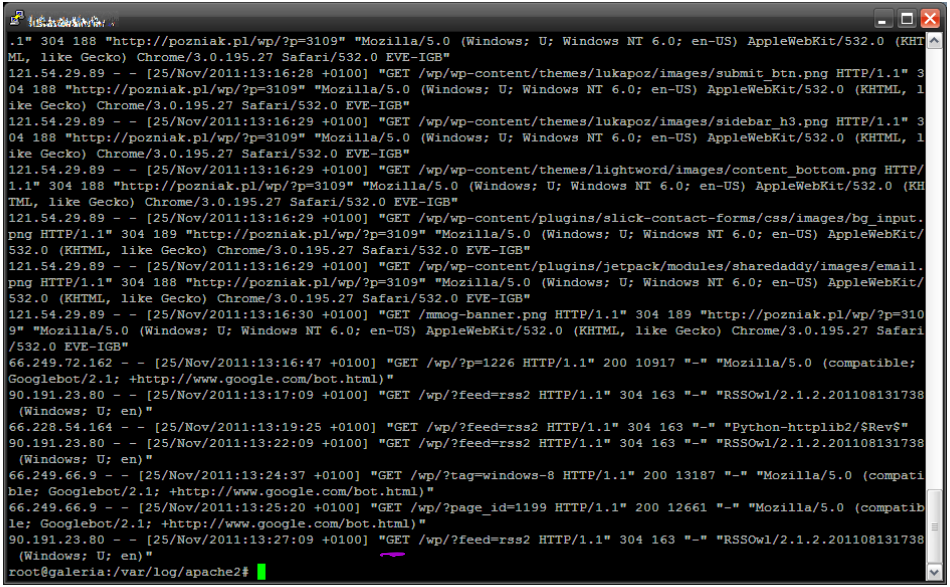
- Ensures that attacker with local access cannot erase logs

Logs are useless if they aren't monitored

Advanced approaches

- Intrusion Detection Systems (IDS)
- Anomaly detection
- Machine learning-based approaches

nginx



```
.1" 304 188 "http://pozniak.pl/wp/?p=3109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHT
ML, like Gecko) Chrome/3.0.195.27 Safari/532.0 EVE-IGB"
121.54.29.89 - - [25/Nov/2011:13:16:28 +0100] "GET /wp/wp-content/themes/lukapoz/images/submit_btn.png HTTP/1.1" 3
04 188 "http://pozniak.pl/wp/?p=3109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, l
ike Gecko) Chrome/3.0.195.27 Safari/532.0 EVE-IGB"
121.54.29.89 - - [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/themes/lukapoz/images/sidebar_h3.png HTTP/1.1" 3
04 188 "http://pozniak.pl/wp/?p=3109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, l
ike Gecko) Chrome/3.0.195.27 Safari/532.0 EVE-IGB"
121.54.29.89 - - [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/themes/lightword/images/content_bottom.png HTTP/
1.1" 304 188 "http://pozniak.pl/wp/?p=3109" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KH
TML, like Gecko) Chrome/3.0.195.27 Safari/532.0 EVE-IGB"
121.54.29.89 - - [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/plugins/slick-contact-forms/css/images/bg_input.
png HTTP/1.1" 304 189 "http://pozniak.pl/wp/?p=3109" "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 EVE-IGB"
121.54.29.89 - - [25/Nov/2011:13:16:29 +0100] "GET /wp/wp-content/plugins/jetpack/modules/sharedaddy/images/email.
png HTTP/1.1" 304 188 "http://pozniak.pl/wp/?p=3109" "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 EVE-IGB"
121.54.29.89 - - [25/Nov/2011:13:16:30 +0100] "GET /mmog-banner.png HTTP/1.1" 304 189 "http://pozniak.pl/wp/?p=310
9" "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 EVE-IGB"
66.249.72.162 - - [25/Nov/2011:13:16:47 +0100] "GET /wp/?p=1226 HTTP/1.1" 200 10917 "-" "Mozilla/5.0 (compatible;
Googlebot/2.1; +http://www.google.com/bot.html)"
90.191.23.80 - - [25/Nov/2011:13:17:09 +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/$Rev$"
90.191.23.80 - - [25/Nov/2011:13:22:09 +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 (compati
ble; 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 (compatib
le; Googlebot/2.1; +http://www.google.com/bot.html)"
90.191.23.80 - - [25/Nov/2011:13:27:09 +0100] "GET /wp/?feed=rss2 HTTP/1.1" 304 163 "-" "RSSOwl/2.1.2.201108131738
 (Windows; U; en)"
root@galeria:/var/log/apache2#
```

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

bcrypt Example

Work factor

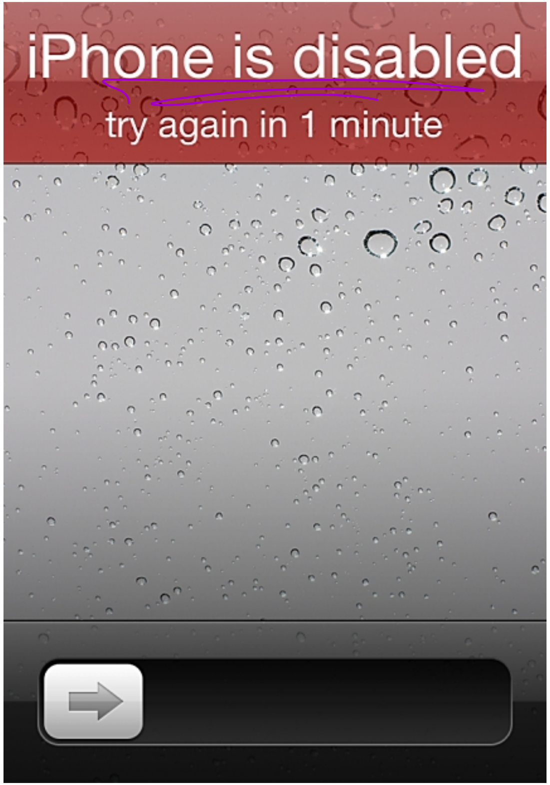
```
[cbw@localhost ~] python
>>> import bcrypt
>>> password = “my super secret password”
>>> fast_hashed = bcrypt.hashpw(password, bcrypt.gensalt(0))
>>> slow_hashed = bcrypt.hashpw(password, bcrypt.gensalt(12))
>>> pw_from_user = raw_input(“Enter your password:”)
>>> if bcrypt.hashpw(pw_from_user, slow_hashed) == slow_hashed:
…           print “It matches! You may enter the system”
…   else:
…           print “No match. You may not proceed”
```

Authentication Rate Limiting

Short delay after each failed authentication attempt

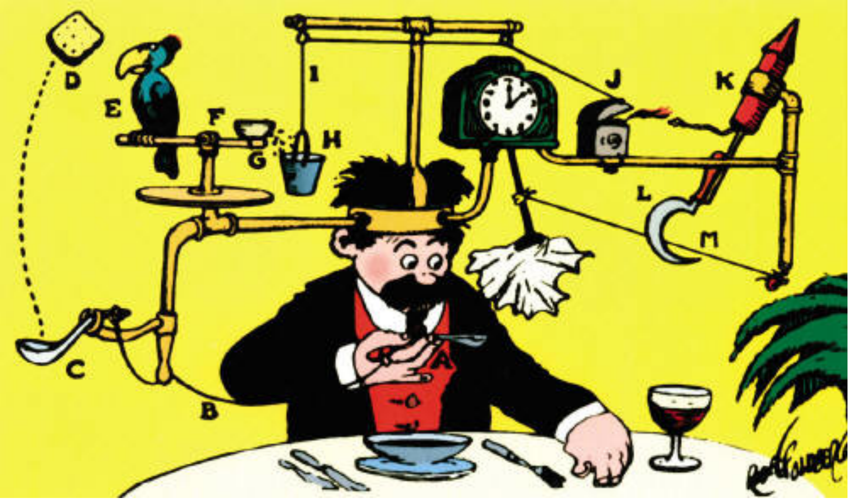
- Delays may increase as the consecutive failed attempts increase

Does not prevent password cracking attempts, but slows them down



Economy of Mechanism

Simplicity



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…

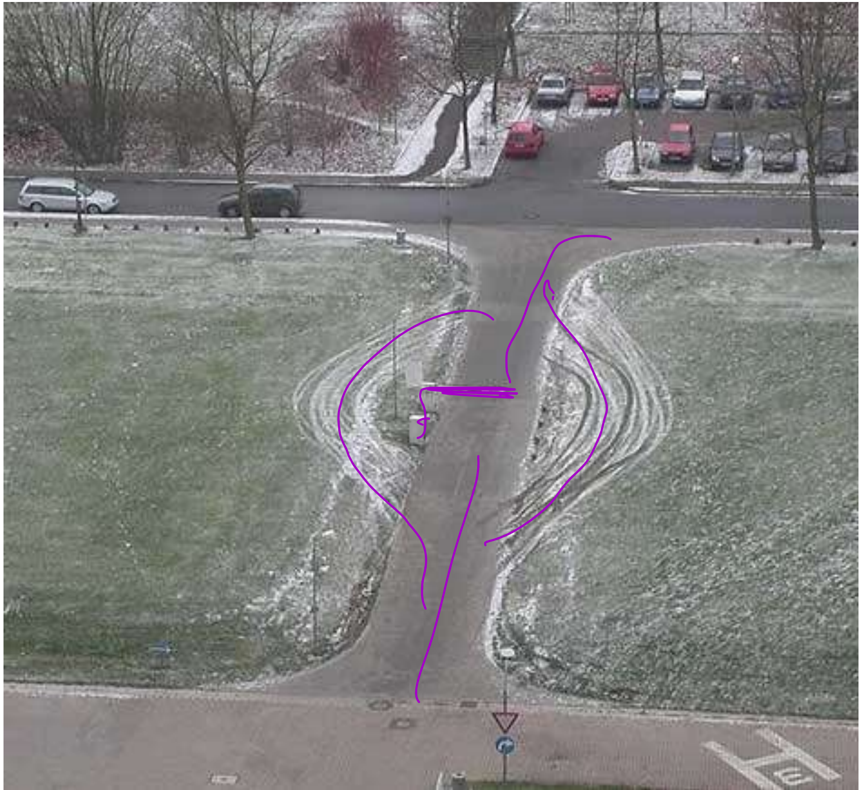
Micro-kernel OS

- 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

- GNU Hurd
- seL4 – formally verified!

Complete Mediation

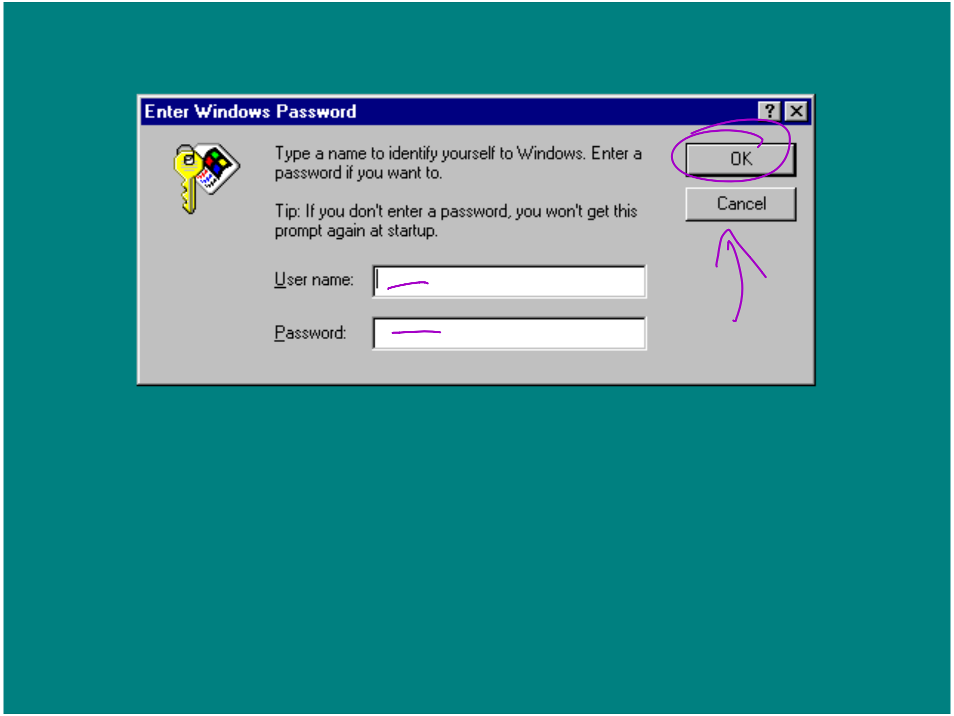


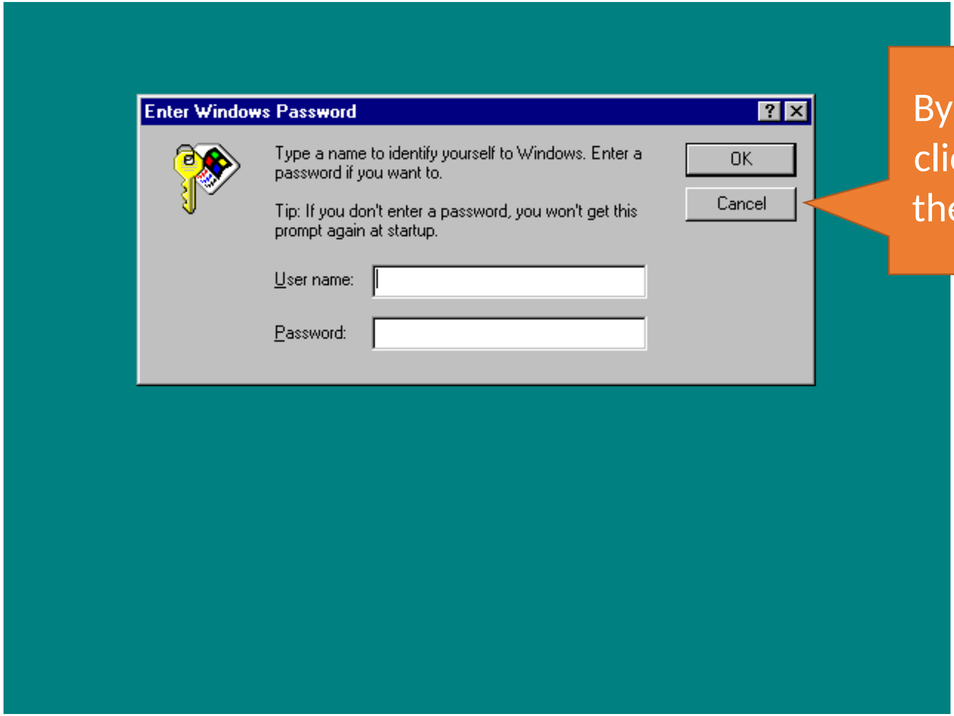
Complete Mediation

Every access to every object must be checked for authorization

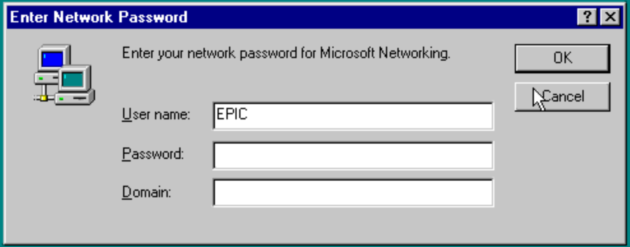
Incomplete mediation implies that a path exists to bypass a security mechanism

In other words, isolation is incomplete

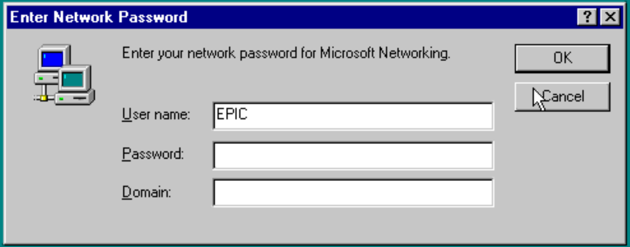

Enter Windows Password
Type a name to identify yourself to Windows. Enter a password if you want to.
Tip: If you don't enter a password, you won't get this prompt again at startup.
User name:
Password:
OK
Cancel



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



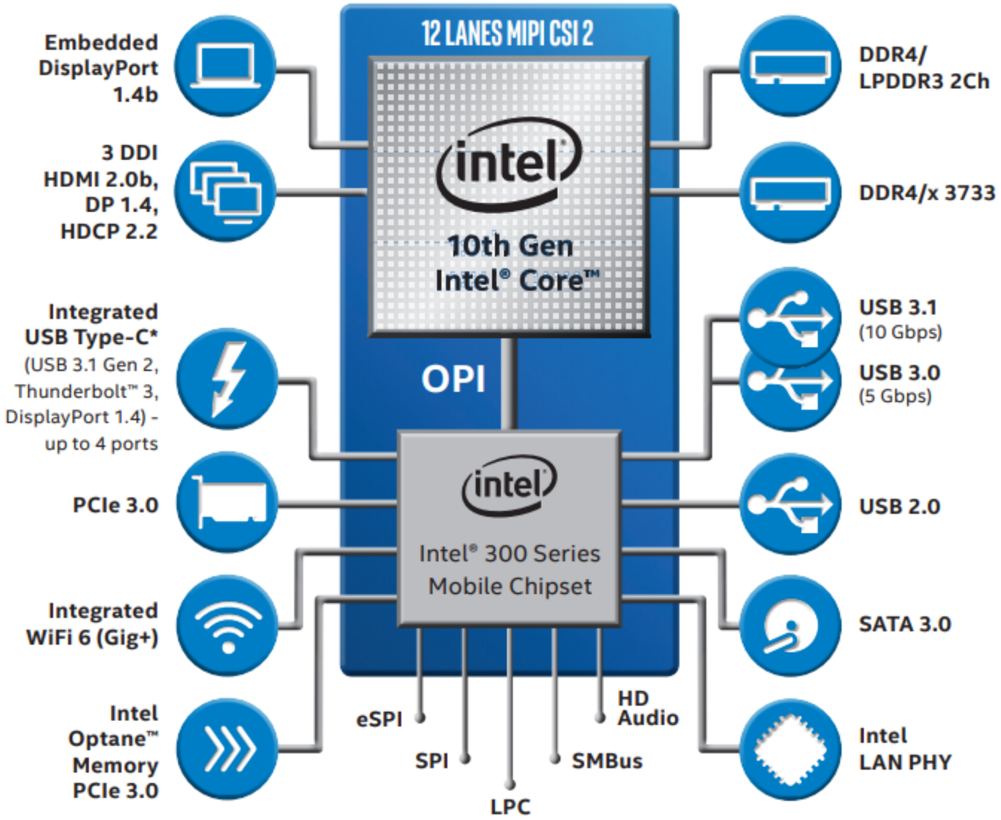
Forgotten you password ?
No problem

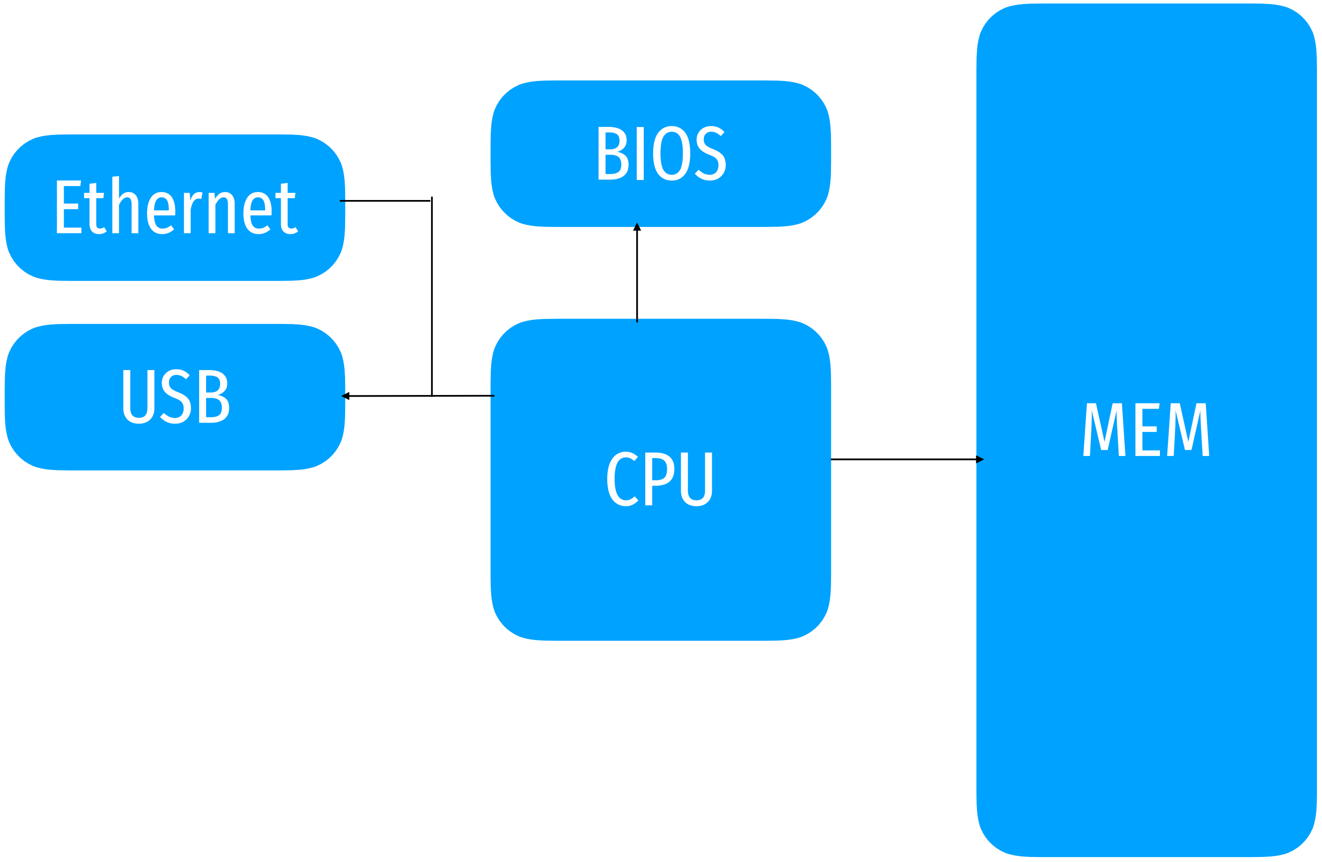


Forgotten you password ?
No problem

Threat Model
Principles
Intro to System Architecture
Hardware Support for Isolation
Examples


IBM
IBM
IBM


12 LANES MIPI CSI 2
intel
10th Gen
Intel® Core™
OPI
intel
Intel® 300 Series
Mobile Chipset
Embedded
DisplayPort
1.4b
3 DDI
HDMI 2.0b,
DP 1.4,
HDCP 2.2
Integrated
USB Type-C*
(USB 3.1 Gen 2,
Thunderbolt™ 3,
DisplayPort 1.4) -
up to 4 ports
PCIe 3.0
Integrated
WiFi 6 (Gig+)
Intel
Optane™
Memory
PCIe 3.0
DDR4/
LPDDR3 2Ch
DDR4/x 3733
USB 3.1
(10 Gbps)
USB 3.0
(5 Gbps)
USB 2.0
SATA 3.0
Intel
LAN PHY
eSPI
SPI
LPC
SMBus
HD
Audio


Ethernet
BIOS
USB
CPU
MEM

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

Address	Contents
114	
113	
112	
111	
110	
109	
108	
107	
106	
105	
104	
103	
102	
101	
100	

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

Integers are typically four bytes

Address	Contents
114	
113	0
112	0
111	0
110	8
109	
108	[illegible]
107	[illegible]
106	[illegible]
105	[illegible]
104	
103	[illegible]
102	[illegible]
101	[illegible]
100	[illegible]

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

Integers are typically four bytes

Each ASCII character is one byte, Strings are null terminated

Address	Contents
114	
113	0
112	0
111	0
110	8
109	
108	0
107	C
106	B
105	A
104	
103	[illegible]
102	[illegible]
101	[illegible]
100	[illegible]

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

Integers are typically four bytes

Each ASCII character is one byte, Strings are null terminated

CPUs understand instructions in assembly language

Address	Contents
114	
113	0
112	0
111	0
110	8
109	
108	0
107	C
106	B
105	A
104	
103	0xAF
102	0x3C
101	0x91
100	0xE3

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

All data and running code are held in memory

```
int my_num = 8;
```

Integers are typically four bytes

Each ASCII character is one byte, Strings are null terminated

CPUs understand instructions in assembly language

Address	Contents
114	
113	0
112	0
111	0
110	8
109	
108	0
107	C
106	B
105	A
104	
103	0xAF
102	0x3C
101	0x91
100	0xE3

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

All data and running code are held in memory

```
int my_num = 8;

String my_str = "ABC";
```

Integers are typically four bytes

Each ASCII character is one byte, Strings are null terminated

CPUs understand instructions in assembly language

Address	Contents
114	
113	0
112	0
111	0
110	8
109	
108	0
107	C
106	B
105	A
104	
103	0xAF
102	0x3C
101	0x91
100	0xE3

What is Memory?

Memory is essentially a spreadsheet with a single column

- Every row has a number, called an address
- Every cell holds 1 byte of data

All data and running code are held in memory

```
int my_num = 8;

String my_str = "ABC";
while (my_num > 0) my_num--;
```

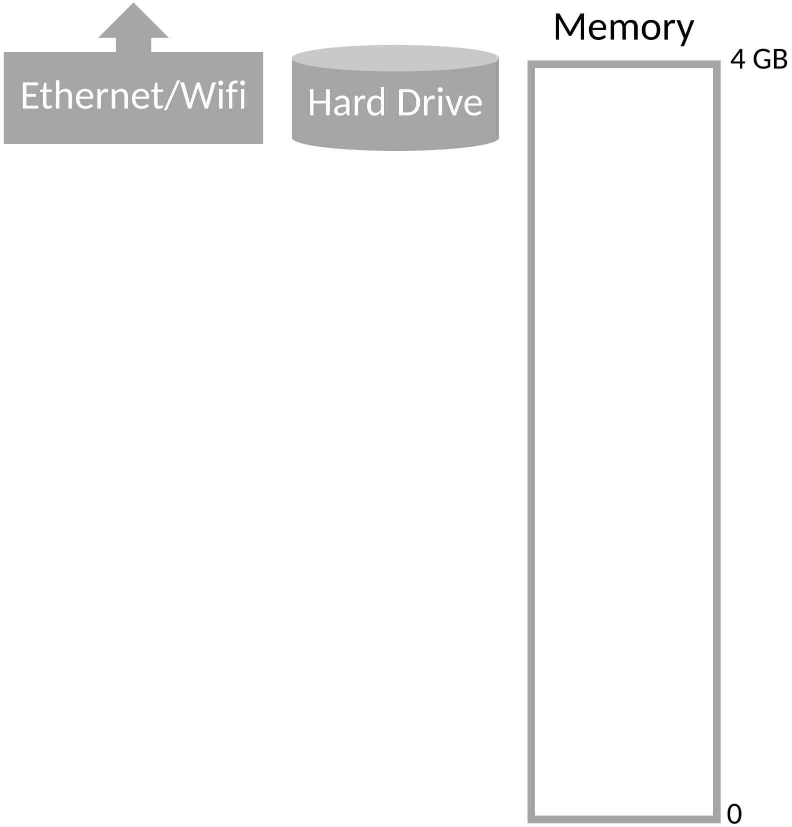
Integers are typically four bytes

Each ASCII character is one byte, Strings are null terminated

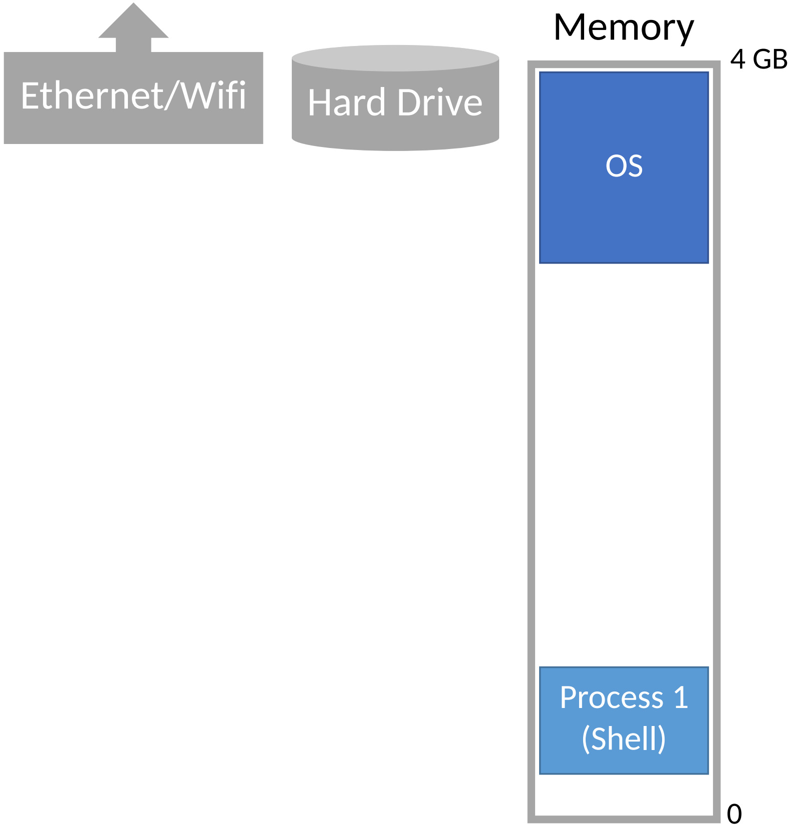
CPUs understand instructions in assembly language

Address	Contents
114	
113	0
112	0
111	0
110	8
109	
108	0
107	C
106	B
105	A
104	
103	0xAF
102	0x3C
101	0x91
100	0xE3

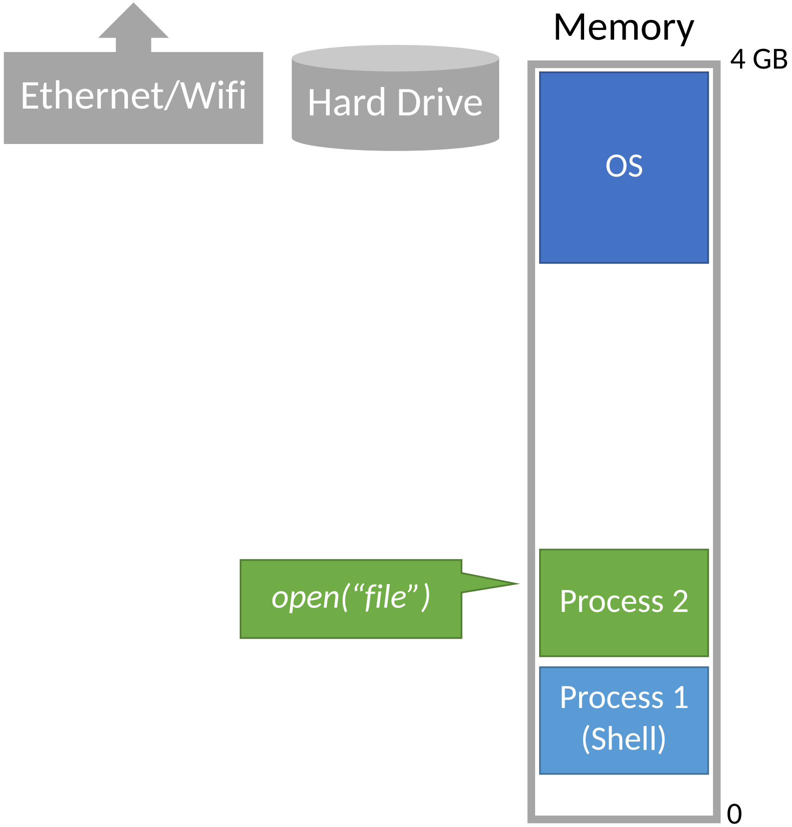
System Model



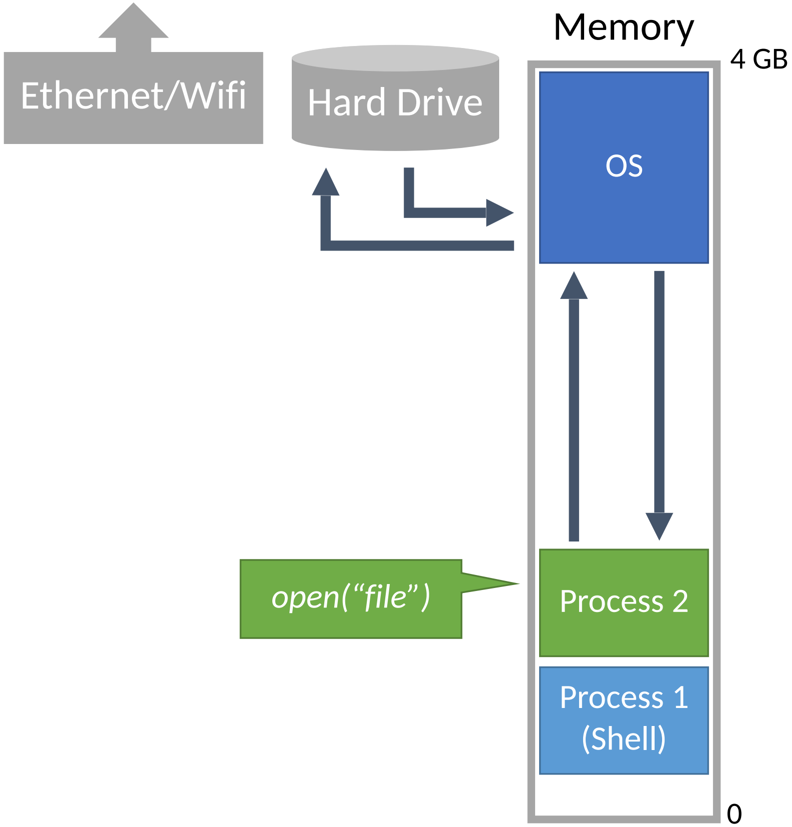
System Model



System Model



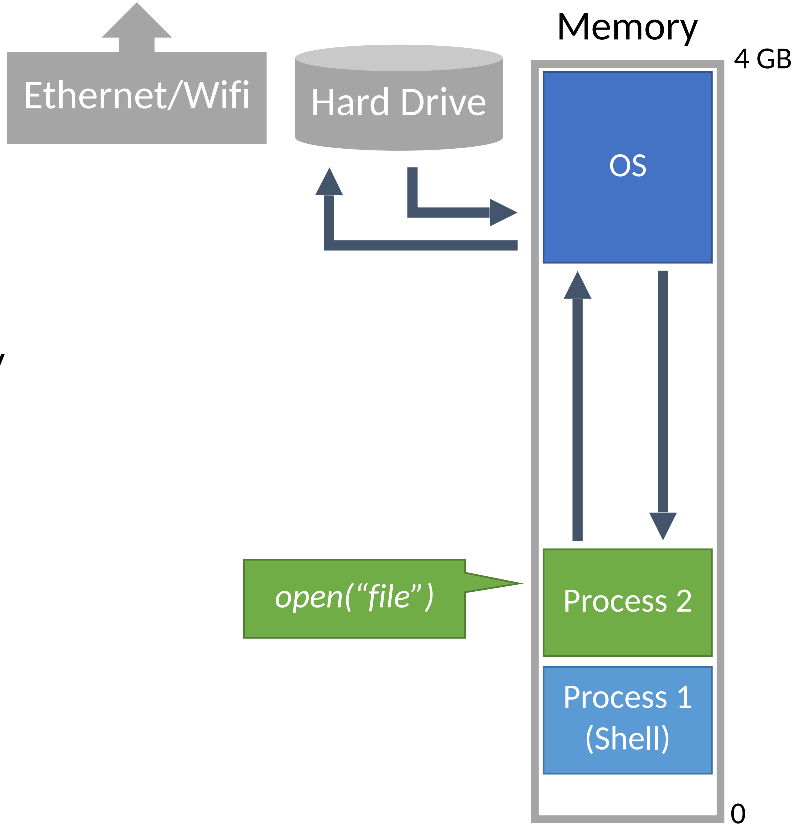
System Model



System Model

On bootup, the Operating System (OS) loads itself into memory

- eg. DOS (before hw isolation)
- Typically places itself in high memory



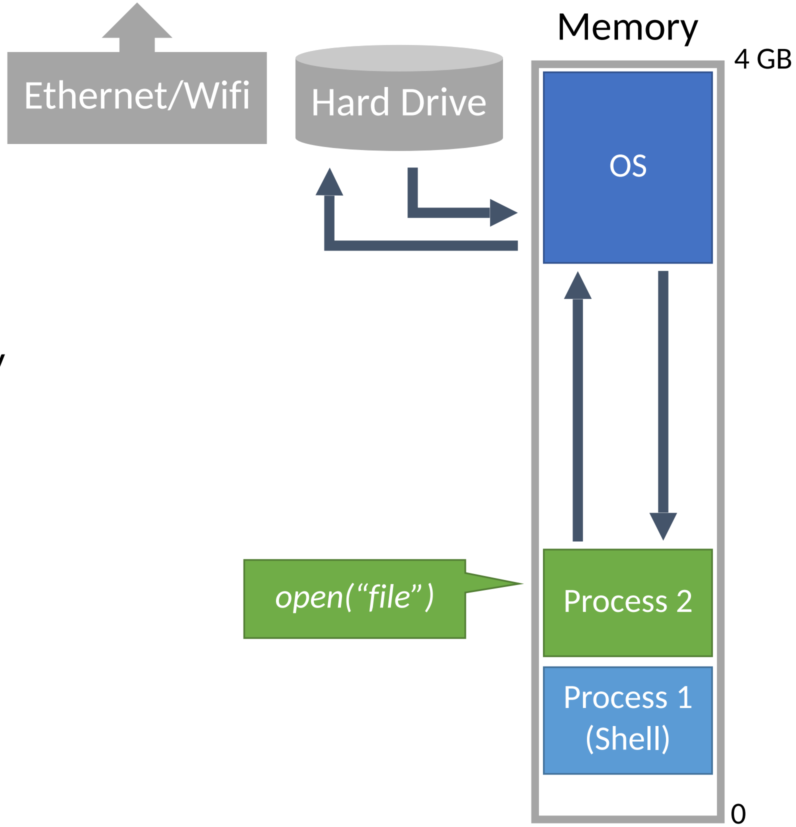
System Model

On bootup, the Operating System (OS) loads itself into memory

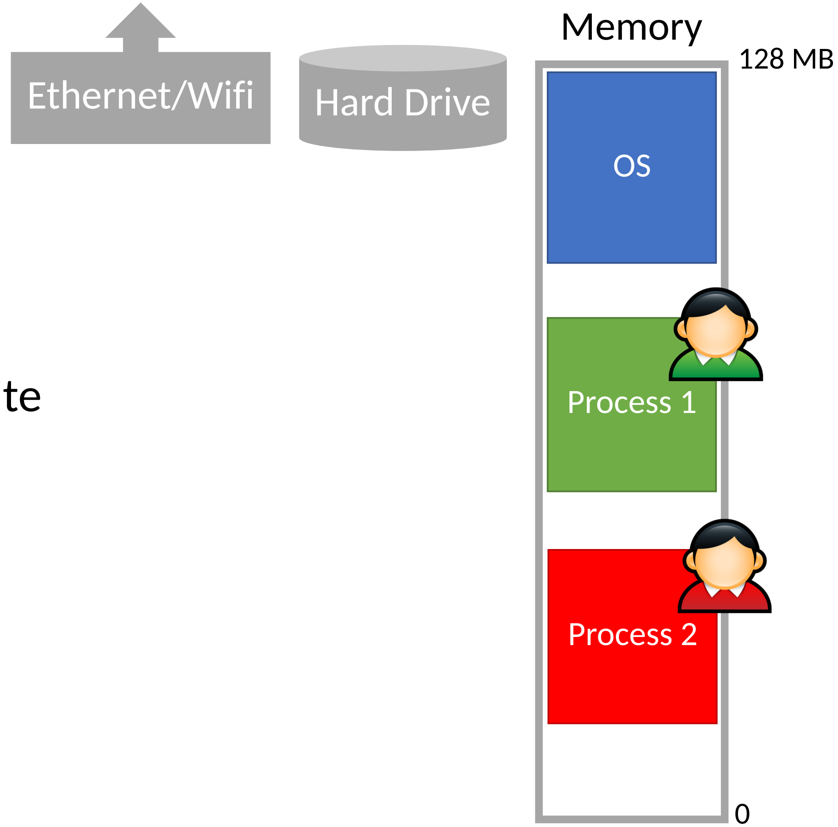
- eg. DOS (before hw isolation)
- Typically places itself in high memory

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



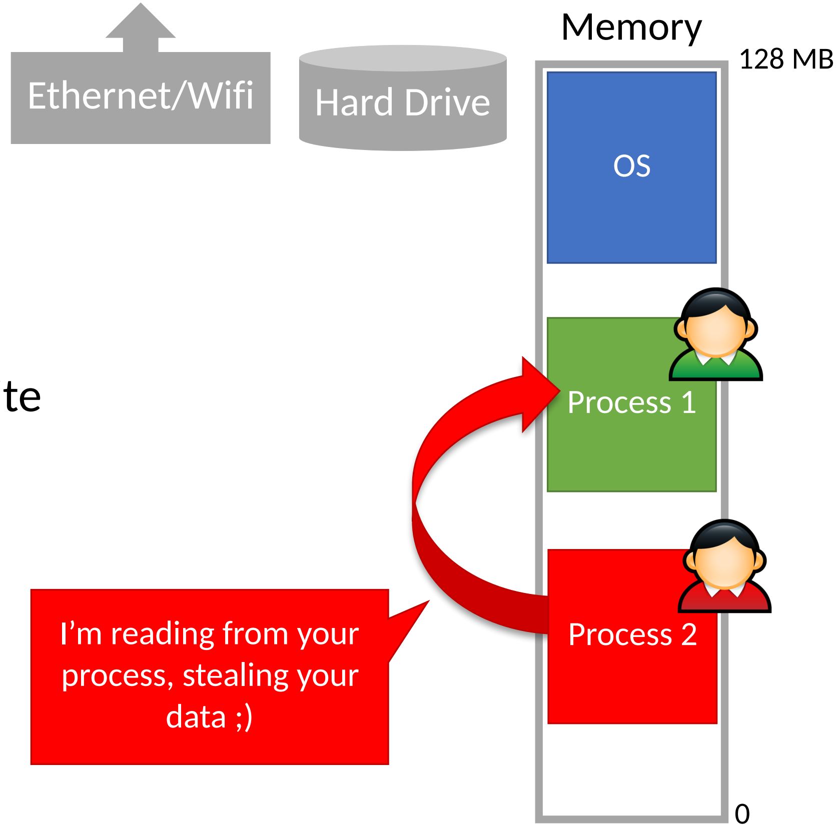
Memory Unsafety



Problem: any process can read/write any memory

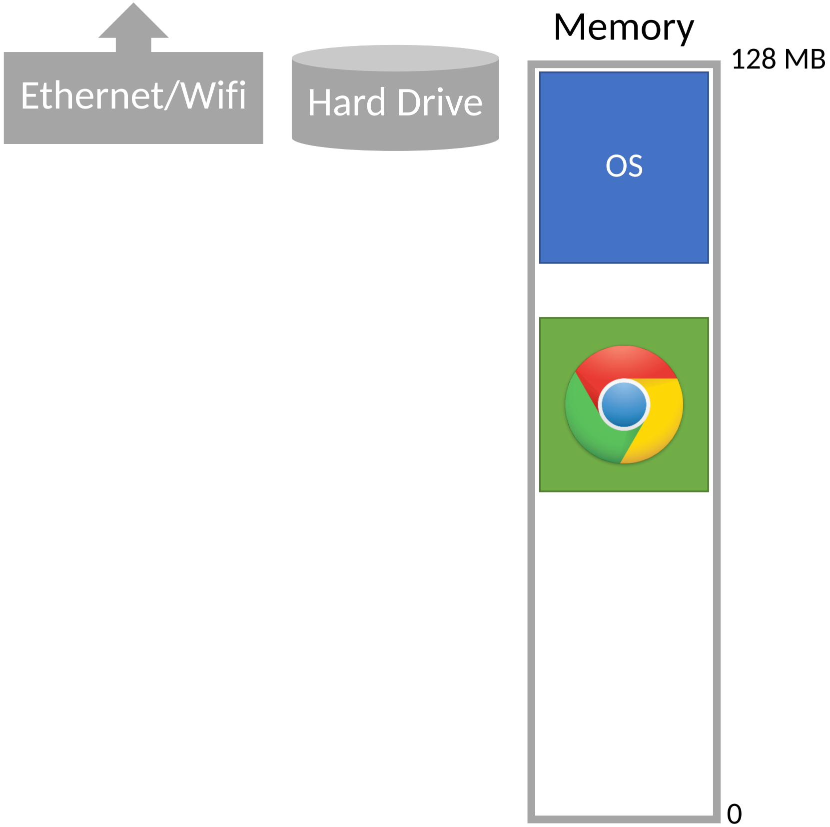
Memory Unsafety

Problem: any process can read/write any memory



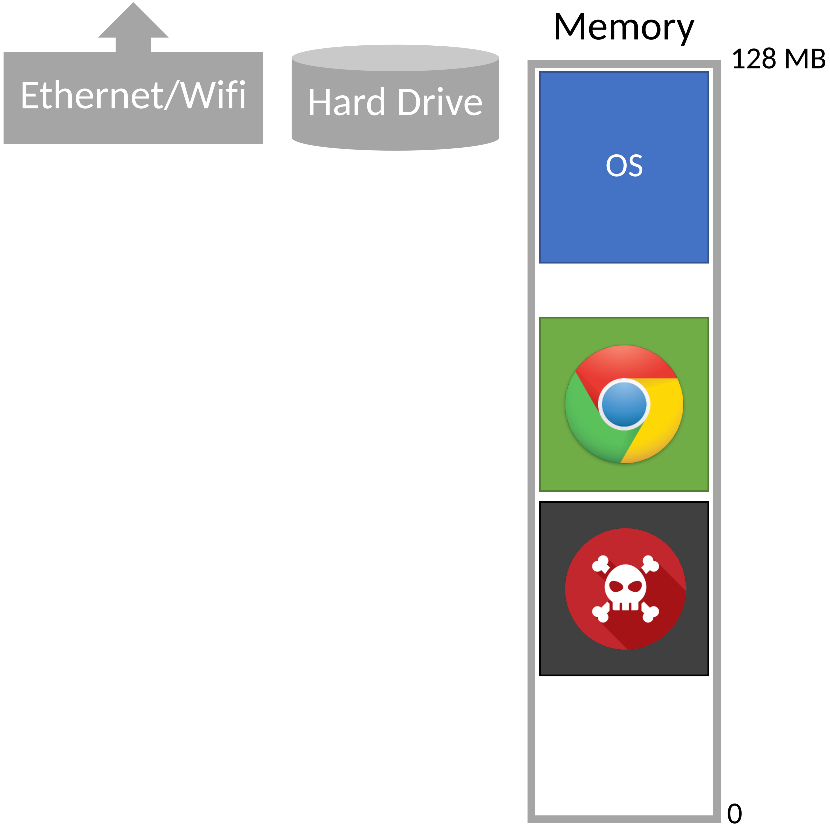
Memory Unsafety

Problem: any process can read/write any memory



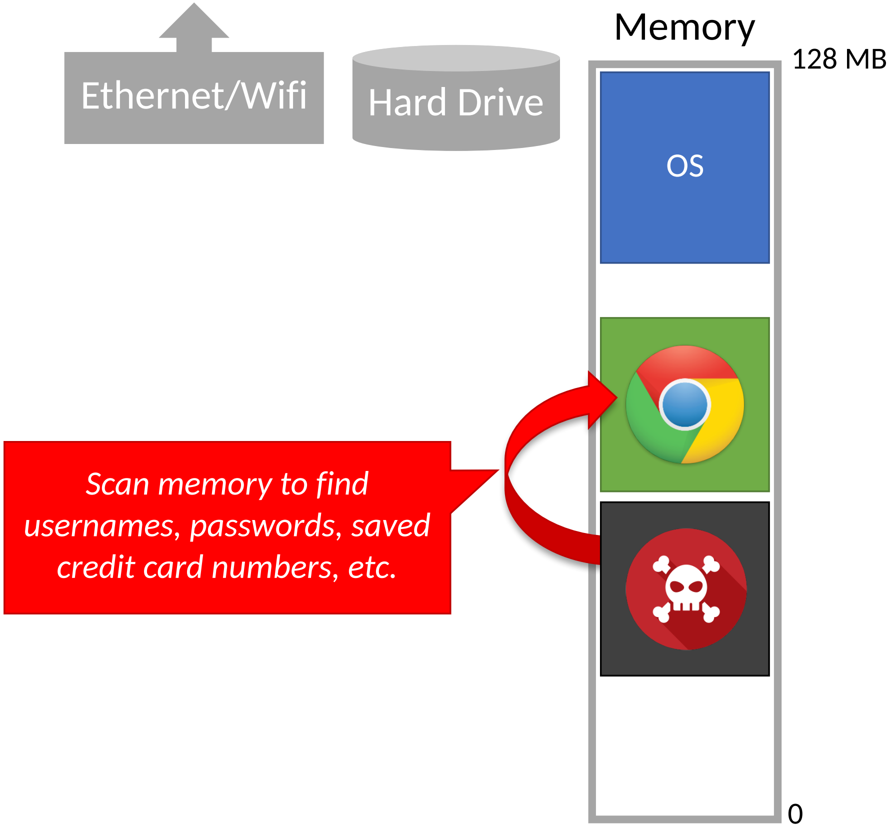
Memory Unsafety

Problem: any process can read/write any memory



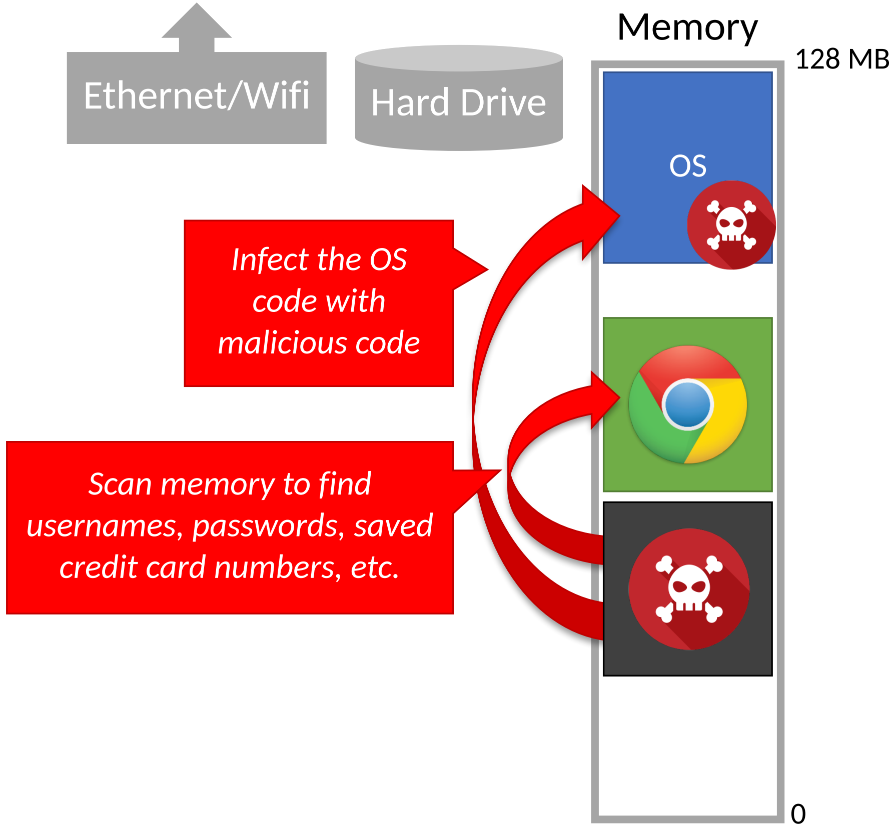
Memory Unsafety

Problem: any process can read/write any memory

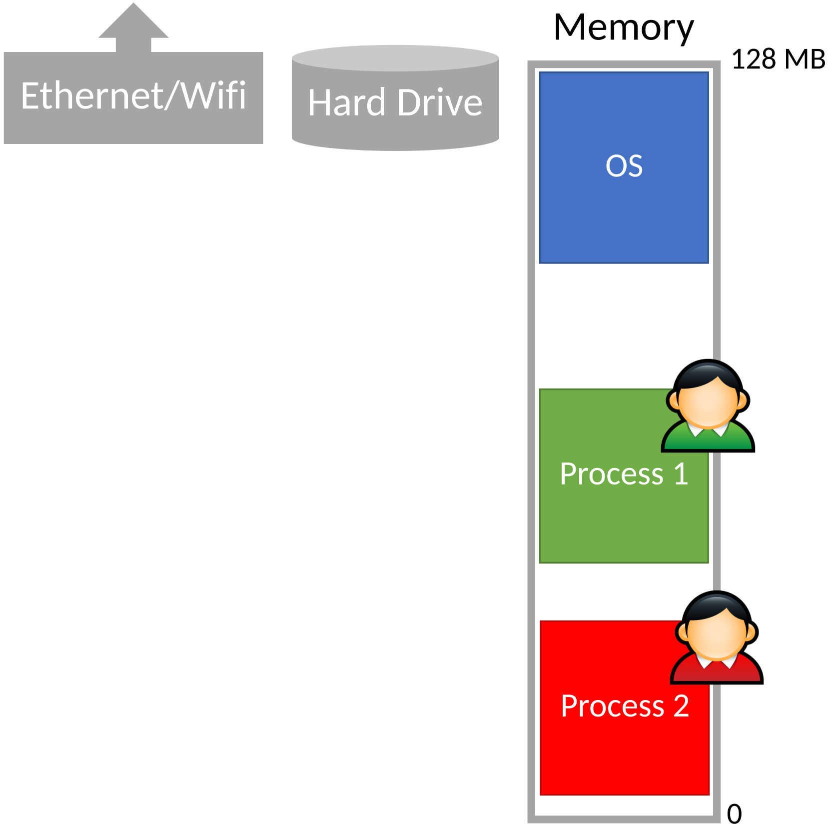


Memory Unsafety

Problem: any process can read/write any memory



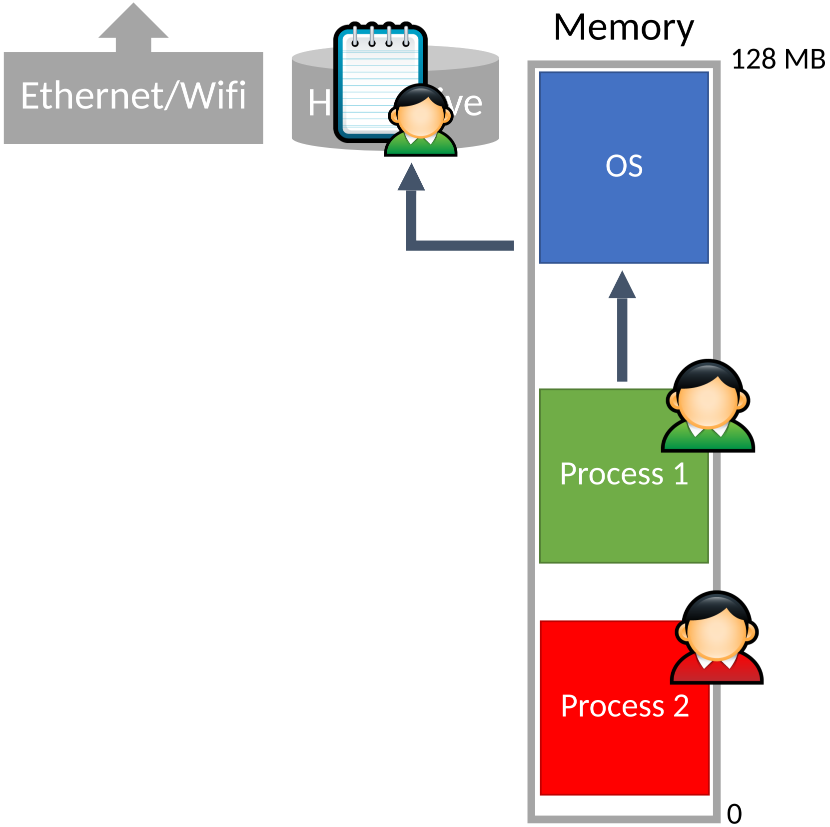
Device Unsafety



Problem: any process can access any hardware device directly

Access control is enforced by the OS, but OS APIs can be bypassed

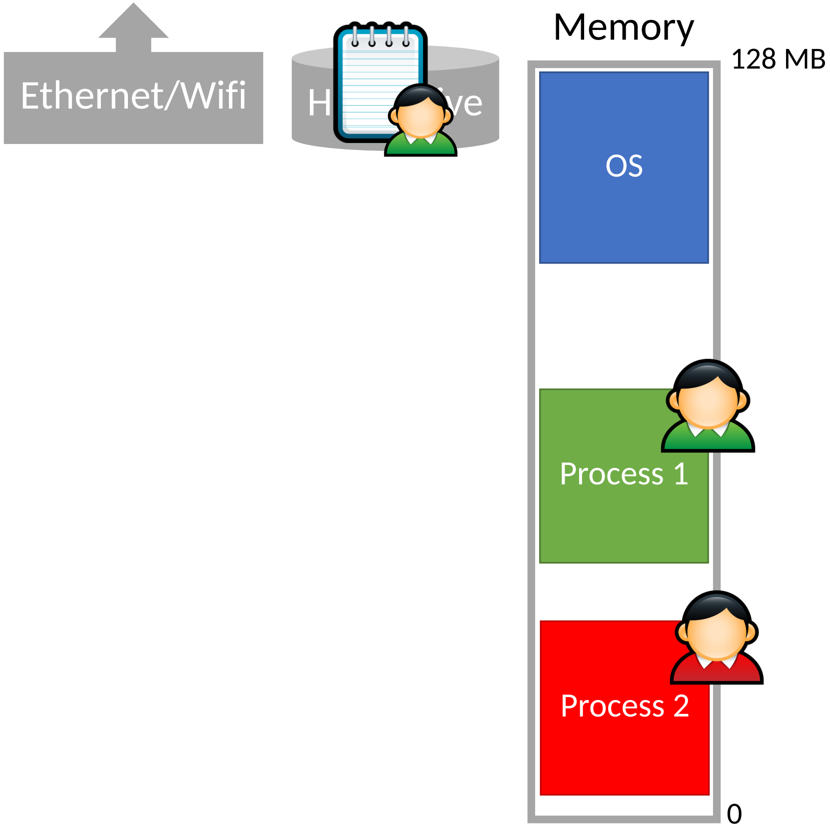
Device Unsafety



Problem: any process can access any hardware device directly

Access control is enforced by the OS, but OS APIs can be bypassed

Device Unsafety



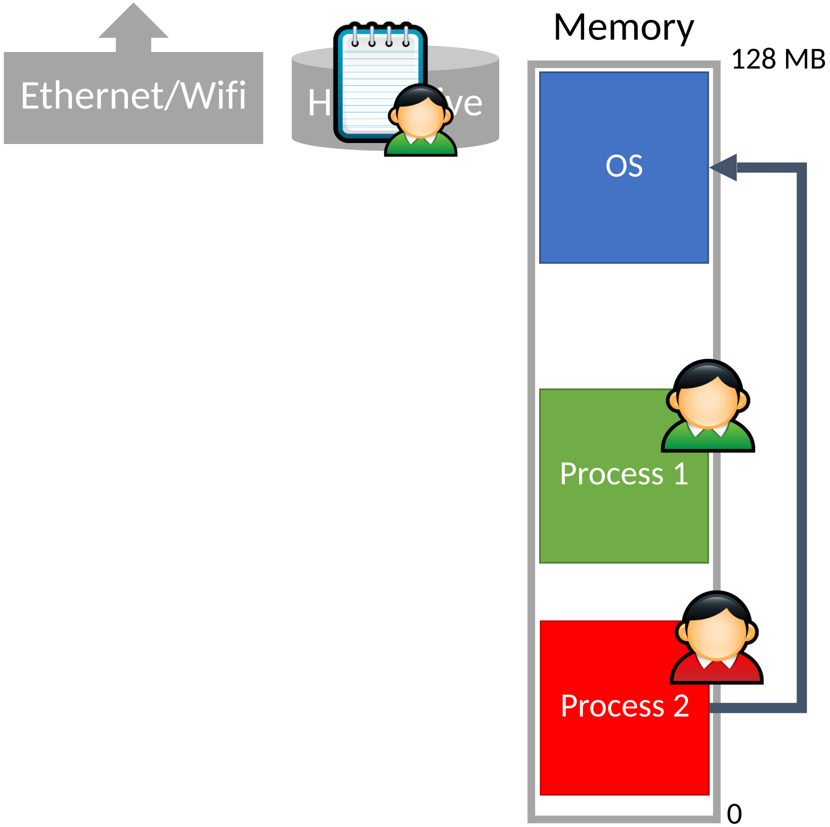
Problem: any process can access any hardware device directly

Access control is enforced by the OS, but OS APIs can be bypassed

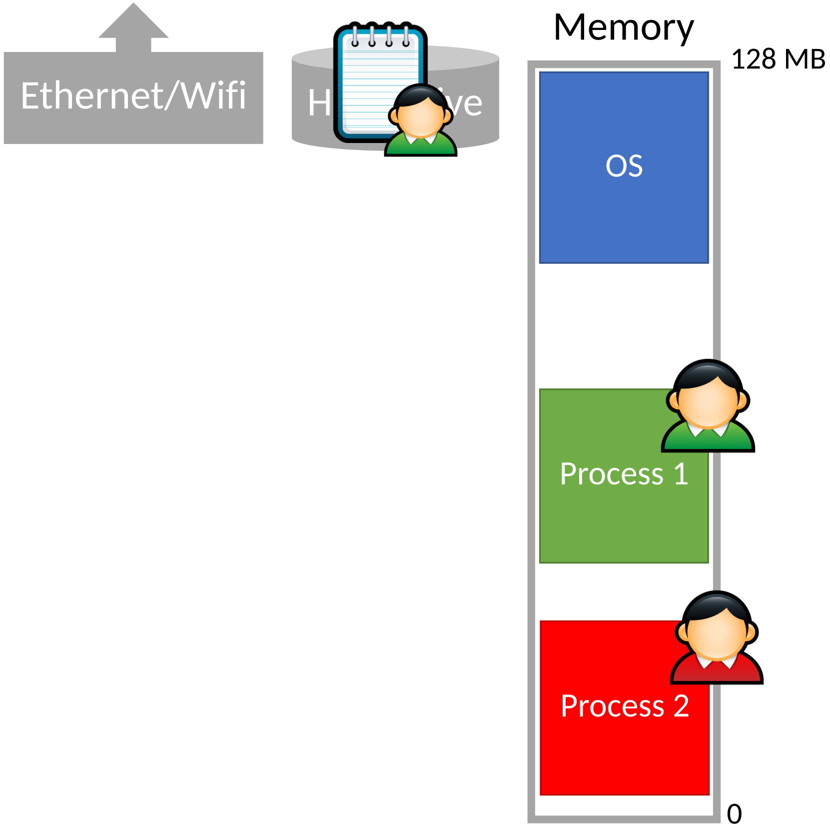
Device Unsafety

Problem: any process can access any hardware device directly

Access control is enforced by the OS, but OS APIs can be bypassed



Device Unsafety



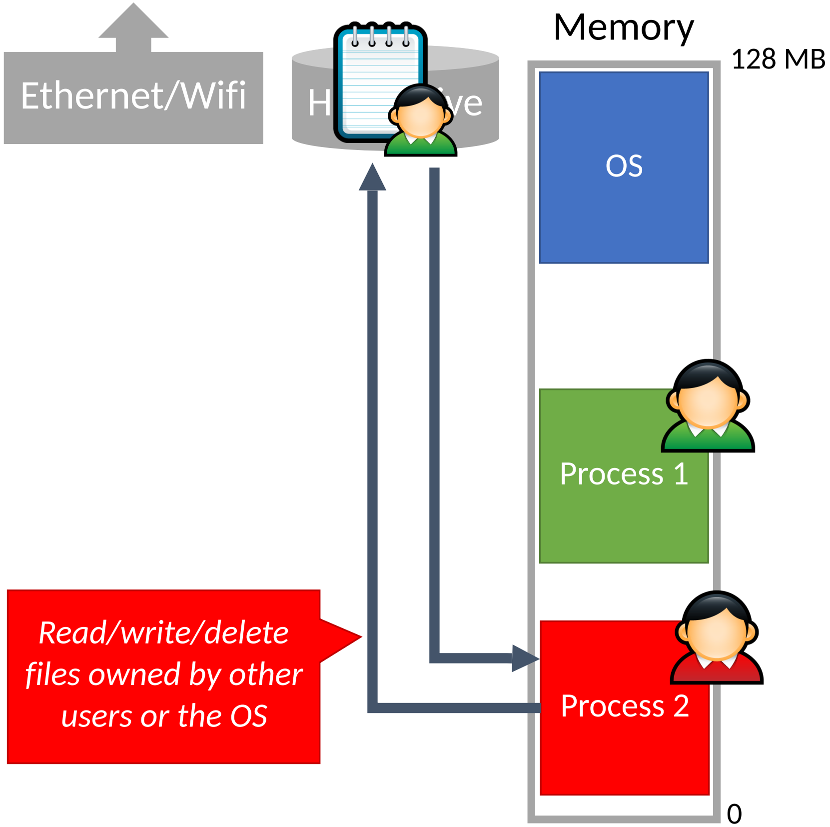
Problem: any process can access any hardware device directly

Access control is enforced by the OS, but OS APIs can be bypassed

Device Unsafety

Problem: any process can access any hardware device directly

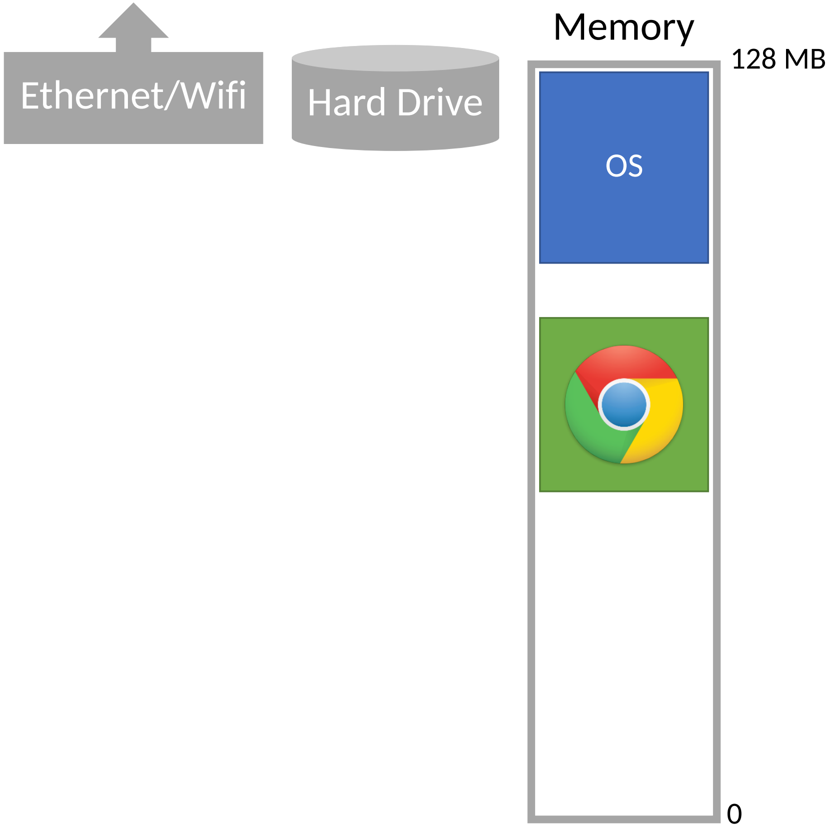
Access control is enforced by the OS, but OS APIs can be bypassed



Device Unsafety

Problem: any process can access any hardware device directly

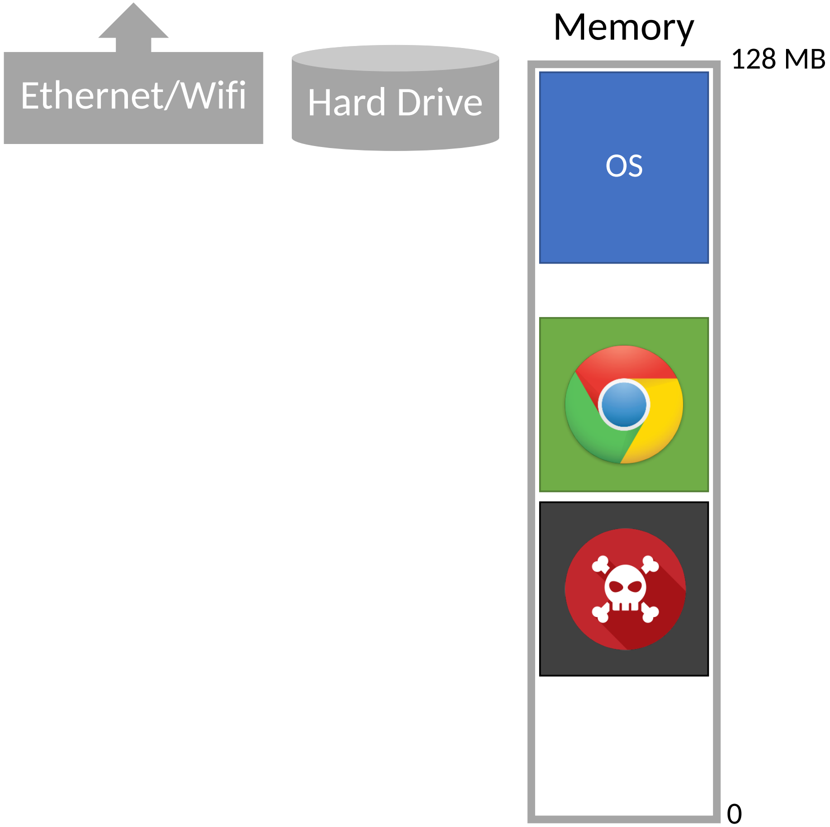
Access control is enforced by the OS, but OS APIs can be bypassed



Device Unsafety

Problem: any process can access any hardware device directly

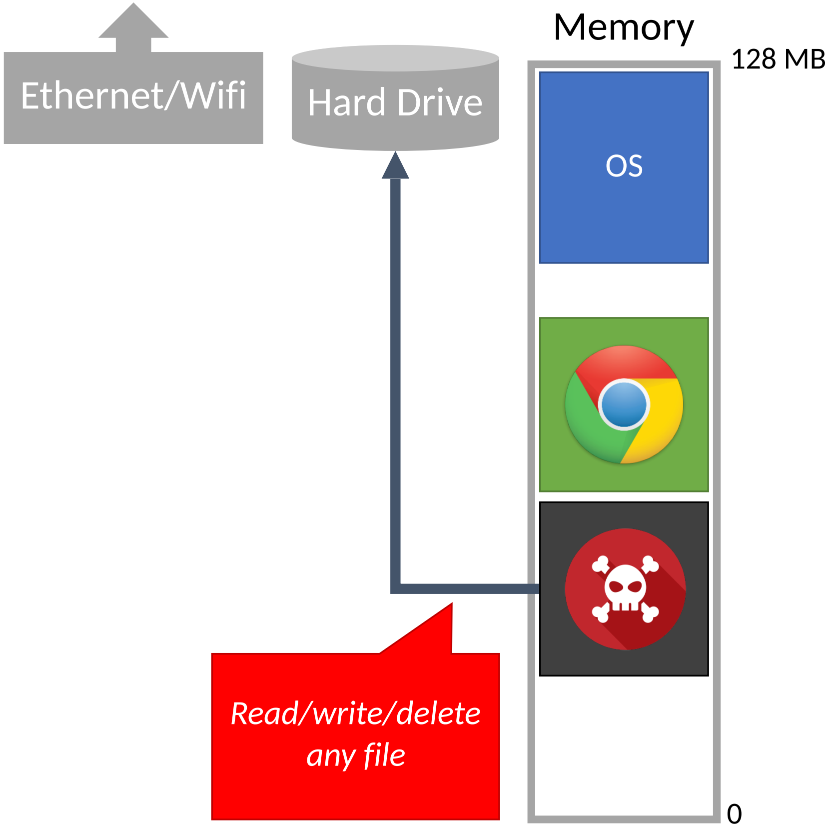
Access control is enforced by the OS, but OS APIs can be bypassed



Device Unsafety

Problem: any process can access any hardware device directly

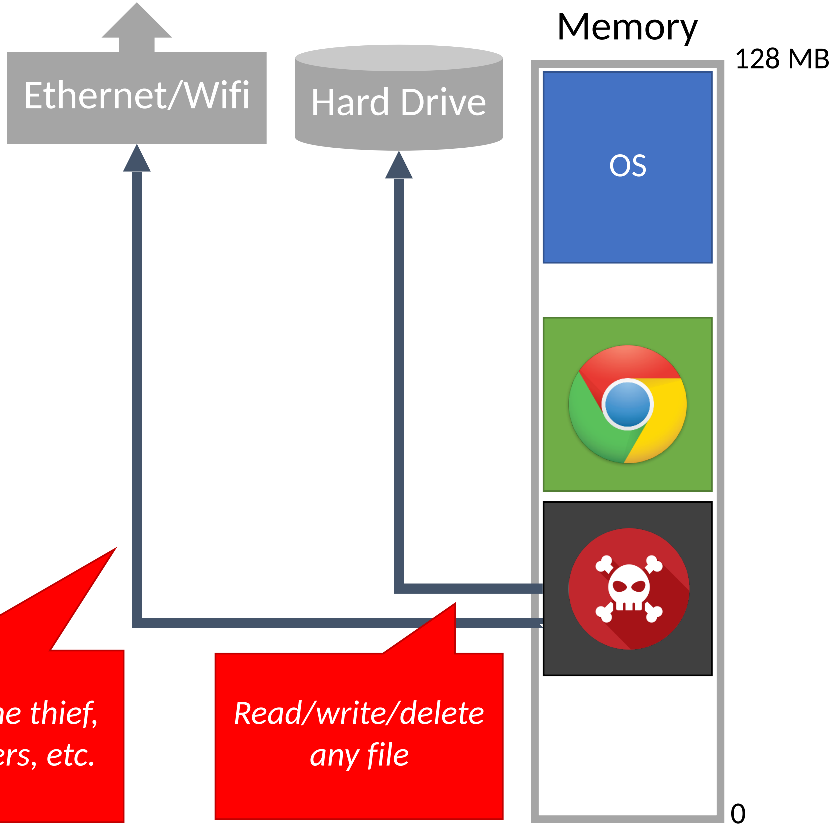
Access control is enforced by the OS, but OS APIs can be bypassed



Device Unsafety

Problem: any process can access any hardware device directly

Access control is enforced by the OS, but OS APIs can be bypassed



Review

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

ISOLATION