

CSCI 4907/6545 Software Security

Fall 2025

Instructor: Jie Zhou

Department of Computer Science

George Washington University



Slides materials are partially credited to Gang Tan of PSU and Mathias Payer of EPFL.

Why does software security matter?


Get the latest software updates from Apple

Keeping your software up to date is one of the most important things you can do to maintain your Apple product's security.

- The latest version of iOS and iPadOS is 18.6.2. Learn how to [update the software on your iPhone or iPad](#).
- The latest version of macOS is 15.6.1. Learn how to [update the software on your Mac](#) and how to allow important [background updates](#).
- The latest version of tvOS is 18.6. Learn how to [update the software on your Apple TV](#).
- The latest version of watchOS is 11.6.1. Learn how to [update the software on your Apple Watch](#).
- The latest version of visionOS is 2.6. Learn how to [update the software on your Apple Vision Pro](#).

Note that after a software update is installed for iOS, iPadOS, tvOS, watchOS, and visionOS, it cannot be downgraded to the previous version.

Apple security updates and Rapid Security Responses

Name and information link	Available for	Release date
iOS 18.6.2 and iPadOS 18.6.2 	iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later	20 Aug 2025
iPadOS 17.7.10	iPad Pro 12.9-inch 2nd generation, iPad Pro 10.5-inch, and iPad 6th generation	20 Aug 2025
macOS Sequoia 15.6.1	macOS Sequoia	20 Aug 2025
macOS Sonoma 14.7.8	macOS Sonoma	20 Aug 2025
macOS Ventura 13.7.8	macOS Ventura	20 Aug 2025
iOS 18.6.1 <small>This update has no published CVE entries.</small>	iPhone XS and later	14 Aug 2025
watchOS 11.6.1 <small>This update has no published CVE entries.</small>	Apple Watch Series 6 and later	14 Aug 2025
Safari 18.6	macOS Ventura and macOS Sonoma	30 Jul 2025
iOS 18.6 and iPadOS 18.6	iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad	29 Jul 2025

iOS 18.6.2 and iPadOS 18.6.2

Released August 20, 2025

ImageIO

Available for: iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later

Impact: Processing a malicious image file may result in memory corruption. Apple is aware of a report that this issue may have been exploited in an extremely sophisticated attack against specific targeted individuals.

Description: An out-of-bounds write issue was addressed with improved bounds checking.

CVE-2025-43300: Apple


Get the latest software updates from Apple

Keeping your software up to date is one of the most important things you can do to maintain your Apple product's security.

- The latest version of iOS and iPadOS is 18.6.2. Learn how to [update the software on your iPhone or iPad](#).
- The latest version of macOS is 15.6.1. Learn how to [update the software on your Mac](#) and how to allow important [background updates](#).
- The latest version of tvOS is 18.6. Learn how to [update the software on your Apple TV](#).
- The latest version of watchOS is 11.6.1. Learn how to [update the software on your Apple Watch](#).
- The latest version of visionOS is 2.6. Learn how to [update the software on your Apple Vision Pro](#).

Note that after a software update is installed for iOS, iPadOS, tvOS, watchOS, and visionOS, it cannot be downgraded to the previous version.

Apple security updates and Rapid Security Responses

Name and information link	Available for	Release date
iOS 18.6.2 and iPadOS 18.6.2	iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later	20 Aug 2025
iPadOS 17.7.10	iPad Pro 12.9-inch 2nd generation, iPad Pro 10.5-inch, and iPad 6th generation	20 Aug 2025
macOS Sequoia 15.6.1	macOS Sequoia	20 Aug 2025
macOS Sonoma 14.7.8	macOS Sonoma	20 Aug 2025
macOS Ventura 13.7.8	macOS Ventura	20 Aug 2025
iOS 18.6.1 <div>This update has no published CVE entries.</div>	iPhone XS and later	14 Aug 2025
watchOS 11.6.1 <div>This update has no published CVE entries.</div>	Apple Watch Series 6 and later	14 Aug 2025
Safari 18.6 	macOS Ventura and macOS Sonoma	30 Jul 2025
iOS 18.6 and iPadOS 18.6	iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad	29 Jul 2025

Safari 18.6

Released July 30, 2025

libxml2

Available for: macOS Ventura and macOS Sonoma

Impact: Processing a file may lead to memory corruption

Description: This is a vulnerability in open source code and Apple Software is among the affected projects. The CVE-ID was assigned by a third party. Learn more about the issue and CVE-ID at [cve.org](#).

CVE-2025-7425: Sergei Glazunov of Google Project Zero

libxslt

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may lead to memory corruption

Description: This is a vulnerability in open source code and Apple Software is among the affected projects. The CVE-ID was assigned by a third party. Learn more about the issue and CVE-ID at [cve.org](#).

CVE-2025-7424: Ivan Fratric of Google Project Zero

Safari

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may lead to an unexpected Safari crash

Description: A logic issue was addressed with improved checks.

CVE-2025-24188: Andreas Jaegersberger & Ro Achterberg of Nosebeard Labs

WebKit

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may lead to universal cross site scripting

Description: This issue was addressed through improved state management.

WebKit Bugzilla: 285927

CVE-2025-43229: Martin Bajanik of Fingerprint, Ammar Askar

WebKit

Available for: macOS Ventura and macOS Sonoma

Impact: Visiting a malicious website may lead to address bar spoofing

Description: The issue was addressed with improved UI.

WebKit Bugzilla: 294374

CVE-2025-43228: Jaydev Ahire

WebKit

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may disclose sensitive user information

Description: This issue was addressed through improved state management.

WebKit Bugzilla: 292888

CVE-2025-43227: Gilad Moav

WebKit

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may lead to memory corruption

Description: The issue was addressed with improved memory handling.

WebKit Bugzilla: 291742

CVE-2025-31278: Yuhao Hu, Yan Kang, Chenggang Wu, and Xiaojie Wei

WebKit Bugzilla: 291745

CVE-2025-31277: Yuhao Hu, Yan Kang, Chenggang Wu, and Xiaojie Wei

WebKit Bugzilla: 293579

CVE-2025-31273: Yuhao Hu, Yan Kang, Chenggang Wu, and Xiaojie Wei

WebKit

Available for: macOS Ventura and macOS Sonoma

Impact: A download's origin may be incorrectly associated

Description: A logic issue was addressed with improved checks.

WebKit Bugzilla: 293994

CVE-2025-43240: Syarif Muhammad Sajjad

Safari 18.6

Released July 30, 2025

libxml2

Available for: macOS Ventura and macOS Sonoma

Impact: Processing a file may lead to memory corruption

Description: This is a vulnerability in open source code and Apple Software is among the affected projects. The CVE-ID was assigned by a third party. Learn more about the issue and CVE-ID at cve.org.

CVE-2025-7425: Sergei Glazunov of Google Project Zero

libxslt

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may lead to memory corruption

Description: This is a vulnerability in open source code and Apple Software is among the affected projects. The CVE-ID was assigned by a third party. Learn more about the issue and CVE-ID at cve.org.

CVE-2025-7424: Ivan Fratric of Google Project Zero

Safari

Available for: macOS Ventura and macOS Sonoma

Impact: Processing maliciously crafted web content may lead to an unexpected Safari crash

Description: A logic issue was addressed with improved checks.

How about Android?

source

Docs

Android Code Search

Search

What's New?

Getting Started

Security

Core Topics

Compatibility

Android Devices

Automotive

Reference

Filter

Overview

Security overview

Android Security Bulletins

Bulletins home

Overview

2025 bulletins

August

July

June

May

April

March

February

January

Android 16

2024 bulletins

2023 bulletins

2022 bulletins

2021 bulletins

2020 bulletins

2019 bulletins

2018 bulletins

2017 bulletins

2016 bulletins

2015 bulletins

Pixel/Nexus bulletins

2025-08-01 security patch level vulnerability details

In the sections below, we provide details for each of the security vulnerabilities that apply to the 2025-08-01 patch level. Vulnerabilities are grouped under the component they affect. Issues are described in the tables below and include CVE ID, associated references, [type of vulnerability](#), [severity](#), and updated AOSP versions (where applicable). When available, we link the public change that addressed the issue to the bug ID, like the AOSP change list. When multiple changes relate to a single bug, additional references are linked to numbers following the bug ID. Devices with Android 10 and later may receive security updates as well as [Google Play system updates](#).

Framework

The most severe vulnerability in this section could lead to local escalation of privilege with no additional execution privileges needed. User interaction is needed for exploitation.

CVE	References	Type	Severity	Updated AOSP versions
CVE-2025-22441	A-376028556	EoP	High	13, 14, 15
CVE-2025-48533	A-383131643	EoP	High	13, 14, 15, 16

System

The vulnerability in this section could lead to remote code execution in combination with other bugs, with no additional execution privileges needed. User interaction is not needed for exploitation.

CVE	References	Type	Severity	Updated AOSP versions
CVE-2025-48530	A-419563680	RCE	Critical	16

Google Play system updates

There are no security issues addressed in Google Play system updates (Project Mainline) this month.

2025-08-05 security patch level vulnerability details

Software security threats are ubiquitous!

Outline of Today

- **Why should we study/research software security?**
- **Course logistics**
- **Principles of Building Secure Software Systems**

Outline of Today's Lecture

- **Why should we study/research software security?**
- Course logistics
- Principles of Building Secure Software Systems

Fact 1: Software Has Bugs

BLACK HAT

Windows Update Flaws Allow Undetectable Downgrade Attacks

Researcher showcases hack against Microsoft Windows Update architecture, turning fixed vulnerabilities into zero-days.



By [Ryan Naraine](#)
August 7, 2024



LAS VEGAS — SafeBreach Labs researcher Alon [redacted] major gaps in Microsoft’s Windows Update archi[redacted] hackers can launch software downgrade attacks meaningless on any Windows machine in the wo[redacted]

ars TECHNICA

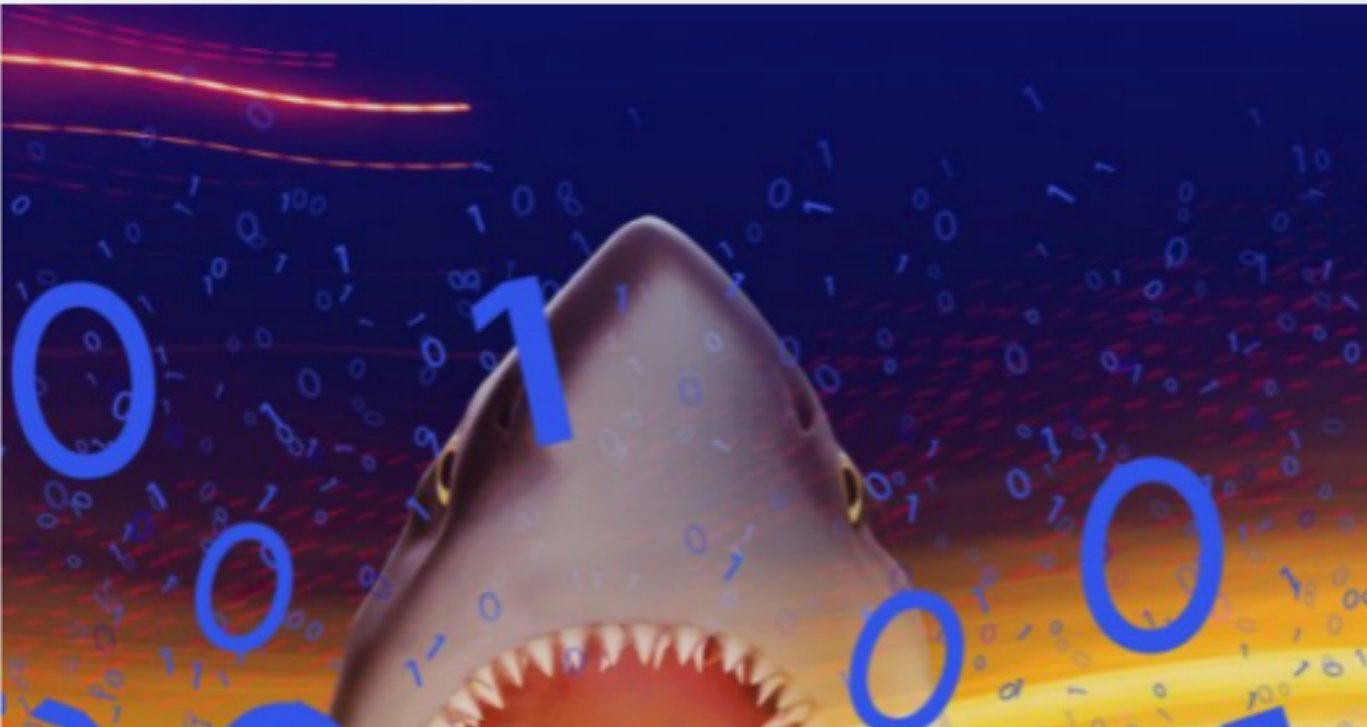
BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

DIRTY PIPE —

Linux has been bitten by its most high-severity vulnerability in years

Dirty Pipe has the potential to smudge people using Linux and Linux derivatives.

DAN GOODIN - 3/7/2022, 6:39 PM





The Mac Security Blog

Search the Blog

Share

PRINT

APPLE

Apple still leaving critical vulnerabilities unpatched in macOS Sonoma

Posted on August 1st, 2024 by [Joshua Long](#)



As we first [noted](#) in November 2023, macOS Sonoma contains some very outdated open-source software components. (Free/libre open-source software is commonly abbreviated as FOSS or FLOSS.) This outdated software puts Mac users at serious risk. We’ve reached out to Apple multiple times about this, and Apple still hasn’t responded. Here’s what we know.

Popular Stories

Porn blackmail "sextortion" emails: Have you been hacked? A new scam

How to Install macOS Sonoma (or Sequoia) on Unsupported Macs, for Security Improvements

The Complete Guide to Apple Watch Bands in 2024: Sizing, Styles, and More

How to run Windows 11 for FREE on a Mac with an M1, M2, or M3 chip

Follow Intego

Recommended

SECURITY & PRIVACY

Definition: Software Bug



Wikipedia: *“A software bug is a bug in computer software.”*

Wikipedia: *“In engineering, a bug is a design defect in an engineered system that causes an undesired result.”*

https://en.wikipedia.org/wiki/Software_bug

[https://en.wikipedia.org/wiki/Bug_\(engineering\)](https://en.wikipedia.org/wiki/Bug_(engineering))

Definition: Software Bug



Wikipedia: *“A software bug is a bug in computer software.”*

Wikipedia: *“In engineering, a bug is a design **defect** in an engineered system that causes an **undesired result**.”*

https://en.wikipedia.org/wiki/Software_bug

[https://en.wikipedia.org/wiki/Bug_\(engineering\)](https://en.wikipedia.org/wiki/Bug_(engineering))

It is Too Easy to Write Bugs

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[]) {
4     char user_name[32];
5     scanf("%s", user_name);
6     printf("Hello, %s!\n", user_name);
7 }
```



What if user's name is longer than 32 characters?



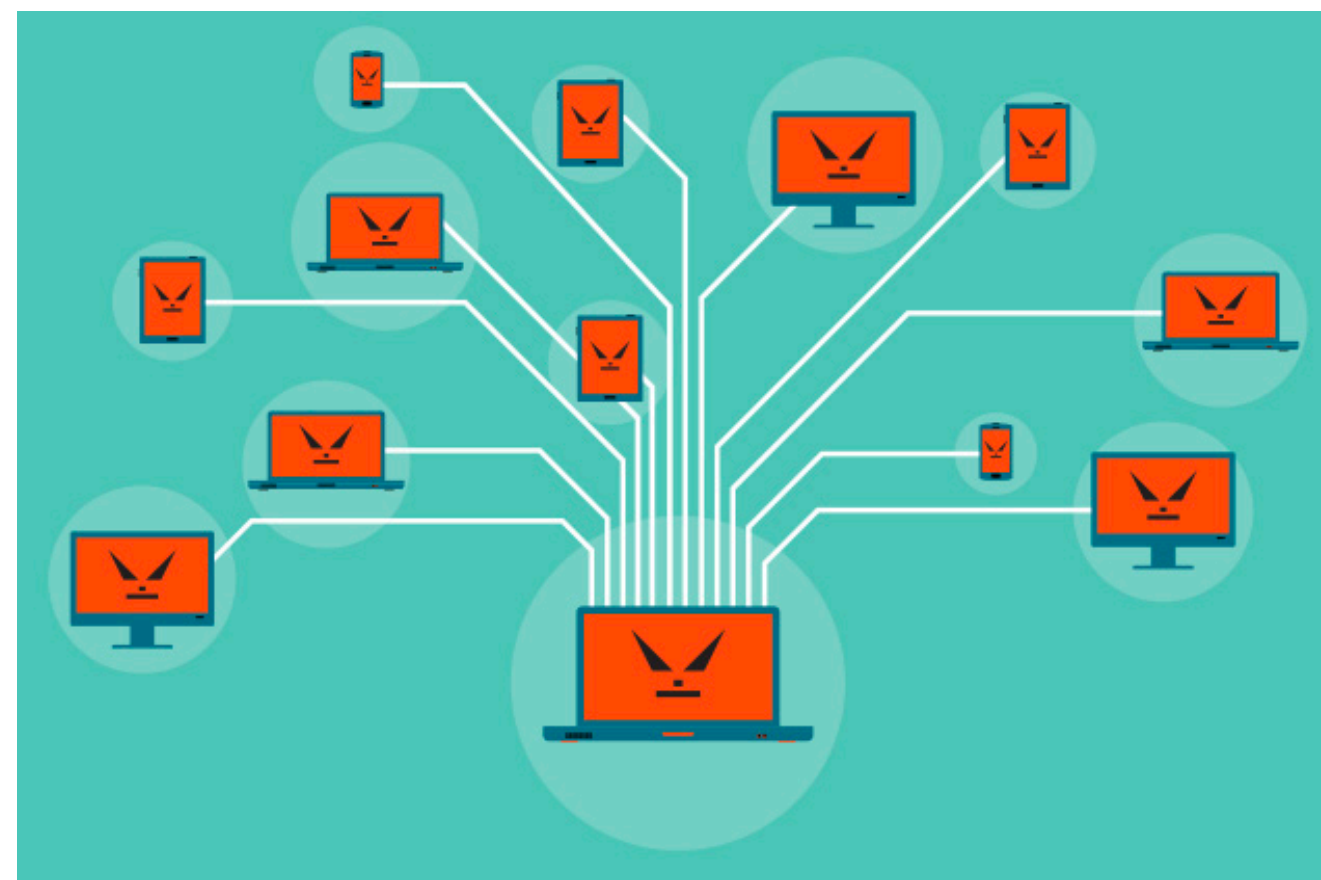
What if the user deliberately/maliciously input something longer than 32 characters?

Fact 2: Many Bugs Are Exploitable (Causing Damage)



Ransomware

e.g. WannaCry



Botnet

e.g. Mirai



Spyware

e.g. Pegasus

Bugs vs. Vulnerabilities



Wikipedia: “A software *bug* is a bug in computer software.”

Wikipedia: “In engineering, a bug is a design **defect** in an engineered system that causes an **undesired result**.”

Bugs vs. Vulnerabilities



Wikipedia: “A software *bug* is a bug in computer software.”

Wikipedia: “In engineering, a bug is a design **defect** in an engineered system that causes an **undesired result**.”



Wikipedia: “Vulnerabilities are **flaws** in a computer system that weaken the overall security of the system.”

Vulnerabilities -> Exploitable Bugs

The Morris Worm



The Morris Worm

Brought down most of the Internet on November 2nd, 1988.

- Buffer overflow in `fingerd`, injected shellcode and commands
- Debug mode in `sendmail` to execute arbitrary commands
- Dictionary attack with frequently used usernames/passwords

 **Buggy worm:** A type of malicious program (malware) that *self-replicates* and spreads across networks to affect as many machines as possible.

OpenSSL Heartbleed Vulnerability

CNN

Business

Markets

Tech

Media

Calculators

Videos

The ‘Heartbleed’ security flaw that affects most of the Internet

Heather Kelly, CNN

🕒 2 minute read · Updated 5:11 PM EDT, Wed April 9, 2014

f

X

✉

🔗



Internet security flaw may affect YOU

🗨 Video Ad Feedback

OpenSSL Heartbleed Vulnerability

- A programming bug in the OpenSSL implementation's HeartBeat mechanism
 - Used in numerous web servers
- The bug: lack of input validation
 - An attacker can send in a HeartBeat request, which contains a message and a length.
 - The length should correspond to the message's size.
- Attacker can
 - Send in a request with a large length, greater than the message's size
 - Hence the attacker can get a slice of data from server's main memory -- one that is up to 64KB in length.
 - That memory could contain the private key of the server (or users' passwords).

Fix for Heartbleed

The fix is equally simple. Just add a bounds check:

```
+    /* Read type and payload length first */
+    if (1 + 2 + 16 > s->s3->rrec.length)
+        return 0; /* silently discard */
+    hbtype = *p++;
+    n2s(p, payload);
+    if (1 + 2 + payload + 16 > s->s3->rrec.length)
+        return 0; /* silently discard per RFC 6520 sec. 4 */
+    pl = p;
```

Fact 3: Software is Incredibly Complex

- **Complexity**

- Software becomes more and more complicated.
- Size is measured in terms of millions lines of code.

- **Connectivity**

- The Internet makes it possible for attackers to exploit software remotely.

- **Extensibility**

- Programs written by untrusted parties

Fact 3: Software is Incredibly Complex

- **Complexity**

- Software becomes more and more complicated.
- Size is measured in terms of millions lines of code.

- **Connectivity**

- The Internet makes it possible for attackers to exploit software remotely.

- **Extensibility**

- Programs written by untrusted parties

Fact 3: Software is Incredibly Complex



Margaret Hamilton with code
for Apollo Guidance Computer
(NASA, '69)



47 million lines!
50 lines/page,
 $0.1 \text{ mm/page} = 94 \text{ m!}$



20.6m



Fact 3: Software is Incredibly Complex

- Complexity

- Software becomes more and more complicated.
- Size is measured in terms of millions lines of code.

- **Connectivity**

- The Internet makes it possible for attackers to exploit software remotely.

- Extensibility

- Programs written by untrusted parties

Connectivity

- It's easy to secure your smartphone if it's off the internet.
 - Attackers cannot get to your phone remotely.
 - You cannot browse malicious webpages or download malware.
- Reality: almost every device is on the internet.
 - Connectivity enables many things
 - But attackers also like it: it allows the possibility of remotely hacking any device on the internet.



Fact 3: Software is Incredibly Complex

- Complexity

- Software becomes more and more complicated.
- Size is measured in terms of millions lines of code.

- Connectivity

- The Internet makes it possible for attackers to exploit software remotely.

- **Extensibility**

- Programs written by untrusted parties

Extensibility

- Software systems are not closed.
- Smartphone app market: allow users to extend the functionality of their phones
- However
 - We don't know who wrote those apps?
 - What if an app steal our credit card info or track our locations?
- Like connectivity, hackers also like extensible systems.
 - Giving them an opportunity to inject malicious code

XZ Utils backdoor

🌐 11 languages ▾

Article [Talk](#)

[Read](#) [Edit](#) [View history](#) [Tools](#) ▾

From Wikipedia, the free encyclopedia

In February 2024, a malicious [backdoor](#) was introduced to the Linux build of the [xz](#) utility within the [liblzma](#) library in versions 5.6.0 and 5.6.1 by an account using the name "Jia Tan".^[b]^[4] ~~The backdoor gives an attacker who possesses a specific [Ed448](#) private key~~ [remote code execution](#) capabilities on the affected Linux system. The issue has been given the [Common vulnerabilities and Exposures](#) number [CVE-2024-3094](#)^[5] and has been assigned a [CVSS](#) score of 10.0, the highest possible score.^[5]

While xz is commonly present in most [Linux distributions](#), at the time of discovery the backdoored version had not yet been widely deployed to [production](#) systems, but was present in development versions of major distributions.^[6] The backdoor was discovered by the software developer Andres Freund, who announced his findings on 29 March 2024.^[7]

Background [\[edit \]](#)

XZ Utils backdoor



Previous XZ logo contributed by Jia Tan

CVE identifier(s)	CVE-2024-3094 ^[5]
Date discovered	at or before 27 March 2024; 8 months ago ^[1] ^[2]
Date of public disclosure	29 March 2024; 8 months ago

It is Too Easy to Write Bugs

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[]) {
4     char user_name[32];
5     scanf("%s", user_name);
6     printf("Hello, %s!\n", user_name);
7 }
```



What if user's name is longer than 32 characters?



What if the user deliberately/maliciously input something than 32 characters?

Fact 3: Software is Incredibly Complex

- **Complexity**

- Software becomes more and more complicated.
- Size is measured in terms of millions lines of code.

- **Connectivity**

- The Internet makes it possible for attackers to exploit software remotely.

- **Extensibility**

- Programs written by untrusted parties

Outline of Today

- Why should we study/research software security?
- **Course logistics**
- Principles of Building Secure Software Systems

Prerequisites

Courses:

- Computer Organizations/Architecture/Systems
- Systems Programming

What it really means:

- Comfortable with C and assembly language
 - Almost no code writing, but some deep code reading/understanding.
- Enjoy working on *low-level* software



This is a very advanced course focusing on low-level systems stuff.

What Are You Supposed to Learn in this Course?

- Understand several fundamental software security threats
 - Causes and Exploitations
- Understand common countermeasures against the threats
- Develop a *secure programming* mindset
 - Build security in

Topics

- Memory Safety
 - Root causes of and exploitations against bugs
 - Deployed and experimental defenses
 - Testing
 - Mitigations
 - Prevention
 - Secure languages
- Type Safety
- Least Privilege Principle

Memory Safety Vulnerabilities Are a Major Threat

Home / Innovation / Security

Microsoft: 70 percent of all security bugs are memory safety issues

Percentage of memory safety issues has been hovering at 70 percent for the past 12 years.



Written by **Catalin Cimpanu**, Contributor on Feb. 11, 2019

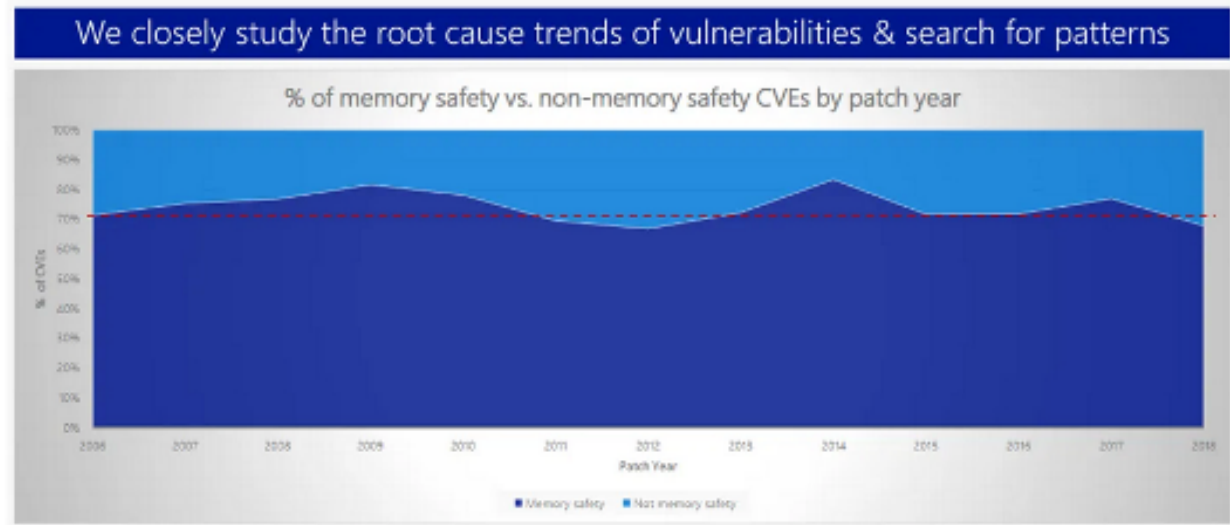


Image: Matt Miller

/ related



Without Dennis Ritchie, there would be no Jobs

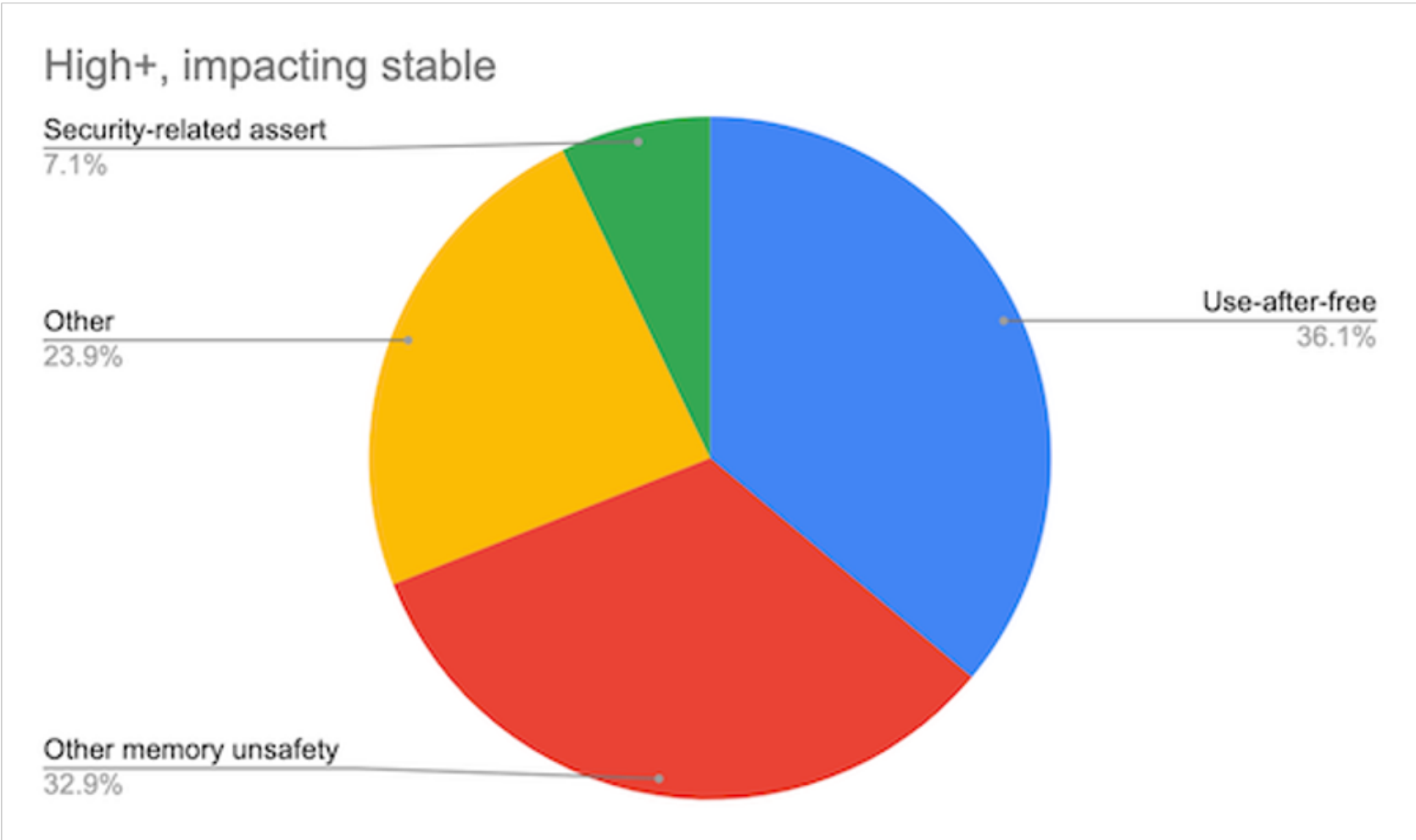


How do the top VPNs compare? Plus, should you try a free VPN?



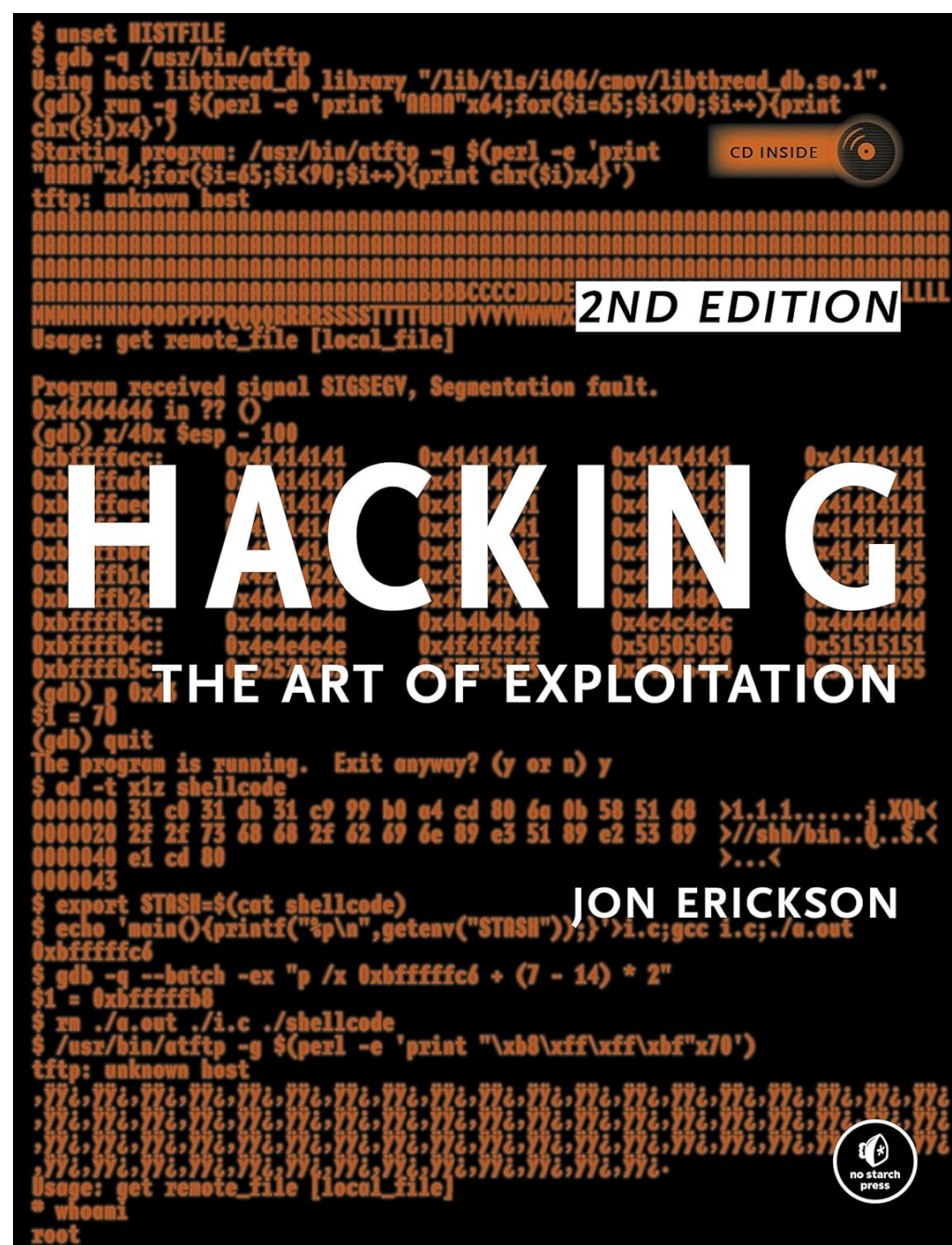
Using a VPN to torrent is a

Around 70% of our high severity security bugs are memory unsafety problems (that is, mistakes with C/C++ pointers). Half of those are use-after-free bugs.



Reading Materials

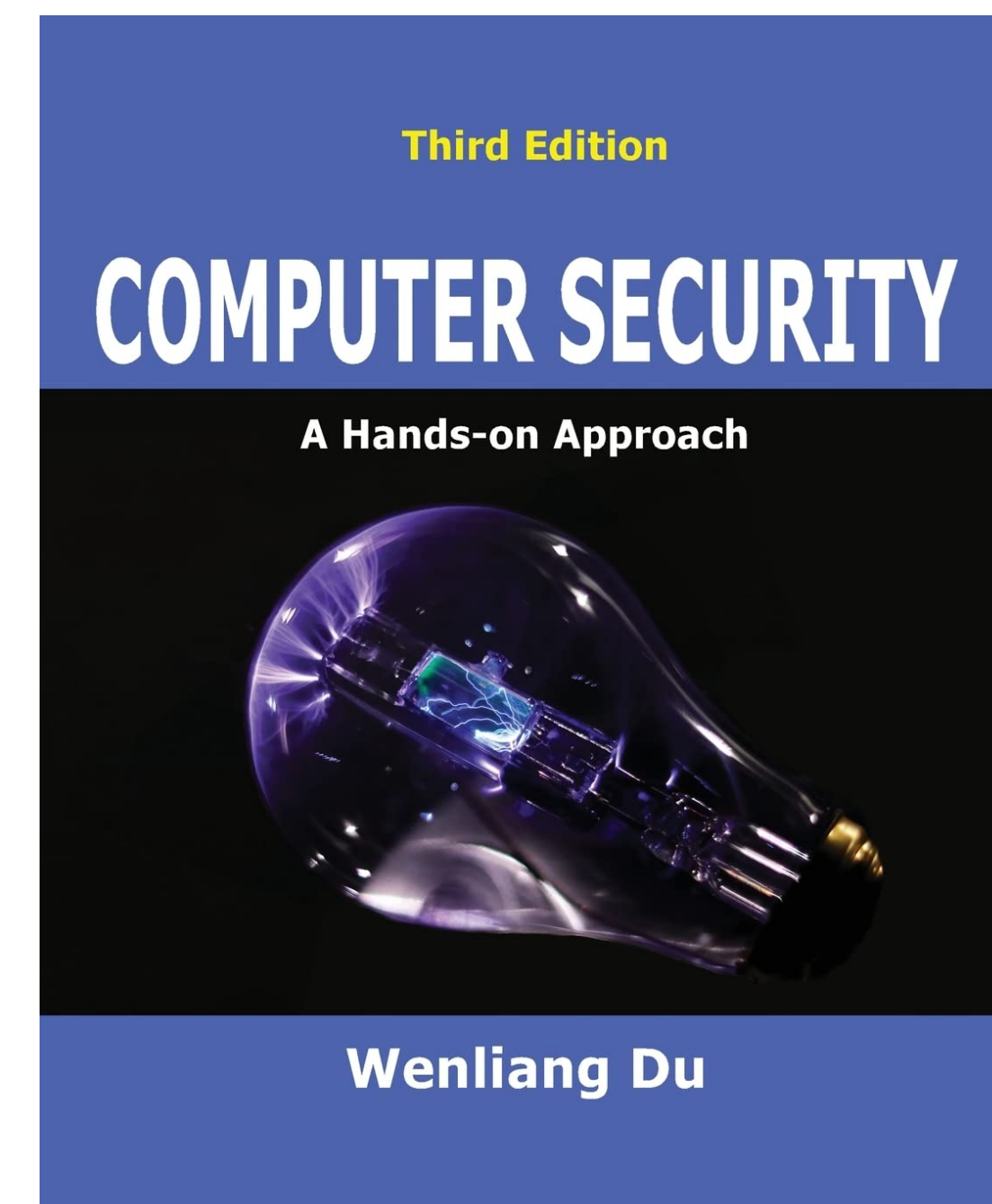
- *Software Security: Principles, Policies, and Protection*, by Mathias Payer
 - Broadly and briefly introduces software security, esp. on low-level software



Low-level hacking techniques



Software security from software engineering's perspective



Hands-on guidance on practicing software security

Lecture Organization

- A quick review of the previous lecture
- A ~10-min in-class break (2.5 hr is a lot!)
- One or two topics

Assignments

- Solve three machine problems (45%)
 - Find and exploit vulnerabilities in buggy programs.
- Read and understand research papers (40%)
 - Summarize and answer questions about papers
- Students-led paper presentation and discussion (10%)
- Class participation (5% + 3% extra credit)
 - Ask and answer questions!
- No Exams

Why Read and Discuss Research Papers?



- The software security area is evolving fast.
 - New defenses and offenses come up everyday.
- “Security is as strong as the weakest link.”
 - Learn how a seemingly minor bug can be deadly.
- Mutual review/critiques is important.

Assignments

- Solve three machine problems (40%)
 - Find and exploit vulnerabilities in buggy programs.
- Read and understand research papers (45%)
 - Summarize and answer questions about papers
- Students-led paper presentation and discussion (10%)
- Class participation (5% + 3% extra credit)
 - Ask and answer questions!
- No Exams

Assignments

- Solve three machine problems (40%)
 - Find and exploit vulnerabilities in buggy programs.
- Read and understand research papers (45%)
 - Summarize and answer questions about papers
- Students-led paper presentation and discussion (10%)
- Class participation (5% + 3% extra credit)
 - Ask and answer questions!
- No Exams

Lateness Policy

- Request for extension must be submitted *before* deadline.
- No late submissions will be accepted.

Use of Generative AI



- Permitted for learning purposes (BE CAREFUL!)
- ***Prohibited*** from generating contents for assignments

Action Items

- Fill out the Google Form if you have not (see the announcement in Blackboard)
- Read the assigned readings for this and next lectures

Tips on Better Learning

- Read the assigned readings *before* each lecture
- Get your hands dirty
- Review regularly what you have learned
- Ask questions!

Course Website

<https://jiezhoucs.github.io/courses/csci-6545/fall-2025/>

- Office hours, syllabus, schedule, etc.
- Assigned readings for lectures on the Schedule
- Firefox on Windows may not render the website correctly!
Use Chrome (or others) if you encounter display issues.

Any Questions?

Take a break!

Outline of Today

- Why should we study/research software security?
- Course logistics
- **Principles of Building Secure Software Systems**

Learning Goals

- Understand security goals
- Know the security triad (CIA) and its limitations
- Reason based on TCB and threat models
- Core concepts: isolation, least privileges, fault compartmentalization
- How abstractions enable reduction of complexity

Bugs vs. Vulnerabilities



Wikipedia: “A software *bug* is a bug in computer software.”

Wikipedia: “In engineering, a bug is a design **defect** in an engineered system that causes an **undesired result**.”



Wikipedia: “Vulnerabilities are **flaws** in a computer system that weaken the overall security of the system.”

Vulnerabilities -> Exploitable Bugs

Definition: Software Security



Allow *intended* use of software and prevent *unintended* use that may cause harm

Goal: Prevent information “mishaps”, but don’t stop good things from happening

- Good things include functionality (e.g. legal information access).
- Tradeoff between functionality and security is the key.



E-Voting

Good things: convenience of voting; fast tallying; voting for the disabled; ...

The convenience comes with risks

- Buggy voting software/hardware
- Changed e-voting software by insiders
- ...

The Sad Reality

- People are obsessed with providing more functionalities.
 - Security is secondary.
 - Security is an after-thought.
 - “We’ll write the software with the required functionalities, then our security team will make it secure.”
- Security perspective: integrate security design into the system design process
 - “*Build Security In*”
- Managing the trade-off between functionality and security from the beginning

Definition: Software Security



Allow *intended* use of software and prevent *unintended* use that may cause harm

Goal: Prevent information “mishaps”, but don’t stop good things from happening

- Good things include functionality or legal information access.
- Tradeoff between functionality and security is the key.



- What is the *intended* behavior?
- How should we define *boundaries*?

Definition: Software Security



Allow *intended* use of software and prevent *unintended* use that may cause harm

Goal: Prevent information “mishaps”, but don’t stop good things from happening

- Good things include functionality or legal information access.
- Tradeoff between functionality and security is the key.

CIA Security Triad (+1)

- **Confidentiality:** An attacker cannot recover protected data.
- **Integrity:** An attacker cannot modify protected data.
- **Availability:** An attacker cannot stop/hinder computation.

Accountability/non-repudiation: Committed changes cannot be undone (as potential fourth fundamental property).



Trust

Fact 3: Software is Incredibly Complex

- Complexity

- Software becomes more and more complicated.
- Size is measured in terms of millions lines of code.

- Connectivity

- The Internet makes it possible for attackers to exploit software remotely.

- Extensibility

- Programs written by untrusted parties



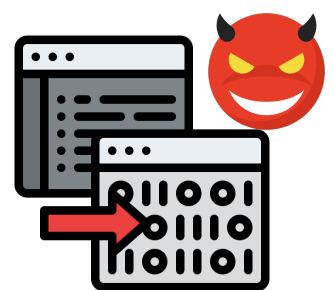
Do you trust computations provided by others?

Thompson: Reflections on Trusting Trust

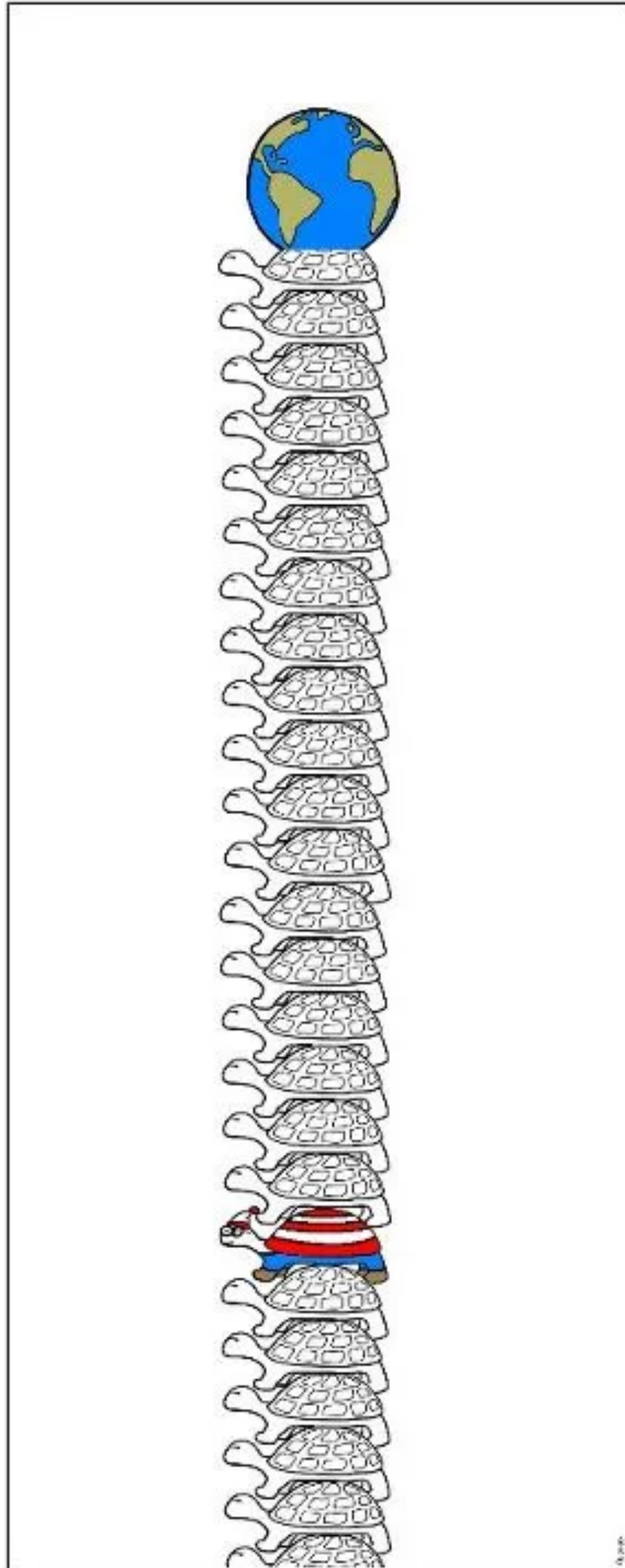


Ken Thompson
Co-creator of the UNIX system
Turing Award Winner

Thompson: Reflections on Trusting Trust

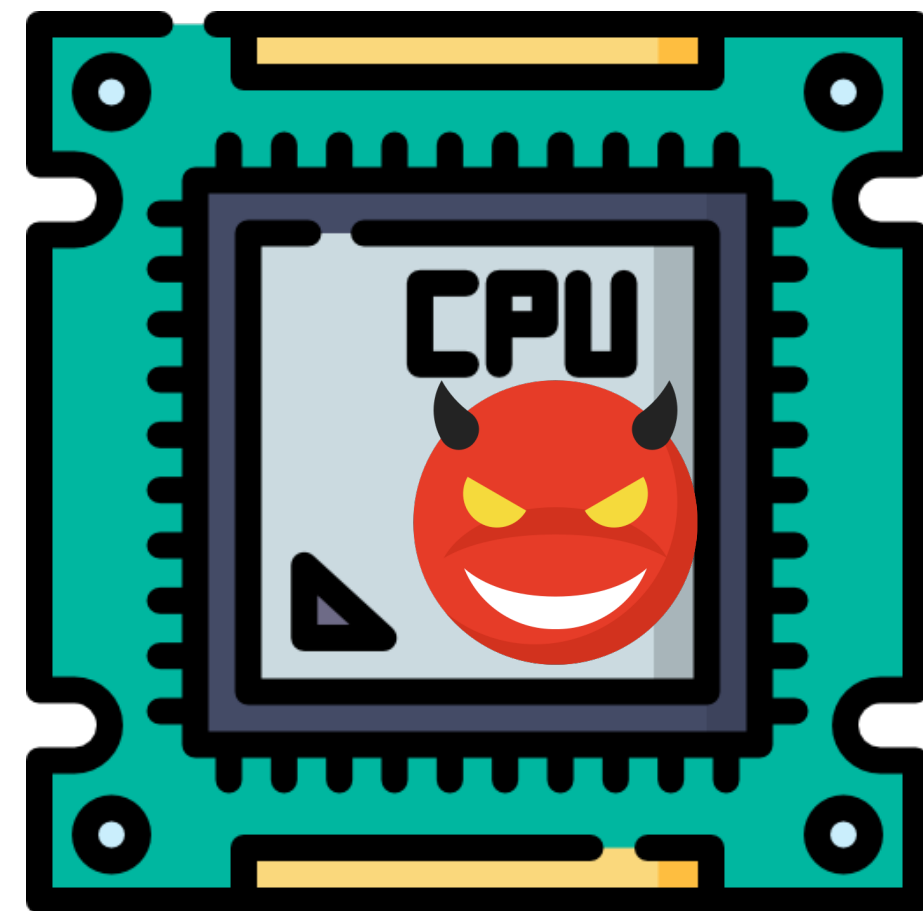
- Ken Thompson's Turing Award lecture describes the importance of making clear what should be trusted.
- Do you trust your compiler? 
 - Thompson described an approach whereby he can generate a compiler that can insert a backdoor.
 - e.g., insert a backdoor when recognizing a login program
- But you can examine the compiler's source code.
 - But, what program compiles the compiler?
 - He puts the malicious code in that program.

Turtles All the Way Down



Takeaway: Thompson states the “obvious” moral that “You cannot trust code that you did not totally create yourself”.

- Creating a basis for trusting is very hard, even today.



But, do you trust your hardware?

Trusted Computing Base (TCB)



A set of trusted hardware, firmware, and software that are critical to the security of a computing system.

- E.g., a conventional e-voting machine: voting software + hardware
- Bugs in the TCB may jeopardize the system's security.
- Components outside of the TCB can misbehave without affecting the security of TCB.



Small or Big TCB?

Trusted Computing Base (TCB)



A set of hardware, firmware, and software that are critical to the security of a computer system.

- Bugs in the TCB may jeopardize the system's security
- E.g., a conventional e-voting machine: voting software + hardware
- Components outside of the TCB can misbehave without affecting the security of TCB.
- In general, a system with a smaller TCB is more trustworthy.
- A lot of security research is about how to move components outside of the TCB (i.e., making the TCB smaller)
 - E.g., Proof-Carrying Code removes the compiler outside of the TCB.

Definition: Threat Model



The abilities and resources of the attacker.

- Threat models enable structured reasoning about the attack surface.
- Awareness of entry points (and associated threats) to break into the target.
- Look at systems from an attacker's perspective:
 - Decompose application: identify structure
 - Determine and rank threats
 - Determine countermeasures and mitigations

Further reading:

https://owasp.org/www-community/Threat_Modeling

Security Analysis



How do you decide if a system is secure?

- What are the assets? (What could an attacker gain?)
- What are the goals? (What drives the attacker?)
- What is the attack surface?

Threat Model: Example of a Safe/Lockbox

You protect your valuables by locking them in a safe.
What is the attack surface?



- In trust land, you don't need to lock your safe.
- An attacker may pick your lock.
- An attacker may use a torch to open your safe.
- An attacker may use advanced technology (x-ray) to open it.
- An attacker may get access (or copy) your key.
- An attacker may steal the whole safe.
- An attacker may replace the safe with a copy.
-

Cost of Security

There is no free lunch. Security incurs overhead.

Security is

- expensive to develop
- expensive to maintain
- may have (high) performance overhead
- may be inconvenient to users

Principles for Building Secure Software Systems

- Isolation
- Least Privilege
- Fault Compartmentalization
- Trust and Correctness

Principle: Isolation



Isolate two components from each other

- One component cannot access data/code of the other component except through a well-defined API.



User-space application may only access the disk through the filesystem API (i.e., the OS prohibits direct block access to raw data). The OS isolates the user-space process from the disk.



Isolation incurs overhead due to switching cost between components.

Principle of Least Privilege




A component has the least privileges needed to function


- Any further removed privilege reduces functionality
- Any added privilege will not increase functionality (according to the specifications)
- This property constraints an attacker in the obtainable privileges.



Rendering in Chromium executes in an encapsulated sandbox where only minimal system calls are allowed.

Principle: Fault Compartmentalization

- 
- Separate individual components into smallest functional entity possible.
 - These units contain faults to individual components.
 - Allows abstraction and permission checks at boundaries.



A chatting app's image processing module and audio processing module are compartmentalized so that bugs in one module will not affect another.

This property builds on least privilege and isolation. Both properties are most effective in combination: many small components that are running and interacting with least privileges.

Principle: Trust and Correctness



Specific components are assumed to be trusted or correct according to a specification.



Formal verification ensures that a component correctly implements a given specification and can therefore be trusted. However,

- It is generally not computationally possible to formally verify arbitrarily complex software.
- Also, the specification may be buggy.

Software and Hardware Abstractions



Abstraction is the act of representing essential features without including the background details or explanations.

- Allow encapsulation of ideas without having to go into implementation details.
 - Require an explicit definition on how to interoperate between layers
-
- In software, an API abstracts the underlying implementation by defining how a library can be used.
 - In operating systems, the system call interface abstracts low level implementations.
 - In hardware, an ISA abstracts the underlying implementation of the instructions into logic and state.

Abstractions: Operating Systems (OS)

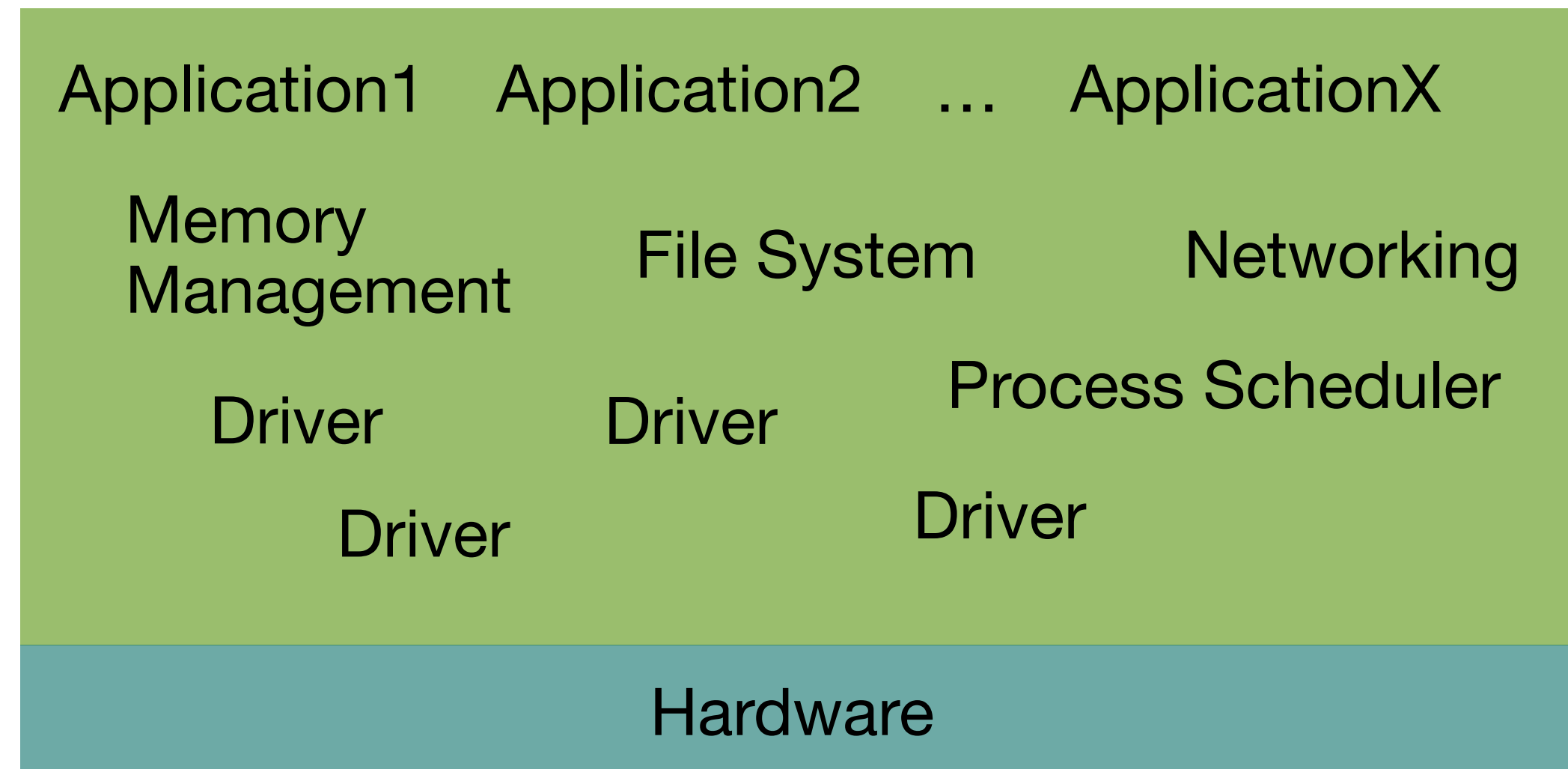
- Provides process abstraction
- Well-defined API to access hardware resources
- Enforces mutual exclusion to resources
- Enforces access permissions for resources
- Restrictions based on user/group/ACL
- Restricts attacker

Abstractions: OS Process Isolation

- Address space: working memory of one process
- Memory protection: protect the memory (code and data such as heap, stack, or globals) of one process from other processes
- Today's systems implement address spaces (virtual memory) through page tables with the help of an Memory Management Unit (MMU).

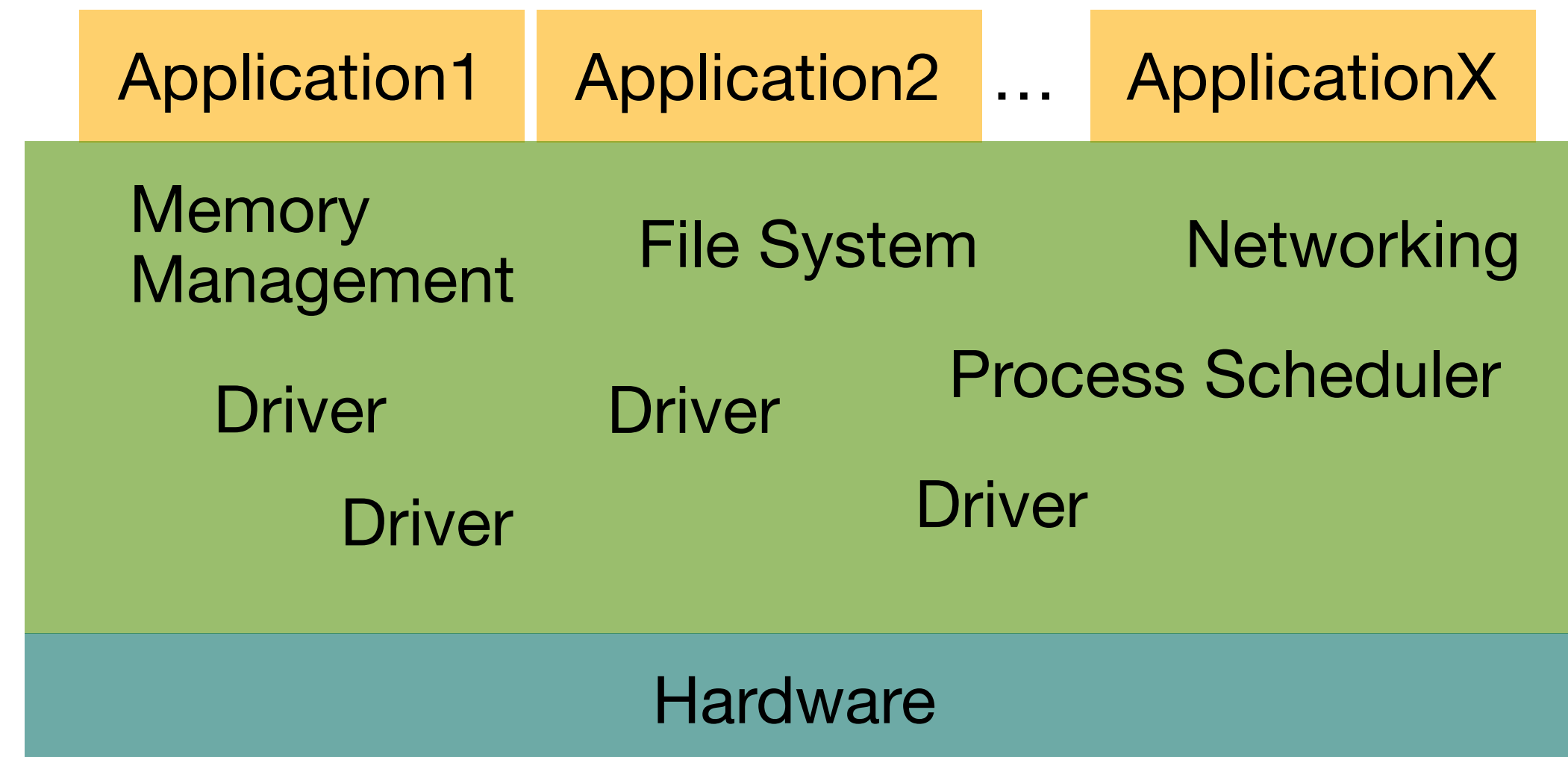
Abstractions: Single-domain OS

- A single layer, no isolation or compartmentalization
- All code runs in the same domain: the application can directly call into drivers.
- High performance, often used in embedded systems



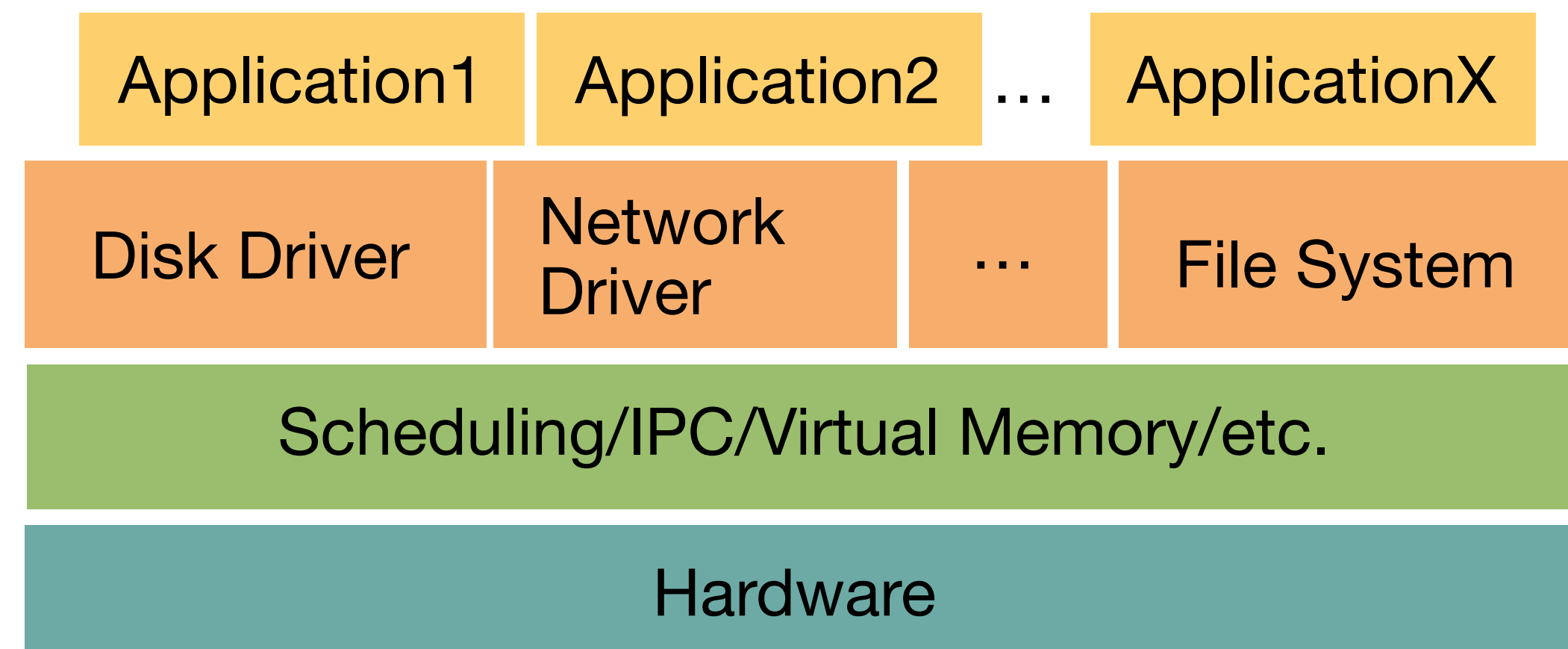
Abstractions: Monolithic OS

- Two layers: the operating system and applications
- The OS manages resources and orchestrates access
- Applications are unprivileged, must request access from the OS
- Linux fully and Windows mostly follow this approach for performance (isolating individual components is expensive)



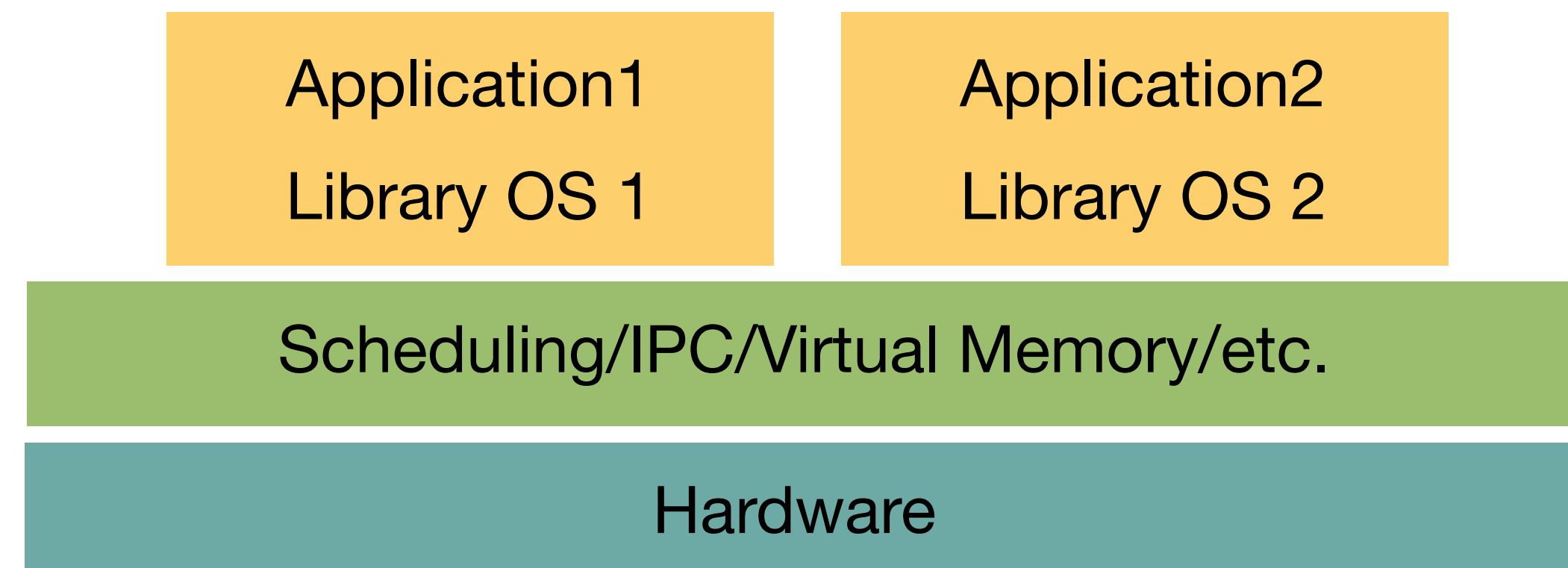
Abstractions: Micro-kernel

- Many layers: each component is a separate process
- Only essential parts are privileged
 - Process abstraction (address spaces)
 - Process management (scheduling)
 - Process communication (IPC)
- Applications request access from different OS processes



Abstractions: Library OS

- Few thin layers; flat structure
- Micro-kernel exposes bare OS services
- Each application brings all necessary OS components



Hardware Abstraction

- Virtual memory through MMU/OS
- Only OS has access to raw physical memory
- DMA for trusted devices
- ISA enforces privilege abstraction (ring 0/3 on x86)

Hardware abstractions are fundamental for performance.

Case Study: Mail Server

- Mail Transfer Agents (MTA) need to do a plethora of tasks:
 - Send/receive data from the network
 - Manage a pool of received/unsent messages
 - Provide access to stored messages for each user

Sendmail uses a typical Unix approach with a large monolithic server and is known for the high complexity and previous security vulnerabilities.

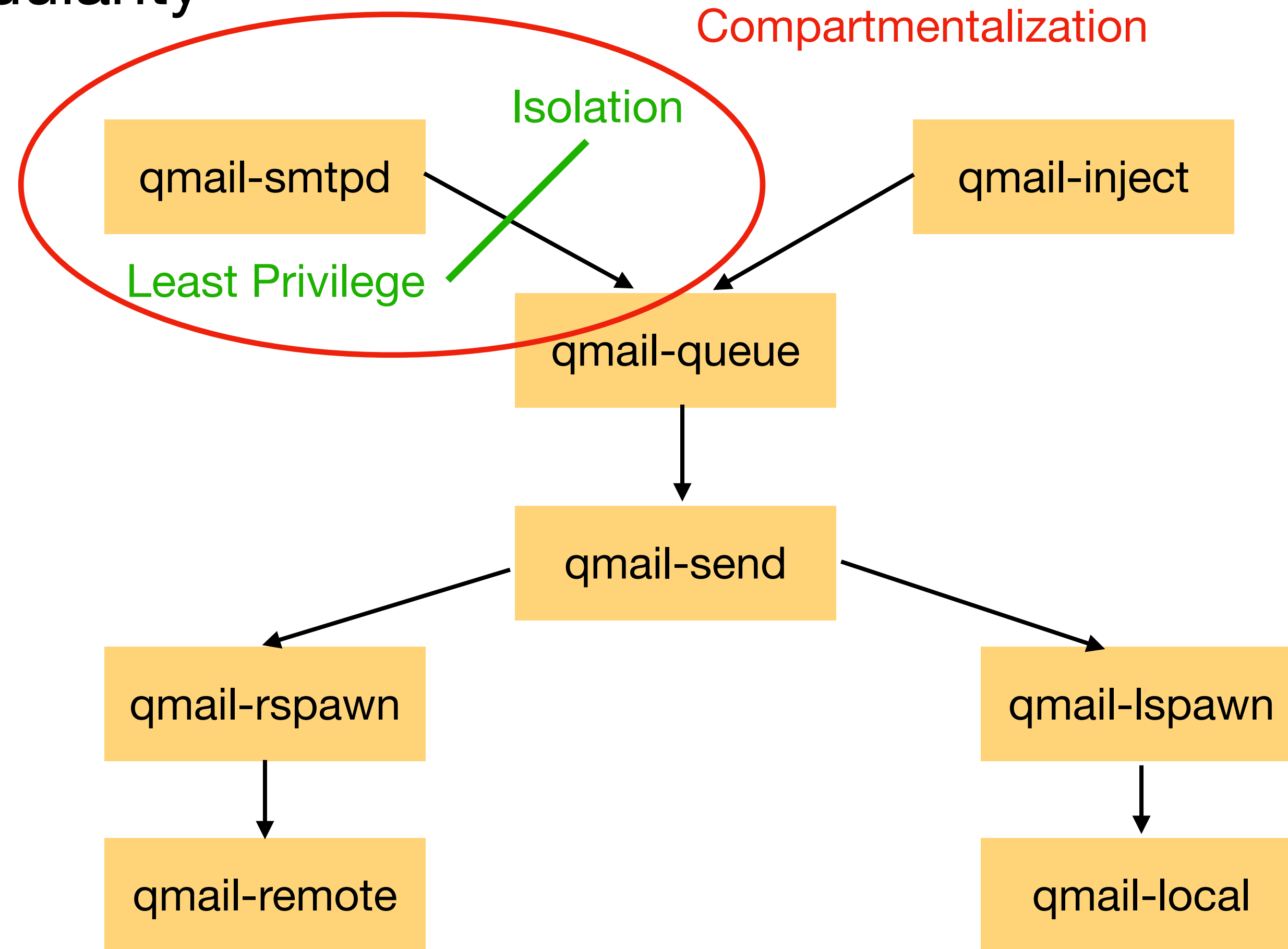


How would you compartmentalize a mail server?

Case Study: Mail Server

qmail: An mail MTA designed with security in mind.

- Key enabler: modularity



Case Study: Mail Server



What can we do to further reduce potential exploits?

- Separate modules run under separate user IDs.

