CSCI 4907/6545 Software Security

Fall 2026

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 26.6.1. Learn how to update the software on your iPhone or iPad.
- The latest version of macOS is 26.6.2. Learn how to update the software on your Mac and how to allow important background updates.
- The latest version of tvOS is 26.6. Learn how to update the software on your Apple TV.
- The latest version of watchOS is 26.6. Learn how to update the software on your Apple Watch.
- The latest version of visionOS is 26.6.1. 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

Name and information link	Available for	Release date
Safari 26.6.1	macOS Sonoma and macOS Sequoia	18 Aug 2026
iOS 26.6.1 and iPadOS 26.6.1	iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later	17 Aug 2026
iOS 18.7.10 and iPadOS 18.7.10	iPhone XS, iPhone XS Max, iPhone XR, iPad 7th generation	17 Aug 2026
macOS Tahoe 26.6.2	macOS Tahoe	17 Aug 2026
visionOS 26.6.1	Details coming soon	17 Aug 2026
macOS Tahoe 26.6.1	macOS Tahoe	06 Aug 2026
macOS Sequoia 15.7.9	macOS Sequoia	06 Aug 2026
macOS Sonoma 14.8.9	macOS Sonoma	06 Aug 2026
iOS 26.6 and iPadOS 26.6	iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and	27 Jul 2026

iOS 26.6.1 and iPadOS 26.6.1

Released August 17, 2026

Audio

Available for: iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later

Impact: An app may be able to leak sensitive user information

Description: A logic issue was addressed with improved checks

CVE-2026-65339: Meta Red Team X - Nik Tsytsarkin

ImageIO

Available for: iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later

Impact: Processing an image may lead to a denial-of-service

Description: The issue was addressed with improved checks.

CVE-2026-65347: Geonha Lee (@leegn4a)

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 26.6.1. Learn how to update the software on your iPhone or iPad.
- The latest version of macOS is 26.6.2. Learn how to update the software on your Mac and how to allow important background updates.
- The latest version of tvOS is 26.6. Learn how to update the software on your Apple TV.
- The latest version of watchOS is 26.6. Learn how to update the software on your Apple Watch.
- The latest version of visionOS is 26.6.1. 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

Name and information link	Available for	Release date
Safari 26.6.1	macOS Sonoma and macOS Sequoia	18 Aug 2026
iOS 26.6.1 and iPadOS 26.6.1	iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later	17 Aug 2026
iOS 18.7.10 and iPadOS 18.7.10	iPhone XS, iPhone XS Max, iPhone XR, iPad 7th generation	17 Aug 2026
macOS Tahoe 26.6.2	macOS Tahoe	17 Aug 2026
visionOS 26.6.1	Details coming soon	17 Aug 2026
macOS Tahoe 26.6.1	macOS Tahoe	06 Aug 2026
macOS Sequoia 15.7.9	macOS Sequoia	06 Aug 2026
macOS Sonoma 14.8.9	macOS Sonoma	06 Aug 2026
iOS 26.6 and iPadOS 26.6	iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and	27 Jul 2026

macOS Tahoe 26.6.2

Released August 17, 2026

Audio

Available for: macOS Tahoe

Impact: An app may be able to leak sensitive user information

Description: A logic issue was addressed with improved checks.

CVE-2026-65339: Meta Red Team X - Nik Tsytsarkin

ImageIO

Available for: macOS Tahoe

Impact: Processing an image may lead to a denial-of-service

Description: The issue was addressed with improved checks.

CVE-2026-65347: Geonha Lee (@leegn4a)

ImageIO

Available for: macOS Tahoe

Impact: Processing an image may lead to arbitrary code execution

Description: An integer overflow was addressed with improved input validation.

CVE-2026-65346: Meta Red Team X - Nik Tsytsarkin

IOGPUFamily

Available for: macOS Tahoe

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

Description: The issue was addressed with improved memory handling.

CVE-2026-64788: f00l (@PPPF00L) and 3ndy1(@_3ndy1) and Minghao Lin@Y1nkoc and 云散花折, Arjanit Isufi

Kernel

Available for: macOS Tahoe

Impact: A remote attacker may be able to cause unexpected system termination

Description: A use after free issue was addressed with improved memory management.

CVE-2026-65343: Drinor Selmanaj (Sentry), Surya Narayan Kushwaha

Kernel

Available for: macOS Tahoe

Impact: An app may be able to cause unexpected system termination or read kernel memory

Description: An out-of-bounds read was addressed with improved input validation.

CVE-2026-65349: an anonymous researcher

Kernel

Available for: macOS Tahoe

Impact: An app may be able to cause unexpected system termination or corrupt kernel memory

Description: The issue was addressed with improved memory handling.

CVE-2026-65330: Bhaswanth Chigurupati, Billy Jheng Bing Jhong and Pan Zhenpeng (@Peterpan0927) of STAR Labs SG Pte. Ltd.

WebKit

Available for: macOS Tahoe

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

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

WebKit Bugzilla: 317632

CVE-2026-64784: Janggoon Lee of Out of Bounds, OpenAI Codex Security - Amy Burnett

WebKit

Available for: macOS Tahoe

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

Description: The issue was addressed with improved memory handling.

WebKit Bugzilla: 313452

CVE-2026-43795: wwwlk

WebKit Bugzilla: 318348

CVE-2026-65338: OpenAI Codex Security - Amy Burnett

WebKit

Available for: macOS Tahoe

ImageIO

Available for: macOS Tahoe

Impact: Processing an image may lead to arbitrary code execution

Description: An integer overflow was addressed with improved input validation.

CVE-2026-65346: Meta Red Team X - Nik Tsytsarkin

IOGPUFamily

Available for: macOS Tahoe

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

Description: The issue was addressed with improved memory handling.

CVE-2026-64788: f00l (@PPPF00L) and 3ndy1(@_3ndy1) and Minghao Lin@Y1nkoc and 云散花折, Arjanit Isufi

Kernel

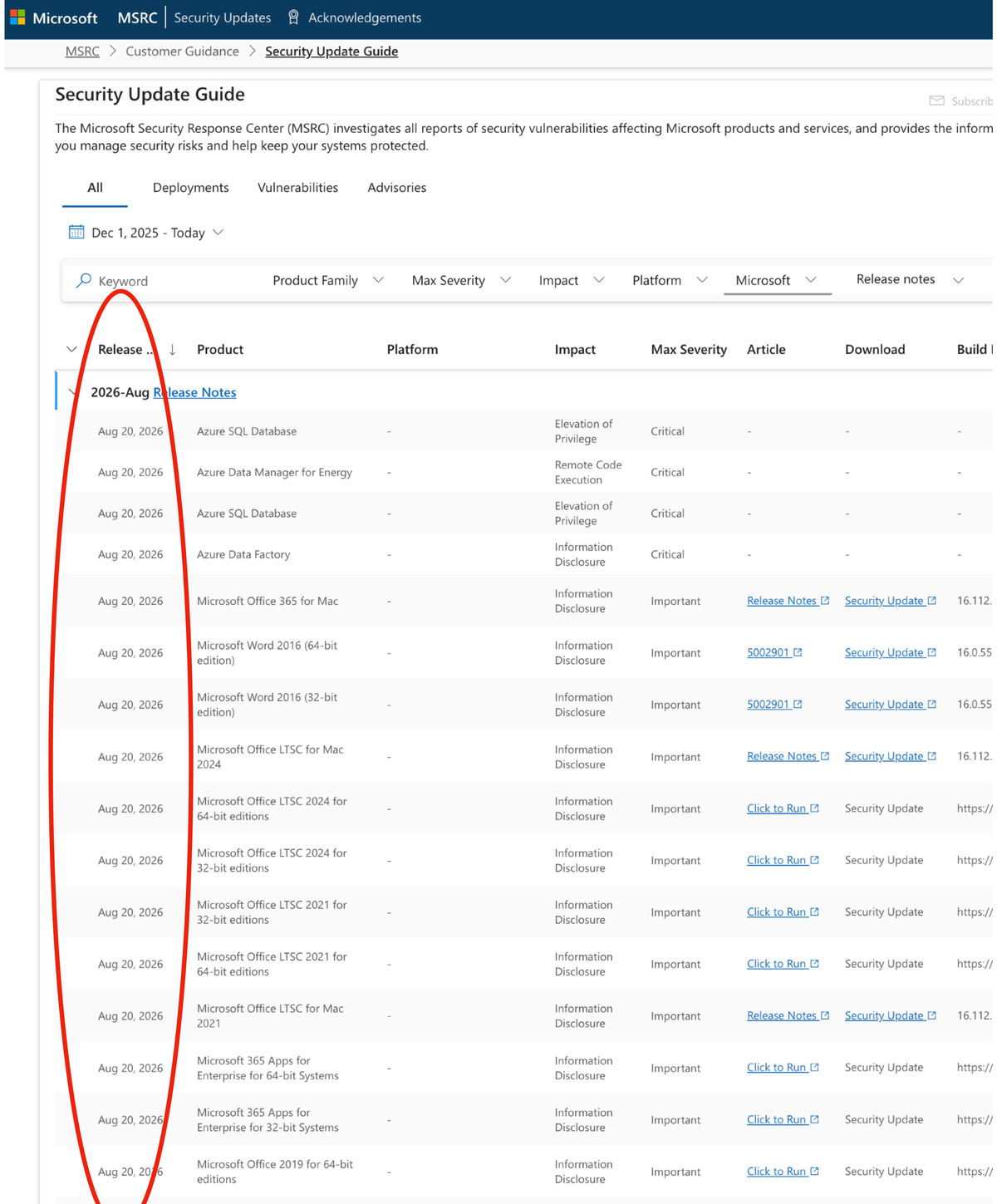
Available for: macOS Tahoe

Impact: A remote attacker may be able to cause unexpected system termination

Description: A use after free issue was addressed with improved memory management.

CVE-2026-65343: Drinor Selmanaj (Sentry), Surya Narayan Kushwaha

How about Windows?


Microsoft MSRC | Security Updates Acknowledgements
MSRC > Customer Guidance > Security Update Guide
Security Update Guide
Subscrib
The Microsoft Security Response Center (MSRC) investigates all reports of security vulnerabilities affecting Microsoft products and services, and provides the inform
you manage security risks and help keep your systems protected.
All
Deployments
Vulnerabilities
Advisories
Dec 1, 2025 - Today
Keyword
Product Family
Max Severity
Impact
Platform
Microsoft
Release notes
Release ..
Product
Platform
Impact
Max Severity
Article
Download
Build
2026-Aug Release Notes
Aug 20, 2026 Azure SQL Database - Elevation of Privilege Critical - - -
Aug 20, 2026 Azure Data Manager for Energy - Remote Code Execution Critical - - -
Aug 20, 2026 Azure SQL Database - Elevation of Privilege Critical - - -
Aug 20, 2026 Azure Data Factory - Information Disclosure Critical - - -
Aug 20, 2026 Microsoft Office 365 for Mac - Information Disclosure Important Release Notes Security Update 16.112.
Aug 20, 2026 Microsoft Word 2016 (64-bit edition) - Information Disclosure Important 5002901 Security Update 16.0.55
Aug 20, 2026 Microsoft Word 2016 (32-bit edition) - Information Disclosure Important 5002901 Security Update 16.0.55
Aug 20, 2026 Microsoft Office LTSC for Mac 2024 - Information Disclosure Important Release Notes Security Update 16.112.
Aug 20, 2026 Microsoft Office LTSC 2024 for 64-bit editions - Information Disclosure Important Click to Run Security Update https://
Aug 20, 2026 Microsoft Office LTSC 2024 for 32-bit editions - Information Disclosure Important Click to Run Security Update https://
Aug 20, 2026 Microsoft Office LTSC 2021 for 32-bit editions - Information Disclosure Important Click to Run Security Update https://
Aug 20, 2026 Microsoft Office LTSC 2021 for 64-bit editions - Information Disclosure Important Click to Run Security Update https://
Aug 20, 2026 Microsoft Office LTSC for Mac 2021 - Information Disclosure Important Release Notes Security Update 16.112.
Aug 20, 2026 Microsoft 365 Apps for Enterprise for 64-bit Systems - Information Disclosure Important Click to Run Security Update https://
Aug 20, 2026 Microsoft 365 Apps for Enterprise for 32-bit Systems - Information Disclosure Important Click to Run Security Update https://
Aug 20, 2026 Microsoft Office 2019 for 64-bit editions - Information Disclosure Important Click to Run Security Update https://

Android?

Filter
Overview
Security overview
Android Security Bulletins
Bulletins home
Overview
2026 bulletins
August
July
June
May
April
March
February
January
Android 17
2025 bulletins
2024 bulletins
2023 bulletins
2022 bulletins
2021 bulletins
2020 bulletins
2019 bulletins
2018 bulletins
2017 bulletins
2016 bulletins

2026-06-01 security patch level vulnerability details

In the sections below, we provide details for each of the security vulnerabilities that apply to the 2026-06-
level. Vulnerabilities are grouped under the component they affect. Issues are described in the tables belo
include CVE ID, associated references, type of vulnerability, severity, and updated AOSP versions (where a
When available, we link the public change that addressed the issue to the bug ID, like the AOSP change lis
multiple changes relate to a single bug, additional references are linked to numbers following the bug ID. D
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 remote escalation of privilege with no additional
privileges needed. User interaction is not needed for exploitation.

CVE	References	Type	Severity	Updated AOSP versi
CVE-2025-65018	A-463998243 [2] [3] [4] [5] [6]	EoP	Critical	14, 15, 16, 16-qpr2
CVE-2025-64720	A-463995203 [2] [3] [4] [5] [6]	DoS	Critical	14, 15, 16, 16-qpr2
CVE-2025-22424	A-350456241	EoP	High	14, 15, 16, 16-qpr2
CVE-2025-22426	A-365086157	EoP	High	14, 15, 16, 16-qpr2
CVE-2025-48570	A-376048041 [2]	EoP	High	14
CVE-2025-48595	A-430889718 [2] [3] [4] [5]	EoP	High	14, 15, 16, 16-qpr2
CVE-2025-48615	A-433250316 [2]	EoP	High	14, 15, 16, 16-qpr2
CVE-2025-48649	A-339109116	EoP	High	14, 15, 16, 16-qpr2

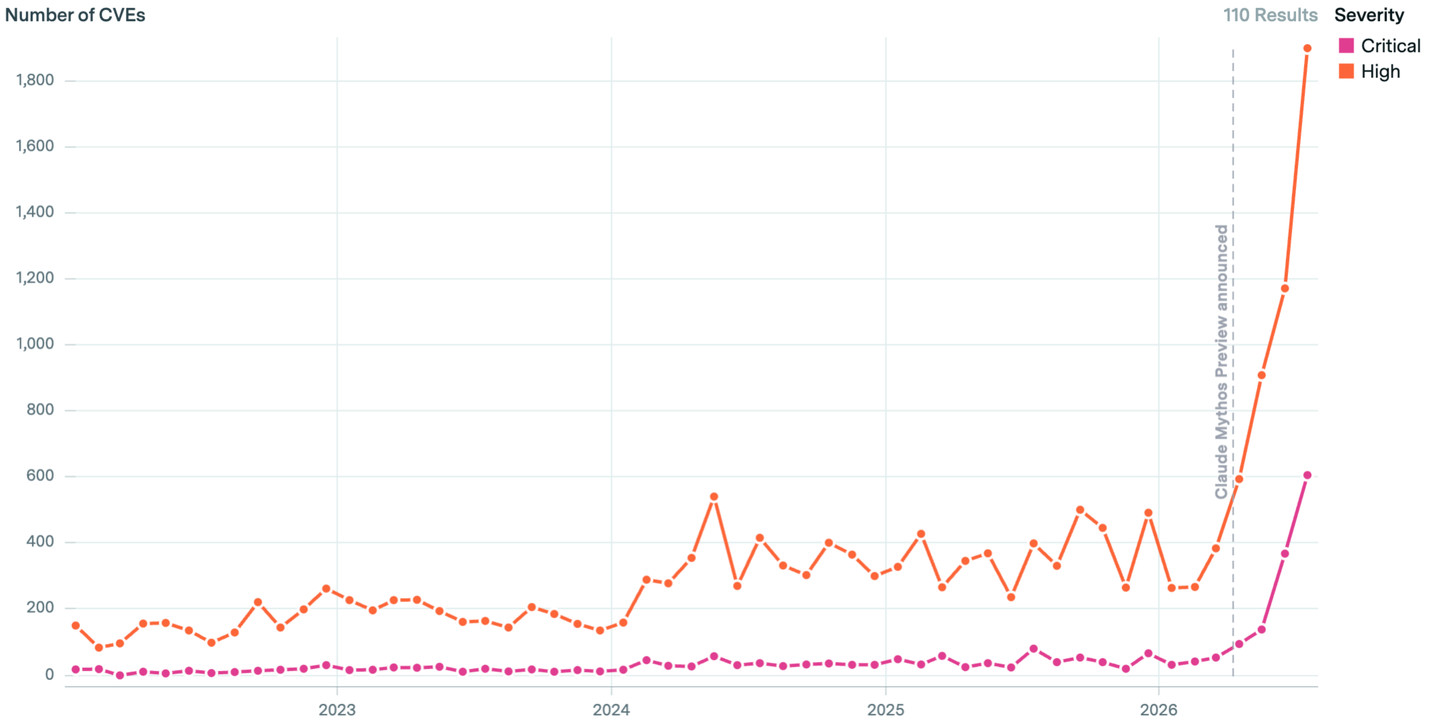
Updated Aug. 26, 2026

Cyber Vulnerabilities

Our interactive explorer for cybersecurity vulnerability trends, built on Common Vulnerabilities and Exposures (CVE) reported to cve.org since 2022.

Download this data

Critical and high severity vulnerabilities from 21 notable organizations



CVEs from Adobe, AMD, Apache, Apple, AWS, Cisco, GitHub, Google, IBM, Intel, Linux, Microsoft, Mozilla, NVIDIA, OpenSSL, Oracle, Qualcomm, Red Hat, Samsung, SAP, and VMware. Reporting procedures, labeling, and cadence vary substantially between organizations.

2022 — 2026

EPOCH AI | CC-BY

Share Download graph

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

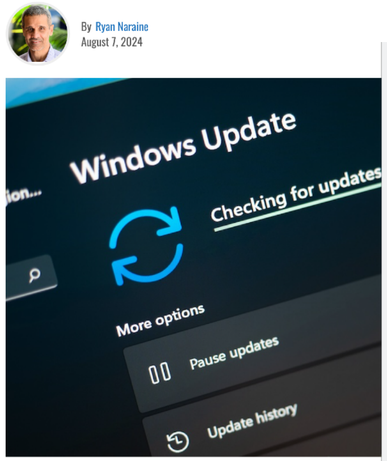
- Why should we study/research software security?
- Course logistics
- Principles of Building Secure Software Systems

Fact 1: Software Has Bugs

BLACK HAT

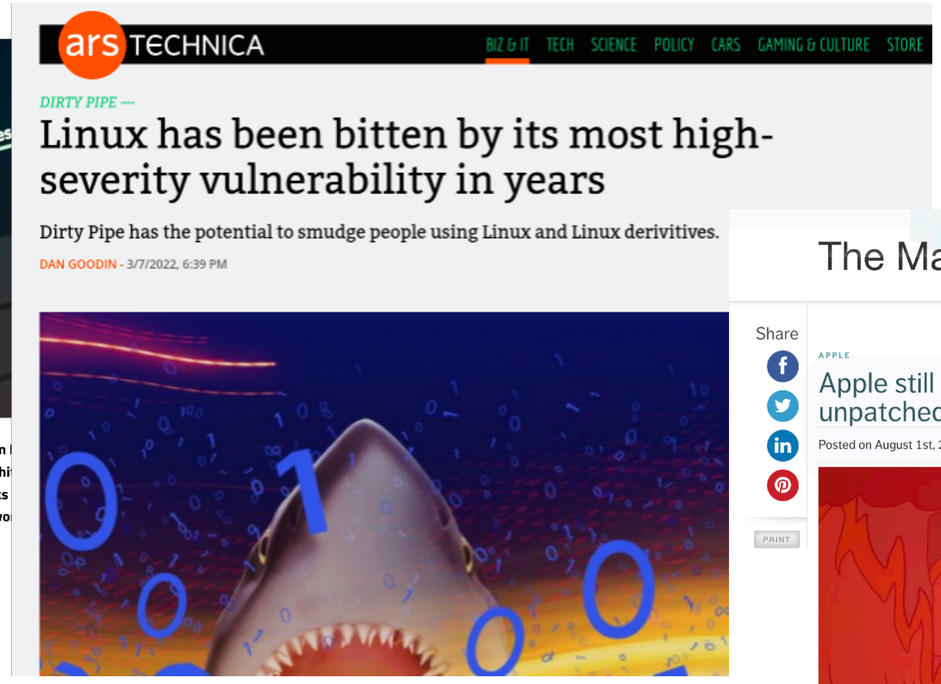
Windows Update Flaws Allow Undetectable Downgrade Attacks

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
major gaps in Microsoft's Windows Update archi
hackers can launch software downgrade attacks
meaningless on any Windows machine in the wo



The Mac Security Blog

Search the Blog

Share

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)

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)

It is Too Easy to Write Bugs

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    char user_name[32];
    scanf("%s", user_name);
    printf("Hello, %s!\n", user_name);
}
```

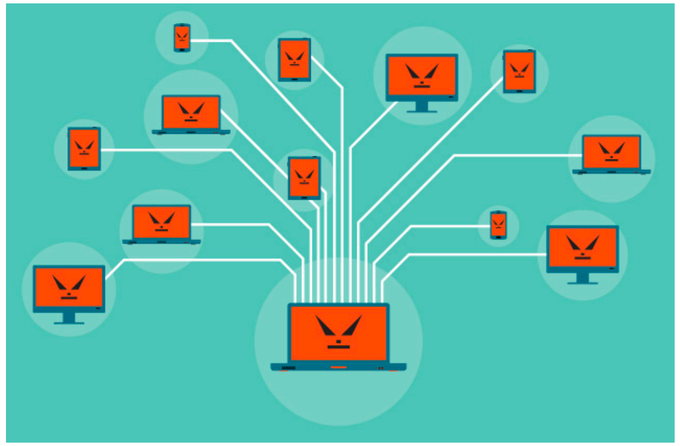
What if user's name is longer than 32 characters?

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

Fact 2: Many Bugs Are Exploitable (Causing Damage)

Ransomeware

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

https://en.wikipedia.org/wiki/Vulnerability_(computer_security)

The Morris Worm

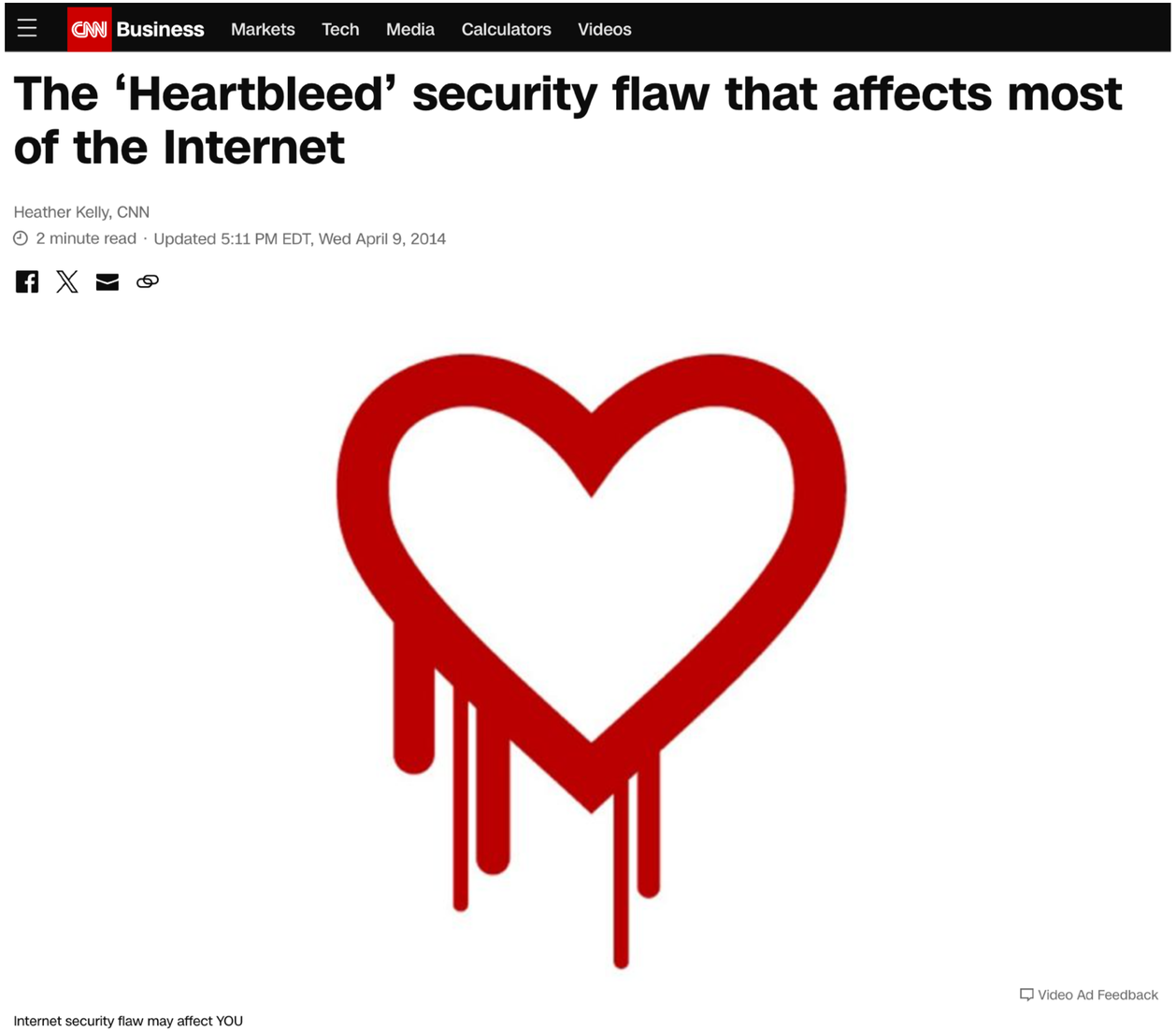
The Morris Worm

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



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 process's 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;
```

https://blog.cryptographyengineering.com/2014/04/08/attack-of-the-week-openssl-heartbleed/

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)

51 million lines!
50 lines/page,
0.1 mm/page = 101 m!

20.6m

https://openhub.net/p/chrome/analyses/latest/languages_summary

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.


NOKIA
ABC
MENU
R
M
C
1
2 abc
3 def
4 ghi
5 jkl
6 mno
7 pqrs
8 tuv
9 wxyz
* +
0
#


DEF
ABC
GHI
JKL
MNO
PRS
TUV
WXY
392-6818


Block Caller Button
Scam Call
Block caller button
Same scam with a different number
Block Caller Button

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 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
Date discovered	at or before 27 March 2024; 8 months ago[1][2]
Date of public disclosure	29 March 2024; 8 months ago

CVE-2021-3156: Heap-Based Buffer Overflow in Sudo (Baron Samedit)

Himanshu Kathpal, Vice President, Product Management, Platform and Technologies
August 3, 2026 - 2 min read

433 Share

Table of Contents

Update Feb 3, 2021: It has been reported that macOS, AIX, and Solaris are also vulnerable to CVE-2021-3156, and that others may also still be vulnerable. Qualys has not independently verified the exploit.

Original Post: The Qualys Research Team has discovered a heap overflow vulnerability in sudo, a near-ubiquitous utility available on major Unix-like operating systems. Any unprivileged user can gain root privileges on a vulnerable host using a default sudo configuration by exploiting this vulnerability.

Sudo is a powerful utility that's included in most if not all Unix- and Linux-based OSes. It allows users to run programs with the security privileges of another user. The vulnerability itself has been hiding in plain sight for nearly 10 years. It was introduced in July 2011 (commit 8255ed69) and affects all legacy versions from 1.8.2 to 1.8.31p2 and all stable versions from 1.9.0 to 1.9.5p1 in their default configuration.

It is Too Easy to Write Bugs

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    char user_name[32];
    scanf("%s", user_name);
    printf("Hello, %s!\n", user_name);
}
```

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 advanced course focusing on low-level systems stuff.

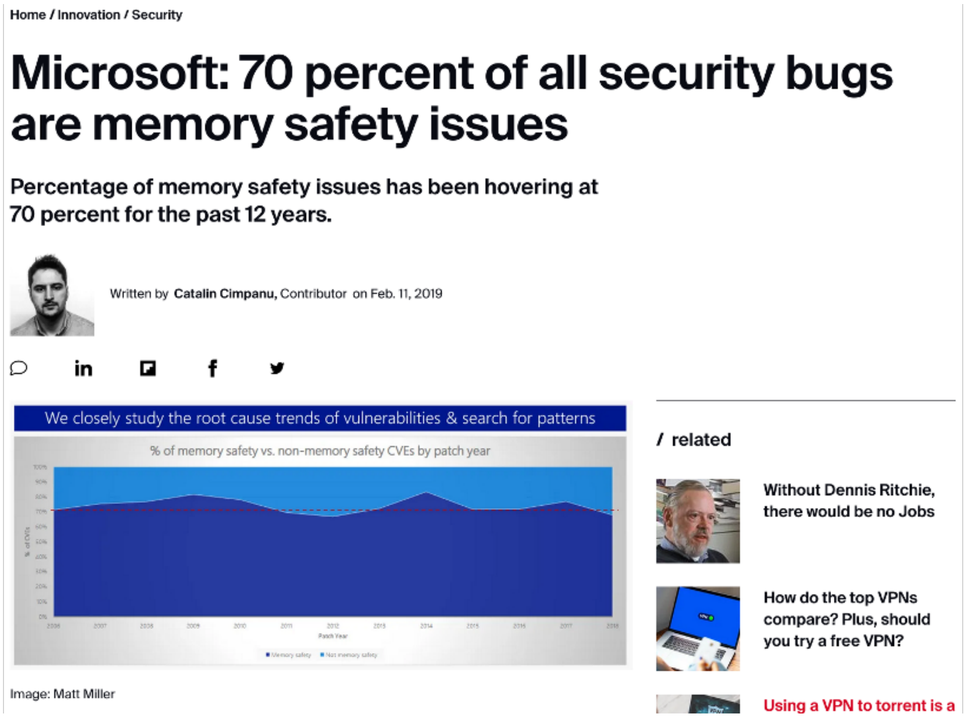
What Are You Supposed to Learn in this Course?

- 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
  - Memory-safety programming languages
- Type Safety
- Least Privilege Principle
- AI for Software Security

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

We closely study the root cause trends of vulnerabilities & search for patterns

% of memory safety vs. non-memory safety CVEs by patch year

Image: Matt Miller

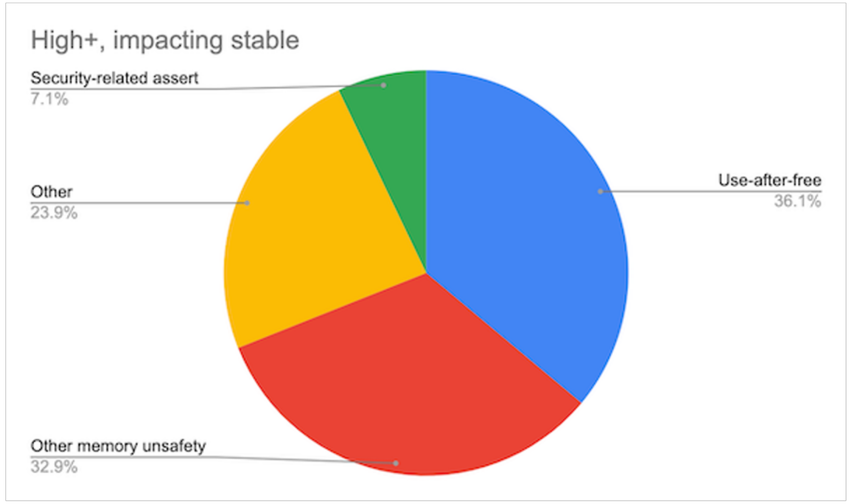
/ related

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.



Memory Safety Vulnerabilities Are a Major Threat

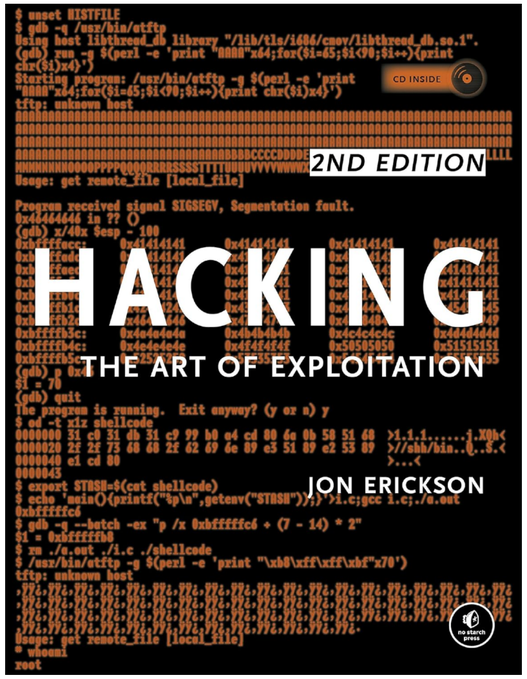
2025 CWE Top 10 KEV Weaknesses

Top 25 Home | Share via: 𝕏 | View in table format | KEV Key Insights | KEV Methodology

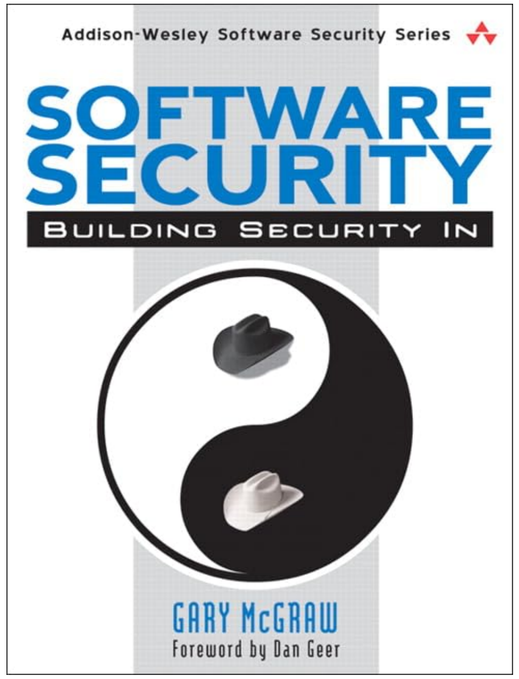
1. Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
   CWE-78 | CVEs in KEV: 20 | Rank Last Year: 3 (up 2) ▲
2. Use After Free
   CWE-416 | CVEs in KEV: 14 | Rank Last Year: 9 (up 7) ▲
3. Out-of-bounds Write
   CWE-787 | CVEs in KEV: 12 | Rank Last Year: 1 (down 2) ▼
4. Missing Authentication for Critical Function
   CWE-306 | CVEs in KEV: 11 | Rank Last Year: 7 (up 3) ▲
5. Deserialization of Untrusted Data
   CWE-502 | CVEs in KEV: 11 | Rank Last Year: 5
6. Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
   CWE-22 | CVEs in KEV: 10 | Rank Last Year: 6
7. Improper Control of Generation of Code ('Code Injection')
   CWE-94 | CVEs in KEV: 7 | Rank Last Year: 4 (down 3) ▼
8. Authentication Bypass Using an Alternate Path or Channel
   CWE-288 | CVEs in KEV: 6 | Rank Last Year: 14 (up 6) ▲
9. Heap-based Buffer Overflow
   CWE-122 | CVEs in KEV: 6 | Rank Last Year: 11 (up 2) ▲
10. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
    CWE-79 | CVEs in KEV: 7 | Rank Last Year: 21 (up 11) ▲

Reading Materials

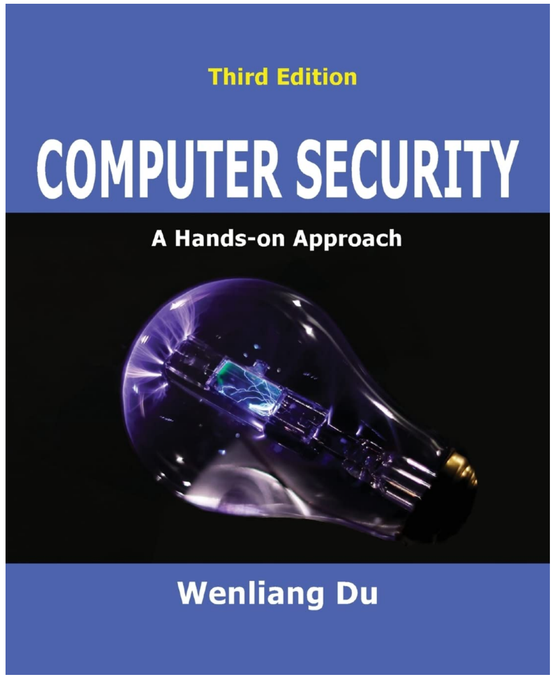
- *Software Security: Principles, Policies, and Protection*, by Mathias Payer
  - Broadly and briefly introduces software security, esp. on low-level software



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 and Grading

- Solve three machine problems (40%)
  - Find and exploit vulnerabilities in buggy programs.
- Read and understand research papers (20%)
  - Summarize and answer questions about papers
- Students-led paper presentation and discussion (10%)
- Class participation (10% + 5% extra credit)
  - Ask and answer questions!
- One final exam (20%)

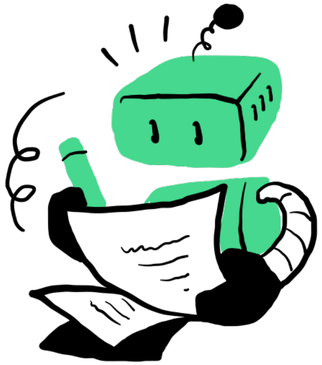
Why Read and Discuss Research Papers?

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

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-2026/

- 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/Review 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

https://en.wikipedia.org/wiki/Vulnerability_(computer_security)

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 *unintended* 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?

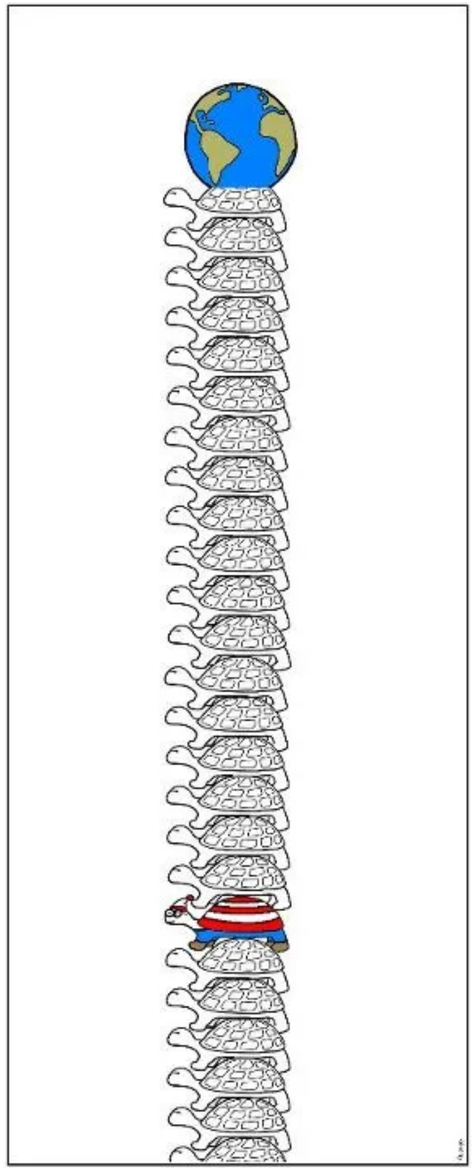
Thompson: Reflections on Trusting Trust

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

Thompson: Reflections on Trusting Trust

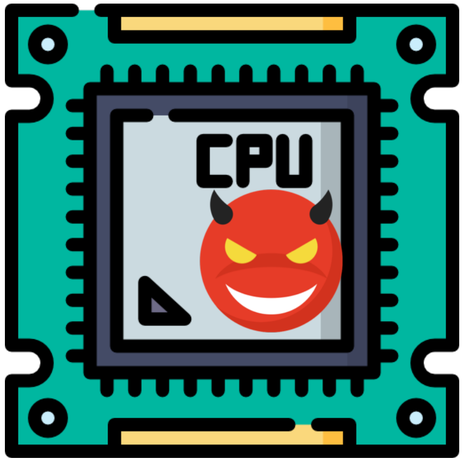
- Ken Thompson's Turing Award lecture describes the importance of making clear what should be trusted.
- Do you trust your compiler? 
  - Thompson described an approach whereby he can generate a compiler that can insert a backdoor.
    - e.g., insert a backdoor when recognizing a login program
- But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

Assume a program is trustworthy, does it need to adhere to this principle?

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 principles 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.

Is a program secure if all its specifications are formally verified?

The specification may be buggy!

Abstractions

“All problems in computer science can be solved by another level of indirection.”

- David Wheeler

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)

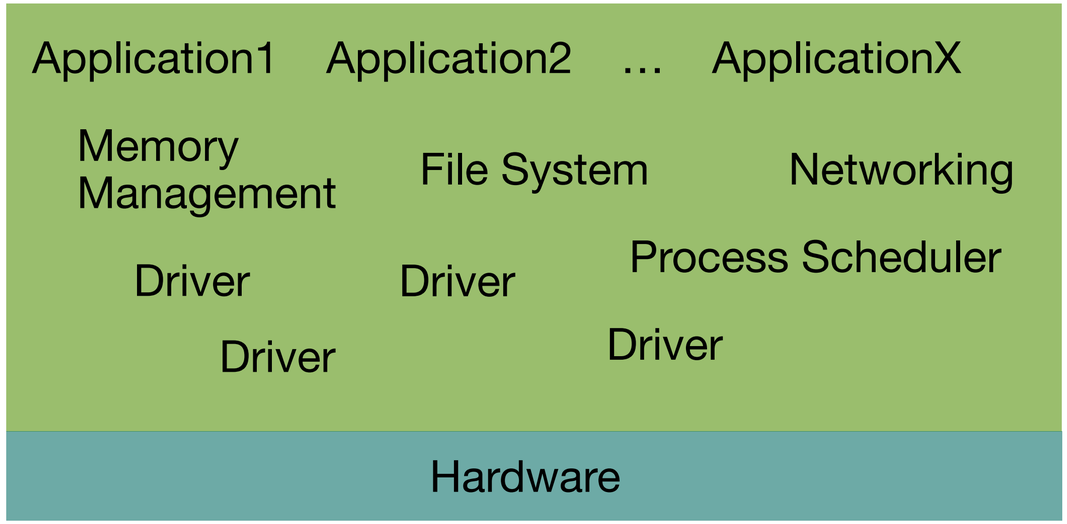
- Provides process abstraction
- Well-defined API to access hardware resources
- Enforces mutual exclusion to resources
- Enforces access permissions for resources
- Restrictions based on user/group/ACL
- Restricts attacker

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).

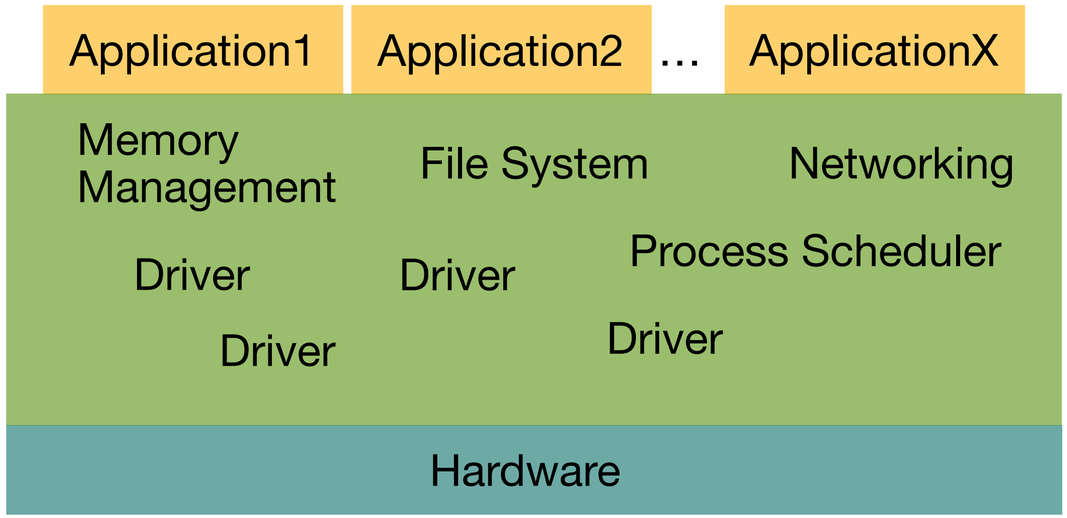
Abstractions: Single-domain OS

- 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



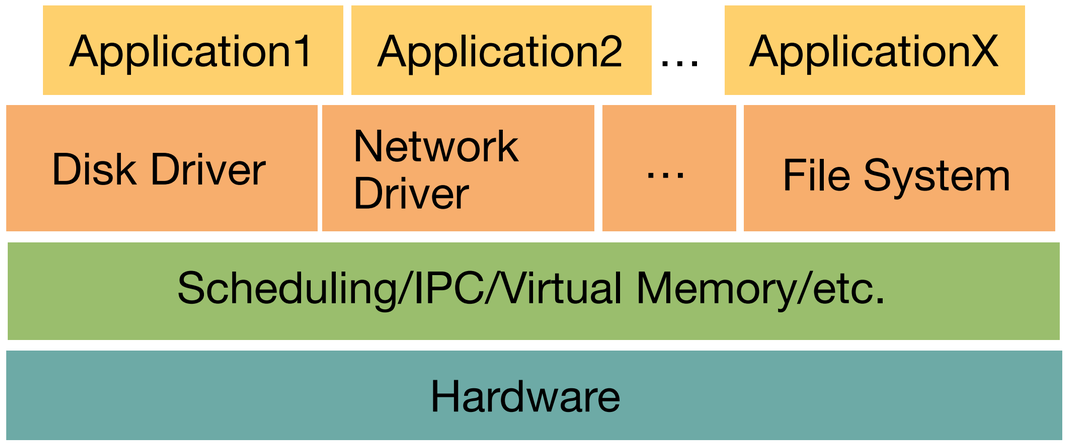
Abstractions: Monolithic OS

- 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)



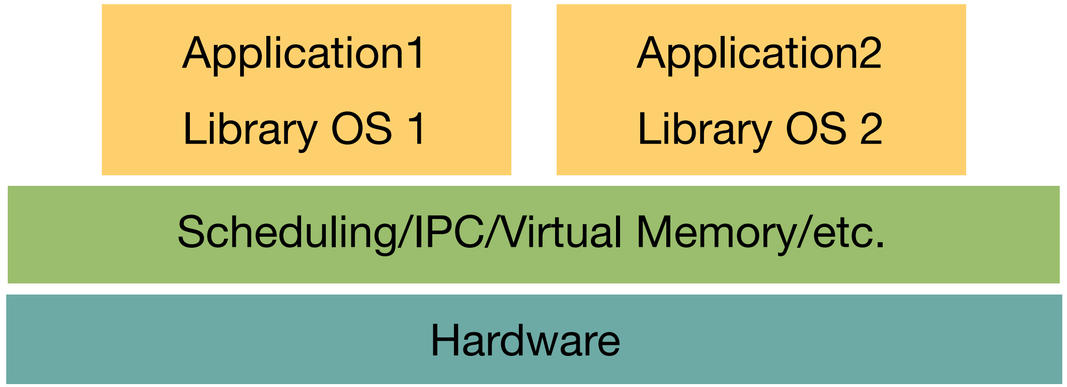
Abstractions: Micro-kernel

- 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.
- IOMMU restricts and virtualizes device access to memory.
- 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

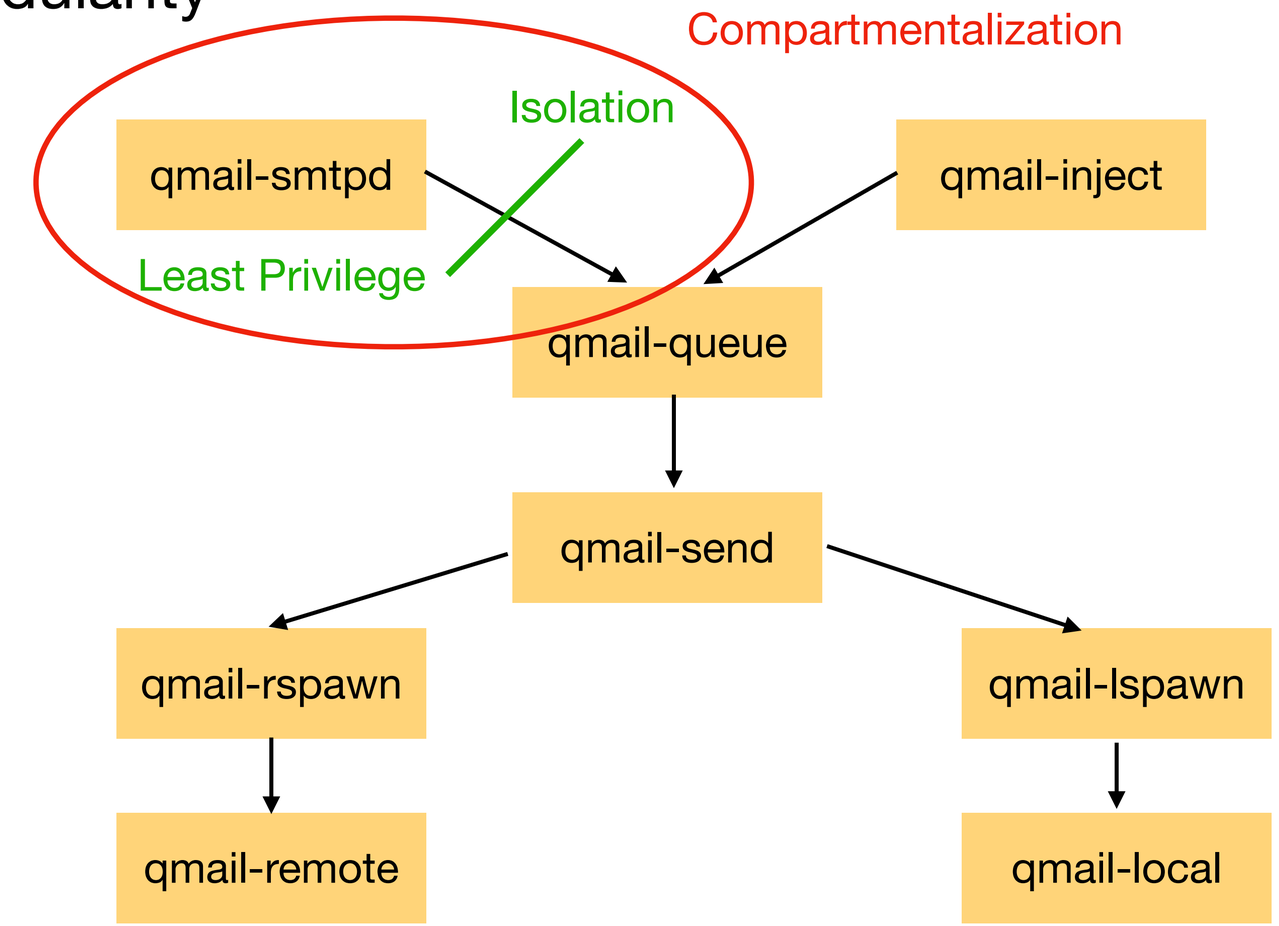
`Sendmail` uses a typical Unix approach with a large monolithic server and is known for the high complexity and previous security vulnerabilities.

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.