Introduction

ICS332
Operating Systems

Henri Casanova (henric@hawaii.edu)

Course Goal

- At this point in your life you:
  - Have used at least one OS
  - Know which OS runs on your computer
  - Know that without the OS you couldn’t use your computer
- Yet, for most of you, the OS is pretty mysterious
- Say your art major friend asks: “What really happens when I double click on an icon to run an application on my computer?”
- Could you give a decent answer besides: “Amazingly, it all works?”
- As a Computer Scientist it’s ok to not know much about how a car/fridge/airplane works, but it’s not ok to be clueless about how an OS works

Motivation to Study OSes?

- Let's be clear, very few of you will develop an actual OS
- **But**, most of you will work on complex systems that couple together many components
- **Obvious Motivation:** These systems all use the OS heavily
  - Important to know how to the use the OS
  - Important to know that the OS can and cannot do
  - Important to know what happens under the cover to understand bug, security, performance, etc.
- **Meta Motivation (you have to trust me on this one):** Knowing OS principles makes you a better software/system designer
  - OS concepts are massively re-usable in your own projects
    - Asking oneself "how does the OS do this?" is always useful
  - Studying "operating systems" makes you better at "systems" :)

OS in the News

- If you follow the news, general or tech-oriented, you know there are quite a few OS-related item each month
  - Some about new “exciting” features targeted at consumers, often very vague on details
  - Some about virus/problems/bugs, typically more targeted at computer professionals
- After taking this course you should be able to understand these, or at least the OS side to them
  - Note that understanding computer architecture is also needed, as vulnerabilities/attacks are typically at the software/hardware interface

A Few Example

- General news:
  - Articles about Specter/Meltdown (2018)
  - Articles about TLBleed (2018)
- Technical content:
  - New Virtual Memory feature in Linux (2022)
  - New Scheduling feature in Windows/Intel (2025)
  - A Windows vulnerability report (2024)
  - A Linux/Android vulnerability report (2022)
- I highlight terms we’ll learn about in this course
  - Note that there are mistakes in some of the general news content!!

The New York Times

The software patches could slow the performance of affected machines by 20 to 30 percent, said Andres Freund, an independent software developer who has tested the new Linux code. The researchers who discovered the flaws voiced similar concerns

What's a kernel?

PCWorld FROM IDG

The kernel inside a chip is basically an invisible process that facilitates the way apps and functions work on your computer. It has complete control over your operating system. Your PC needs to switch between user mode and kernel mode thousands of times a day, making sure instructions and data flow seamlessly and instantaneously. Here's how The Register puts it: "Think

The Register® Biting the hand that feeds IT

Think of the kernel as God sitting on a cloud, looking down on Earth. It's there, and no normal being can see it, yet they can pray to it.

These KPTI patches move the kernel into a completely separate address space, so it's not just invisible to a running process, it's not even there at all. Really, this shouldn't be needed, but clearly there is a flaw in Intel's silicon that allows kernel access protections to be bypassed in some way.

THE VERGE

THURSDAY, JANUARY 4, 2018 | CHIPOCALYPSE NOW

FLAW IS RELATED TO KERNEL MEMORY ACCESS

The exact bug is related to the way that regular apps and programs can discover the contents of protect kernel memory areas. Kernels in operating systems have complete control over the entire system, and connect applications to the processor, memory, and other hardware inside a computer. There appears to be a flaw in Intel's processors that lets attackers bypass kernel access protections so that regular apps can read the contents of kernel memory. To protect against this, Linux programmers have been separating the kernel's memory away from user processes in what's being called "Kernel Page Table Isolation."

January '18 "Spectre, Meltdown"

The Register®
Biting the hand that feeds IT

Security

Meet TLBleed: A crypto-key-leaking CPU attack that Intel reckons we shouldn't worry about

How to extract 256-bit keys with 99.8% success

By Chris Williams, Editor in Chief 22 Jun 2018 at 22:44 60 SHARE ▼

ars TECHNICA SUBSCRIPTIONS SIGN IN

LOST IN TRANSLATION —

Hyperthreading under scrutiny with new TLBleed crypto key leak

A new attack prompted OpenBSD's developers to disable hyperthreading by default.

Apple
Apple iOS 12: How to update your iOS device and get the latest features

TECH2
Wednesday, September 19, 2018
Back to FIRSTPOST.

Xiaomi Redmi
Xiaomi Redmi
review: A great all-rou
with best in seg
di

HOME NEWS REVIEWS SCIENCE AUTO GAMING PHOTOS VIDEOS APPLE 2

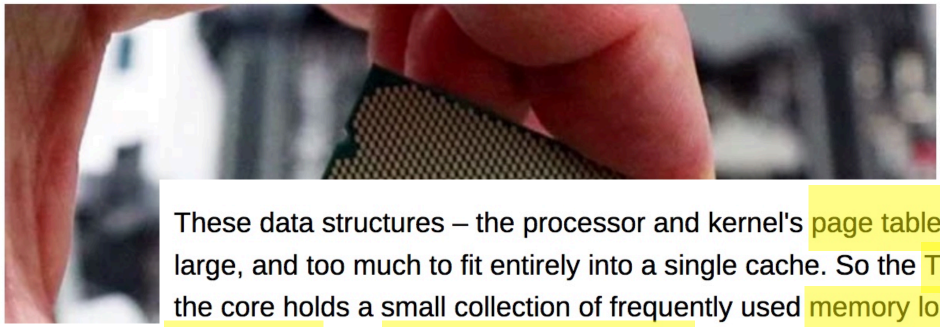
Technology News / News-Analysis

ANIRUDH REGIDI 26 JUNE, 2018 13:24 IST

TLBLEED JOINS SPECTRE AND MELTDOWN AS YET ANOTHER MAJOR CPU FLAW TO WORRY ABOUT

June '18
"TLBleed"

Intel Has No Plans To Patch TLBleed Hyper-Threading CPU Exploit, Here's Why



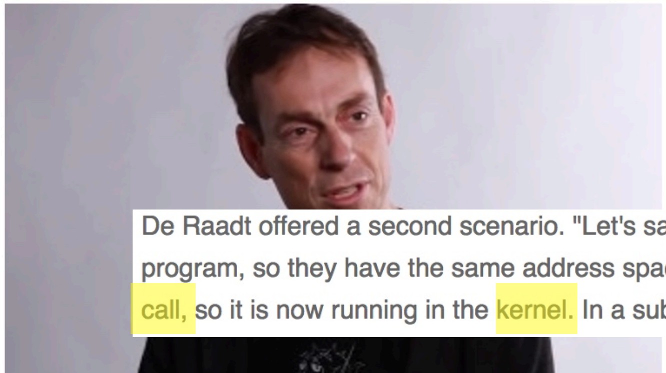
These data structures – the processor and kernel's page tables – are large, and too much to fit entirely into a single cache. So the TLB within the core holds a small collection of frequently used memory location translations. If a virtual-to-physical lookup is performed, and it's not in the TLB, it will be brought in so subsequent translations are almost instant. This may involve pushing out a less frequently used lookup from the buffer.

By observing the processor recycling TLB slots with new translations, it is possible to work out how the other thread running on the same core is operating.

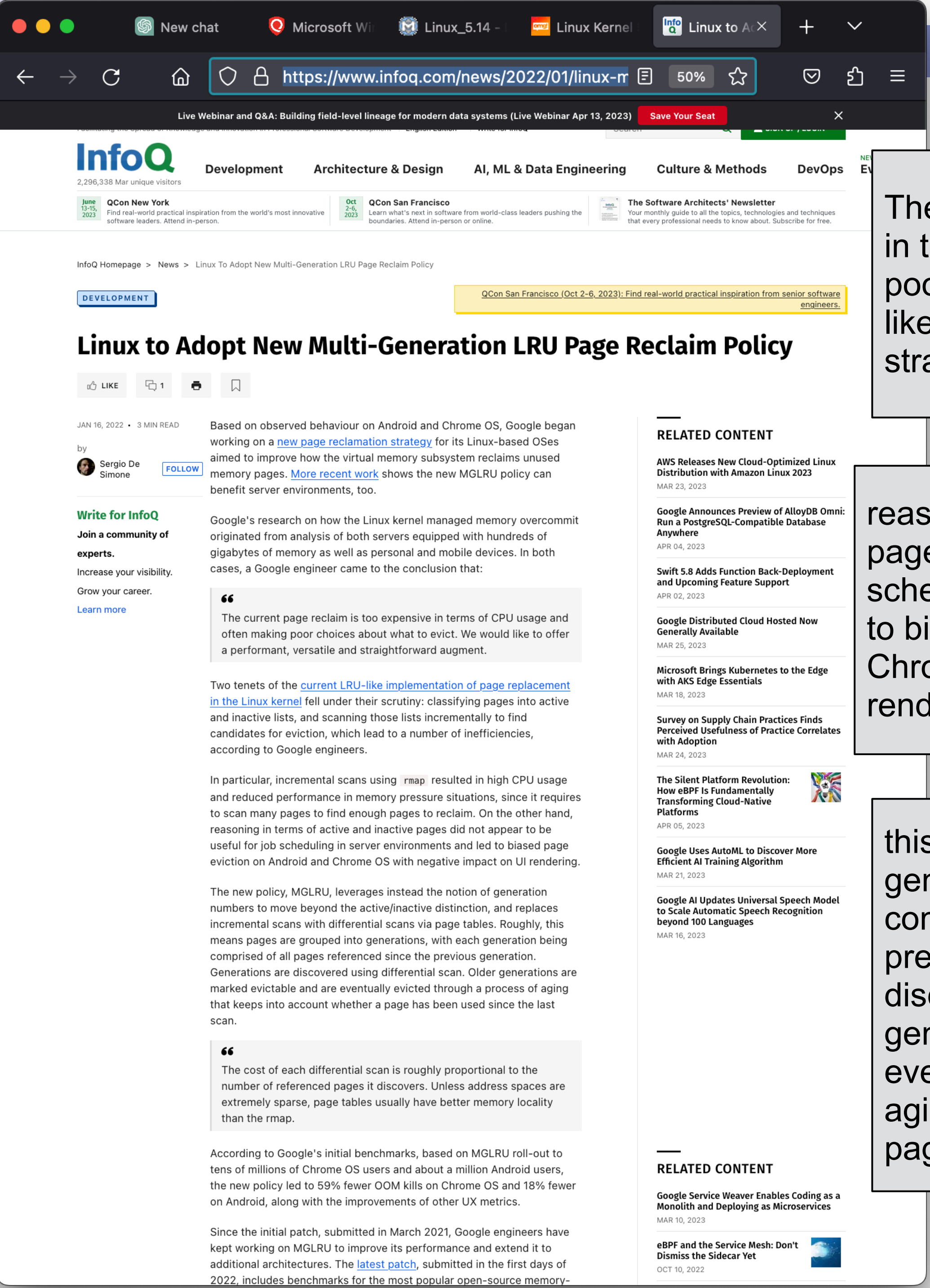
OpenBSD chief de Raadt says no easy fix for new Intel CPU bug

Share Tweet Share Share Share

iTWire
m Varghese | Published in Security | 2 Comments



De Raadt offered a second scenario. "Let's say you just run threads of the same program, so they have the same address space. One of the threads does a system call, so it is now running in the kernel. In a subtle way, the rule has just been broken."



Live Webinar and Q&A: Building field-level lineage for modern data systems (Live Webinar Apr 13, 2023) Save Your Seat

InfoQ
2,296,338 Mar unique visitors

Development | Architecture & Design | AI, ML & Data Engineering | Culture & Methods | DevOps

QCon New York (June 13-15, 2023) Find real-world practical inspiration from the world's most innovative software leaders. Attend in-person.

QCon San Francisco (Oct 2-6, 2023) Learn what's next in software from world-class leaders pushing the boundaries. Attend in-person or online.

The Software Architects' Newsletter Your monthly guide to all the topics, technologies and techniques that every professional needs to know about. Subscribe for free.

InfoQ Homepage > News > Linux To Adopt New Multi-Generation LRU Page Reclaim Policy

DEVELOPMENT

QCon San Francisco (Oct 2-6, 2023): Find real-world practical inspiration from senior software engineers.

Linux to Adopt New Multi-Generation LRU Page Reclaim Policy

LIKE 1

JAN 16, 2022 • 3 MIN READ

by Sergio De Simone FOLLOW

Write for InfoQ
Join a community of experts. Increase your visibility. Grow your career.
Learn more

Based on observed behaviour on Android and Chrome OS, Google began working on a new page reclamation strategy for its Linux-based OSes aimed to improve how the virtual memory subsystem reclaims unused memory pages. More recent work shows the new MGLRU policy can benefit server environments, too.

Google's research on how the Linux kernel managed memory overcommit originated from analysis of both servers equipped with hundreds of gigabytes of memory as well as personal and mobile devices. In both cases, a Google engineer came to the conclusion that:

> The current page reclaim is too expensive in terms of CPU usage and often making poor choices about what to evict. We would like to offer a performant, versatile and straightforward augment.

Two tenets of the current LRU-like implementation of page replacement in the Linux kernel fell under their scrutiny: classifying pages into active and inactive lists, and scanning those lists incrementally to find candidates for eviction, which lead to a number of inefficiencies, according to Google engineers.

In particular, incremental scans using `rmap` resulted in high CPU usage and reduced performance in memory pressure situations, since it requires to scan many pages to find enough pages to reclaim. On the other hand, reasoning in terms of active and inactive pages did not appear to be useful for job scheduling in server environments and led to biased page eviction on Android and Chrome OS with negative impact on UI rendering.

The new policy, MGLRU, leverages instead the notion of generation numbers to move beyond the active/inactive distinction, and replaces incremental scans with differential scans via page tables. Roughly, this means pages are grouped into generations, with each generation being comprised of all pages referenced since the previous generation. Generations are discovered using differential scan. Older generations are marked evictable and are eventually evicted through a process of aging that keeps into account whether a page has been used since the last scan.

> The cost of each differential scan is roughly proportional to the number of referenced pages it discovers. Unless address spaces are extremely sparse, page tables usually have better memory locality than the rmap.

According to Google's initial benchmarks, based on MGLRU roll-out to tens of millions of Chrome OS users and about a million Android users, the new policy led to 59% fewer OOM kills on Chrome OS and 18% fewer on Android, along with the improvements of other UX metrics.

Since the initial patch, submitted in March 2021, Google engineers have kept working on MGLRU to improve its performance and extend it to additional architectures. The latest patch, submitted in the first days of 2022, includes benchmarks for the most popular open-source memory-

RELATED CONTENT

AWS Releases New Cloud-Optimized Linux Distribution with Amazon Linux 2023
MAR 23, 2023

Google Announces Preview of AlloyDB Omni: Run a PostgreSQL-Compatible Database Anywhere
APR 04, 2023

Swift 5.8 Adds Function Back-Deployment and Upcoming Feature Support
APR 02, 2023

Google Distributed Cloud Hosted Now Generally Available
MAR 25, 2023

Microsoft Brings Kubernetes to the Edge with AKS Edge Essentials
MAR 18, 2023

Survey on Supply Chain Practices Finds Perceived Usefulness of Practice Correlates with Adoption
MAR 24, 2023

The Silent Platform Revolution: How eBPF Is Fundamentally Transforming Cloud-Native Platforms
APR 05, 2023

Google Uses AutoML to Discover More Efficient AI Training Algorithm
MAR 21, 2023

Google AI Updates Universal Speech Model to Scale Automatic Speech Recognition beyond 100 Languages
MAR 16, 2023

RELATED CONTENT

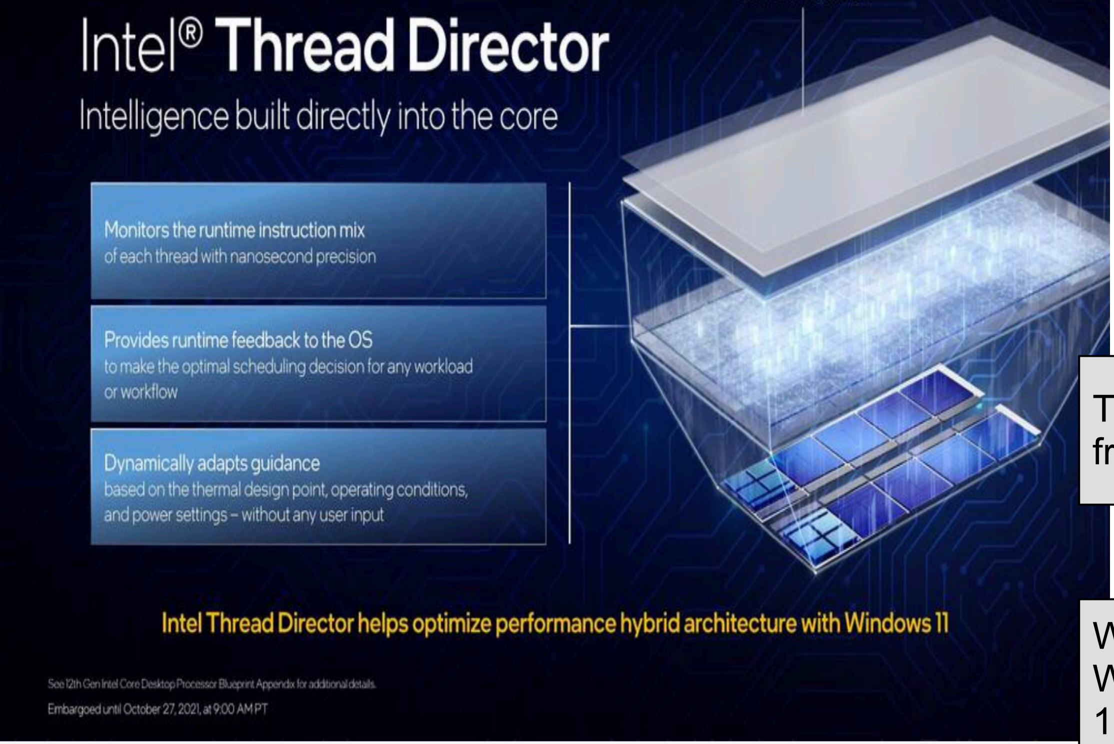
Google Service Weaver Enables Coding as a Monolith and Deploying as Microservices
MAR 10, 2023

eBPF and the Service Mesh: Don't Dismiss the Sidecar Yet
OCT 10, 2022

The current page reclaim is too expensive in terms of CPU usage and often making poor choices about what to evict. We would like to offer a performant, versatile and straightforward augment.

reasoning in terms of active and inactive pages did not appear to be useful for job scheduling in server environments and led to biased page eviction on Android and Chrome OS with negative impact on UI rendering.

this means pages are grouped into generations, with each generation being comprised of all pages referenced since the previous generation. Generations are discovered using differential scan. Older generations are marked evictable and are eventually evicted through a process of aging that keeps into account whether a page has been used since the last scan.



Intel® Thread Director

Intelligence built directly into the core

Monitors the runtime instruction mix
of each thread with nanosecond precision

Provides runtime feedback to the OS
to make the optimal scheduling decision for any workload or workflow

Dynamically adapts guidance
based on the thermal design point, operating conditions, and power settings – without any user input

Intel Thread Director helps optimize performance hybrid architecture with Windows 11

See 12th Gen Intel Core Desktop Processor Blueprint Appendix for additional details.
Embargoed until October 27, 2021, at 9:00 AM PT

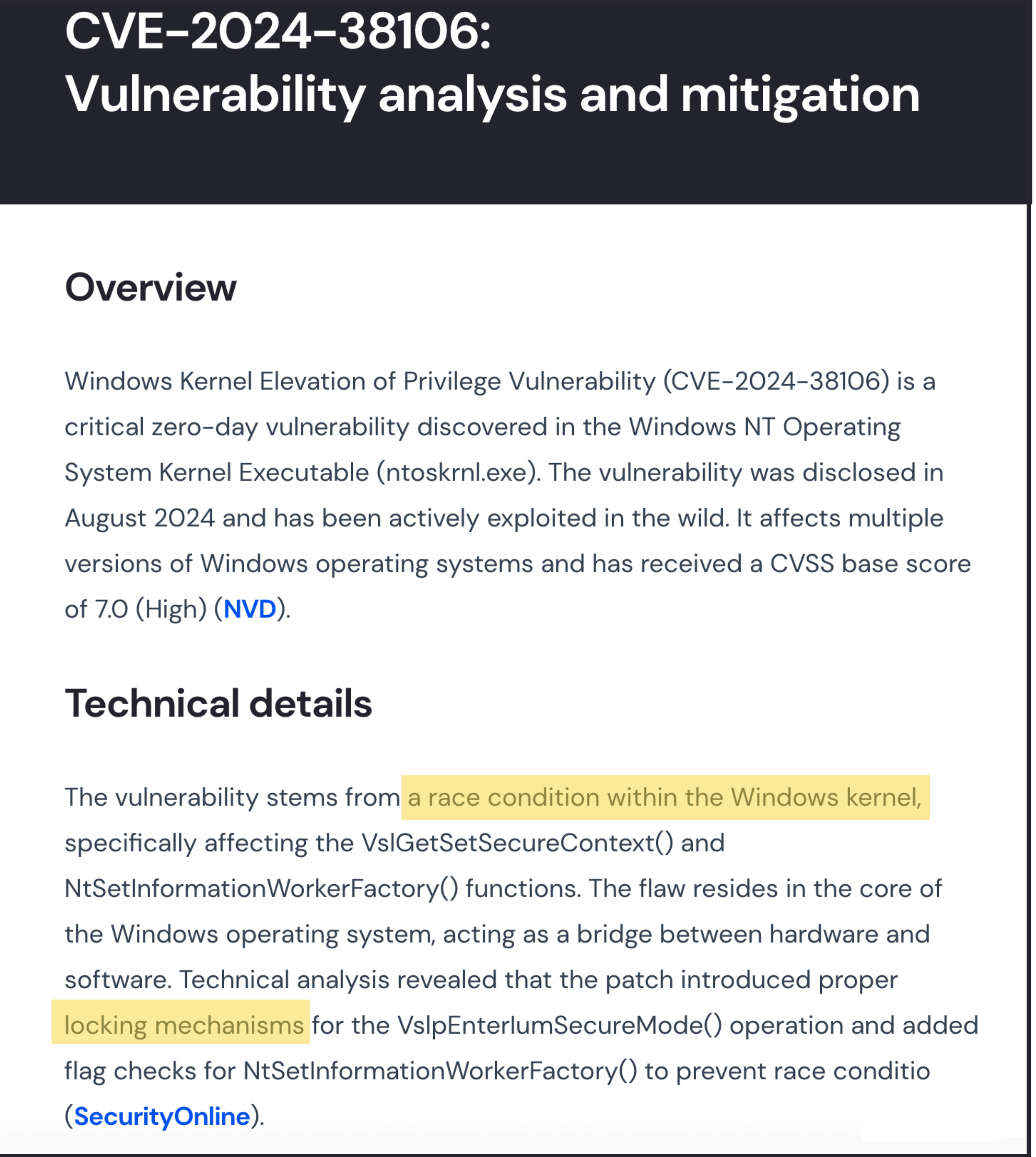
In order to ensure that the cores are used to their maximum, Intel had to work with Microsoft to implement a new hybrid-aware scheduler, and th
one interacts with an on-board microcontroller on the CPU for more information about what is actually going on. The microcontroller on the CPU
what we call Intel Thread Director. It has a full scope view of the whole processor – what is running where, what instructions are running, and what appears to be the most important. It monitors the instructions at the nanosecond level, and communicates with the OS on the microsecond level. It takes into account thermals, power settings, and identifies which threads can be promoted to higher performance modes, or those that can be bumped if something higher priority comes along. It can also adjust recommendations based on frequency, power, thermals, and additional sensory data not immediately available to the scheduler at that resolution. All of that gets fed to the operating system. The scheduler takes all of the information from Thread Director, constantly, as a guide. So if a user comes in with a more important workload, Thread Director tells the schedul
which cores are free, or which threads to demote. The scheduler can override the Thread Director, especially if the user has a specific request, s
as making background tasks a higher priority.

What makes Windows 11 better than Windows 10 in this regard is that Windows 10 focuses more on the power of certain cores, whereas Windows 11 expands that to efficiency as well. While Windows 10 considers the E-cores as lower performance than P-cores, it doesn't know how well each core does at a given frequency with a workload, whereas Windows 11 does. Combine that with an instruction prioritization model, and Intel states that under Windows 11, users should expect a lot better consistency in performance when it comes to hybrid CPU designs. Thread Director is running a pre-trained algorithm based on millions of hours of data gathered during the development of the feature. It identifies the effective IPC of a given workflow, and applies that to the performance/efficiency metrics of each core variation. If there's an obvious potential for better IPC or better efficiency, then it suggests the thread is moved.

The scheduler takes all the information from Thread Director, constantly as a guide

What makes Windows 11 better than Windows 10 in this regard is that Windows 10 focuses more on the power of certain cores, whereas Windows 11 expands that to efficiency as well.

If there's an obvious potential for better IPC or better efficiency, then it suggests the thread is moved



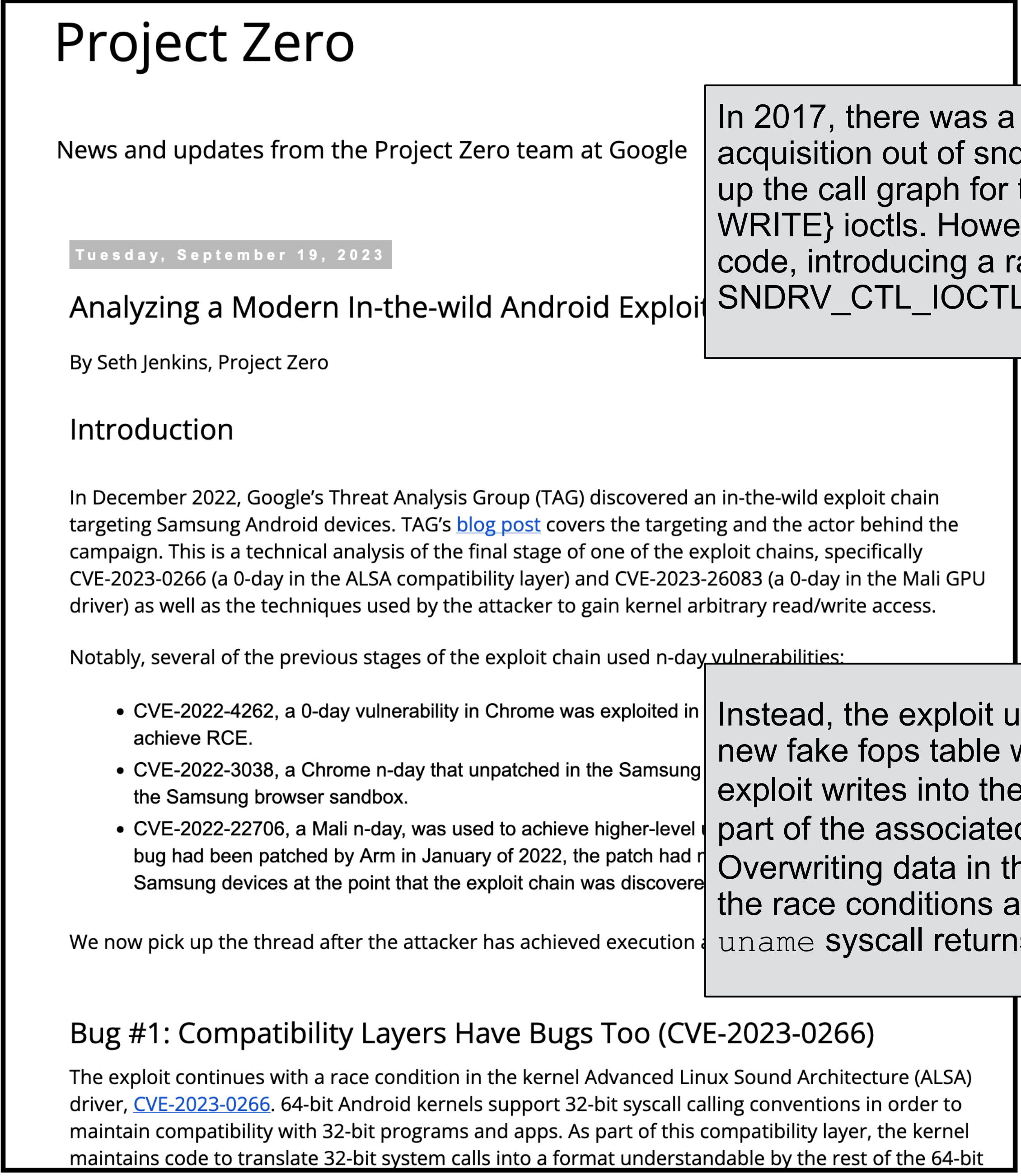
CVE-2024-38106: Vulnerability analysis and mitigation

Overview

Windows Kernel Elevation of Privilege Vulnerability (CVE-2024-38106) is a critical zero-day vulnerability discovered in the Windows NT Operating System Kernel Executable (ntoskrnl.exe). The vulnerability was disclosed in August 2024 and has been actively exploited in the wild. It affects multiple versions of Windows operating systems and has received a CVSS base score of 7.0 (High) (**NVD**).

Technical details

The vulnerability stems from a race condition within the Windows kernel, specifically affecting the VslGetSetSecureContext() and NtSetInformationWorkerFactory() functions. The flaw resides in the core of the Windows operating system, acting as a bridge between hardware and software. Technical analysis revealed that the patch introduced proper locking mechanisms for the VslpEnterIumSecureMode() operation and added flag checks for NtSetInformationWorkerFactory() to prevent race conditio (**SecurityOnline**).


Project Zero

News and updates from the Project Zero team at Google

Tuesday, September 19, 2023

Analyzing a Modern In-the-wild Android Exploit

By Seth Jenkins, Project Zero

Introduction

In December 2022, Google's Threat Analysis Group (TAG) discovered an in-the-wild exploit chain targeting Samsung Android devices. TAG's blog post covers the targeting and the actor behind the campaign. This is a technical analysis of the final stage of one of the exploit chains, specifically CVE-2023-0266 (a 0-day in the ALSA compatibility layer) and CVE-2023-26083 (a 0-day in the Mali GPU driver) as well as the techniques used by the attacker to gain kernel arbitrary read/write access.

Notably, several of the previous stages of the exploit chain used n-day vulnerabilities:

- CVE-2022-4262, a 0-day vulnerability in Chrome was exploited in
  achieve RCE.
- CVE-2022-3038, a Chrome n-day that unpatched in the Samsung
  the Samsung browser sandbox.
- CVE-2022-22706, a Mali n-day, was used to achieve higher-level
  bug had been patched by Arm in January of 2022, the patch had
  Samsung devices at the point that the exploit chain was discovere

We now pick up the thread after the attacker has achieved execution

Bug #1: Compatibility Layers Have Bugs Too (CVE-2023-0266)

The exploit continues with a race condition in the kernel Advanced Linux Sound Architecture (ALSA) driver, CVE-2023-0266. 64-bit Android kernels support 32-bit syscall calling conventions in order to maintain compatibility with 32-bit programs and apps. As part of this compatibility layer, the kernel maintains code to translate 32-bit system calls into a format understandable by the rest of the 64-bit

In 2017, there was a refactoring in the ALSA driver to move the lock acquisition out of snd_ctl_elem_{write|read}() functions and further up the call graph for the SNDRV_CTL_IOCTL_ELEM_{READ|WRITE} ioctls. However, this commit only addressed the 64-bit ioctl code, introducing a race condition into the 32-bit compatibility layer SNDRV_CTL_IOCTL_ELEM_{READ|WRITE}32 ioctls.

Instead, the exploit uses their initial arbitrary write to construct a new fake fops table within the `.data` section of the kernel. The exploit writes into the `init_uts_ns` kernel symbol, in particular the part of the associated structure that holds the `uname` of the kernel. Overwriting data in this structure provides a clear indicator of when the race conditions are won and the arbitrary write succeeds (the `uname` syscall returns different data than before).

Disclaimer (1)

- It would be great for us to develop an OS during the semester
- Sadly, it’s not feasible:
  - Too hard, time-consuming, programming-heavy for most students at this stage in their education, especially students who may struggle with low-level programming (Assembly, C, C++, Rust, etc.)
  - Or at least given the expected number of hours a student should put into an undergraduate course...
- As a result, **undergraduate** OS courses are often less “hands-on” than what some students expect
  - Typically a few of you come into this thinking we’re going to do awesome hacks in the Linux kernel….and will be disappointed
- If your dream is to get your hands dirty with the OS kernel you can:
  - Do an internship at a company that does low level / OS-related work, do an OS-related Google Summer of code, ..
  - Take the OS graduate course
  - Do it on your own (each semester there is at least one student who does this and there are tons of on-line resources!)

Disclaimer (2)

- I am not an "OS geek"
- If you've read the source code of some OS, if you have worked on an OS (internship), if you have any useful knowledge, you are more than welcome to share with the class whenever relevant
  - And I'm always happy to have the course content evolve dynamically based on students suggested topics within some reasonable bounds
- This said, the class is more on general principles than specific implementations (since those change often, as we will see)
  - You can learn a lot about OSes without necessarily spending hours looking at OS code

Teaching OS is not easy

- A significant part of the material is of the “here is how it works and why it’s a good idea” kind
  - Precisely because it’s not feasible to have the full-fledged hands-on experience at the undergraduate level
  - Don’t fear, there will be plenty of programming / hands-on activities in this course
- Although I try to make the course as interactive as possible, there is just a limit to what any instructor can do for some of this material at the undergraduate level
- Bottom line:
  - The course is fun when students are engaged and ask questions
    - And when we answer questions by writing code in class
  - The course is dull when students are silent

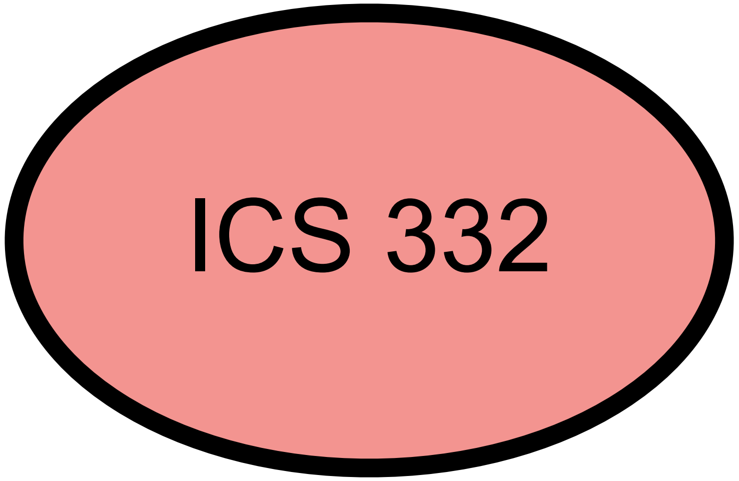
What we will learn (1)

- Roles of an operating system
- Fundamental principles of operating system design and kernel implementation
- Key features of operating systems of practical importance
  - The course content is not specific to a particular OS
  - Many OSes do things in similar way, but they also have key differences
  - We will often reference Unix derivatives (Mac OS, Linux, iOS, Android, ...) and Windows
  - We will mention ”historical” OSes whenever relevant
  - We will not study special-purpose OSes (e.g., real-time, network operating systems, ...)

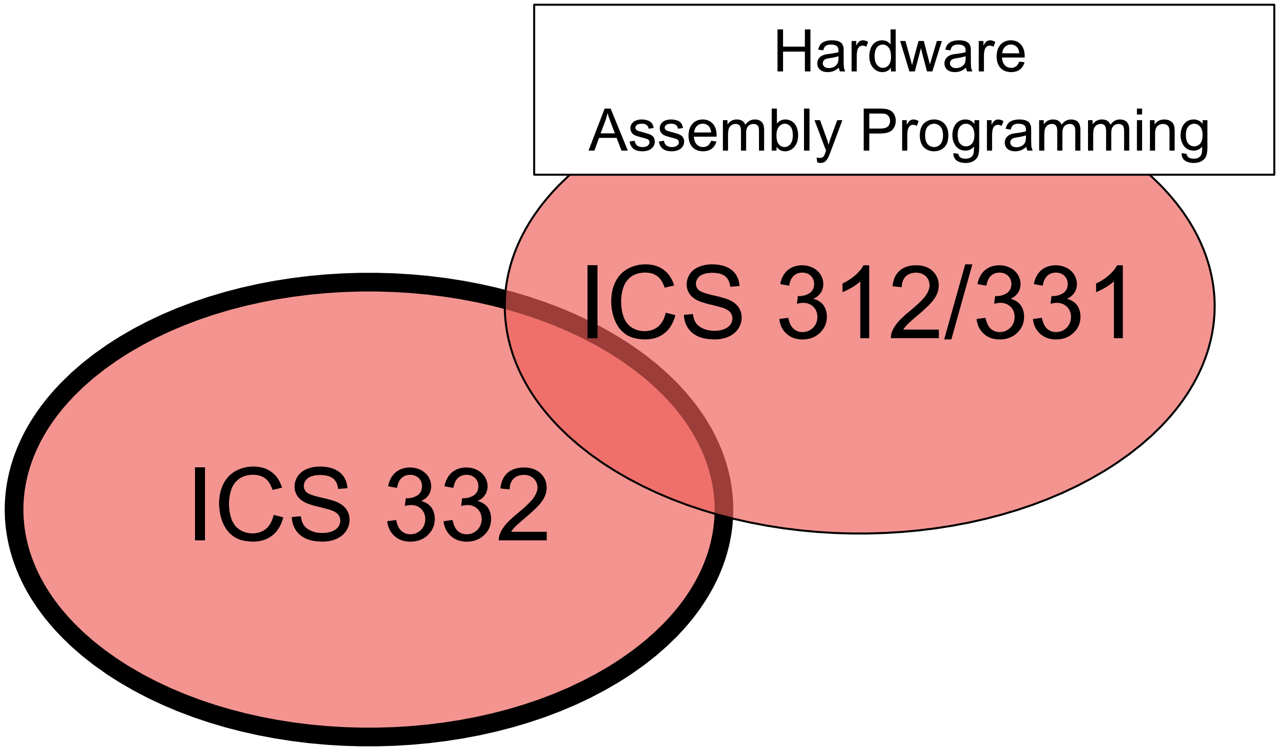
What we will learn (2)

- Fundamental components and principles of modern operating systems:
  - Processes and Threads Management
    Scheduling
  - Synchronization (barely scratching the surface here)
  - Memory and Virtual Memory Management
  - Storage and, if time, File Systems
  - Virtual Machines and Containers
- The programming assignments are more about “using” the OS than about “implementing” the OS
  - i.e., what most of you will need most in your profession

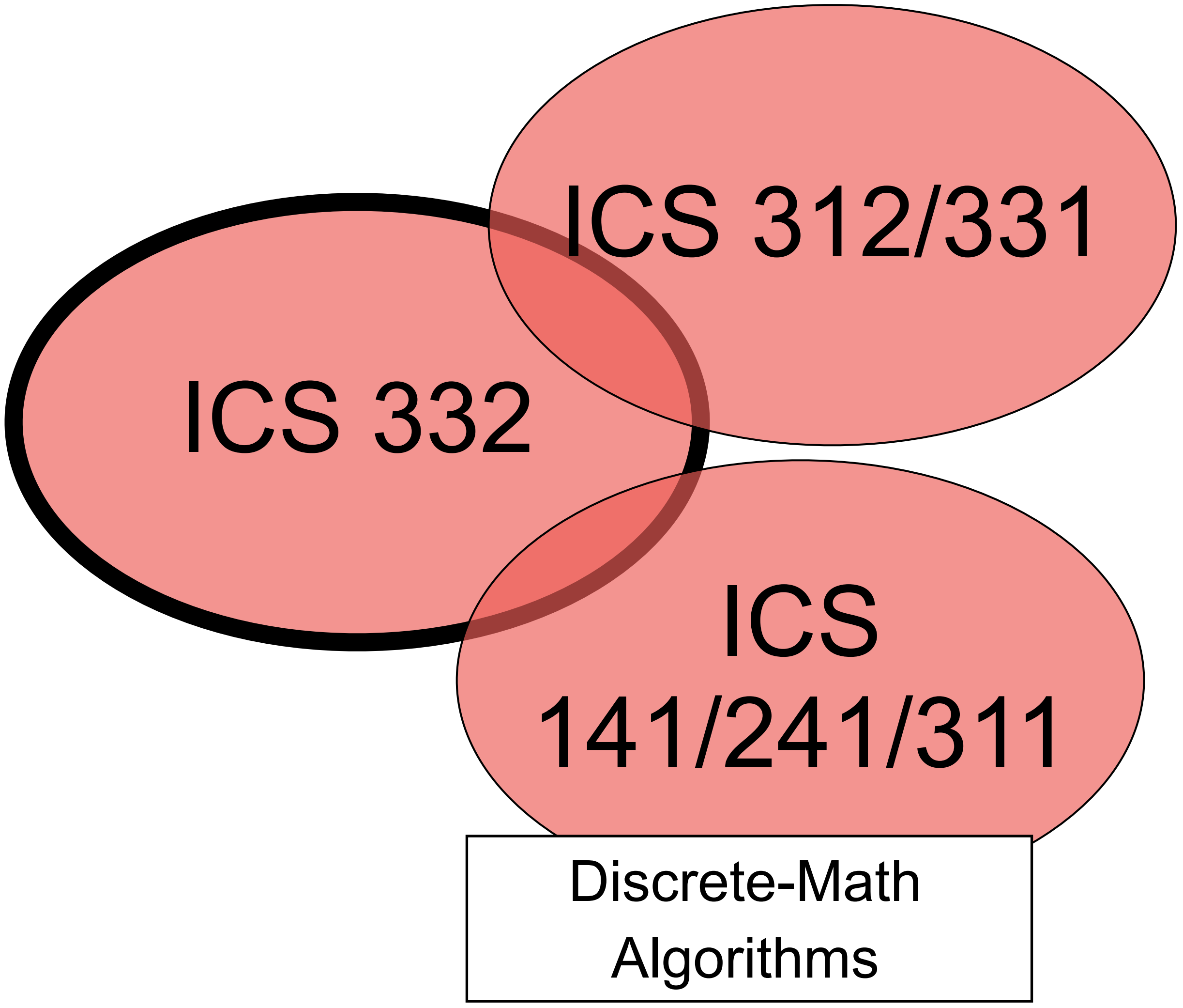
ICS332 and the ICS Curriculum



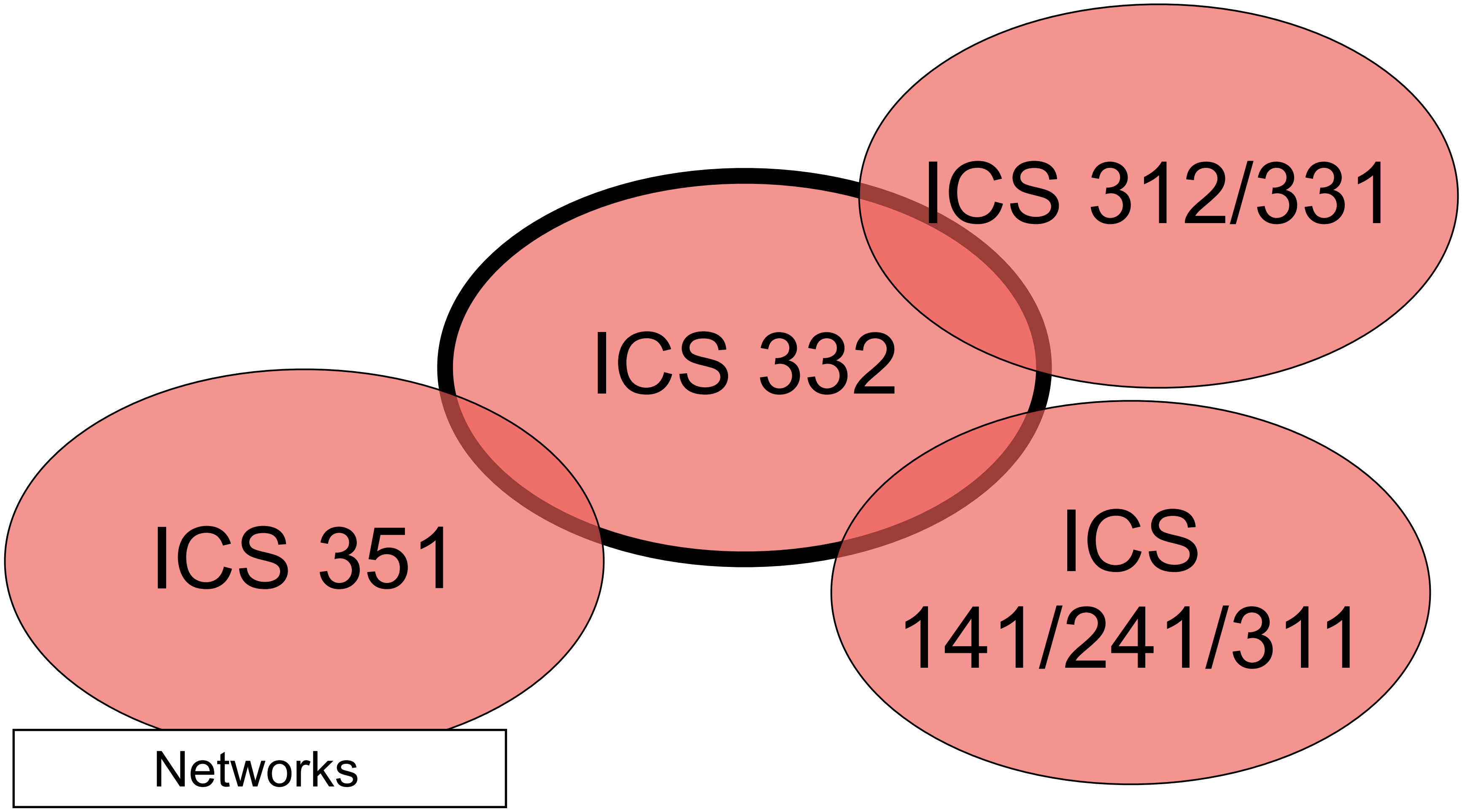
ICS332 and the ICS Curriculum



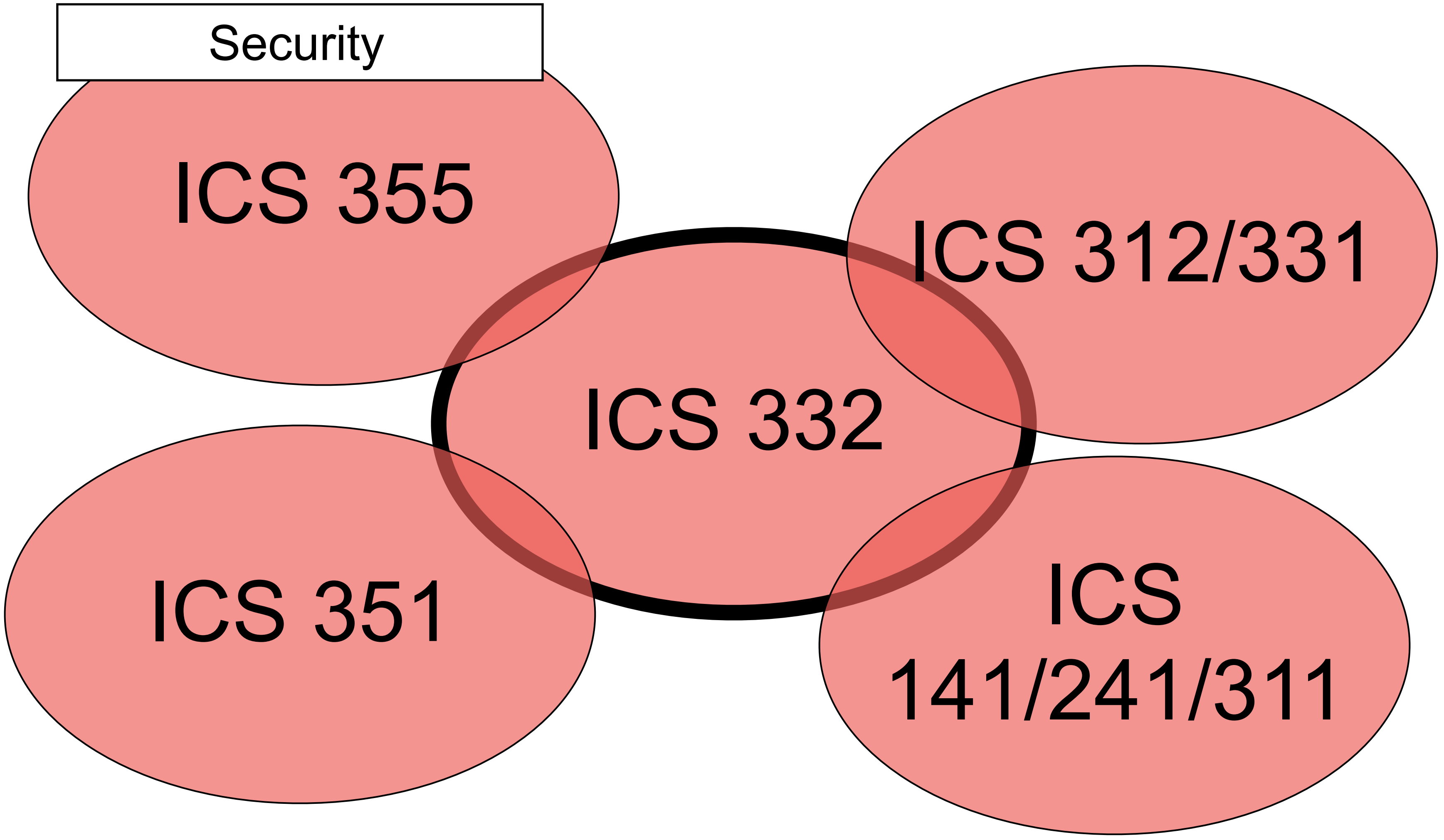
ICS332 and the ICS Curriculum



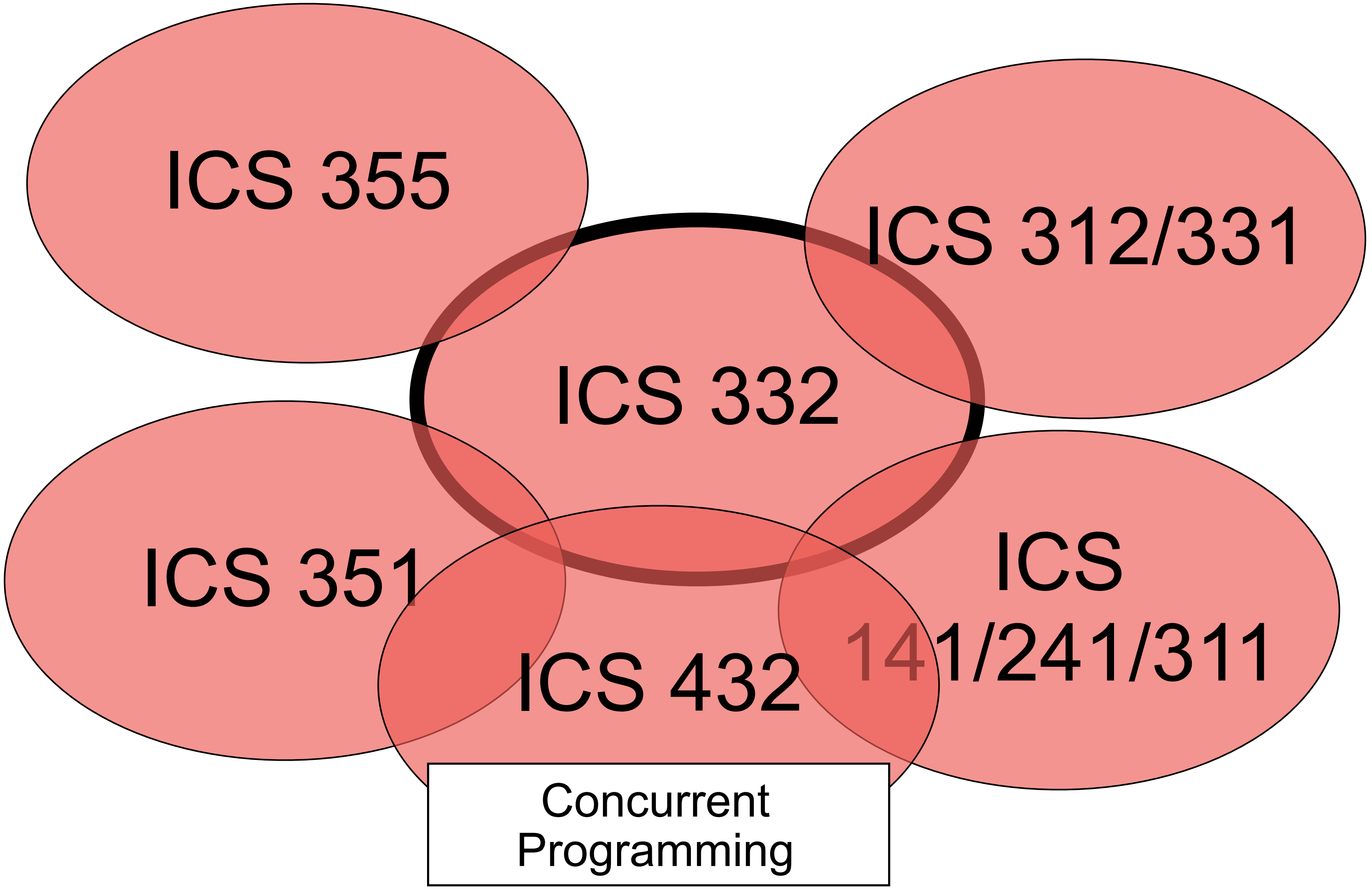
ICS332 and the ICS Curriculum



ICS332 and the ICS Curriculum



ICS332 and the ICS Curriculum



Course Website

- Located at:
  - https://courses.ics.hawaii.edu/ics332_spring2026/
  - Linked from my personal homepage
    - Google for “Henri Casanova”
- Organized as Modules
  - All lecture notes as PDF files
  - Pointers to useful on-line material
  - All assignments
  - Announcements
  - A link to the Syllabus
    - Which we’re going over now in these slides
- Let’s look at the Web site...

Textbook

- Operating Systems: Three Easy Pieces (a.k.a. OSTEP) 1.00 by Arpaci-Dusseau, R. H. and Arpaci-Dusseau, A. C
  - Freely available!
- Lectures are tightly connected to particular chapters therein
- There will be reading assignments from this textbook, as indicated on the lecture notes
  - Up to you whether you prefer to read them before or after our lectures....
- Some exam questions and assignments will be directly from or inspired by the textbook
- There are several classic texts for Operating Systems (see the “syllabus” page on the course Web site)

Lecture Notes

- Lecture notes are posted on the course’s Web site regularly
  - You can read them before or after the lecture, up to you really
  - I am notorious for spacing out on putting the notes up on the site, so just drop me a one-line e-mail

Inverted Lectures

- *A few* lectures will be "inverted"
  - You watch a screencast at your own pace
  - The lecture period is for questions and practice exercises
- I do this for a few topics in the course that are more "mechanical" or "difficult"
- You must watch the screencast ahead of time!
  - E-mails reminders will be sent out
- Scheduling may be imperfect
  - Out-of-order and/or overlapping modules
  - We might end a few lecture periods early

Screencast Lectures

- A few lectures will be screencast
  - This is because I am sometimes required to travel

- More information later…

Course Content

- In spite of my best efforts it happens that the course Web site could have small problems (typos, missing link, etc.)

- Anytime you see anything strange/broken on the Web site, please let me know right away!
  - A one-line e-mail, a DM on Discord, etc.

Grading on 1000 points

- Sample and optional homework assignments for 0 points

- Three exams
  - Midterm #1 exam (300 points)
  - Midterm #2 exam (300 points)
  - Final exam (330 points)

- Quizzes (70 points)
  - 8 10-point quizzes, worst grade is discarded

Homework Assignments

- **All homework assignments in this course are either "sample homework" or "optional homework", worth zero points**
- **Sample homework assignments:**
  - Posted as regular assignments would be
  - Solutions are provided on the assignment's page
  - You cannot turn them in
- **Optional homework assignments:**
  - Posted as regular assignments would be
  - You can turn them in and there is a due date
  - You will receive feedback
  - Solutions are available upon request after the due date
- Why? …

Homework Assignments

- Rationale for 0 points on homework assignments:
  - The use of LLMs has rendered homework unfair across students (blatantly seen last semester)
  - Students in this course used to “write a lot of code and struggle somewhat on assignments”, but this is no longer a thing
- The whole point of the course has always been to teach key concepts, not to attempt to make you assembly programming pros
  - Writing code was only a means-to-an-end for learning in this course anyway
  - Graded homework assignments used to be a way to force students to be prepared for exams

Homework Assignments

- **Can students learn the important concepts without going through the works of doing homework assignments?**
- Nobody really knows, some people think “absolutely” and some think “absolutely not”
- I think it completely depends on the student:
  - For some of you, not attempting the homework assignments or practice problems on your own will lead to catastrophic results
  - Some of you will ace all exams regardless
- The assumption in this “new world” is now that students are adults and know what they need to do to pass exams 😬
- We will do quite a bit of practice and live-coding in class!
  - And you should never hesitate asking “can we live-code this?” during lecture

Exams

- Exams are taken in class, closed-note
- Pocket calculators, not programmable calculators or phones, allowed (but not needed)
- Exams are randomly generated and students have different exams
  - So don’t cheat with your neighbors, it’s super obvious if you do (and it always happens!)
- Each exam will have mostly exercises that match exactly homework assignments and practice problems
- The final exam will have one selected exercise from each of the previous midterm exams
  - And I’ll tell you which kind of exercise

Quizzes

- 8 Quizzes in the semester
- Taken on the first lecture day of the week
  - Always on a Tuesday, unless that Tuesday is a holiday, in which case it will be on a Thursday
- Always **announced the previous week ON THE COURSE'S WEB SITE**
- Taken at the **beginning of the lecture period, in the first 10 minutes**
  - You **cannot take the quiz if you show up more than 5 minutes late** to the lecture
- No make-up quizzes, unless a documented reason
- But **the worst quiz grade is discarded**

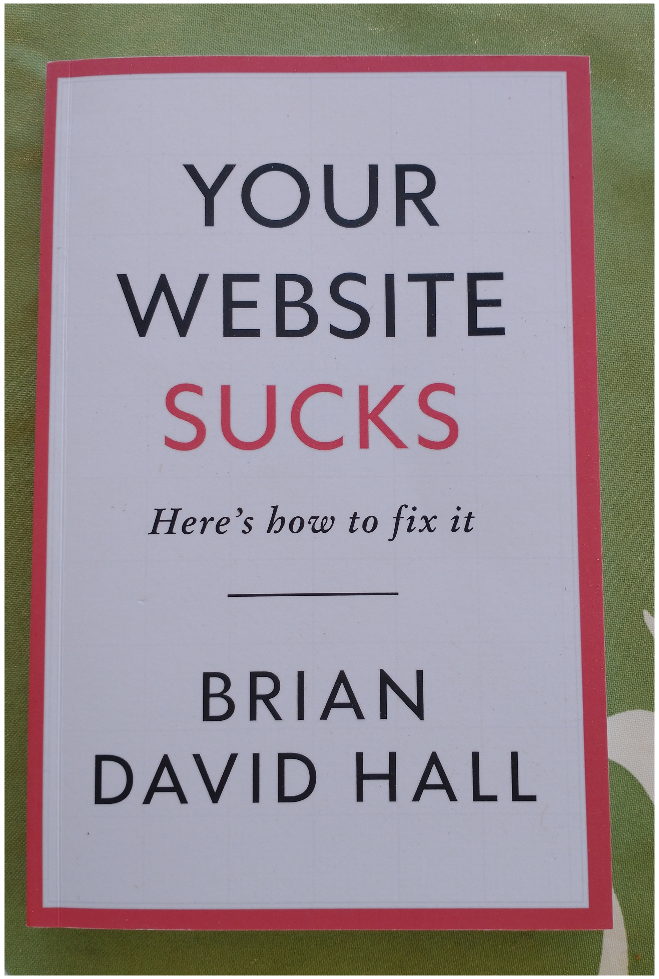
CES Evaluation

- Extra credit given to all students:
  - 0 points if CES completion rate is < 80%
  - 5 points if CES completion rate is >= 80%
  - 10 points if CES completion rate is 90%
- Why?
  - I do look at the evaluation every semester and evolve the course accordingly
    - Even if you love the course, it’s important for me to hear what things didn’t work
  - These evaluations have more impact than you may think and are taken seriously
    - Impact for individual faculty, for the whole department, for future students, etc.

The Linux CLI (Shell)

- Knowledge of the UNIX/Linux CLI (Shell) is needed in this course
- Show of hand: who feels somewhat familiar with the Unix/Linux CLI?
- A huge potential side-benefit of taking this course is to become decent/good with the command-line
  - About 25% of students passing the course tell me that they are forever grateful that they forced themselves to learn/use the Shell
  - This is also something we hear from alumni who, once in a job, “discover” that 99% of back-end systems are Linux based and using the Shell is a daily activity
- This module (Getting Started) contains some pointers. Let’s look at them now...
  - Many of you really want to look at this material to prepare for upcoming programming assignments
- In case you’re not convinced …..

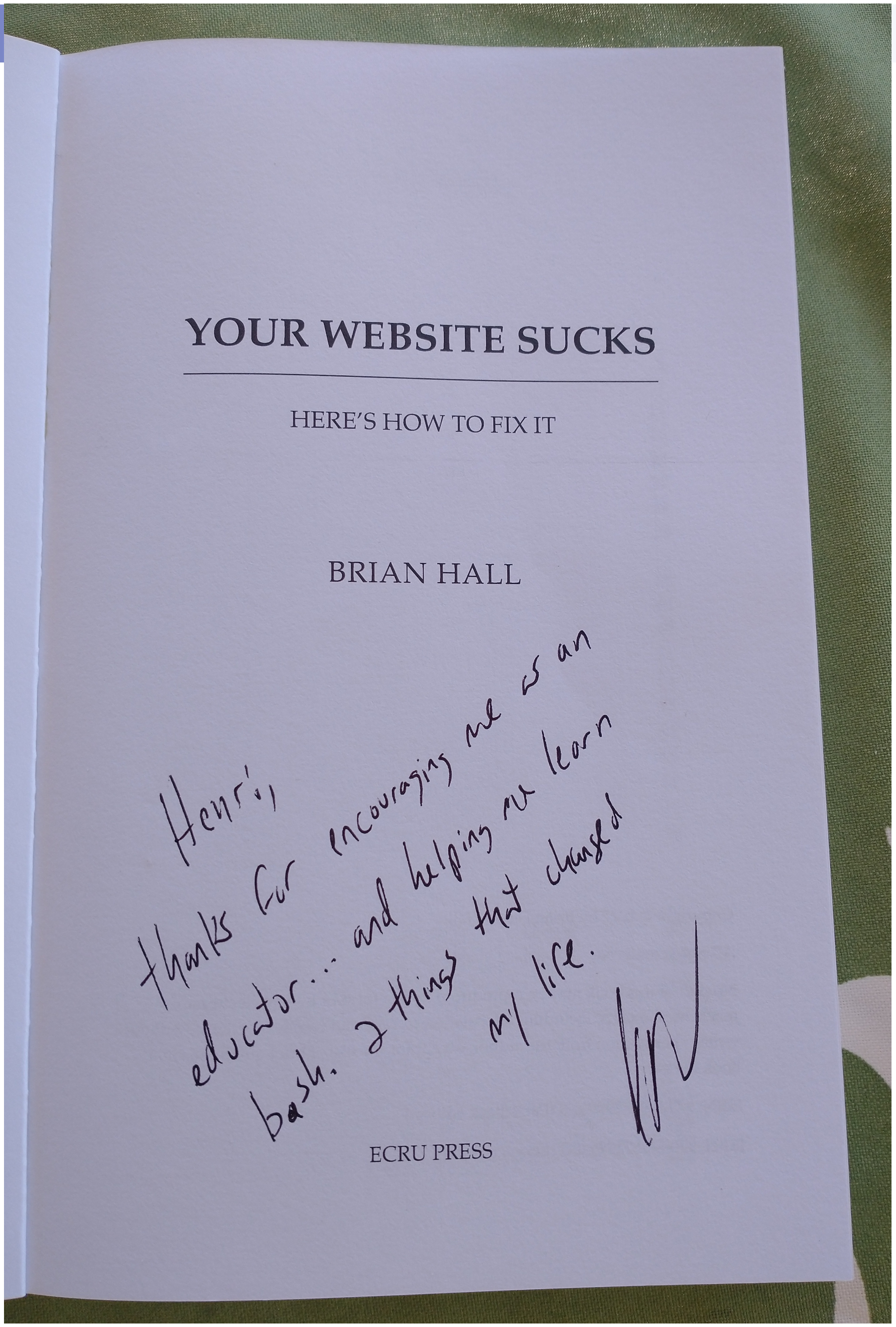
ICS Alumnus Brian Hall



YOUR WEBSITE SUCKS

Here's how to fix it

BRIAN DAVID HALL



YOUR WEBSITE SUCKS

HERE'S HOW TO FIX IT

BRIAN HALL

Henri,
thanks for encouraging me as an educator... and helping me learn bash. 2 things that changed my life.

[illegible]

ECRU PRESS

Brian Hall <bdh@briandavidhall.com>
Email him to request access to his Udemy course on the CLI (for free!)

On the Importance of Terminology

- As we learn about Operating Systems we will encounter a lot of terminology
- Recognizing and using the correct terminology is part of what we learn in this course
- Knowing the terminology is very important (e.g., in a job interview, in a professional context)
- So, in class, whenever a student asks a question (which is highly encouraged!), I might rephrase the question using proper terminology
  - This is not to be annoying or belittling, it’s to make sure we all come out of this course speaking the language of Operating Systems (and of Computer Science)
- Of course, terminology will be part of the quizzes/exams

How to not do well in this course?

- **Don’t come to class (“the slides are nice”)**
  - We do a LOT of stuff in class, including live coding, and I give a lot of explanations, examples
- **Don’t attempt the homework assignments or practice problems**
  - Putting in time on those is the best way to learn the material
- **Don’t come to office hours (“The instructor is scary because he shows ‘how to no do well in this course?’ slides”)**
  - After you struggle for a while on something, drop by
  - Instructor and TA office hours are an amazing service provided to you, and yet, they go mostly unused

How to not do well in this course?

- **Cheat**
  - Cheating is bad for many reasons, including hurting the reputation of ICS graduates!
  - If you are caught cheating or enabling cheating:
    - zero on the assignment/exam
    - overall grade lowered by a step (i.e., a “B” becomes a “C”)
    - reported to UH’s Office of Judicial Affairs (as required)
- **Expect that “what can I do for extra credit?” will be met with a positive response**
- **Don’t study for the quizzes**
  - “It’s only a small fraction of the grade”
  - But studying for quizzes is a HUGE help to prepare for exams
  - When I don’t do quizzes, the average grade drops!

Software/Hardware for ICS332

- You'll have to use a Linux "machine"
- Let's look at Homework Assignment #0, which is ungraded but which you should do as soon as possible in the semester
  - If you intend to do any programming at all (which you really should!)

Questions?

- Any questions on the syllabus?
- Any questions on the course in general?

Participation Verification

- As you know, each instructor has to report on “Student Participation” and certify the class roster
  - If you have not “participated”, you could be dropped from the course

- **IMPORTANT:** Do the ungraded “Participation Verification” Assignment **posted on Lamakū**

Conclusion

- If no more questions, let’s adjourn for today
- But before let’s do an ungraded quiz!