L1: Course Intro & Logistics

Bernard Nongpoh

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Monday, 27 July 2026

Today

- **Why** we analyze programs
- A first look at **security, vulnerabilities & exploits**
- The **state of the art** in finding bugs and writing exploits
- **What** the course covers
- **How** it runs: logistics, grading, the project

Software runs the world.

Software runs the world.

Finance. Healthcare. Defence. Infrastructure. Communication.

Big Software, Big Risk

Linux Kernel: **36.9 million** lines of C, 4,037 contributors[1]

Chromium: **36.2 million** lines of C++, 2,109 contributors[2]

[1] openhub.net/p/linux [2] openhub.net/p/chrome [3] CISA CSAC Report, Dec 2023

Big Software, Big Risk

Linux Kernel: **36.9 million** lines of C, 4,037 contributors[1]

Chromium: **36.2 million** lines of C++, 2,109 contributors[2]

More code → more complexity → **more bugs**

And some of those bugs are security vulnerabilities.

[1] openhub.net/p/linux [2] openhub.net/p/chrome [3] CISA CSAC Report, Dec 2023

Big Software, Big Risk

Linux Kernel: **36.9 million** lines of C, 4,037 contributors[1]

Chromium: **36.2 million** lines of C++, 2,109 contributors[2]

More code → more complexity → **more bugs**

And some of those bugs are security vulnerabilities.

70% of security issues are memory-safety bugs[3]

[1] openhub.net/p/linux [2] openhub.net/p/chrome [3] CISA CSAC Report, Dec 2023

Real-World Impact

Heartbleed (2014): OpenSSL buffer over-read. Leaked private keys worldwide.[1]

[1] heartbleed.com [2] CVE-2021-44228 [3] Times of India, 2025

Real-World Impact

Heartbleed (2014): OpenSSL buffer over-read. Leaked private keys worldwide.[1]

Log4Shell (2021): Java logging library. Remote code execution on millions of servers.[2]

[1] heartbleed.com [2] CVE-2021-44228 [3] Times of India, 2025

Real-World Impact

Heartbleed (2014): OpenSSL buffer over-read. Leaked private keys worldwide.[1]

Log4Shell (2021): Java logging library. Remote code execution on millions of servers.[2]

IIT Roorkee (2025): 30,000 student and alumni records exposed for years.[3]

[1] heartbleed.com [2] CVE-2021-44228 [3] Times of India, 2025

Real-World Impact

Heartbleed (2014): OpenSSL buffer over-read. Leaked private keys worldwide.[1]

Log4Shell (2021): Java logging library. Remote code execution on millions of servers.[2]

IIT Roorkee (2025): 30,000 student and alumni records exposed for years.[3]

One insecure line can compromise **millions** of systems.

[1] heartbleed.com [2] CVE-2021-44228 [3] Times of India, 2025

What Is Software Security?

Ensuring a program behaves correctly and safely even under **adversarial** conditions — not just on well-behaved inputs.

[1] NIST CSRC — CIA triad

What Is Software Security?

Ensuring a program behaves correctly and safely even under **adversarial** conditions — not just on well-behaved inputs.

Classically framed as protecting the **C-I-A triad**[1]:

Confidentiality (no leaks) · **Integrity** (no tampering) · **Availability** (stays up)

[1] NIST CSRC — CIA triad

What Is Software Security?

Ensuring a program behaves correctly and safely even under **adversarial** conditions — not just on well-behaved inputs.

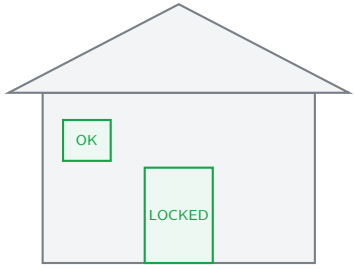
Classically framed as protecting the **C-I-A triad**[1]:

Confidentiality (no leaks) · **Integrity** (no tampering) · **Availability** (stays up)

The goal: find and fix the flaws **before an attacker does**.

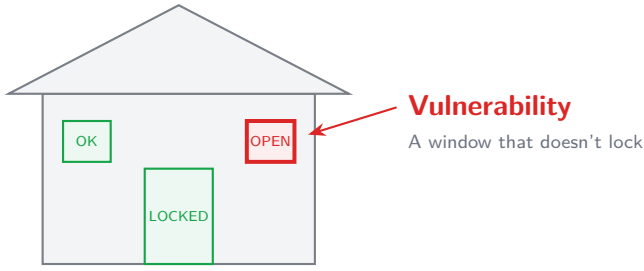
[1] NIST CSRC — CIA triad

A Vulnerability Is a Flaw in Software



Your Software

A Vulnerability Is a Flaw in Software



A **weakness** in software that *could* be exploited to cause harm.

Unintentional. But attackers hunt for exactly these.

Cataloging Vulnerabilities: CWE & CVE

The community tracks flaws systematically so we can name, count, and study them:

[1] cwe.mitre.org [2] cve.org

Cataloging Vulnerabilities: CWE & CVE

The community tracks flaws systematically so we can name, count, and study them:

- **CWE** (Common Weakness Enumeration) — **classes** of weakness *(buffer overflow, SQL injection, use-after-free, . . .)*[1]

[1] cwe.mitre.org [2] cve.org

Cataloging Vulnerabilities: CWE & CVE

The community tracks flaws systematically so we can name, count, and study them:

- **CWE** (Common Weakness Enumeration) — **classes** of weakness *(buffer overflow, SQL injection, use-after-free, . . .)*[1]
- **CVE** (Common Vulnerabilities and Exposures) — **specific** disclosed vulnerabilities in real products[2]

[1] cwe.mitre.org [2] cve.org

Cataloging Vulnerabilities: CWE & CVE

The community tracks flaws systematically so we can name, count, and study them:

- **CWE** (Common Weakness Enumeration) — **classes** of weakness *(buffer overflow, SQL injection, use-after-free, . . .)*[1]
- **CVE** (Common Vulnerabilities and Exposures) — **specific** disclosed vulnerabilities in real products[2]

CWE is the taxonomy of *kinds* of bugs; CVE is the ledger of *actual* ones found in software/hardware.

[1] cwe.mitre.org [2] cve.org

From Vulnerability to Exploit to Impact

Vulnerability

The flaw

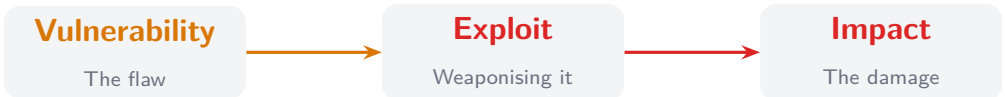
The vulnerability is the **unlocked window**.

From Vulnerability to Exploit to Impact

The vulnerability is the **unlocked window**.

The exploit is the **burglar climbing through**.

From Vulnerability to Exploit to Impact



The vulnerability is the **unlocked window**.

The exploit is the **burglar climbing through**.

The impact is **everything they steal**.

The State of the Art: Finding Bugs

Scale. ∼40,000 new CVEs were published in 2025 — a record, and rising.[1]

[1] cve.org/About/Metrics [2] OSS-Fuzz [3] Google Project Zero, 2024

The State of the Art: Finding Bugs

Scale. ∼40,000 new CVEs were published in 2025 — a record, and rising.[1]

Automation. Google's **OSS-Fuzz** has found 10,000+ vulnerabilities across 1,000+ open-source projects.[2]

[1] cve.org/About/Metrics [2] OSS-Fuzz [3] Google Project Zero, 2024

The State of the Art: Finding Bugs

Scale. ∼40,000 new CVEs were published in 2025 — a record, and rising.[1]

Automation. Google's **OSS-Fuzz** has found 10,000+ vulnerabilities across 1,000+ open-source projects.[2]

AI arrives. In 2024, Google's LLM agent "**Big Sleep**" found a real, previously-unknown vulnerability in SQLite — automatically.[3]

[1] cve.org/About/Metrics [2] OSS-Fuzz [3] Google Project Zero, 2024

Three Ways to Analyze Code

Manual review **doesn't scale**. We build automated tools instead:

Three Ways to Analyze Code

Manual review **doesn't scale**. We build automated tools instead:

- **Static analysis** — inspect code *without* running it

Three Ways to Analyze Code

Manual review **doesn't scale**. We build automated tools instead:

- **Static analysis** — inspect code *without* running it
- **Dynamic analysis** — *run* the code and watch what happens

Three Ways to Analyze Code

Manual review **doesn't scale**. We build automated tools instead:

- **Static analysis** — inspect code *without* running it
- **Dynamic analysis** — *run* the code and watch what happens
- **Hybrid** — combine the two

What This Course Is About

A course on **static, dynamic, and hybrid** program analysis — and using it to find bugs in software.

What This Course Is About

A course on **static, dynamic, and hybrid** program analysis — and using it to find bugs in software.

Hands-on assignments on **LLVM**, **Static, Dynamic Analysis**, and modern fuzzers — the same tools used in industry and research.

Course Roadmap — 9 Topics

#	Topic	Lectures
1	**Foundations & Program Representation**	L1–L8
2	**Testing** (fuzzing, coverage-guided, concurrency)	L9–L12
3	**Dataflow Analysis**	L13–L20
4	**Pointer Analysis**	L21–L24
5	**Constraint-Based Analysis**	L25–L28
6	**Type Systems**	L29–L32
7	**Symbolic Execution**	L33–L36
8	**Automated Test Generation**	L37–L39
9	**LLM-Assisted Software Analysis**	L40–L42

Meets Mon / Tue / Wed · 23 Jul – 12 Nov 2026 · full schedule on the course page

A Glimpse of the Frontier: LLMs Finding Vulnerabilities

LLM **agents** can now read code, hypothesise flaws, and confirm them — finding **real, previously-unknown** bugs on their own.

[1] Google, Big Sleep [2] xbow.com

A Glimpse of the Frontier: LLMs Finding Vulnerabilities

LLM **agents** can now read code, hypothesise flaws, and confirm them — finding **real, previously-unknown** bugs on their own.

- **Big Sleep** (Google) — caught a live-exploited SQLite vulnerability before attackers could use it (2025).[1]

[1] Google, Big Sleep [2] xbow.com

A Glimpse of the Frontier: LLMs Finding Vulnerabilities

LLM **agents** can now read code, hypothesise flaws, and confirm them — finding **real, previously-unknown** bugs on their own.

- **Big Sleep** (Google) — caught a live-exploited SQLite vulnerability before attackers could use it (2025).[1]
- **XBOW** — an autonomous AI hacker that topped the HackerOne US leaderboard (2025).[2]

[1] Google, Big Sleep [2] xbow.com

How the Course Runs

Grading

End-semester exam	**30%**
Mid-semester exam	**25%**
Quizzes ($2 \times 5\%$)	**10%**
Term project	**20%**
H/W assignments	**10%**
Attendance	**5%**
Total	**100%**

The Term Project

Pick a **widely-used** open-source target, build or adapt a tool from the course, and find bugs that get **accepted upstream**.

e.g. Linux, Chromium, FFmpeg, OpenSSL, SQLite, Postgres, CPython, LLVM...

The Term Project

Pick a **widely-used** open-source target, build or adapt a tool from the course, and find bugs that get **accepted upstream**.

e.g. Linux, Chromium, FFmpeg, OpenSSL, SQLite, Postgres, CPython, LLVM...

Teams: **TBA**. Deliverable: a short FSE-format paper + a reproducible GitHub artifact.

The Term Project

Pick a **widely-used** open-source target, build or adapt a tool from the course, and find bugs that get **accepted upstream**.

e.g. Linux, Chromium, FFmpeg, OpenSSL, SQLite, Postgres, CPython, LLVM...

Teams: **TBA**. Deliverable: a short FSE-format paper + a reproducible GitHub artifact.

Proposal due right after the mid-sem (**20 Sep 2026**); get your target approved before you start.

Logistics & Policies

Classes. Mon / Tue / Wed in Room 5101, Core 5 (Mon 5:00, Tue 4:00, Wed 3:00 PM — 55 min each).

Thursday 2:00 PM slot reserved for quizzes & industry talks.

Prerequisites. Data structures; C/C++ or Java; basic OS & compilers.

Contact. Office hours by appointment (email instructor or TAs); Microsoft Teams for discussion.

Website. `https://bernardnongpoh.github.io/cs-5187/`

Policies. Discuss freely, write up individually, cite your sources. LLM use must be **disclosed**

Your Teaching Team

Instructor. Bernard Nongpoh bnongpoh@iitg.ac.in

Teaching Assistants.

Shruti Yadav	shruti.yadav@iitg.ac.in
Kunal Upadhyay	k.upadhyay@iitg.ac.in
Rhythm Jamwal	rhythm.jamwal@iitg.ac.in
Yatharth Sonteke	y.sonteke@iitg.ac.in
Leisel Rangsuo Koireng	leisel.koireng@iitg.ac.in

Welcome to CS 5187.

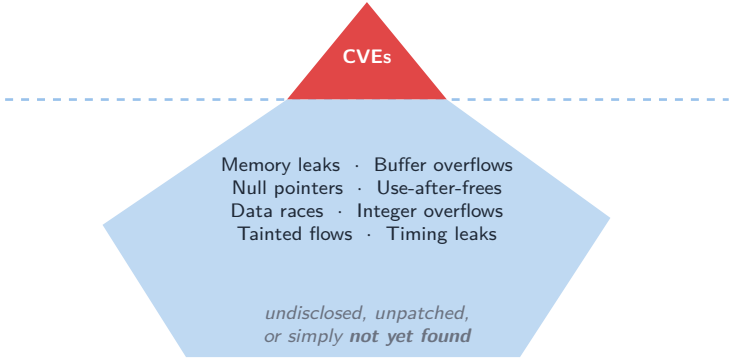
Welcome to CS 5187.

By the end of the semester, your tool will have found a bug in software that millions of people use.

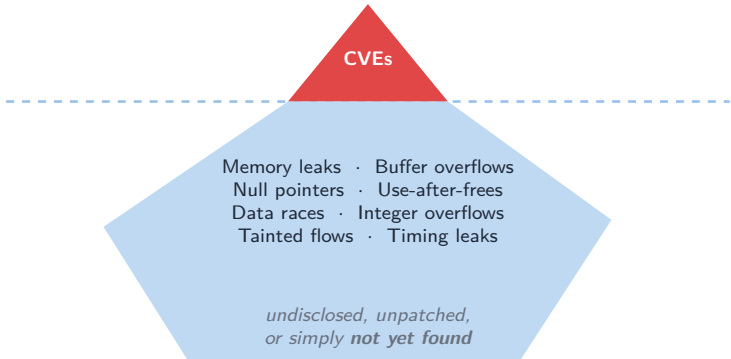
Why analysis matters more than ever

Public Vulnerabilities Are the Tip of the Iceberg

Public Vulnerabilities Are the Tip of the Iceberg



Public Vulnerabilities Are the Tip of the Iceberg



∼40,000 CVEs published in 2024[1] — and that is only what somebody **found**, **reported**, and **published**.

[1] cve.org/About/Metrics

The Shift: AI Agents Are Writing the Code Now

- Coding agents — **Claude Code**, **Codex**, **OpenCode** — now scaffold entire web and mobile apps from a prompt.

[1] Stack Overflow Developer Survey 2025 [2] State of Code 2025

The Shift: AI Agents Are Writing the Code Now

- Coding agents — **Claude Code**, **Codex**, **OpenCode** — now scaffold entire web and mobile apps from a prompt.
- **84%** of developers use or plan to use AI tools — up from 76% in 2024.[1]

[1] Stack Overflow Developer Survey 2025 [2] State of Code 2025

The Shift: AI Agents Are Writing the Code Now

- Coding agents — **Claude Code**, **Codex**, **OpenCode** — now scaffold entire web and mobile apps from a prompt.
- **84%** of developers use or plan to use AI tools — up from 76% in 2024.[1]
- Roughly **41–42%** of code written in 2025 was AI-generated or AI-assisted.[2]

[1] Stack Overflow Developer Survey 2025 [2] State of Code 2025

The Shift: AI Agents Are Writing the Code Now

- Coding agents — **Claude Code**, **Codex**, **OpenCode** — now scaffold entire web and mobile apps from a prompt.
- **84%** of developers use or plan to use AI tools — up from 76% in 2024.[1]
- Roughly **41–42%** of code written in 2025 was AI-generated or AI-assisted.[2]
- Result: **more code than ever**, produced **faster than ever**, and understood by **nobody in particular**.

[1] Stack Overflow Developer Survey 2025 [2] State of Code 2025

Meme :)

Activity	Before AI
Writing code	2 hours
Debugging	6 hours

Meme :)

Activity	Before AI	After AI
Writing code	2 hours	5 minutes
Debugging	6 hours	24 hours

The Trust Gap

Developers are adopting AI **faster** than they trust it. From the 2025 Stack Overflow survey of ~49,000 developers:[1]

[1] Stack Overflow Developer Survey 2025

The Trust Gap

Developers are adopting AI **faster** than they trust it. From the 2025 Stack Overflow survey of ~49,000 developers:[1]

- **66%** — biggest frustration is "AI solutions that are **almost right, but not quite**".

[1] Stack Overflow Developer Survey 2025

The Trust Gap

Developers are adopting AI **faster** than they trust it. From the 2025 Stack Overflow survey of ~49,000 developers:[1]

- **66%** — biggest frustration is "AI solutions that are **almost right, but not quite**".
- **45%** — "debugging AI-generated code is **more time-consuming**".

[1] Stack Overflow Developer Survey 2025

The Trust Gap

Developers are adopting AI **faster** than they trust it. From the 2025 Stack Overflow survey of ~49,000 developers:[1]

- **66%** — biggest frustration is "AI solutions that are **almost right, but not quite**".
- **45%** — "debugging AI-generated code is **more time-consuming**".
- **46%** actively **distrust** AI accuracy; only 33% trust it. Trust *fell* year on year.

[1] Stack Overflow Developer Survey 2025

The Trust Gap

Developers are adopting AI **faster** than they trust it. From the 2025 Stack Overflow survey of ~49,000 developers:[1]

- **66%** — biggest frustration is "AI solutions that are **almost right, but not quite**".
- **45%** — "debugging AI-generated code is **more time-consuming**".
- **46%** actively **distrust** AI accuracy; only 33% trust it. Trust *fell* year on year.

"Almost right" is the **worst** failure mode — it passes review and fails in production.

[1] Stack Overflow Developer Survey 2025

Principled Analysis Matters More Than Ever

In the age of AI coding agents, principled ways to **scan**, **check**, and **reason about** code are more important than ever.

Principled Analysis Matters More Than Ever

In the age of AI coding agents, principled ways to **scan**, **check**, and **reason about** code are more important than ever.

We must **assess** the security, correctness, and limitations of automatically generated code — rather than simply **trusting what the AI produces**.

Principled Analysis Matters More Than Ever

In the age of AI coding agents, principled ways to **scan**, **check**, and **reason about** code are more important than ever.

We must **assess** the security, correctness, and limitations of automatically generated code — rather than simply **trusting what the AI produces**.

The generator got faster. The **verifier** did not get less necessary — it got **more**.

Automated Code Analysis and Verification

Automatically analysing and assuring the behaviour of programs with respect to a property — *correctness*, *robustness*, *safety*, *liveness*.

Software Analysis

(this course)

Find the **existence** of bugs.

If **a path exists** where a bug may be triggered, report it.

Automated Code Analysis and Verification

Automatically analysing and assuring the behaviour of programs with respect to a property — *correctness*, *robustness*, *safety*, *liveness*.

Software Analysis

(this course)

Find the **existence** of bugs.

If **a path exists** where a bug may be triggered, report it.

Software Verification

(the stronger claim)

Prove the **absence** of bugs.

For **all paths**, the specification holds and no bug triggers.

Automated Code Analysis and Verification

Automatically analysing and assuring the behaviour of programs with respect to a property — *correctness*, *robustness*, *safety*, *liveness*.

Software Analysis

(this course)

Find the **existence** of bugs.

If **a path exists** where a bug may be triggered, report it.

Software Verification

(the stronger claim)

Prove the **absence** of bugs.

For **all paths**, the specification holds and no bug triggers.

"*There exists a path...*" versus "*For all paths...*" — one quantifier apart.

Are We Building the Right Thing — and Building It Right?

Analysis and verification both serve one question: **have we made what we were trying to make?**

Are We Building the Right Thing — and Building It Right?

Analysis and verification both serve one question: **have we made what we were trying to make?**

- Are we building the system **right**?

Are We Building the Right Thing — and Building It Right?

Analysis and verification both serve one question: **have we made what we were trying to make?**

- Are we building the system **right**?
- Does our design meet **user expectations**?

After Sui, COMP6131, UNSW

Are We Building the Right Thing — and Building It Right?

Analysis and verification both serve one question: **have we made what we were trying to make?**

- Are we building the system **right**?
- Does our design meet **user expectations**?
- Does the implementation conform to the **specification**?

After Sui, COMP6131, UNSW

Are We Building the Right Thing — and Building It Right?

Analysis and verification both serve one question: **have we made what we were trying to make?**

- Are we building the system **right**?
- Does our design meet **user expectations**?
- Does the implementation conform to the **specification**?

A program with no crashes that does the **wrong thing** is still broken. Correctness is always *relative to a specification*.

After Sui, COMP6131, UNSW