VERACODE

State of Software Security: Addressing the Threat of Security Debt

Chris Wysopal
Chief Security Evangelist

World AI Summit 2024

October 10, 2024

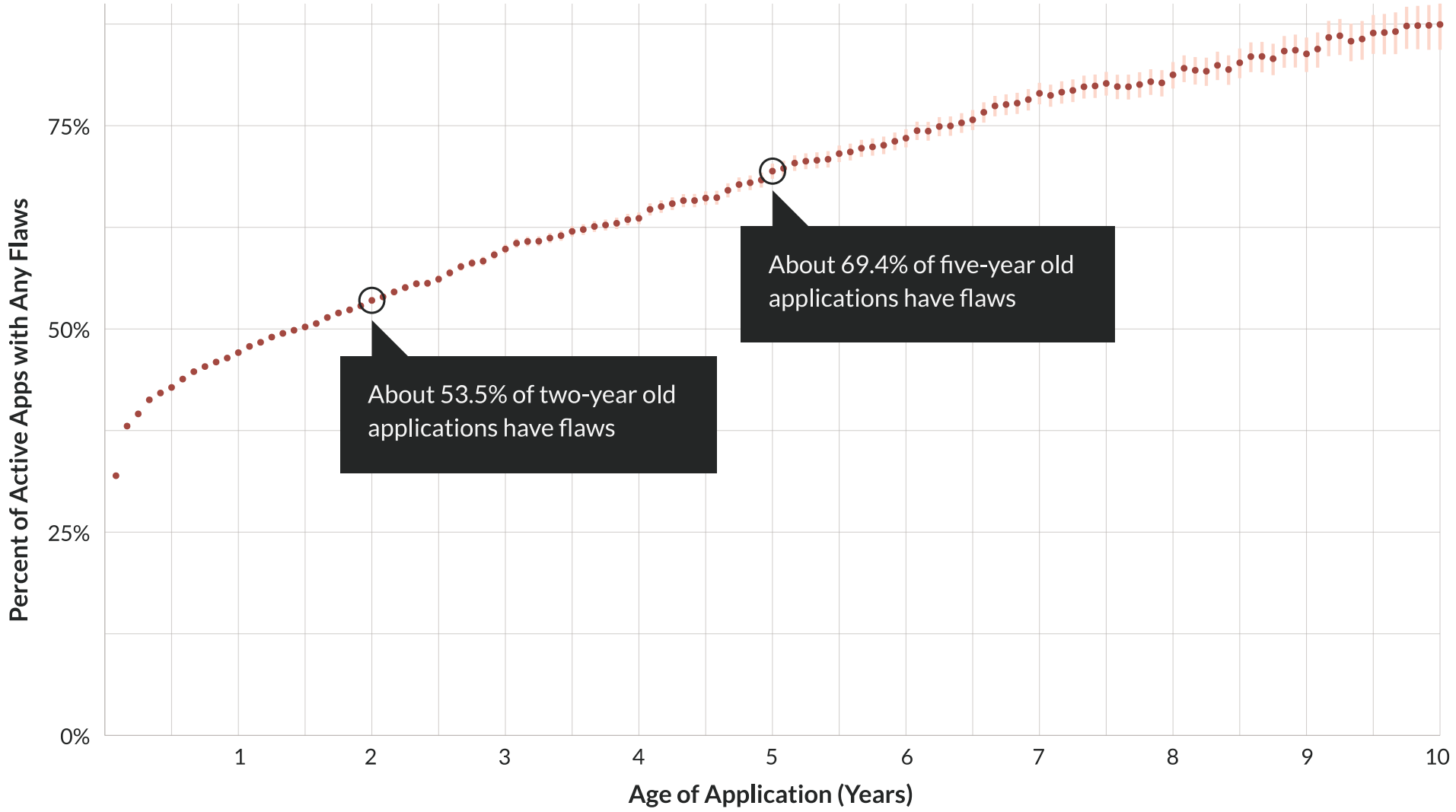
One of the 1st vulnerability researchers, member of hacker think tank, L0pht in 1990s

Unites States Senate testimony - 19 May 1998

VERACODE

Today we are finding software security flaws faster than we can fix them

Flaws accumulate faster than they're fixed



VERACODE

organizations are drowning in **security debt**

70.8%

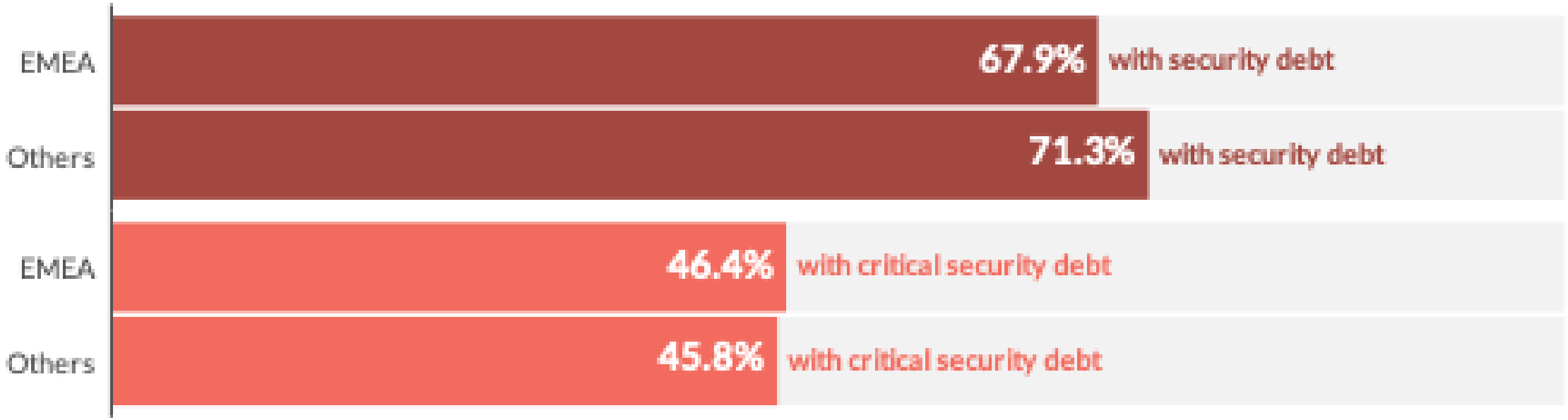
of organizations have security debt

45%

of organizations have critical security debt

* We are defining all flaws that remain unremediated for over one year, regardless of severity, as security debt.
**Critical debt: High-severity flaws that remain unremediated for over one year.

Our EU customers



VERACODE

why software security is **hard**

- security knowledge gaps
- increased application complexity
- incomplete view of risk
- evolving threat landscape

Let’s add the exciting potential of large language models that can write code!

Developer GenAI use right now

Generating code

Understanding code/Code review

Remediating defects

Translating programming languages

Creating and maintaining unit tests

Writing documentation

Emerging dev uses for GenAI

Learning about the code base

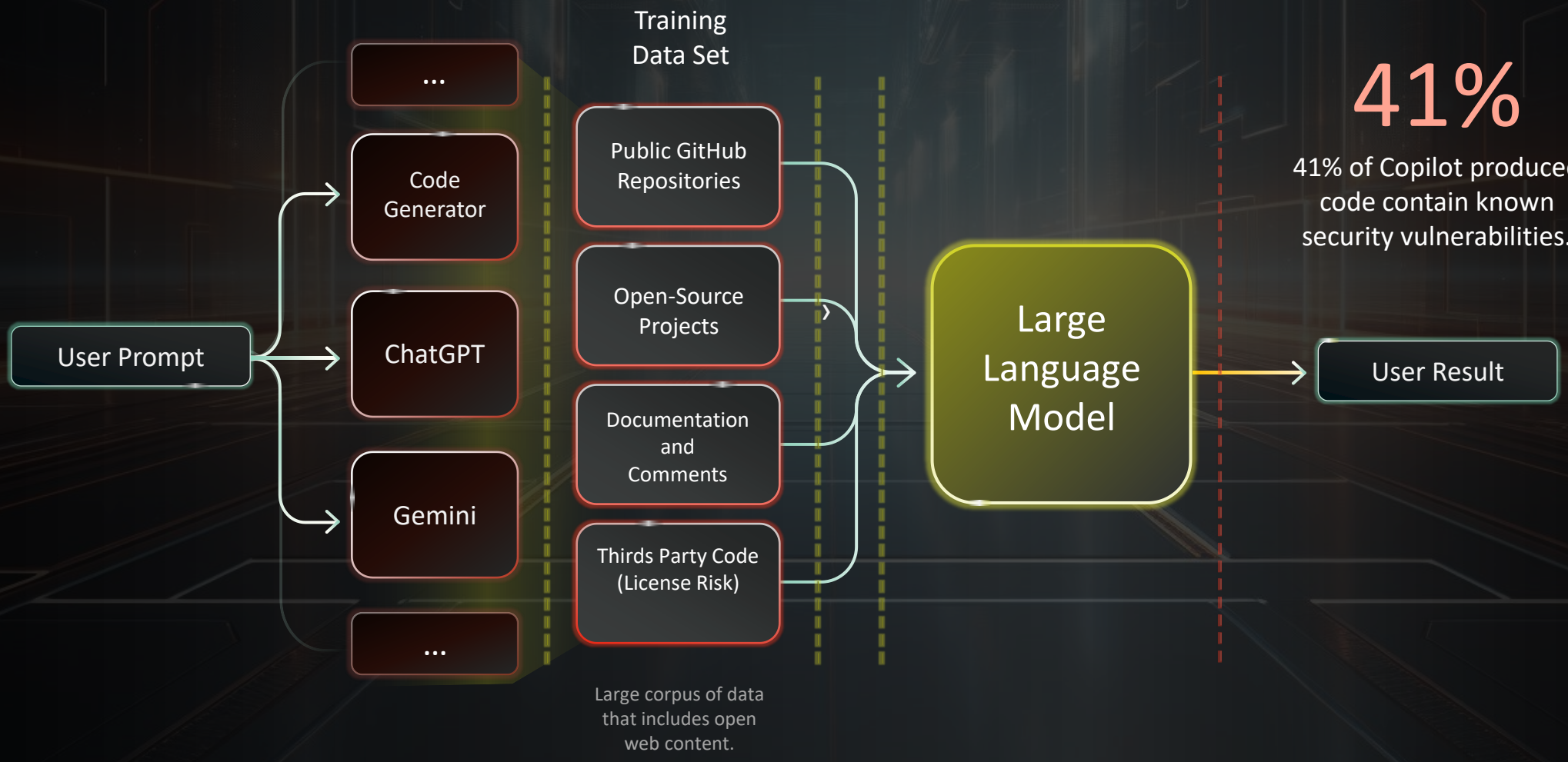
Searching for answers to avoid reinventing the wheel

Reading log files to find a root cause

Creating and running functional & non-functional tests

Remediating security vulnerabilities

Large Language Models used for coding



Security Implications of LLMs

Wuhan University Study
on AI Code Generators

36%

Out of the **435 Copilot** generated code snippets found in repos **36%** contain security weaknesses, across **6** programming languages.


Security Weaknesses of Copilot Generated Code in GitHub

Yujia Fu, Peng Liang, Amjed Tahir, Zengyang Li, Mojtaba Shahin, Jiaxin Yu, Jinfu Chen

arXiv:2310.02059v1 [cs.SE] 3 Oct 2023

New York University Study
on GitHub Copilot

41%

Of 1689 generated programs 41% of Copilot produced programs contained vulnerabilities


Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri

arXiv:2108.09293v3 [cs.CR] 16 Dec 2021

Stanford University Study
on AI Code Generators

Developers using LLMs were more likely to write insecure code.

They were more confident their code was secure.


Do Users Write More Insecure Code with AI Assistants?

Neil Perry, Megha Srivastava, Deepak Kumar, Dan Boneh

arXiv:2211.03622v3 [cs.CR] 18 Dec 2023

Purdue University
on ChatGPT accuracy

52%

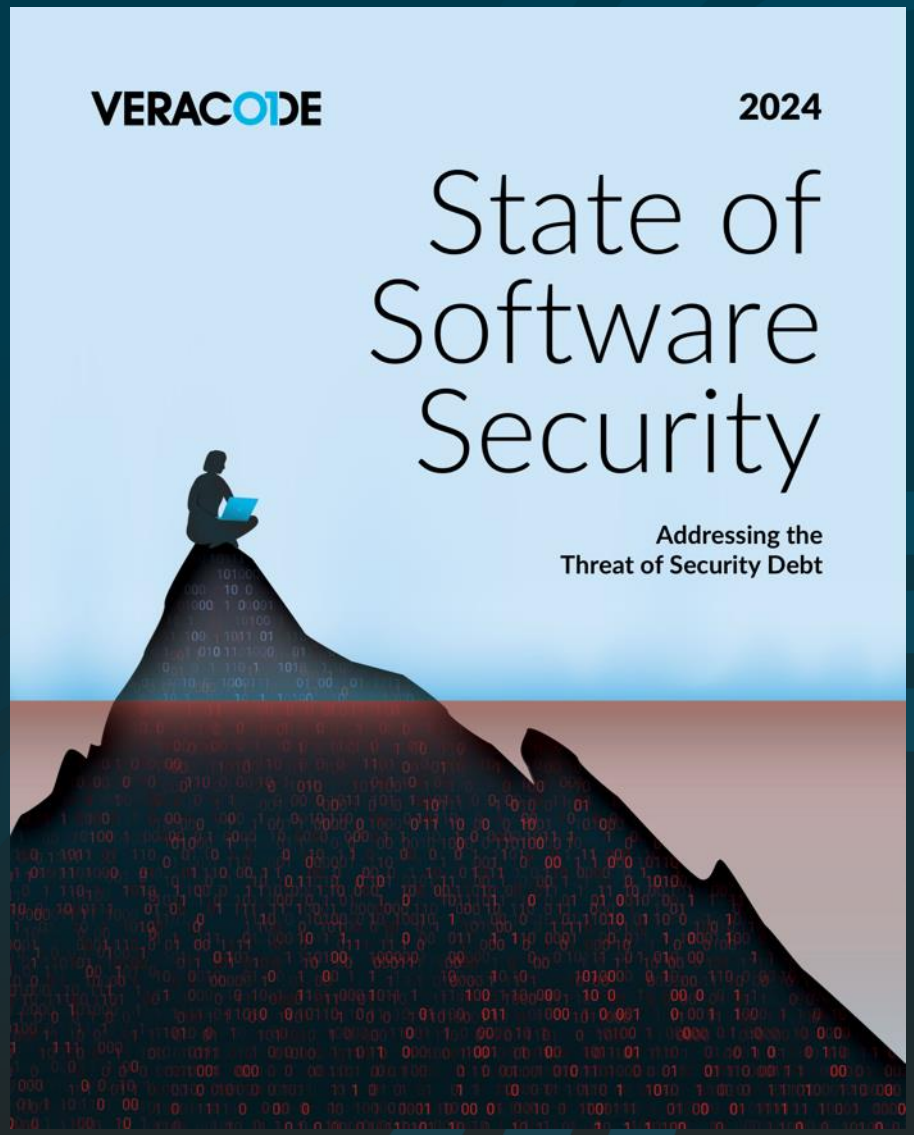
52% of ChatGPTs answers were incorrect.
Developers preferred them 35% of the time yet 77% of those answers were wrong


Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions

Samia Kabir, David N. Udo-Imeh, Bonan Kou, Tianyi Zhang

arXiv:2308.02312v3 [cs.SE] 10 Aug 2023

What is Veracode seeing across our customer base?



VERACODE

This research draws from the following:

1,007,133
applications across all scan types

1,553,022
dynamic analysis scans

11,429,365
static analysis scans

All those scans produced:

96.0 million
raw static findings

4.0 million
raw dynamic findings

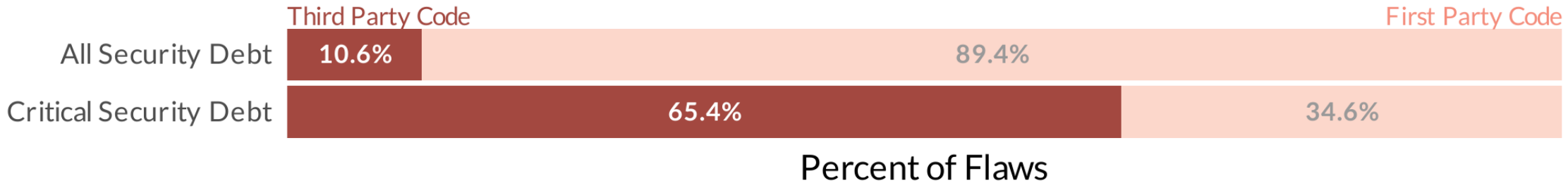
12.2 million
raw software composition analysis findings

Our approach and methodology

Where is the security debt?

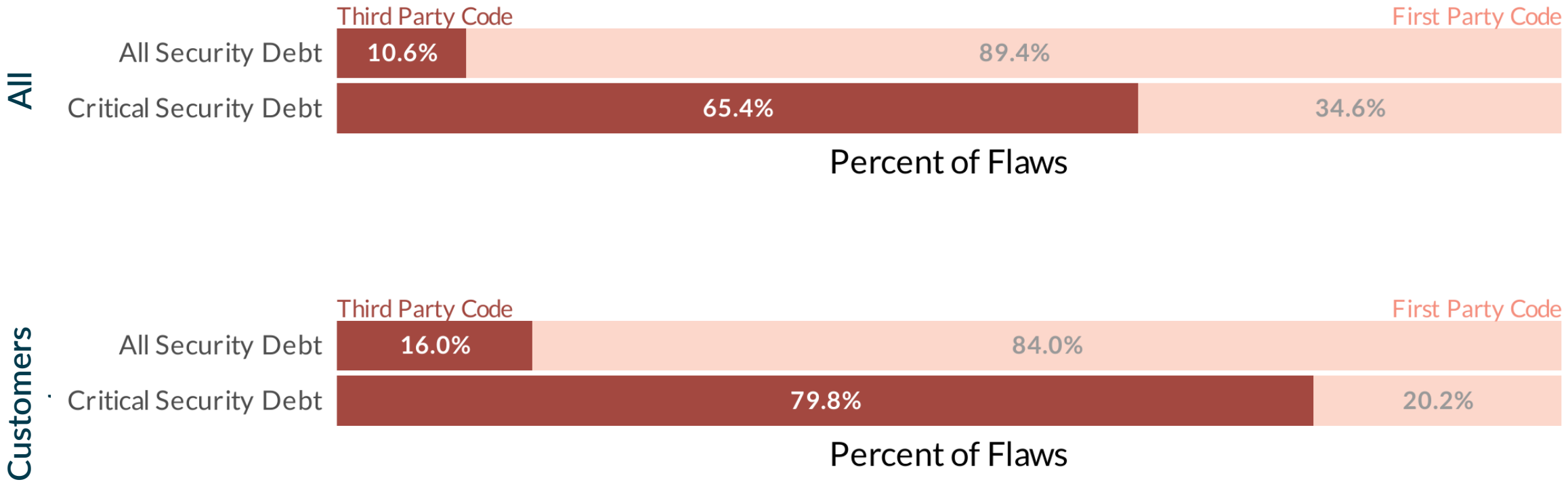
While first-party code constitutes almost **90% of all security debt**

65% of critical debt comes from **third-party code** in open-source libraries



VERACODE

EU customer breakdown is similar



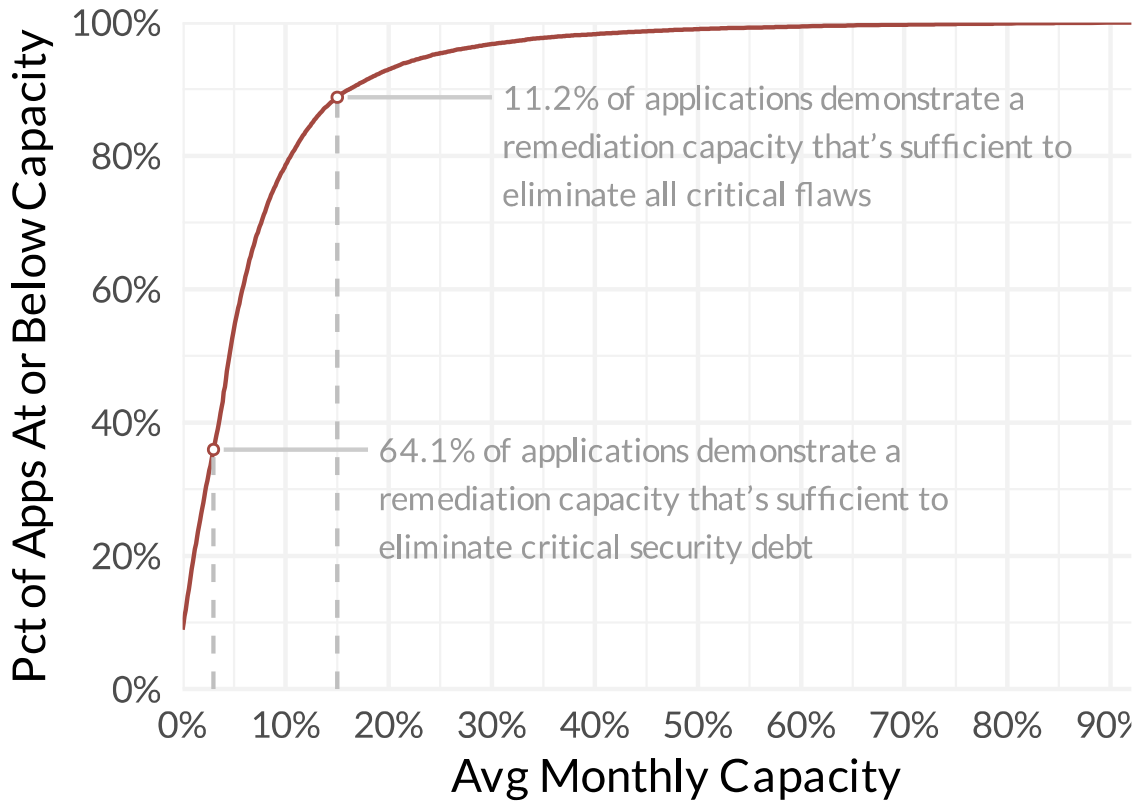
VERACODE

Remediation capacity is constrained

Only 64% of applications demonstrate a sustained capacity to eliminate all critical security debt.

Only two out of ten applications show an average monthly fix rate that exceeds ten percent of all security flaws.

This means few teams bail fast enough to reverse the tide of debt once it starts rising.



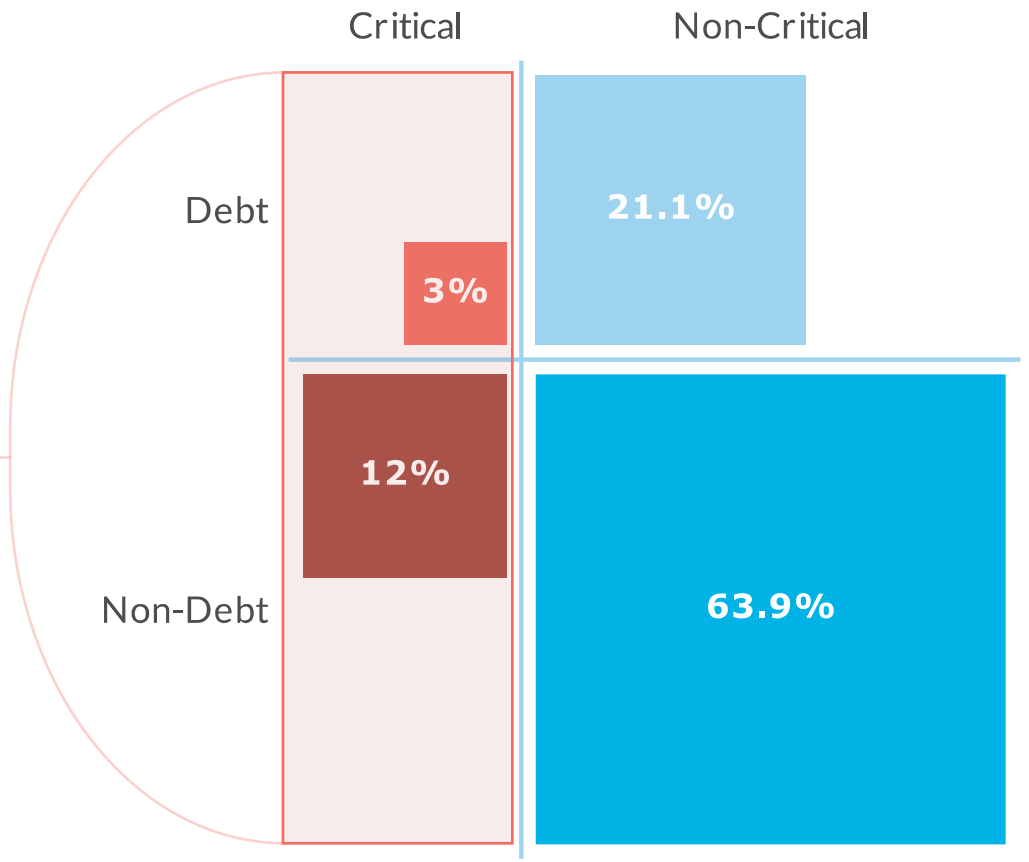
VERACODE

Prioritization is the key

If the rate of new and existing flaws exceeds the capacity to remediate them, then prioritizing which flaws to remediate is essential.

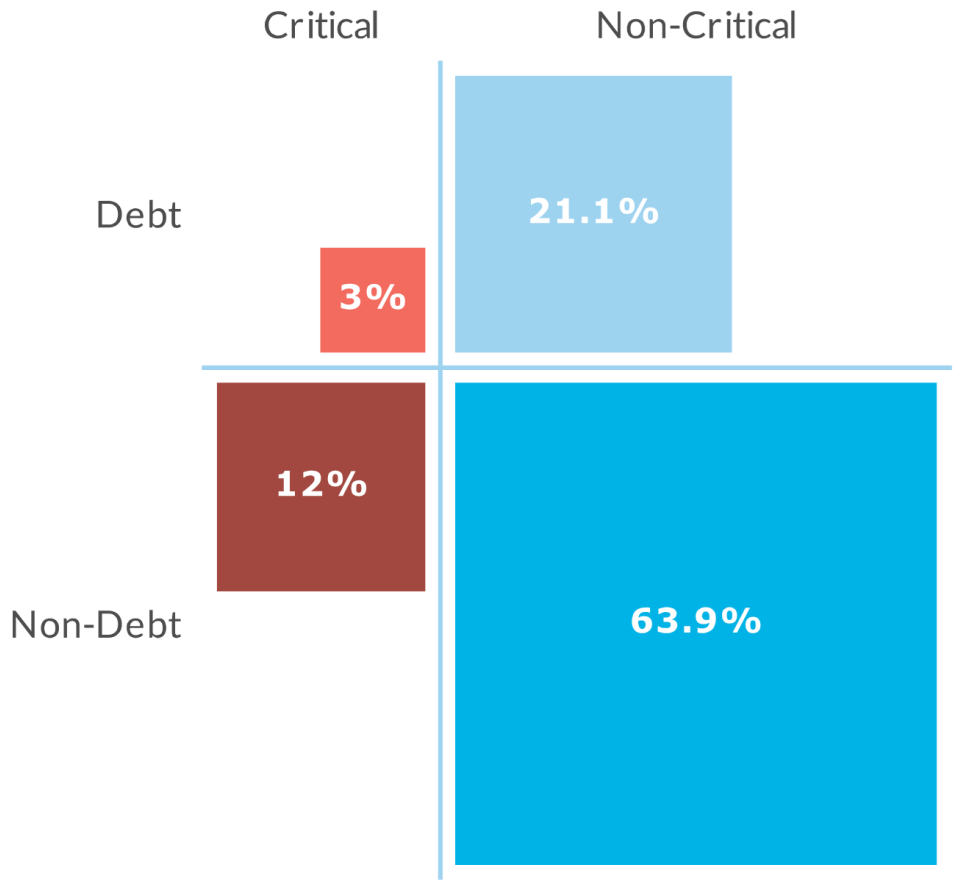
Only 15 percent of all flaws are *critical* flaws.

This subset of flaws represents pound-for-pound the greatest risk exposure to your applications. Prioritize that 15 percent, and, while you won't eliminate all security debt, you will achieve a goal of maximum risk reduction with focused effort.

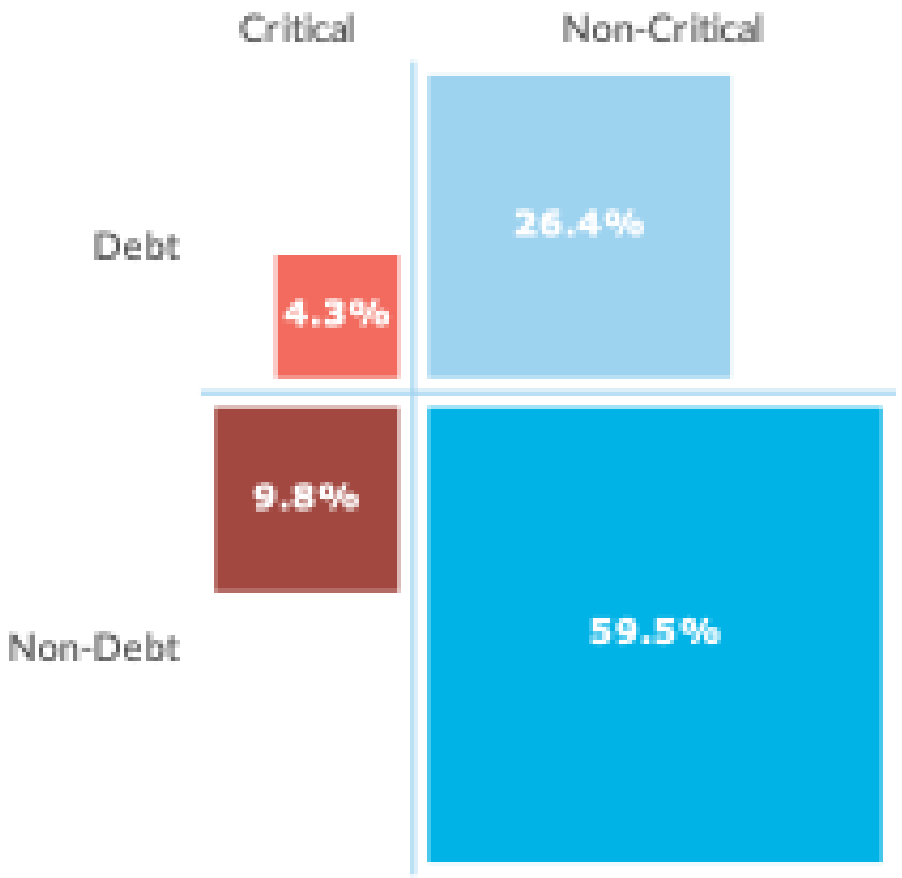


VERACODE

EU apps may require more fix capacity



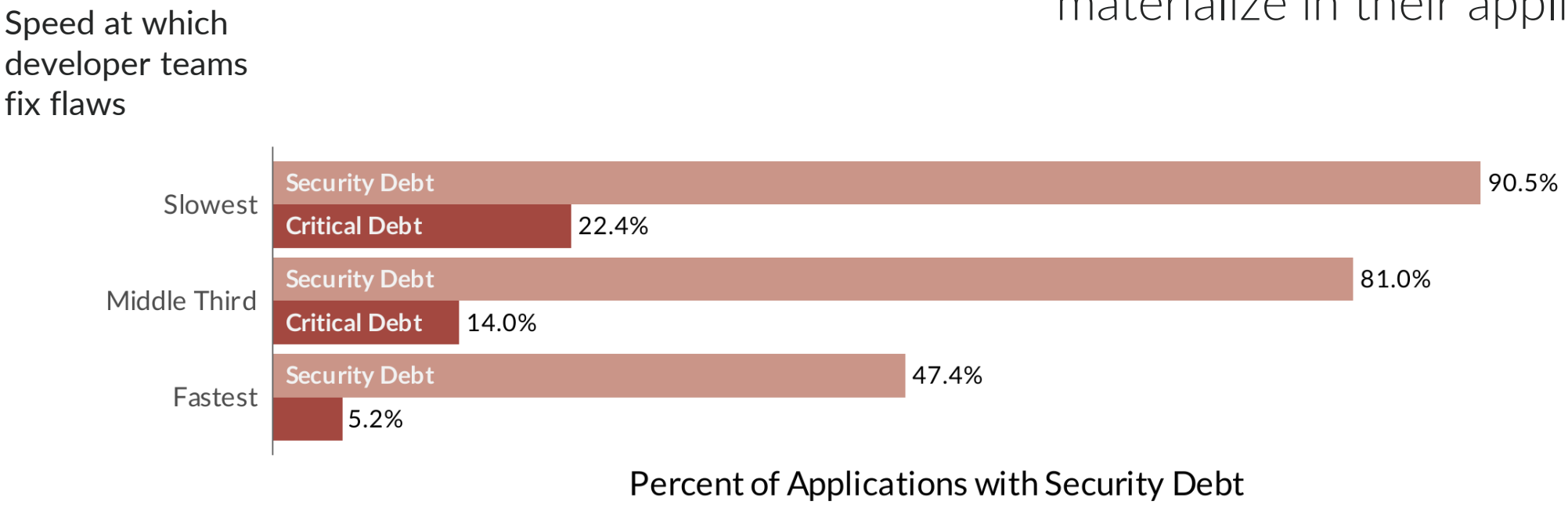
All Customers



EU Customers

Managing security debt: fix flaws faster!

Development teams that fix flaws fastest are **four times less likely** to let critical security debt materialize in their applications.



VERACODE

Takeaways

Key learnings from the SoSS report

- Code velocity is **on the rise**, in part thanks to generative AI
- More code will result in **more security debt** because generated code exhibits all of the same security weaknesses as human-written code
- Development teams…
  - …allocate **very little capacity** to fixing security flaws
  - …and often do not prioritize the **most critical** flaws

Techniques for tackling security debt

- **Increase capacity**: the amount of time development teams dedicate to fixing security flaws ***is a choice*** not an inherent limitation
- **Prioritize wisely**: fix critical flaws (debt and non-debt) before non-critical flaws to ***reduce the most risk***
- **Build security habits**: scan and fix regularly; teams that fix flaws the fastest accumulate ***4x less*** critical security debt
- **Fix faster**: AI-assisted fixing has the potential to help developers fix ***more flaws*** in the ***same amount of time***

Thank You!

Visit Veracode at booth W36