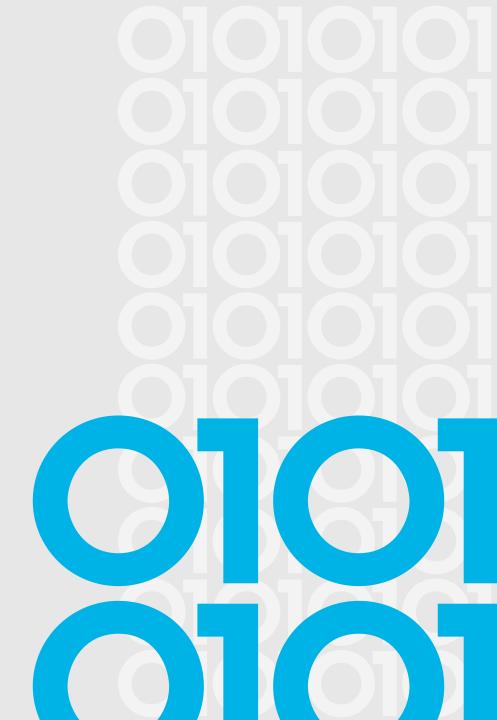


#### State of Software Security: Addressing the Threat of Security Debt

Chris Wysopal Chief Security Evangelist

World AI Summit 2024

October 10, 2024



# One of the 1<sup>st</sup> vulnerability researchers, member of hacker think tank, L0pht in 1990s



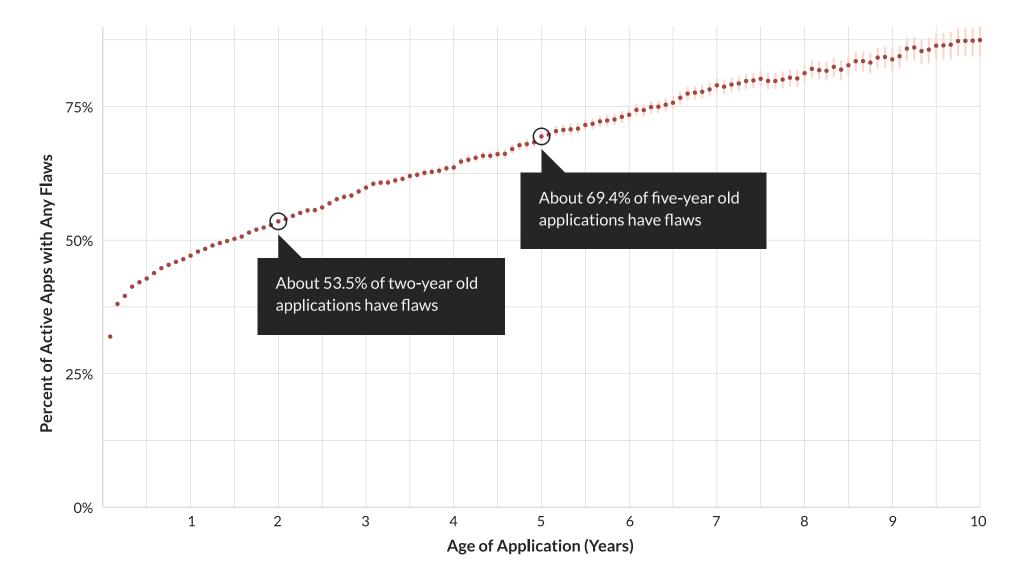
#### Unites States Senate testimony - 19 May 1998



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



#### Flaws accumulate faster than they're fixed





# organizations are drowning in security debt

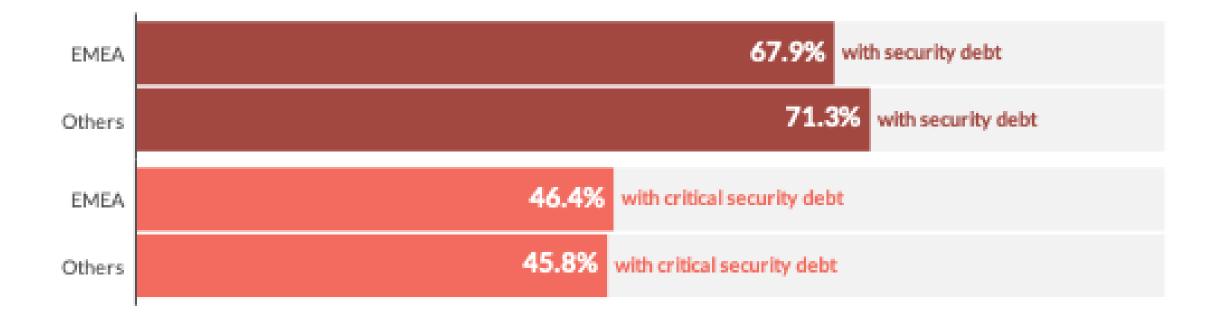
### 70.8%

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





## 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 GenAl use right now

#### Generating code

Understanding code/Code review Remediating defects Translating programming languages Creating and maintaining unit tests Writing documentation

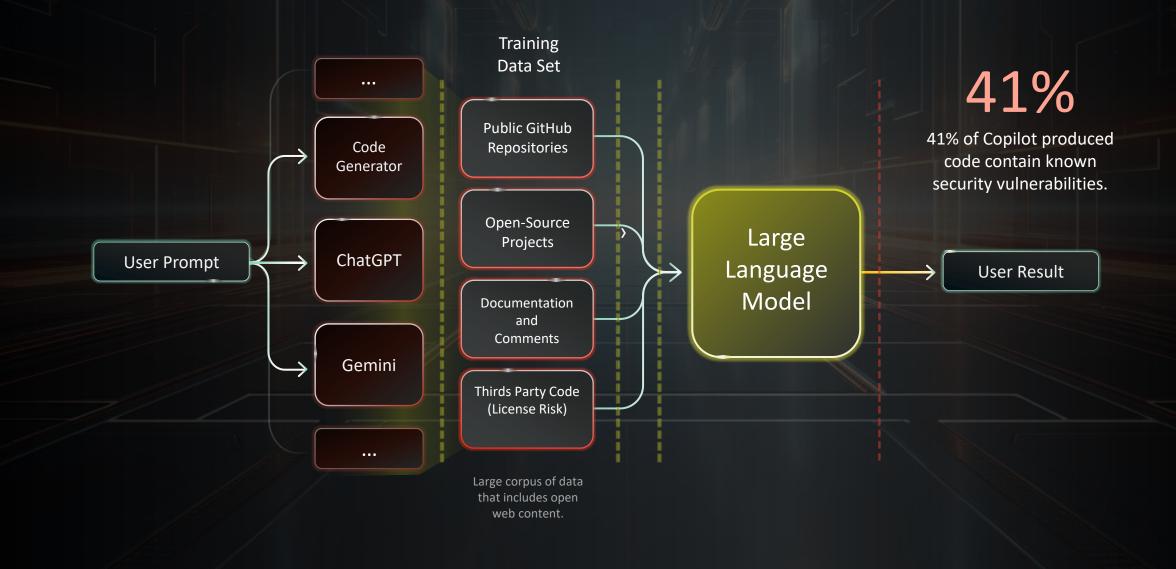


#### **Emerging dev uses for GenAl**

- 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 Al Code Generators

36%

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

M. U. W.			A				
Yujia Fu		Liang	Amjed Tahir				
School of Computer Science	School of Computer Science		School of Mathematical and				
Wahan University	Wuhan University Wohan, China		Computational Sciences				
Wuhan, China yuzia, failwhu.edu.cu	Wulsan, China liangp@whu.edu.cn		Maney University Palmerston North, New Zealand				
John hellowing services of			a tahirijimasovy.ac.na				
Zengyang Li	Mojtaba Shahin		Jianin Yu				
School of Computer Science	School of Computing Technologies		School of Computer Science				
Central China Normal University	EMIT University		Wahan University				
Wuhan, China		r, Australia	Wohan, China				
sengyangli@cons.edu.on	mojtaba shahi	nijemiteduau	jianinyu@whu.edu.cn				
ABSTRACT Modern code generations tools use Al models, particularly Large- Language Models (LLM), in generate functional and complete orde. While such tools are browning popular and woldely available for developers, using these tools is often accompanied by security challenges. Insidig to insurance order merging not the odde has.		<ul> <li>mggented code. It also shows that practitioners should cultivate corresponding security awareness and skills.</li> <li>CCS CONCEPTS</li> </ul>					
					- Software and its engineering Software development tech		
		niques - Security	niques Security and privacy Software socarity engineer				
		Therefore, it is important to assess the quality of the generated		ing			
code, especially in toims of its neurity. Researchers have investly - explored varians aspects of code generation tools, including se-		KEYWORDS					
exposed various aspects of code generation toos, menating se- curity. However, many open questions about the security of the		Code Generation, Security Weeknesses, CWEe, CitHub Copdot					
merated code require furthet investigation, e		ACM Reference Fue					
eues of automatically generated code in the		Suite For Peng Liang, Anged Talte: Designing Li, Motulus Shahim, and Janin					
conducted an empirical study by analyting the security weaknesses in code unippets generated by GitHub Copilot that are found as		No. 2023. Recordy Wa	Ya. 2020. Recently Windonessen of Copilist Generated Code in Gildhih. In				
art of publicly available projects hosted on G			informer (Conference 17). ACM, New York, NY, USA 49-70, 1145 (contempt communit)				
vestigate the types of security neuro and the	is scale in real-would	14 pages. https://doi.or	og 31.1145 mennen demonse				
scenarios (softer than cealed scenarios). To this end, we identi- field 451 code mippers governed by Gd9bib Copilet from publicly available projects. We then conducted extensive security analysis to identify Common Winkness Transmission (CWE) instances in the identify Common Winkness Transmission (CWE) instances in		<ol> <li>INTRODUCTION Code generation tools aim to automatically generate functional code based on prompts, which can include text descriptions (rom)</li> </ol>					
				use code snippets. The results abow that (1)	33.8% of Copilet gen-		as function signatures, expressions, variable
				rated code anippets contain CWEs, and the			onbination of text and code [34]. After writing
rosi sultple languages, (2) the socarity we		an initial code or comment, developers can tely on code genera- tion tools to complete the censaining code. This approach can serve					
and minted to 42 different CWEs, in which CWE 78:03 Command Japerton, CWE 500 Die of bengfictently Random Values, and CWE 193 Japapper Check or Handling of Exceptional Conditions occurred the most Despatisly, and (2) among the 42 CWEs identified, 11 of		development time and accelerate the software development pro- cess. Automated code generation tools have always here an active research discussion topic [22, 30]. Some of the staffast work can					
				use belong to the currently recognized 202			te 1940s when Waldinger and Lee proposed a
				findings confirm that developers should be careful when adding code generated by Copiliri (and similar Al code generation tools) and should also run appropriate security checks as they access the		program synthesizer called PROW, which automatically generated LBP programs based on opecifications precided by users in the	
form of perdicate calculus [46]. As computer languages continued							
and the second sec	and the state of	to evolve, more and	more programming languages began to support making automated code arminition technology				
	the week for personal se-		leading around to cold great and to classing				
common to ender digital or bard organs of all or part of	chamous are a granted without he perioded that open an not hade or datebuild. for perfit or composing advectings and that open bear for settor and the full citation.		of artificial intelligence technologies, particularly in the form of				
corrow are a granted without he provided that report a courth or composition advantage and that reports hear the	setter-and the full citation						
converse use a granted without he previated that report c prefit or conserver al solventage and that require bear three the first page. Copyrights for congression of this work of		esachine learning a	nd deep learning models, has accelerated the				
comment one is generated without for previously drags report r perify no contractional advantage and that couples bear from the floor page. Cognitylifies for components of this study a and for flooraged. Attention imp with could in period. At the period on percent on the study with a study in period.	would be others than ACM	esachine learning a development of cod	ed deep learning models, has accelerated the le generation technologies.				
concerns on a granted without he provided that report careful to considering advantage and that report hear the the data page. Cognitights for components of this work or or for bottom (d) biotecome and control to consider the	would be others than ACM	machine learning a development of cod Encost advances	nd deep learning models, has accelerated the				

New York University Study on GitHub Copilot



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 Pource Bakegh Atmad Benjamin Tan Brendan Delae-Gavit Ramoh Ku Deparatoso et REL: Deparations of ICE: Deparations of IC

 $\alpha$  - There is the second to foreign of the second to the

present an obsolute interview in the second second

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.

Purdue University on ChatGPT accuracy

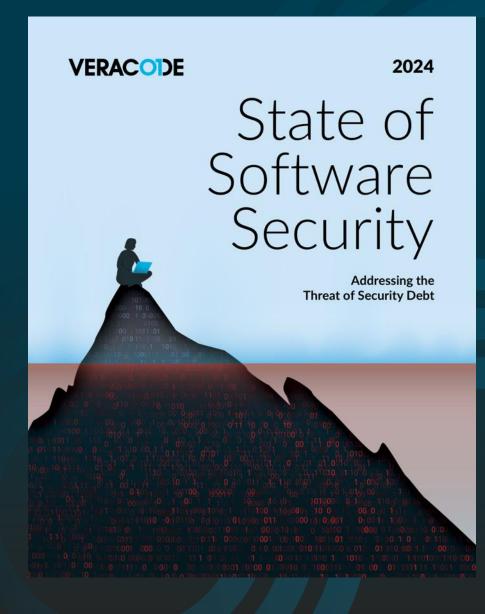


52% of ChatGPTs answers were incorrect. Developers preferred them 35% of the time yet 77% of those answers were wrong

	Do Users W	rite More Insecu	are Code with Al	Assistants?
	Neil Perry* Starford University	Megha Srivastava' Stanford University	Deepak Kumar Stanford University / UC San Diego	Dan Boneh Stanford University
arXiv:2211.03622v3 [cs.CR] 18 Dec 2023	<section-header><text><text><section-header><section-header><section-header><section-header></section-header></section-header></section-header></section-header></text></text></section-header>	In and an improve damping the object of the	<text><text><list-item><list-item><text></text></list-item></list-item></text></text>	maintaints twelfs means sub?" maintaints twelfs means sub?" which depend on the second sec

	s to Software Engineering Questions
Samia Kabir	David N. Udo-Imeh
Purdue University	Purdue University
West Lafepotte, USA	West Lafayette, USA
kabirs@purdue.edu	dudoimeht@purdue.edu
Bonan Kou	Tianvi Zhang
Purdue University	Purdue University
West Lafavette, USA	West Lafavette, USA
koub@pardue.edu	tianyi@pundue.edu
ABSTRACT	1 INTRODUCTION
Over the last decade, QAA platforms have played a crue	cial role in Software developers often resort to online resources for a variety
how programmers seek help online. The emergence of	ChatGPT. of software engineering tasks, e.g., API learning, bug fixing, com-
however, is causing a shift in this pattern. Despite ChatC	IPT's pop-prehension of code or concepts, etc. [53, 57, 63]. A wast majority
ularity, there have't been a thorough investigation into t	
and usability of its responses to software engineering q	
address this gap, we undertook a compethensive analys	
GPT's replies to 517 questions from Stack Overflow (5	
sessed the correctness, consistency, comprehensiveness	
ciseness of these responses. Additionally, we conducted	
sive linguistic analysis and a user study to gain insight	
inguistic and human aspects of ChatGPT's answers. Ou	
tion revealed that 52% of ChatGPT's answers contain in	
and 77% are verbose. Nevertheless, users still prefer Cha	
sponses 39.54% of the time due to their comprehensiv	
articulate language style. These findings underscore the	
meticulous error correction in ChatGPT while also raisi	
ness among users about the potential risks associated v	
ingly accurate answers.	was introduced as an open-access Chatbot, which surpassed the
	popularity of other models in its category. ChatGPT's capacity to
CCS CONCEPTS	engage in human-like conversations, promptly learn from contin-
- Software and its engineering General and refs	normer nous human fordback, and accessibility to the general public, have
Empirical studies:	all contributed to its popularity. Consequently, ChatGPT's popu-
	larity has ignited numerous debates among academics, researchers,
KEYWORDS	and industry professionals on Twitter and other social media plat-
nack overflow; q&a, large language model, chargpt	forms [17, 51]. These discussions revolve around the scenarios wherei ChatGPT could potentially replace prominent search engines (e.g.,
ACM Reference Format:	Google) or widely used Q&A platforms (e.g., Stack Overflow).
Samia Kabir, David N. Udo-Imeh, Boman Kou, and Tianyi Zhang	
Answers 3: Better? An In Depth Analysis of ChatOPT and Stat	
Answers to Software Engineering Questions . In Proceedings of J	
enter the correct conference title from your rights confermation or ence accord on XO, ACM, New York, NY, USA, 13 pages, https://	
eter actorym. 200 ACHUNEW TORCNELTING, 12 pages, impacte	
	often generate fabricated texts that mimic truthful information,
Premission to make signal or band copies of all or part of this work for chanceson use is granted without for provided that copies are not make or	
for profit or commercial advantage and that copies bear this notice and	
tion on the first page. Copyrights for components of this work owned to ACM must be beneard. Abstracting with could is permitted. To copy of	
publish, to post an across or to auditabate to late, requires prior specifi	
and/or a los. Resparsé premissiona leurs premissional/hauss.org.	has prompted Stack Overflow to impose a ban on posting answers
	generated by ChatGPT [44].
Conference accorgon XX, Jane 17-48, 2018, Wandshold, NT	
<ul> <li>DDI Association for Computing Machineses ACM Director 2010 Association for Computing Machineses</li> </ul>	

#### What is Veracode seeing across our customer base?





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

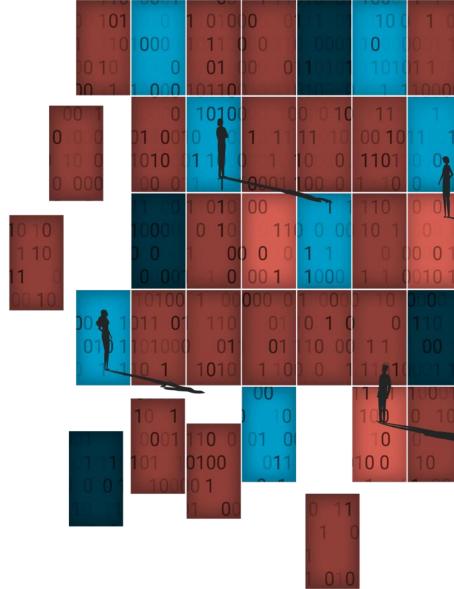
All those scans produced:

**96.0 million** raw static findings

**4.0 million** raw dynamic findings

**12.2 million** 

raw software composition analysis findings



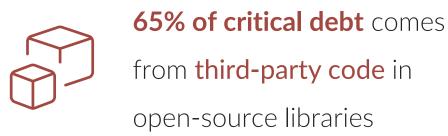
### Our approach and methodology

### Where is the security debt?



While **first-party code** constitutes almost

90% of all security debt



 Third Party Code
 First Party Code

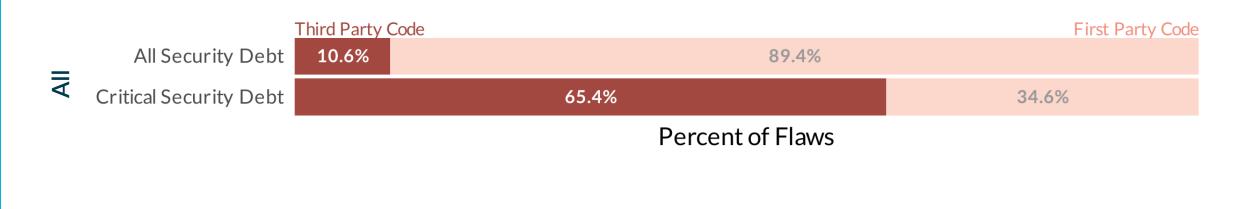
 All Security Debt
 10.6%
 89.4%

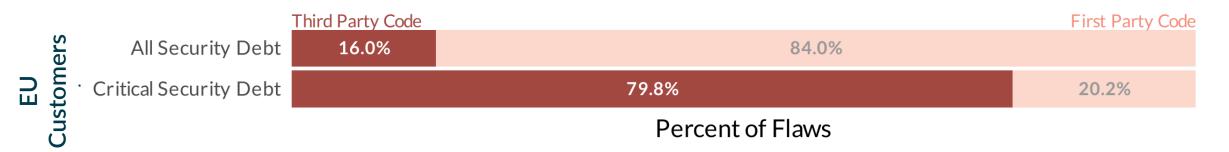
 Critical Security Debt
 65.4%
 34.6%

Percent of Flaws



#### EU customer breakdown is similar





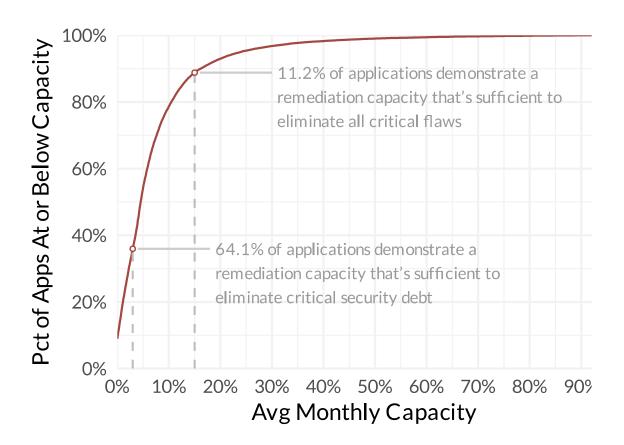


Only **64%** of applications demonstrate a sustained capacity to eliminate all critical security debt.

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.

# Remediation capacity is constrained

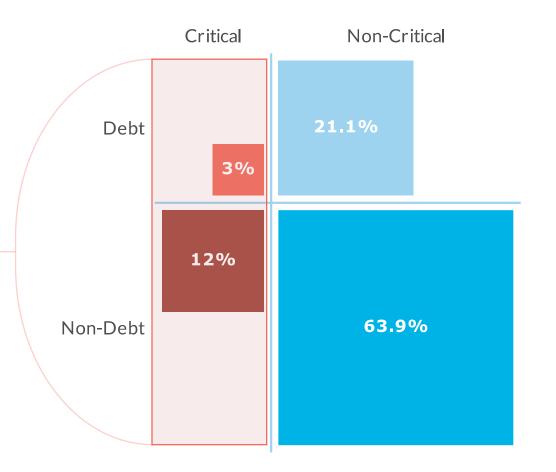


### Prioritization is the key

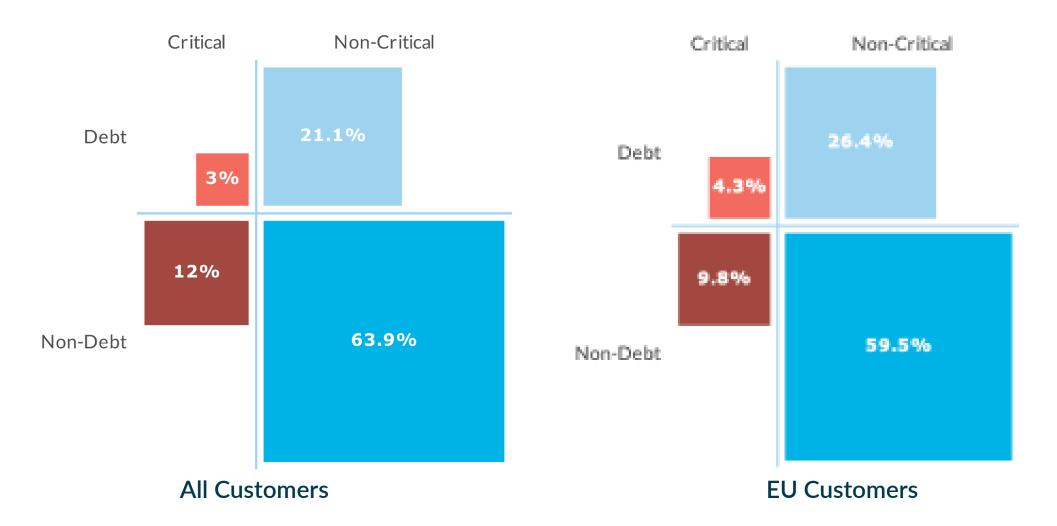
If the rate of new and existing flaws **exceeds** the capacity to remediate them, then **prioritizing which flaws to remediate is essential**.

#### 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.



### EU apps may require more fix capacity

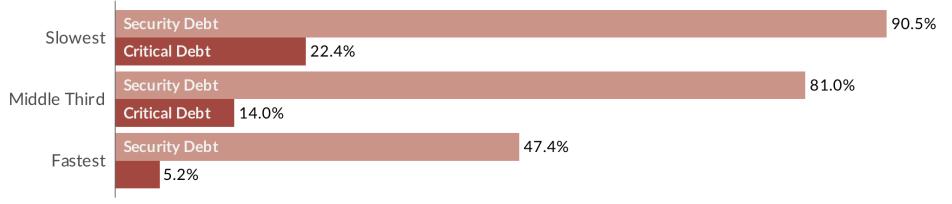




# Managing security debt: fix flaws faster!

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

Speed at which developer teams fix flaws



Percent of Applications with Security Debt



#### 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

