From HAL to HALT: Thwarting Skynet's Siblings in the GenAI Coding Era

Chris Wysopal

Co-founder & CTO, Veracode

VERACODE

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

Unites States Senate testimony - 19 May 1998

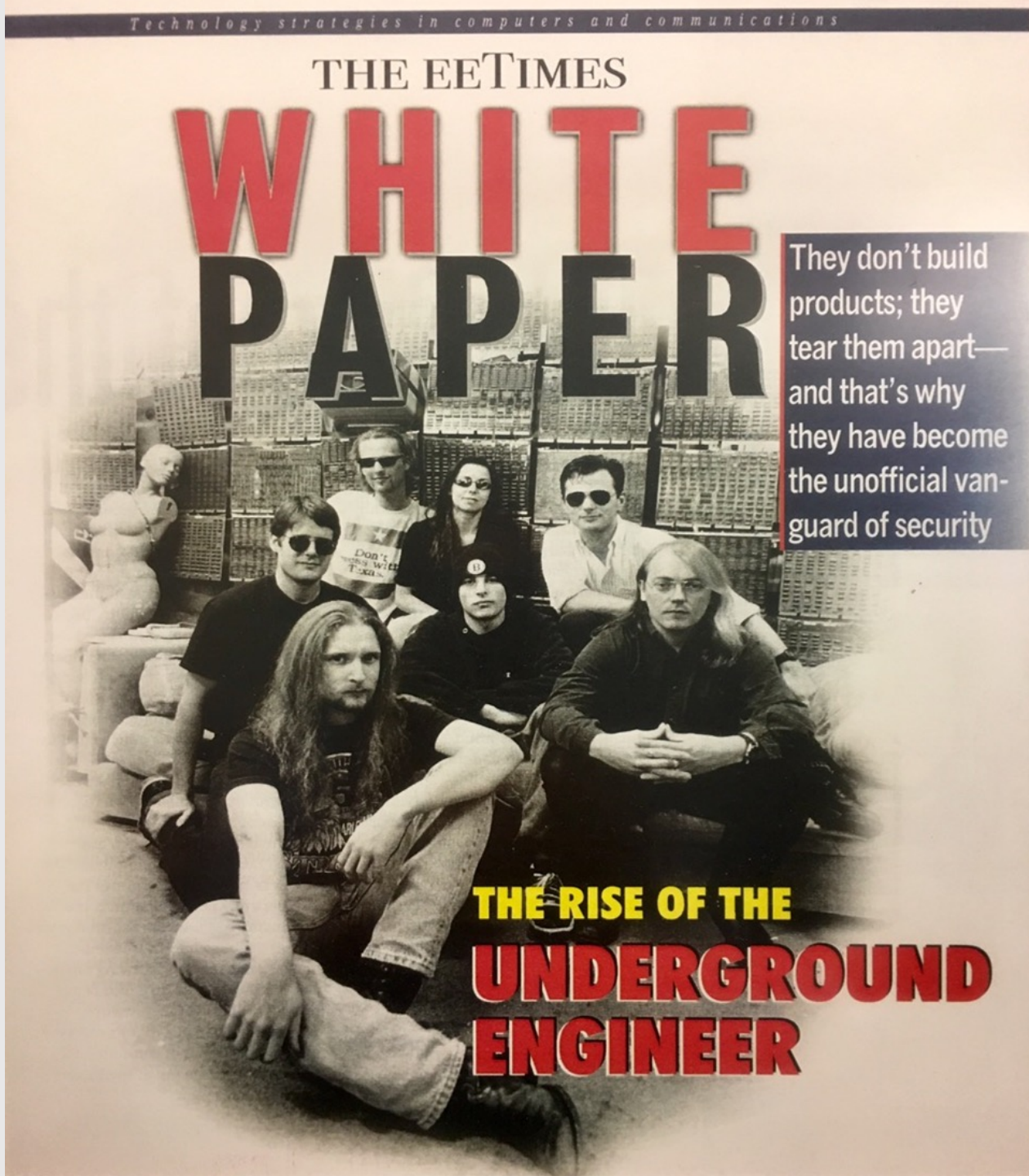
Into the light: *Once shadowy computer code warriors like Kingpin are going legit*

Using Good Hackers to Battle Bad Hackers

IF YOU HAVE A MURKY PAST AND DOUBT you could become a dot-com millionaire, think again. Last week a scraggly band of hackers known as "L0pht Heavy Industries" joined with some straitlaced tech execs to form @Stake, an Internet-security consulting firm.

Newsweek, January 17, 2000

Improve the Security of Your *Product* by Breaking Into It

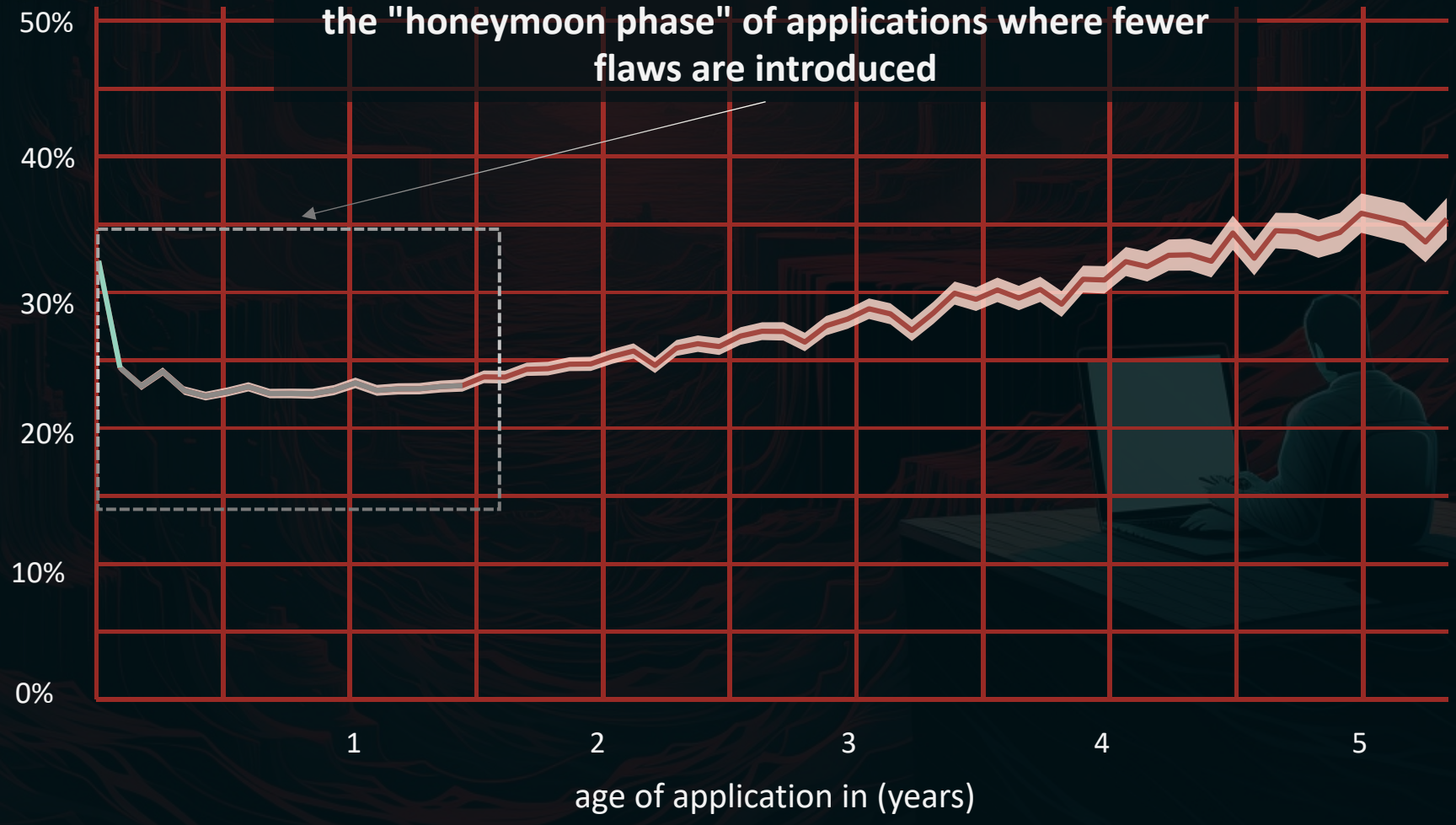


Founded @stake security research team and then Veracode to build security into SDLC

State of Software Security 2024

Addressing the Threat of Security Debt

new flaws introduced by application age



organizations are drowning in **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.


2 out of 10
applications show an average monthly fix rate that exceeds ten percent of all security flaws.
few teams fix flaws fast enough to reduce security risk at a meaningful pace

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

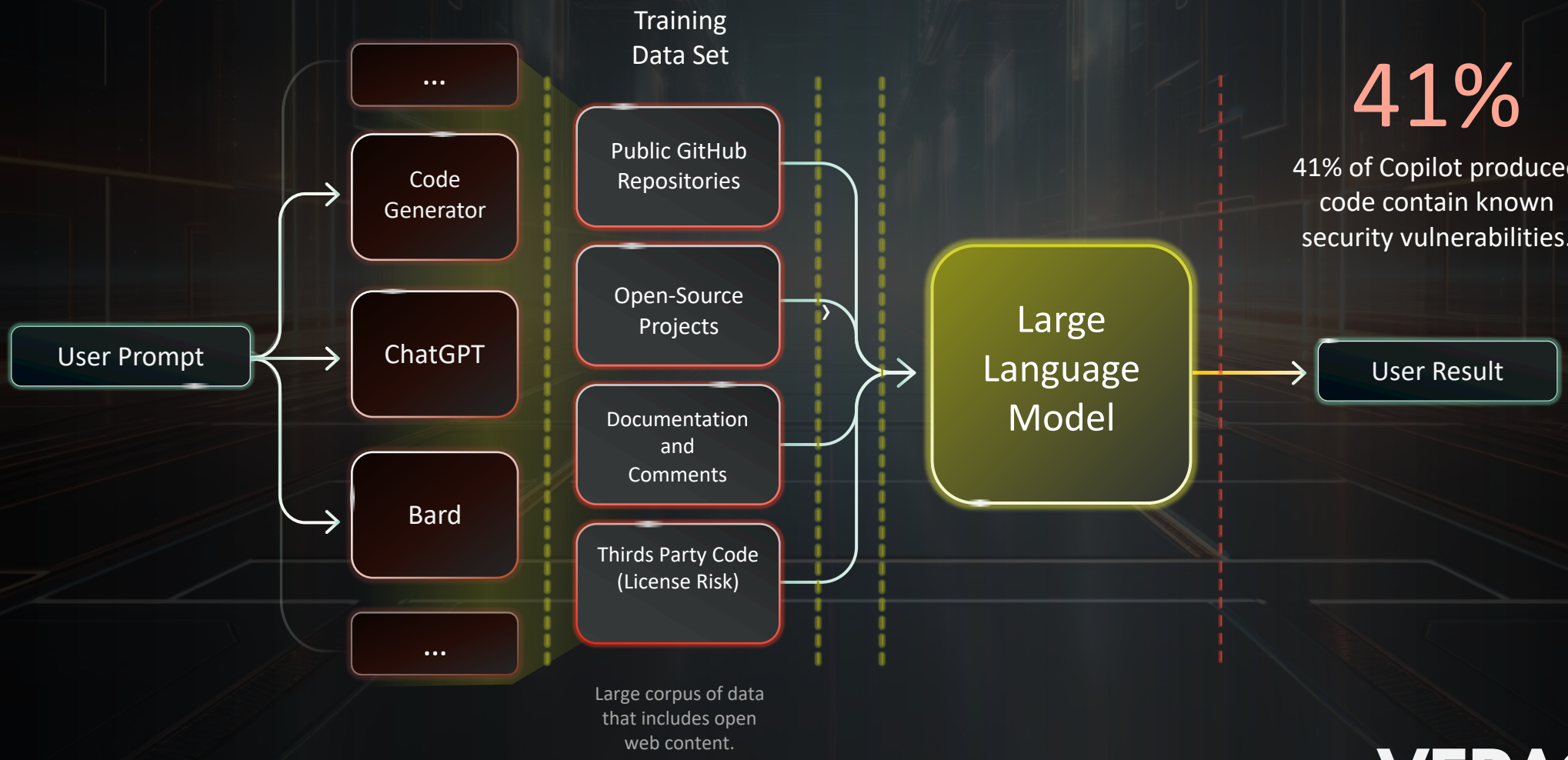
Searching for answers to avoid reinventing the wheel

Reading log files to find a root cause

Creating and running functional & non-functional tests

Remediating security vulnerabilities

Large Language Models



41%

41% of Copilot produced code contain known security vulnerabilities.

VERACODE

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

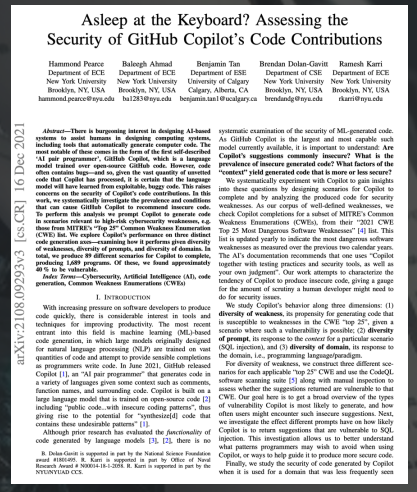
Yujia Fu, Peng Liang, Amjed Tahir, Zengyang Li, Mojtaba Shahin, Jiaxin Yu, 2023. Security Weaknesses of Copilot Generated Code in GitHub. In *Proceedings of ACM Conference (Conference'17). ACM*, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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, 2021. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*, May 2022, San Francisco, CA, USA. IEEE Computer Society, Los Alamitos, CA, USA, 754-768. https://doi.org/10.1109/SP46214.2022.9833571

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, 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark. ACM*, New York, NY, USA, 16 pages. https://doi.org/10.1145/3576915.3623157

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, 2023. Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX). ACM*, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

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

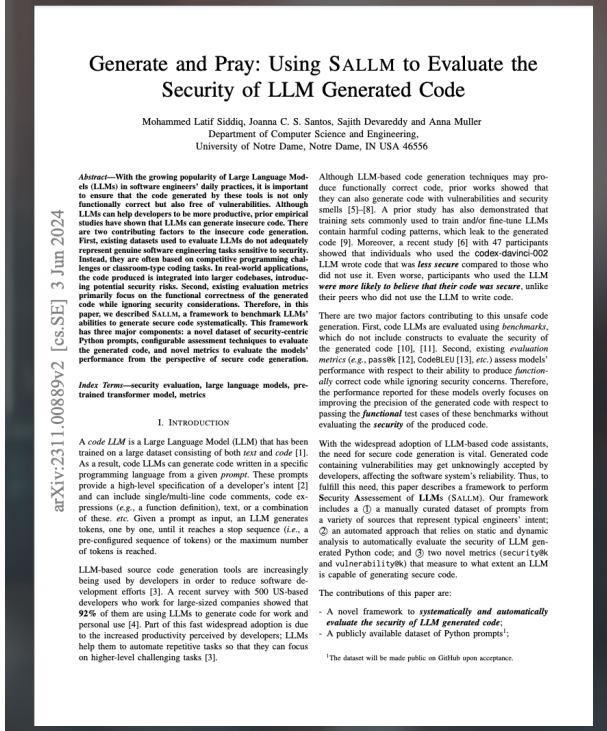
SALLM Framework For measuring LLM vulnerability generation - Notre Dame

Vulnerable@k metric best to worst:

StarCoder
GPT-4:
GPT-3.5:.
CodeGen-2.5-7B:
CodeGen-2B:

VULNERABILITIES FOUND IN THE CHATGPT-GENERATED PYTHON CODES

CWE Name	CWE Top-25 Rank	# Vuln. Samples
CWE-312 Cleartext Storage of Sensitive Information	-	14
CWE-798 Use of Hard-coded Credentials	18	5
CWE-208 Observable Timing Discrepancy	-	3
CWE-215 Insertion of Sensitive Information Into Debugging Code	-	3
CWE-338 Use of Cryptographically Weak Random Generator	-	3
CWE-79 Cross-site Scripting	2	2
CWE-209 Generation of Error Message Containing Sensitive Information	-	2
CWE-287 Improper Authentication	13	1
CWE-295 Improper Certificate Validation	-	1
CWE-918 Server-Side Request Forgery	19	1



Generate and Pray: Using SALLM to Evaluate the Security of LLM Generated Code

Mohammed Latif Siddiq, Joanna C. S. Santos, Sajith Devareddy and Anna Muller
Department of Computer Science and Engineering,
University of Notre Dame, Notre Dame, IN USA 46556

arXiv:2311.00889v2 [cs.SE] 3 Jun 2024

https://arxiv.org/abs/2311.00889

Implications of LLM code generation

Code reuse goes down

Code velocity goes up

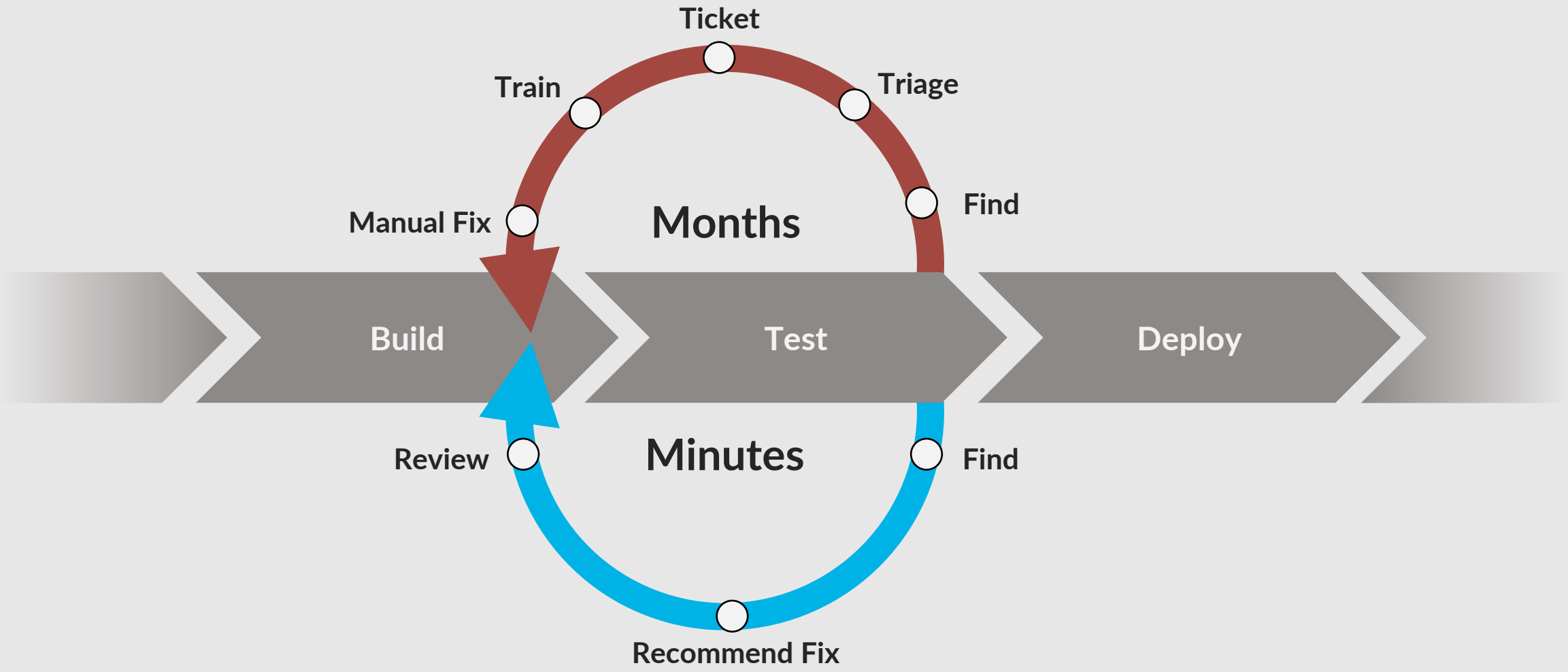
Vulnerability density similar

=

Increased Vulnerability Velocity

How can we apply AI to the problem of insecure code, but in a more accurate and trustworthy manner?

We need a faster test and fix workflow



Training data set: Java XSS

```
public void doGet(HttpServletRequest req, HttpServletResponse resp) {
    String name = req.getParameter("name");
    String[] array = new String[10];
    array[0] = name;
    PrintWriter writer = resp.getWriter();
    writer.println("Hello " + array[0]);
}
```

← Cross-site scripting (CWE 80)

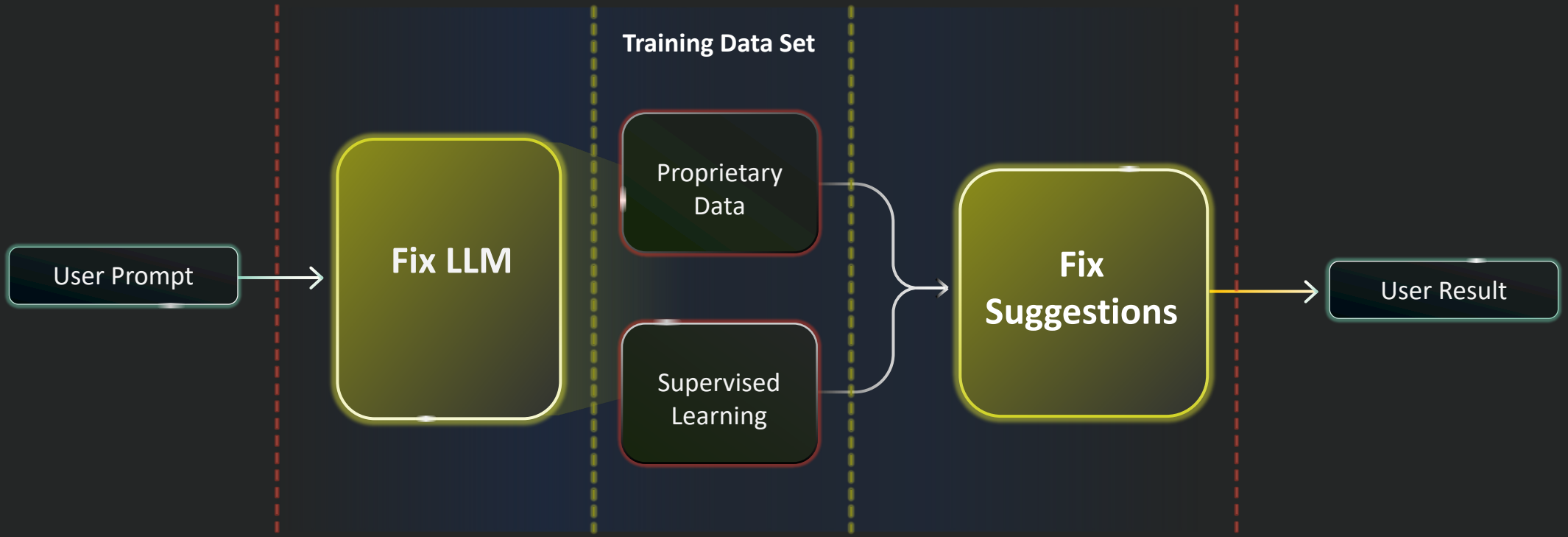
```
public void doGet(HttpServletRequest req, HttpServletResponse resp) {
    String name = req.getParameter("name");
    String[] array = new String[10];
    array[0] = name;
    PrintWriter writer = resp.getWriter();
    writer.println("Hello " + StringEscapeUtils.escapeHtml4(array[0]));
}
```

Fix Approach

Curated Dataset

Code Provenance Assurance

Coverage all that matter



VERACODE

Recommendations for AI and code security

Consider the implementation details before leveraging AI for developing and/or securing code

- What does the ML model use for training data?
- Is that training data trustworthy/vetted?
- Are there licensing issues with generated code?
- Is any of my intellectual property being leaked?
- How accurate are the generated fixes?

Be aware of human biases that trick us into feeling overly confident about the correctness of AI-generated content

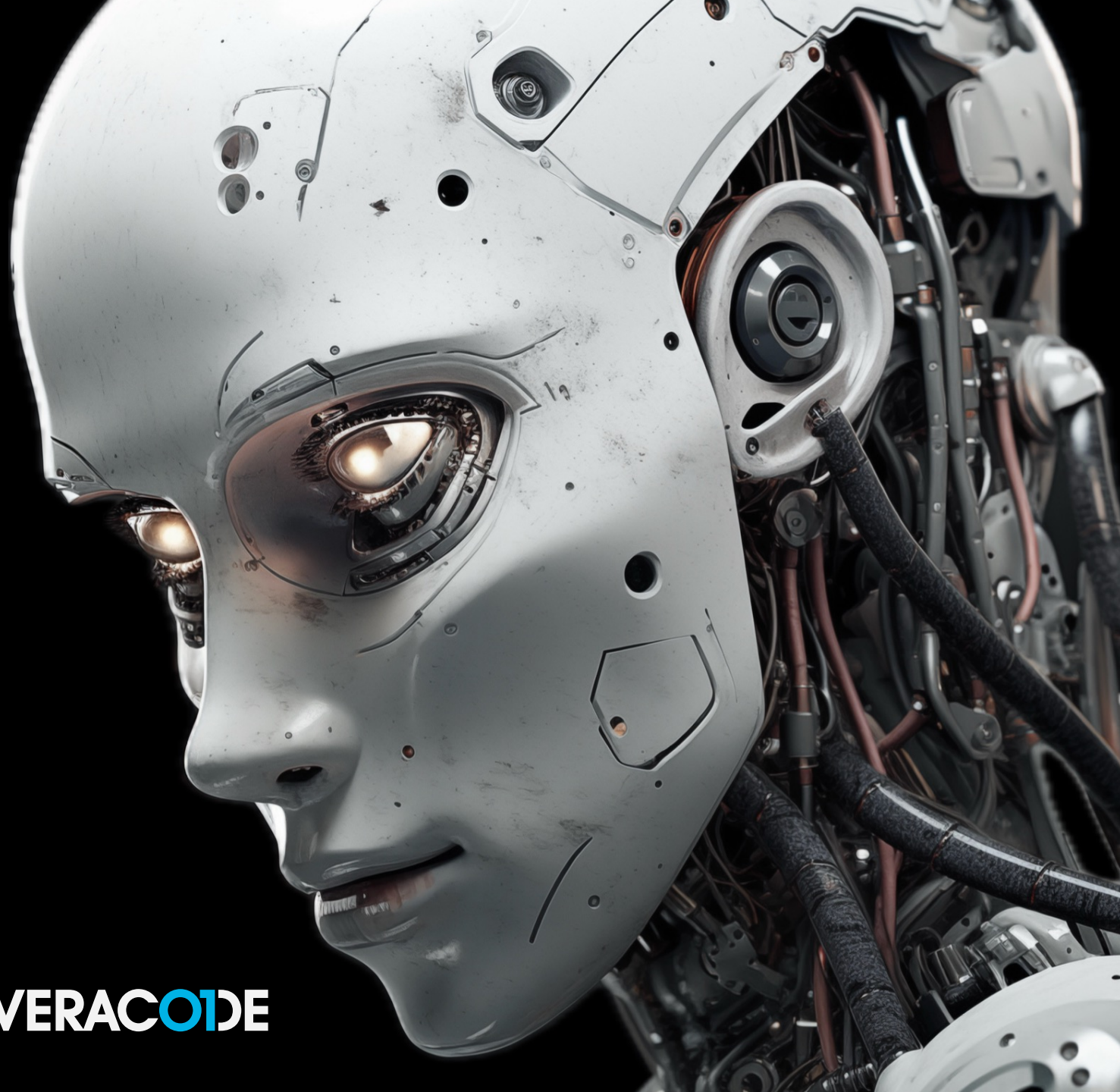

Other Risks to GenAI Code
Data Poisoning
IP Infringement
Recursive Learning
Bias & Fairness
Propagation of Deprecated Practices
Hallucinated & Squatted Packages
VERACODE

GenAI in dev is a powerful tool that requires the same level of security scrutiny and best practices as any other aspect of software development

Include security considerations in GenAI prompts

Automate as much of security process as possible, including automated fixing

Chris Wysopal
Co-founder & CTO Veracode
@weldpond



VERACODE