Just another copy and paste?
Comparing the security vulnerabilities of ChatGPT generated code and StackOverflow answers

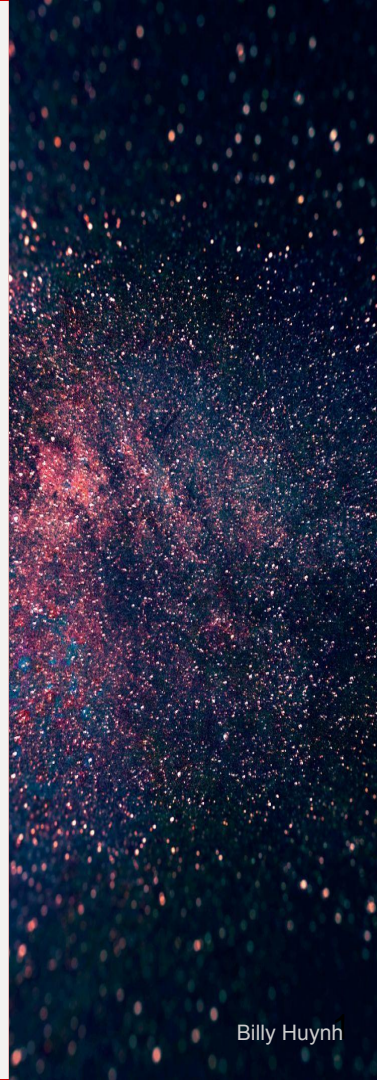
Sivana Hamer, Marcelo d'Amorim,
Laurie Williams

NC STATE UNIVERSITY

S3C2
SECURE SOFTWARE SUPPLY CHAIN CENTER

Realsearch
Real People · Real Projects · Real Impact

WSPR



Billy Huynh

97%

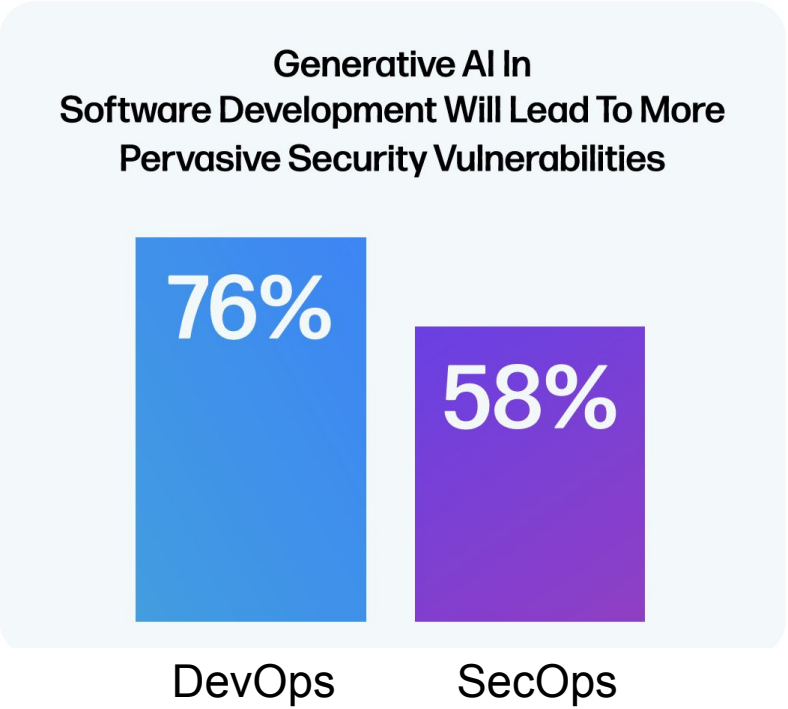
developers and security leads are using generative AI

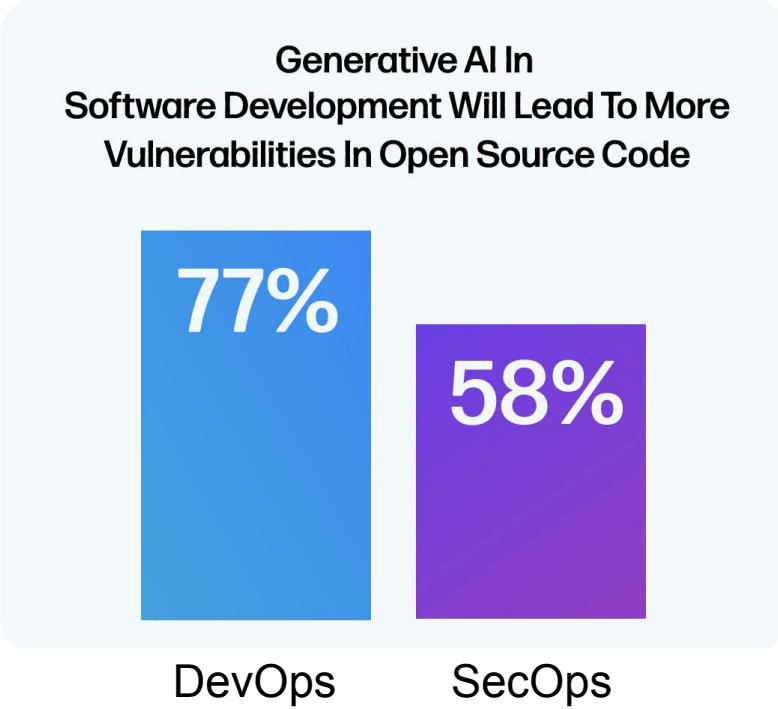
sonatype

74%

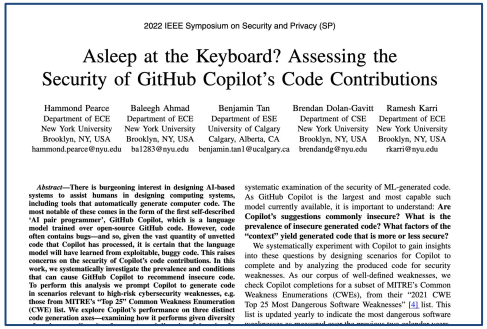
feel pressured to use generative AI despite security risks

sonatype


Generative AI In
Software Development Will Lead To More
Pervasive Security Vulnerabilities
76%
58%
DevOps
SecOps


Generative AI In
Software Development Will Lead To More
Vulnerabilities In Open Source Code
77%
58%
DevOps
SecOps

sonatype



2022 IEEE Symposium on Security and Privacy (SP)

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

Hammond Pearce, Department of ECE, New York University, Brooklyn, NY, USA, hammond.pearce@nyu.edu
Baleegh Ahmad, Department of ECE, New York University, Brooklyn, NY, USA, ba1283@nyu.edu
Benjamin Tan, Department of ESE, University of Calgary, Calgary, Alberta, CA, benjamin.tan1@ucalgary.ca
Brendan Dolan-Gavitt, Department of CSE, New York University, Brooklyn, NY, USA, brendandg@nyu.edu
Ramesh Karri, Department of ECE, New York University, Brooklyn, NY, USA, rkarri@nyu.edu

***Abstract*—There is burgeoning interest in designing AI-based systems to assist humans in designing computing systems, including tools that automatically generate computer code. The most notable of these comes in the form of the first self-described 'AI pair programmer', GitHub Copilot, which is a language model trained over open-source GitHub code. However, code often contains bugs—and so, given the vast quantity of unvetted code that Copilot has processed, it is certain that the language model will have learned from exploitable, buggy code. This raises concerns on the security of Copilot's code contributions. In this work, we systematically investigate the prevalence and conditions that can cause GitHub Copilot to recommend insecure code. To perform this analysis we prompt Copilot to generate code in scenarios relevant to high-risk cybersecurity weaknesses, e.g. those from MITRE's "Top 25" Common Weakness Enumeration (CWE) list. We explore Copilot's performance on three distinct code generation axes—examining how it performs given diversity**

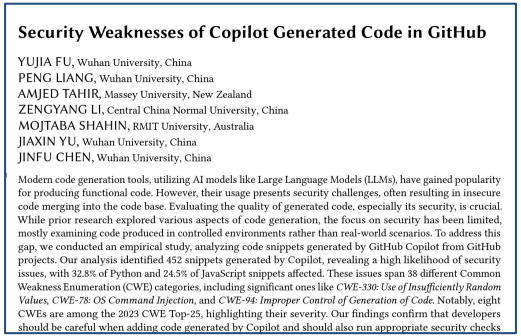
systematic examination of the security of ML-generated code. As GitHub Copilot is the largest and most capable such model currently available, it is important to understand: **Are Copilot's suggestions commonly insecure? What is the prevalence of insecure generated code? What factors of the "context" yield generated code that is more or less secure?**

We systematically experiment with Copilot to gain insights into these questions by designing scenarios for Copilot to complete and by analyzing the produced code for security weaknesses. As our corpus of well-defined weaknesses, we check Copilot completions for a subset of MITRE's Common Weakness Enumerations (CWEs), from their "2021 CWE Top 25 Most Dangerous Software Weaknesses" [4] list. This list is updated yearly to indicate the most dangerous software

(Pearce et al., 2022)

…we prompt Copilot to generate code in scenarios relevant to high-risk cybersecurity weaknesses …we found ***approximately 40% to be vulnerable.***

…analyzing code snippets generated by GitHub Copilot from GitHub projects …***29.6% of Copilot-generated code snippets*** have security weaknesses.



Security Weaknesses of Copilot Generated Code in GitHub

YUJIA FU, Wuhan University, China
PENG LIANG, Wuhan University, China
AMJED TAHIR, Massey University, New Zealand
ZENGYANG LI, Central China Normal University, China
MOJTABA SHAHIN, RMIT University, Australia
JIAXIN YU, Wuhan University, China
JINFU CHEN, Wuhan University, China

Modern code generation tools, utilizing AI models like Large Language Models (LLMs), have gained popularity for producing functional code. However, their usage presents security challenges, often resulting in insecure code merging into the code base. Evaluating the quality of generated code, especially its security, is crucial. While prior research explored various aspects of code generation, the focus on security has been limited, mostly examining code produced in controlled environments rather than real-world scenarios. To address this gap, we conducted an empirical study, analyzing code snippets generated by GitHub Copilot from GitHub projects. Our analysis identified 452 snippets generated by Copilot, revealing a high likelihood of security issues, with 32.8% of Python and 24.5% of JavaScript snippets affected. These issues span 38 different Common Weakness Enumeration (CWE) categories, including significant ones like *CWE-330: Use of Insufficiently Random Values*, *CWE-78: OS Command Injection*, and *CWE-94: Improper Control of Generation of Code*. Notably, eight CWEs are among the 2023 CWE Top-25, highlighting their severity. Our findings confirm that developers should be careful when adding code generated by Copilot and should also run appropriate security checks

(Fu et al., 2024)

…still GenAI is not the only security risk

… **97.9%** contain at least one insecure code snippet.

2017 IEEE Symposium on Security and Privacy

Stack Overflow Considered Harmful?
The Impact of Copy&Paste on Android Application Security

Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky*, Yasemin Acar*, Michael Backes*, Sascha Fahl*
Fraunhofer Institute for Applied and Integrated Security; *CISPA, Saarland University

***Abstract*—Online programming discussion platforms such as Stack Overflow serve as a rich source of information for software**

misusing Android's cryptographic API [3], a developer who is seeking help can find solutions for almost any problem. While

(Fischer et al., 2017)

stackoverflow

Android

…use only StackOverflow **produced significantly less secure code**…

2016 IEEE Symposium on Security and Privacy

You Get Where You're Looking For
The Impact of Information Sources on Code Security

Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim†, Michelle L. Mazurek†, Christian Stransky
CISPA, Saarland University; †University of Maryland, College Park

***Abstract*—Vulnerabilities in Android code – including but not limited to insecure data storage, unprotected inter-component communication, broken TLS implementations, and violations of least privilege – have enabled real-world privacy leaks and motivated research cataloguing their prevalence and impact.**

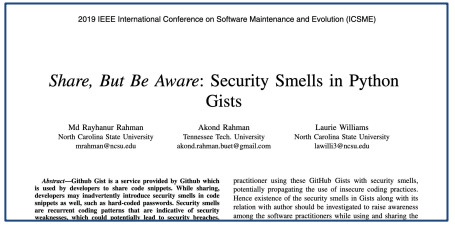
[29], [31], [33], [34], [36], [43], [44], [46]. Developers tend to request more permissions than actually needed, do not use TLS or cryptographic APIs correctly, often use insecure options for Inter Component Communication (ICC), and fail to store

(Acar et al., 2016)

stackoverflow

Documentation Books

…of those Gists, which is around **31%**, have at least one security smell



2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)

Share, But Be Aware: Security Smells in Python Gists

Md Rayhanur Rahman
North Carolina State University
mrahman@ncsu.edu

Akond Rahman
Tennessee Tech. University
akond.rahman.buet@gmail.com

Laurie Williams
North Carolina State University
lawilli3@ncsu.edu

***Abstract*—Github Gist is a service provided by Github which is used by developers to share code snippets. While sharing, developers may inadvertently introduce security smells in code snippets as well, such as hard-coded passwords. Security smells are recurrent coding patterns that are indicative of security weaknesses, which could potentially lead to security breaches.**

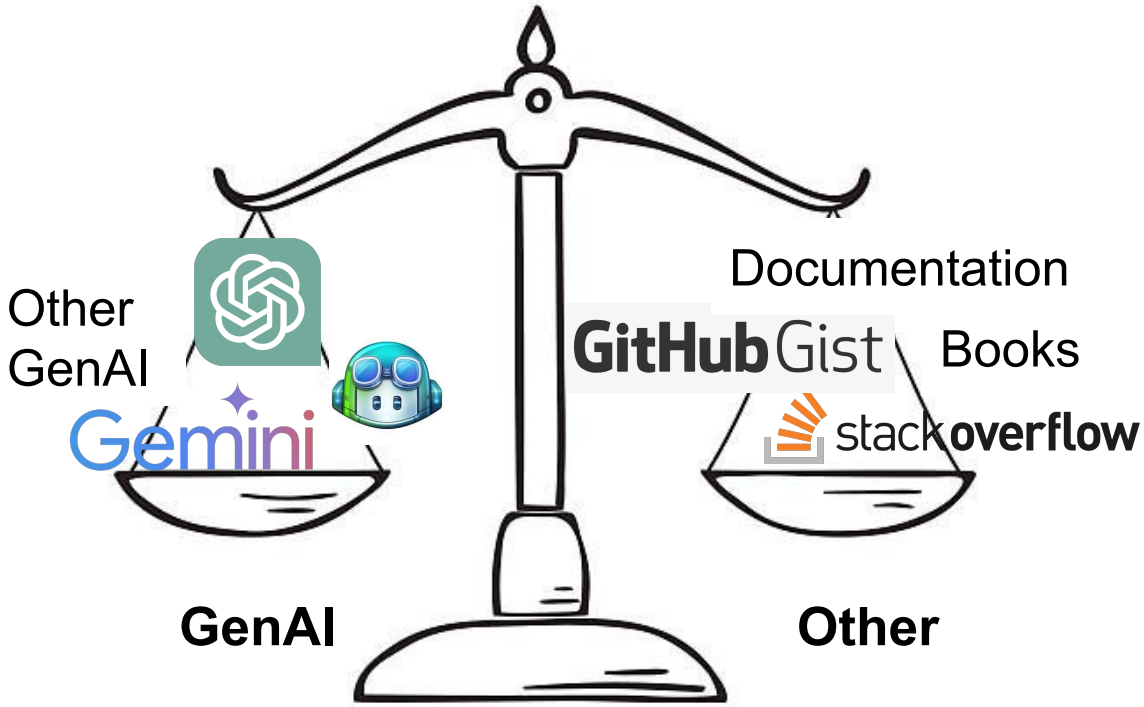
practitioner using these GitHub Gists with security smells, potentially propagating the use of insecure coding practices. Hence existence of the security smells in Gists along with its relation with author should be investigated to raise awareness among the software practitioners while using and sharing the

(Rahman et al., 2019)

GitHub Gist

Empirical data is needed to help developers choose information sources

Confused developer



Goal: to raise software developers' awareness of the security implications when selecting code snippets by empirically comparing the vulnerabilities of ChatGPT and StackOverflow

Security related Q&A in StackOverflow

(Chen et al., 2019)

Detect Java code snippets from the answers

StackOverflow
Gather question and answers

How do I use 3DES encryption/ decryption in Java?

Every method I write to encode a string in Java using 3DES can't be decrypted back to the original string. Does anyone have a simple code snippet that can just encode and then decode the string back to the original string?...

6 answers

Here's a very simply static encrypt/decrypt class biased on the Bouncy Castle no padding example by Jose Luis Montes de Oca....

```
....
  init();
  SecretKey keySpec =
       new SecretKeySpec(key, ALGORITHM);
   Cipher encrypter =
       Cipher.getInstance(
           TRIPLE_DES_TRANSFORMATION,
           BOUNCY_CASTLE_PROVIDER);
  encrypter.init(Cipher.ENCRYPT_MODE, keySpec);
  return encrypter.doFinal(input);
...
```

ChatGPT
Generate answers

You

For all the following questions, generate a compilable code snippet in Java.

Every method I write to encode a string in Java using 3DES can't be decrypted back to the original string. Does anyone have a simple code snippet that can just encode and then decode the string back to the original string?...

ChatGPT

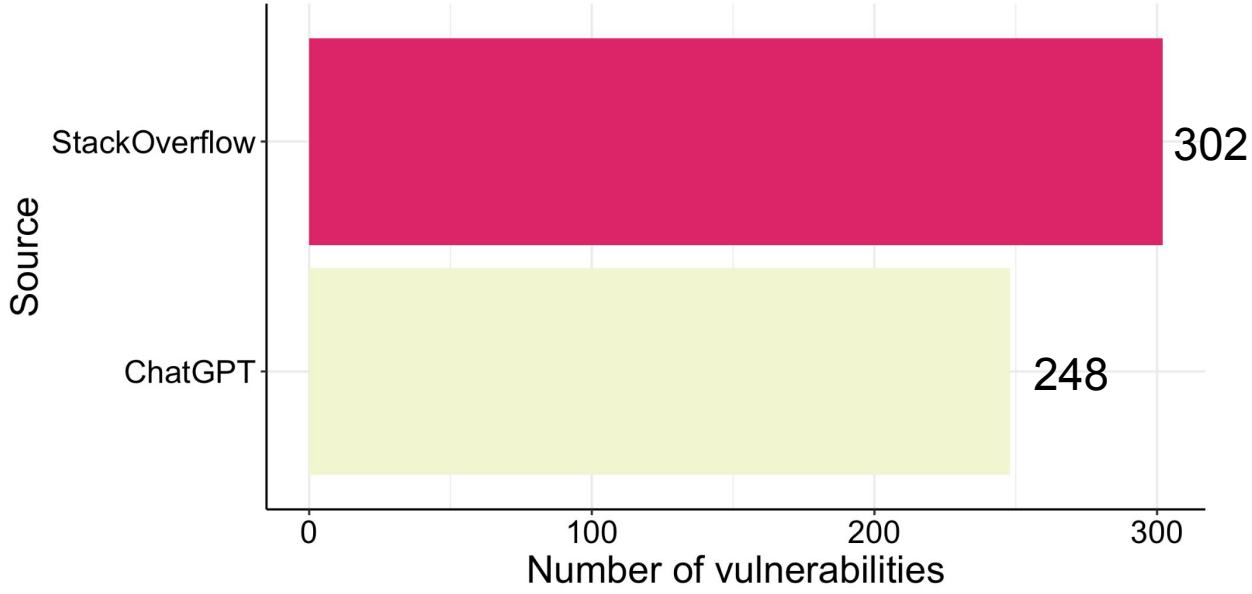
Here is a compilable code snippet in Java that you can use to encode and decode a string using 3DES:

```
....
final SecretKey key =
  new SecretKeySpec(keyBytes, "DESede");
final IvParameterSpec iv =
  new IvParameterSpec(new byte[8]);
final Cipher cipher =
  Cipher.getInstance("DESede/CBC/PKCS5Padding");
  cipher.init(Cipher.ENCRYPT_MODE, key, iv);
...
```

CodeQL
Compare vulnerabilities

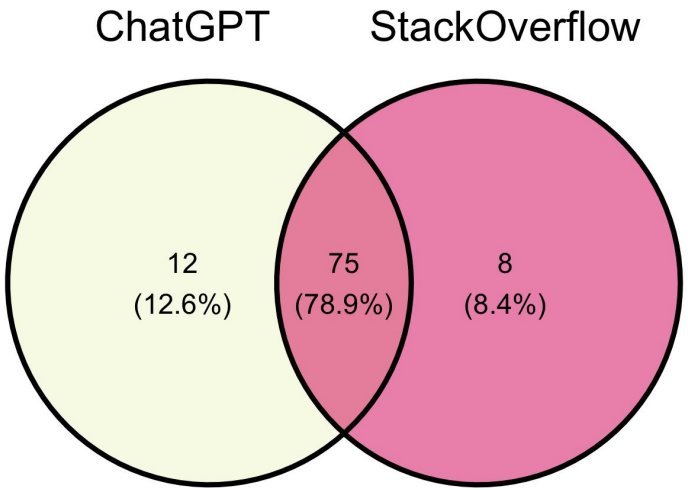
n = 108 snippet pairs

ChatGPT generated less vulnerabilities compared to StackOverflow

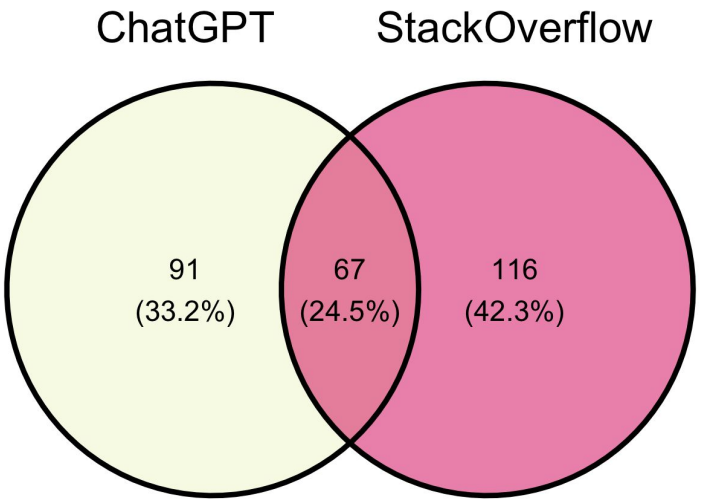


$p < 0.05$

Snippets with vulnerabilities are similar, yet the vulnerabilities are different



Snippets with vulnerabilities



Unique vulnerabilities in the snippets

Differences in some types of CWEs

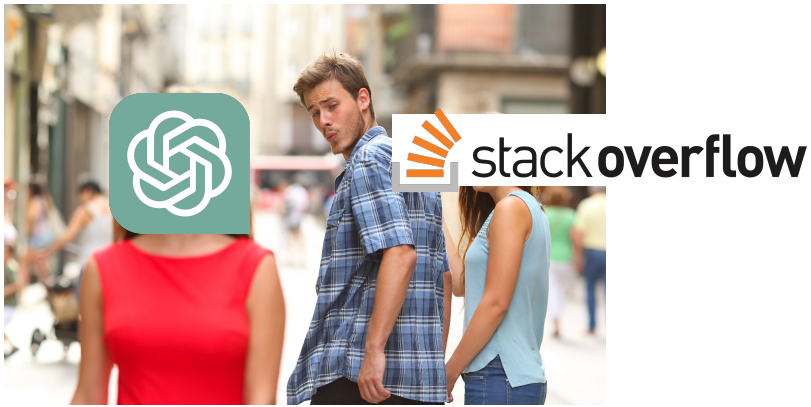
CWE-ID	CWE Name	#GPT	#SO	Top 25
CWE-335***	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)	0	16	-
CWE-835**	Loop with Unreachable Exit Condition (‘Infinite Loop’)	8	0	-
CWE-798*	Use of Hard-coded Credentials	36	20	18
CWE-570*	Expression is Always False	0	4	-
CWE-571*	Expression is Always True	0	4	-

Chi-squared test significance: ‘***’ $p < 0.001$, ‘**’ $p < 0.01$, ‘*’ $p < 0.05$

So… is it just another copy and paste?

No…

Differences between platforms



…But yes?

Security risk from any copy and paste

What should practitioners do?

1.  **Do not blindly trust** any code, AI-generated or human-created
2.  **Use** the platforms, but apply good software practice
3.  **Incorporate** LLMs within the tools or information sources

What should we do as researchers?

1.  **Empower** developers with more information
2.  **Reduce** insecure code propagation with tools
3.  **Understand** why differences occur between sources

ChatGPT generated less vulnerabilities than StackOverflow. Yet any code code copied and pasted, created by AI or humans, cannot be trusted blindly.

Paper

Dataset

Contact me!
Sivana Hamer ✨
sahamer@ncsu.edu

Billy Huynh