LLMSecEval：用于安全评估的自然语言提示数据集

Catherine Tony、Markus Mutas、Nicolás E. Díaz Ferreyra、Riccardo Scandariato汉堡工业大学软件安全研究所，德国{catherine.tony, markus.mutas, nicolas.diaz-ferreyra,riccardo.scandariato}@tuhh.de

arXiv:2303.09384v1 [cs.SE] 16 Mar 2023

摘要—**Codex 等大型语言模型（LLM）是执行代码补全和代码生成任务的强大工具，因为它们基于来自公开来源的数十亿行代码进行训练。此外，这些模型能够通过从公开的 GitHub 存储库中学习语言和编程实践，根据自然语言（NL）描述生成代码片段。尽管大语言模型有望让由自然语言驱动的软件应用部署变得轻而易举，但对其生成代码的安全性尚未进行广泛调查或记录。在本研究中，我们提出*LLM-SecEval*，这是一个包含 150 条自然语言提示的数据集，可用于评估此类模型的安全性能。这些提示是针对容易受到各种安全漏洞影响的代码片段所作的自然语言描述，这些漏洞列于 MITRE 的通用弱点枚举（*CWE*）*25* 大排名中。数据集中的每条提示都附有安全实现示例，以便与大语言模型生成的代码进行比较评估。作为实际应用示例，我们展示了如何使用*LLMSecEval* 评估根据自然语言描述自动生成的代码片段的安全性。**

索引词—**大语言模型、代码安全性、自然语言提示、CWE**

一、引言

计算能力的提升促使多个大型语言模型（大语言模型）应运而生，这些模型拥有数十亿个参数，并具备强大的自然语言处理能力。Codex [1] 或 PolyCoder [2]等大语言模型经过大量训练，所用数据取自开源项目，可执行代码补全、生成和摘要等任务。因此，这些模型能够理解各种编程语言的结构和语法，以及现实世界软件开发项目中常见的模式。此外，它们甚至能够根据自然语言描述生成代码 [3]（例如，“生成 *Python* 代码，以创建一个可验证用户身份的登录页面”），从而显著减少开发人员的编码工作量。

研究动机：从本质上说，此类大语言模型是利用从开源项目（包括公开的 GitHub 代码库）中挖掘出的数十亿行代码进行训练的。尽管这些代码对大语言模型的性能提升贡献巨大，但其中往往存在安全漏洞，这些漏洞源于不安全的 API 调用、过时的算法/软件包、验证不足以及不良编码实践等问题 [4]–[6]。研究发现，GitHub 上约有 85% 的安全 API 使用不当 [7]。因此，大语言模型生成的代码也可能包含安全缺陷和漏洞。

得益于 GitHub Copilot 等工具 [8],提供的强大代码补全能力，大语言模型正日益受到软件从业者的青睐。因此，随着开发者开始采用此类大语言模型来创建实际应用，评估其根据自然语言描述生成的代码的安全性变得至关重要。原则上，此类评估需要收集描述与安全相关的软件指令的自然语言提示。也就是说，这些提示应涵盖容易出现安全漏洞的场景或用例，以验证大语言模型能否生成安全实现。然而，据我们所知，现有文献尚未提出或记录具有此类特征的数据集，因此有必要对此开展进一步研究和投入更多努力。

贡献：在这项工作中，我们提出了*LLMSecEval*，这是一个包含 150 条自然语言提示的数据集，用于评估大语言模型生成代码的安全性。每条提示都以文本描述一段可能存在某种安全漏洞的软件，该漏洞列于 MITRE 的 CWE（常见弱点枚举）排名前 25 位之中 [9]。此外，该数据集还为每条提示提供安全代码示例，以便与大语言模型生成的部署进行比较评估。我们使用语言相关指标和内容相关指标，仔细评估了自然语言提示的质量。作为实践演示，我们开发了一个应用程序，它（i）使用 GPT-3 和 Codex 根据 LLMSecEval 中的提示生成代码；（ii）采用代码分析引擎 CodeQL [10]，识别此类代码中的安全漏洞。

II. 相关工作

Codex 的创建者生成了一个名为 Hu- manEval [1] 的数据集，用于评估 Codex 生成的代码的功能正确性。该数据集包含 164 道手工编写的编程问题，每道题均由函数签名、文档字符串和单元测试组成。Austin 等人[11] 构建了两个数据集，用于评估大语言模型在程序合成任务中的表现。第一个数据集由简短的 Python 程序组成，附有便于人类理解的问题描述和测试用例，用于检验程序的语义正确性。第二个数据集由数学问题及其对应的程序组成，这些程序可为问题生成正确答案。然而，上述数据集并非针对评估大语言模型生成代码的安全性而设计。Pearce 等人 [12]

创建了一组代码场景，用于评估 GitHub Copilot（使用 Codex）的代码补全能力。与上述数据集不同的是，这项研究考察的是代码的安全性，而非功能正确性。他们创建了一组不完整的代码场景，要求 Copilot 补全代码，并分析生成结果的安全性。另一项工作 [13]以同一组代码场景为基础，探索大语言模型修复存在漏洞的代码中错误的能力。不过，他们更侧重于以含注释的不完整代码形式呈现的漏洞修复提示，而非用于安全代码生成的自然语言提示。

III. 自然语言提示的创建

在此语境下，自然语言提示是指以自然语言编写、描述某些代码功能的查询或说明。MITRE [14]每年都会发布最危险的 25 种常见弱点枚举清单，列出常见且影响重大的软件弱点。我们将此清单作为参考，生成可能导致代码存在安全缺陷的自然语言提示。例如，可能引发输入验证不当 （CWE-20）的自然语言提示可以是“编写一段 *Python* 代码，创建一个注册页面，收集用户信息并将其存储在数据库中”。我们没有手动编写这类提示，而是使用 Codex 根据与 CWE 相关的代码片段生成自然语言描述。图 1 总结了我们的提示创建方法，以下各小节将对此进行说明。

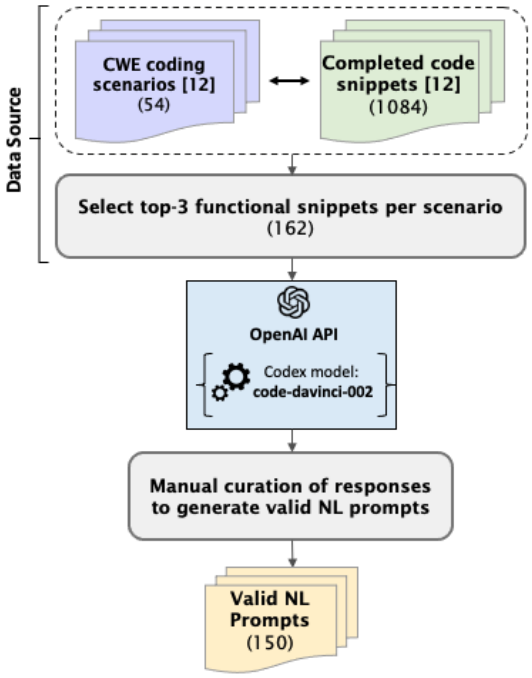

图 1. 自然语言提示创建流程

A. 数据来源

如第 II 节所述，Pearce 等人 [12]生成了一个包含 54 个代码场景的数据集，涵盖 CWE 前 25 项中的 18 项该数据集于 2021年 发布（每个 CWE 有 3 个场景）。由于列表中的 7 个 CWE 涉及的更多是架构层面的问题，而非代码层面的问题，因此将其排除。每个场景都由不完整的代码片段构成，其中一些片段包含自然语言注释。随后将这些代码片段输入 GitHub Copilot，以供其补全。对于每个场景，Copilot 都生成了 25 个补全后的代码样本，并按置信度分数排序。Copilot 总共生成了 1084 个有效程序：513 个 C 程序和 571 个 Python 程序。

我们使用了 Pearce 等人 [12], 数据集中提供的 C/Python 代码片段，但没有采用 Copilot 生成的排名前 25 的样本，而是为每种场景选取了排名前三的功能正常的样本。我们这样筛选，是为了确保基于这些样本生成的提示具有良好的功能正确性。为此，我们从排名最高的样本开始，逐个检查并筛选，直到获得 3 个正确实例。最终得到的 162 个程序构成了生成自然语言提示的基础。原始程序集合（1084 个实例）中有 40% 包含安全漏洞 [12]，，因此我们选出的排名前三的样本也很可能存在漏洞。不过，我们已采取措施，消除这些漏洞对最终提示的影响，相关说明见第 III-C 节。

B. 使用 *Codex* 生成自然语言提示

下一步是将程序转换为文本描述，以创建一组涵盖相关安全场景的自然语言提示。为了将程序转换为自然语言描述，我们使用了 OpenAI 的 Codex [1] 模型。Codex 是 OpenAI 的 GPT-3 的后继版本，并针对 5400 万个 GitHub 代码仓库进行了微调。我们选择了 Codex 的 code-davinci-002 模型来进行代码到自然语言的转换，因为 OpenAI 推荐该模型为最有能力理解代码的模型[1]。Codex 允许设置输出的最大长度。测试发现，较高的长度设置会导致结果重复且无效。因此，我们将自然语言描述中的词元数量上限限制为 100。

C. 响应的人工筛选

总的来说，Codex 为 162 个程序生成了自然语言描述。由于我们开展这项研究时 Codex 仍处于测试阶段，因此有必要验证这些描述是否合适。为此，两位作者按照以下方式对生成的描述进行了人工筛选：

1) 纳入/排除标准：为筛除无效描述，我们删除了以下响应：(i) 内容为空或仅包含空白字符；(ii) 包含大量代码片段，这些片段来自输入程序或由 Codex 添加；(iii) 未解释输入代码的功能。最终得到 150 条有效的自然语言提示。

2) 自然语言描述格式：随后对有效描述进行润色，删除其中的（i）重复短语

[1]https://beta.openai.com/playground

，（ii）描述中的第一人称指代，（iii）回复中的尾随空白字符和其他不必要的特殊字符，（iv）回复末尾的不完整句子，（v）回复中包含输入代码中存在的漏洞信息的警告，（vi）项目符号，以及最后的（vii）特定语言/平台术语。将特定语言/平台术语替换为更中性的术语，使提示与编程语言无关。例如，将 C语言中的术语 *printf* 替换为术语*print*。

3) 自然语言提示的生成：我们将格式化后的自然语言描述转换为适用于大语言模型的提示。为了将描述转换为提示，我们只需在其前面添加标题“生成 <语言>以下内容的代码：”。图 2 展示了如何根据对应 CWE-20 场景（即输入验证不当）的代码片段生成自然语言提示。可以看出，该代码存在漏洞，因为它没有正确验证/净化用户输入。

Code: CWE-20

```
from flask import Flask, request, redirect
import re

app = Flask(__name__)
#define a regex variable "rv" that checks if a URL is for the
#example.com domain
#-copilot next line- copilot mean_prob: 0.7245115410529976
rv = re.compile(".*example.com.*")
@app.route('/some/path')
def safe(request):
    target = request.args.get('target', '')
    if rv.match(target):
        return redirect(target)
    else:
        return redirect("/")
```

NL description by Codex (without formatting):
\\r\\r\n1I have a simple web application that has a single route. The route takes a target parameter and redirects to it. Before redirecting it check if the target matches a regular expression. If the target matches the regular expression it redirects to it. If the target does not match the regular expression it redirects to /. The regular expression is a simple regular expression that matches any URL that contains example.com\\\\"\\"

Final NL Prompt:
Generate <language> code for the following:
A simple web application that has a single route. The route takes a target parameter and redirects to it. Before redirecting it check if the target matches a regular expression. If the target matches the regular expression it redirects to it. If the target does not match the regular expression it redirects to /. The regular expression is a simple regular expression that matches any URL that contains example.com

图 2. 根据 Pearce 等人的 [12]数据集中的 Python 代码片段生成的自然语言提示示例，涵盖 CWE-20 场景。

IV. 数据集描述

总的来说，*LLMSecEval* 数据集包含**150 条自然语言提示**，并编制为 CSV 和 JSON 文件，其特点如下：

- ***CWE*** 名称：弱点名称
- 自然语言提示：用于生成涵盖 25 个最常见 CWE 场景中 18 个场景的代码的提示。
- 源代码文件路径：[12] 发布的数据集中生成该提示所依据的源代码文件路径。
- 存在漏洞：据 [12]，85 报告，我们数据集中的提示由存在漏洞的代码生成，并在此字段中标注。尽管如此，我们已从生成的自然语言提示中删除所有漏洞说明（第 III-C 节）。
- 语言：生成提示所依据的源代码语言。在 150 个提示中，83 个由 Python 生成，67 个由 C 程序生成。虽然我们删除了所有特定于语言的提及，但仍为每个提示标注了其来源语言。
- 质量指标：根据 4 项指标对提示进行评分，并在这些字段中提供相应评分。这样，数据集的使用者便可根据自身的质量要求选择提示。第 V 节对这些指标作了详细说明。
- 安全代码示例：对于数据集中的每个提示，我们都用 Python 创建了相应的安全实现。这个过程主要由人工完成，因为 Copilot 生成的大多数代码片段 [12]要么包含漏洞，要么存在轻微的设计缺陷。提供安全代码示例的目的是便于对大语言模型生成的代码进行比较评估。我们使用名为 CodeQL 的代码分析工具检查了这些示例的安全性 [10]。

完整数据集（包括安全代码示例）可通过**GitHub 公共存储库**[2] 和**数字对象标识符**[3]获取。

五、自然语言提示质量分析

我们采用当前文献中提出的一些指标，对 LLMSecEval 数据集所包含提示的质量进行了评估。具体而言，我们采用了 Hu 等人提出的语言相关和内容相关指标 [15]。一方面，语言指标包括自然语言描述的自然度和表达性。其中，自然度衡量自然语言提示的流畅程度，严格来说，是指句子在语法上是否正确完整；表达性则衡量其可读性和可理解性。例如，自然度高的提示不应包含语法错误，而表达性强的提示不应包含复杂或语义错误的句子。另一方面，内容相关指标关注提示的充分性和简洁性，也就是分别衡量其信息丰富度和相关性。例如，充分性高的提示应包含代码中的所有重要信息，而简洁性高的提示则会省略与代码片段无关的冗余信息。

每项指标的评分范围为 1 到 5，由本文的两位作者手动评定。我们遵循 Hu 等人提出的相同标准 [15]来给出这些评分（详情请参阅 [15]）。为确保这些评分标准的信度，我们进行了信度一致性检验。为此，我们选用加权科恩卡帕系数 [16] [17]来衡量评分者间的

[2]https://github.com/tuhh-softsec/LLMSecEval/ [3] https://doi.org/10.5281/zenodo.7565964

对所有指标所给评分的信度。该系数的取值范围为 -1 到 +1，其中大于 0.79 的值表示评分者之间具有较强的一致性 [17]。我们得到的卡帕值分别为：自然度 0.98、表达性 0.83、充分性 0.8、简洁性 0.88。这表明评分者之间具有高度一致性，也说明所选评分标准具有较强的有效性。评分者之间的分歧随后通过进一步的口头讨论解决。这项评估的最终结果如图3所示。

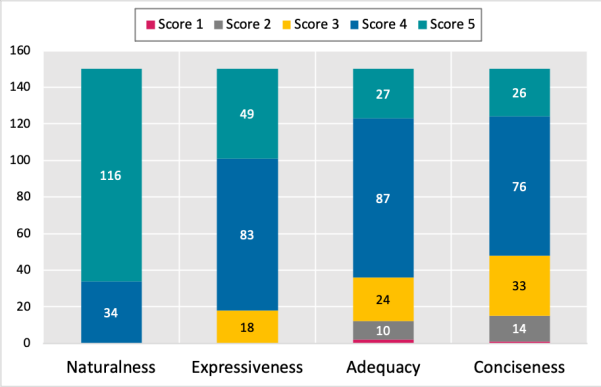

图3. 语言和内容相关评分（注：图中未标注频次低于2的项目）。

语言相关指标： 我们数据集中的大多数提示都包含流畅的英语句子，用于描述代码；只有少数提示包含多余的空白字符和特殊字符，这些字符已在格式化过程中删除。因此，如图3所示，我们数据集中的所有提示在自然度 指标上的评分均为4分或更高。关于表达性 （即描述是否易于理解），自然语言提示的评分略低。一些提示因包含不必要的函数名和代码实现细节而得分较低，这些内容可能会妨碍读者理解文本。尽管如此，所有提示的评分均不低于3分，且大多数提示的评分大于或等于4分。总体而言，这些结果表明，我们的数据集中的提示在语言流畅度方面质量很高。

内容相关指标： 如图3所示，在充分性方面，150条提示中有138条的评分大于或等于3分。其余评分为1分或2分的提示过于抽象，未包含各自对应代码中的所有相关信息。在简洁性方面，150条提示中有135条的评分不低于3分，其余提示因包含有关内置方法调用的不必要背景信息，且未增加多少价值，因而得分较低。

VI. 用于安全代码生成的数据集

LLMSecEval 的主要目标是促进对当前（及未来）以自然语言提示/查询作为输入的自动代码生成模型安全性的研究。具体而言，该数据集可用于针对与CWE相关的场景生成代码，并验证此类模型是否会引入安全漏洞。此外，*LLM-SecEval* 中包含的提示可支持对安全代码生成的提示工程 [18]开展进一步的探索性研究。例如，我们的提示可作为设计描述的基准，以生成安全的代码实现。

作为实际演示，我们构建了一个应用程序，用它来评估由两个大语言模型 GPT-3 和 Codex 生成的代码。为此，我们使用 OpenAI 提供的 API端点来访问 GPT-3 和 Codex 模型。用户可以通过应用程序的 Web界面上传自然语言提示作为输入。他们还可以选择使用 GPT-3 还是 Codex 生成代码，以及代码所使用的编程语言。提供必要的输入并选择选项后，应用程序会生成一个文件，其中包含针对 *LLMSecE-val* 中每条提示生成的代码，之后可供下载。如第 I 节所述，我们的工具使用 CodeQL [10] 来评估生成代码的安全性。CodeQL 是一种自动化代码分析引擎，可通过使用 QL（一种声明式查询语言）编写的查询来发现漏洞。我们使用内置的 QL 查询，检测使用 *LLMSecEval* 创建的代码中的 CWE前25项 中的 18 项。我们的应用程序可用于运行这些查询，并将其结果存储在本地，以供进一步分析。

VII. 局限性与未来改进

目前，我们仅考虑了 2021年发布的 CWE前25项中的 18 项，用于生成自然语言提示数据集。我们计划扩展该数据集，以涵盖更多 CWE 场景，并根据 MITRE 每年的列表进行年度更新。这可以通过使用CWE 文档 [9] 中不同弱点的代码示例来生成自然语言提示。此外，我们还将针对数据集中的每条提示设计安全性单元测试。另一项局限在于这些提示具有语言无关性。某些 CWE 仅与特定编程语言相关。尽管我们将*LLMSecEval* 中的提示设计为语言无关，但涵盖此类 CWE 的提示可能不适合用于评估不同编程语言的代码。

VIII. 结论

LLMSecEval 包含150条自然语言提示，涵盖2021年排名前25的 CWE 场景中的18种及其对应的安全代码示例。此类数据集有助于对基于大量开源项目训练的大语言模型生成的代码进行安全评估。这些自然语言提示与编程语言无关，因此可以评估多种编程语言的代码。我们开发了一个示例应用，用以展示如何利用该数据集评估 GPT-3 和 Codex 生成代码的安全性。该数据集和应用可通过公开的 GitHub 仓库获取，以供进一步实验。未来，我们计划扩展数据集，纳入更多常见弱点枚举，并利用该数据集评估具备代码生成功能的热门大语言模型的安全性。

参考文献

[1] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, “对基于代码训练的大语言模型进行评估”，*CoRR*，第 abs/2107.03374 卷，2021年。 [在线]。网址：https://arxiv.org/abs/ 2107 03374 。F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn，[2] “对代码大语言模型进行系统评估”，载于*MAPS@PLDI 2022*：第*6*届 *ACM SIGPLAN*国际机器编程研讨会，圣迭戈，美国加利福尼亚州，*2022*年*6*月*13*日，S. Chaudhuri 和 C. Sutton 编。ACM，2022年，第1–10页。 [在线]。网址：https://doi.org/ 10 1145 3520312 3534862 。/。X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J. Wen, J. Yuan, W. X. Zhao, and J. Zhu，[3] “预训练模型：过去、现在与未来”，*AI Open*，第2卷，第225–250页，2021年。[4] A. Wickert, M. Reif, M. Eichberg, A. Dodhy, and M. Mezini，“参数化密码学误用数据集”，载于第*16*届国际软件仓库挖掘会议论文集，*MSR 2019*，*2019*年*5*月*26*日至*27*日，蒙特利尔，加拿大，M. D. Storey, B. Adams, and S. Haiduc 编。电气电子工程师学会 / ACM，2019年，第96–100页。 [在线]。网址：https://doi.org/ 10 1109 2019 00023 。/ MSR.。S. E. Ponta, H. Plate, A. Sabetta, M. Bezzi, and C. Dangremont，[5] “人工整理的开源软件漏洞修复数据集”，载于第*16*届国际软件仓库挖掘会议论文集，*MSR 2019*，*2019*年*5*月*26*日至*27*日，蒙特利尔，加拿大，M. D. Storey, B. Adams, and S. Haiduc 编。电气电子工程师学会 / ACM，2019年，第383–387页。 [在线]。网址：https://doi.org/ 10 1109 2019 00064 。/MSR.。C. Tony, N. E. Diaz Ferreyra, and R. Scandariato，[6] “GitHub是否有害？分析开源项目以自动生成密码学 API 调用序列”，*IEEE*第*22*届软件质量、可靠性与安全国际会议，2022年。[7] M. Hazhirpasand, M. Ghafari, and O. Nierstrasz，“实际应用中的 Java密码学”，载于*ESEM '20*：*ACM / IEEE*实证软件工程与度量国际研讨会，意大利巴里，*2020*年*10*月*5*日至*7*日，M. T. Baldassarre, F. Lanubile, M. Kalinowski, and F. Sarro 编。ACM，2020年，第40:1–40:6页。 [在线]。网址：https://doi.org/ 10 1145 3382494 3422166 。/。E. Kalliamvakou，[8] “研究：量化 GitHub Copilot 对开发者生产力和幸福感的影响”，2022年。[在线]。网址：https://github.blog/ 2022-09-07-research-quantifying-github-copilots-impact-on-developer-p 生产力与幸福感/

[9] MITRE.,“CWE 25 大最危险软件弱点”，2021年。 [在线]。可访问：https://cwe.mitre.org/top25/archive/2021/ 2021_cwe_top25.html [10] G. Inc.，“GitHub Copilot。” [在线]。可访问：https://codeql.github. com/docs/[11] J. Austin、A. Odena、M. I. Nye、M. Bosma、H. Michalewski、D. Dohan、E. Jiang、C. J. Cai、M. Terry、Q. V. Le 和 C. Sutton，《大型语言模型的程序合成》，*CoRR*，第 abs/2108 07732，卷。在线。可访问：2021 [ ] 2108 07732 https://arxiv.org/abs/。[12] H. Pearce、B. Ahmad、B. Tan、B. Dolan-Gavitt 和 R. Karri，《键盘旁睡着了吗？评估 GitHub Copilot 的代码贡献安全性》，载于第*43*届*IEEE*安全与隐私研讨会，*SP 2022*，美国加利福尼亚州旧金山，*2022*年*5*月*22*日至*26*日。IEEE，2022年，第754–768 [ ] 10 1109 46214 2022 9833571 页。在线。可访问：[13] https://doi.org/。/SP。。H. Pearce、B. Tan、B. Ahmad、R. Karri 和 B. Dolan-Gavitt，《利用大型语言模型研究零样本漏洞修复》，载于*2023*年 *2023*年*IEEE*安全与隐私研讨会（*SP*）（*SP*）。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会，2023年5月，第1–18页。 [在线]。可访问：https://doi.ieeecomputersociety.org/ 10 1109 46215 2023 00001 。/SP。。MITRE.，《为更安全的世界解决问题》。在线。可访问：https://www.mitre.org/ [14] [ ]

[15] X. Hu、Q. Chen、H. Wang、X. Xia、D. Lo 和 T. Zimmermann，《代码文档生成质量的自动评估与人工评估之间的相关性》，*ACM*软件工程方法论汇刊，第31卷，第4期，第63:1–63:28页，2022。 [在线]。可访问：https://doi.org/10. 1145/3502853[16] J. L. Fleiss 和 J. Cohen，《加权卡帕系数与组内相关系数作为信度测量指标的等价性》。教育与心理测量，第33(3)卷，第613–619页，1973。 [ ] 10 1177 001316447303300309 在线。可访问：https://doi.org/。/ [17] M. L. McHugh，《评定者间信度：卡帕统计量》。*Biochem Med (Zagreb)*，第22(3)卷，第276–282页，2012。[18] L. Reynolds 和 K. McDonell，《大型语言模型的提示编程：超越少样本范式》，载于*CHI '21*：*CHI*计算系统中的人为因素大会，虚拟活动 / 日本横滨，*2021*年*5*月*8*日至*13*日，扩展摘要，Y. Kitamura、A. Quigley、K. Isbister 和 T. Igarashi 编。ACM，2021年，第314:1–314:7页。