OWASP
Open Web Application
Security Project

代码安全大模型的设计、应用和实践

李涛

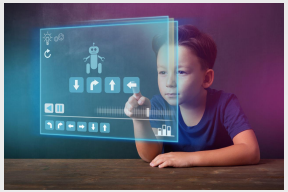
代码安全的挑战来源于软件本身的复杂性、多样性、动态性

01 复杂性

- 现实世界映射
- 复杂系统
- 生态系统

02 多样性

- 业务多样性
- 语言多样性
- 逻辑多样性
- 形式多样性



03 动态性

- 演化
- 执行路径
- 生命周期

软件安全的技术发展路径

规则引擎

- 用规则描述专家经验
- 成本高、完备性弱

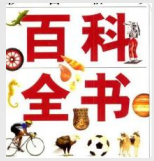

机器学习、深度学习

- 大数据驱动模型建模
- 针对特定问题，泛化性差

- （1）软件代码度量
- （2）代码属性挖掘
- （3）代码相似性漏洞挖掘
- （4）代码模式挖掘


大模型

- **大**：参数大、规模达
- **多**：支持多个复杂任务
- **智**：涌现，举一反三



生成代码　漏洞检测

自动测试　自动注释

现有代码审计检测工具痛点分析

规则引擎误报高导致大量人工开销

较高的误报率：

现有的规则引擎工具误报率在30%以上，甚至达到了50%或者更高。大量的误报严重降低了工作效率。

规则泛化性差：

规则的制定和调整往往依赖于专家经验，难以覆盖所有可能的代码模式和漏洞场景。

代码漏洞缺乏解释和自动修复能力

漏洞解释不充分：

漏洞缺乏可解释性，有时候难以理解，需要投入大量的时间进行人工分析。审计效率低。

依赖人工修复：

代码修复主要依赖人工，漏洞知识库仅仅给出案例参考，学习成本较高，漏洞修复耗时费力。

通用大模型审计工具检测效率低

扫描速度慢：

通用大模型扫描大型项目时速度较慢，影响开发效率，不适应敏捷开发等。

硬件资源依赖：

检测效率受限于硬件资源，难以在资源有限的环境中高效运行。

传统SAST工具基于规则检测误报率高、扫描速度慢，需要耗费大量人力投入分析和修复缺陷。

通用大模型生成代码安全性不高

Security Weaknesses of Copilot Generated Code in GitHub

YUJIA FU, Wuhan University, China
PENG LIANG, Wuhan University, China
AMJED TAHIR, Massey University, New Zealand

Is GitHub's Copilot as Bad as Humans at Introducing Vulnerabilities in Code?

Owura Asare*, Meiyappan Nagappan and N. Asokan

Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1.

*Corresponding author(s). E-mail(s): oasare@uwaterloo.ca;
Contributing authors: mei.nagappan@uwaterloo.ca;
asokan@acm.org;

Table 3. The number and percentage of code snippets with security weaknesses generated by Copilot

Language	# Snippets	# Snippets containing security weaknesses	%
Python	277	91	32.8%
JavaScript	175	43	24.6%
Total	**452**	**134**	**29.6%**

Table 8. CWEs and security issue types in Copilot-generated code

Type of Security Issue	Relevant CWEs
Data Neutralization Issues	CWE-94, CWE-78, CWE-117, CWE-89
Resource Management Errors	CWE-772, CWE-502
Error Conditions, Return Values	CWE-396, CWE-617
Bad Coding Practices	CWE-628, CWe-563
Credentials Management Errors	CWE-798
Information Management Errors	CWE-312, CWE-215
Authentication Errors	CWE-295
Concurrency Issues	CWE-367
File Handling Issues	CWE-22

现有代码大模型的安全性分析---生成代码不安全

自回归模型（Autoregressive Model）
输入：语言序列
输出：下一个单词或字符的概率分布

The best thing about AI is its ability to	learn	4.5%
	predict	3.5%
	make	3.2%
	understand	3.1%
	do	2.9%

The best thing about AI is its ability to,
The best thing about AI is its ability to learn,
The best thing about AI is its ability to learn from,
The best thing about AI is its ability to learn from experience,
The best thing about AI is its ability to learn from experience.,
The best thing about AI is its ability to learn from experience. It,
The best thing about AI is its ability to learn from experience. It's,
The best thing about AI is its ability to learn from experience. It's not

每一次输出都计算下一个单词（token）的概率分布

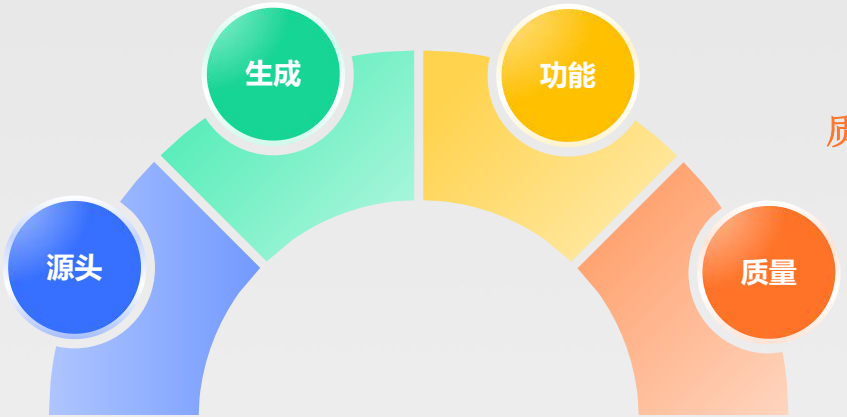
软件安全大模型的设计思路

生成安全

生成代码无漏洞

功能性

代码解释、代码补全、代码注释等



源头安全

保障训练代码数据无漏洞

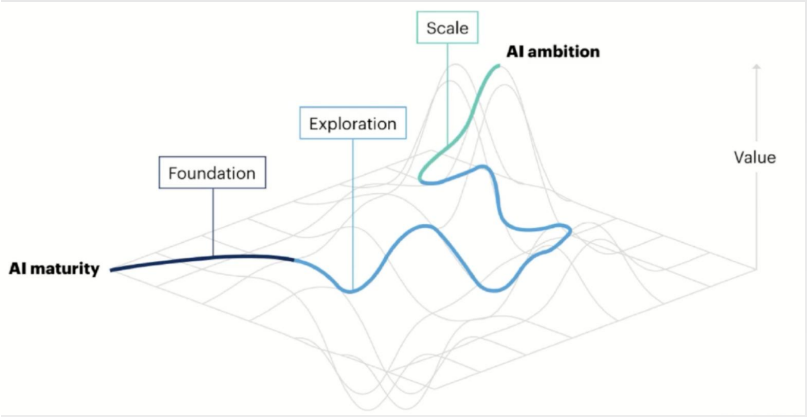
质量评价

连贯性、采纳率、完整性、安全性

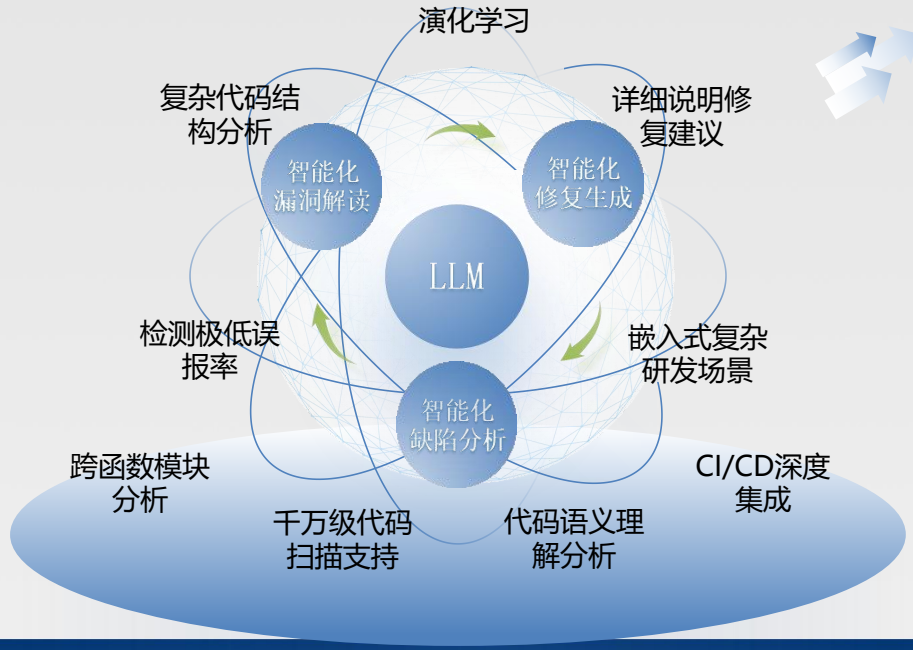
企业级代码安全大模型的设计目标

通用大模型缺乏行业高质量数据训练，**难以解决代码安全生成落地的“最后一公里”**，需要聚焦企业级大模型在代码生成垂直领域和企业的应用场景落地。

	通用大模型	专用大模型
泛化性	好	较好
性能	差	好
准确性	通用场景好，专用场景差	专用场景好
成本	初期低，长期高	初期高，长期低
安全性	差	高
行业数据	无或者较少	高质量、海量行业数据
部署	云端	私有化
集成	不能集成	集成生产环境



代码安全“小”模型



降低误报，漏洞解读

大模型通过综合运用数据处理、特征提取、上下文分析、语义理解、逻辑推理等多种技术手段，实现漏洞精准确认和深入解读，**解决传统工具误报高的痛点与难点，降低审计成本。**

生成安全代码和自动修复代码

大模型能够更全面地理解漏洞并生成更具针对性的修复代码。**降低对人员能力的要求，提成漏洞的修复效率。**

代码缺陷分析

大模型通过深度语义理解、并行处理能力、智能化特征提取、自适应学习与优化、资源优化利用以及定制化解决方案等创新方式，实现了对代码缺陷分析的高效赋能。

软件安全专用大模型

软件安全专用大模型在自有数据集训练，实现软件开发全生命周期安全检测和管理，涵盖安全设计、安全需求、代码编程、漏洞检测、单元测试和数据安全，具有高性能、低误报、多场景的优势。

- AIdesign，针对设计文档，运用大模型检测是否符合安全标准和规范
- AISecurity：高性能代码漏洞检测和自动修复工具
- AITest：运用大模型自动生成测试用例
- AIaudit：运用大模型进行敏感数据检测

代码安全大模型架构图


应用层
快速创建项目
检测项目管理
检测队列管理
报告管理
项目管理
检测方案配置
数据统计
模型微调
漏洞解读插件
生成修复方案
增量扫描
定时扫描
模型配置
依赖配置
账号管理
个人中心
生态集成
帮助中心
权限管理
IDE插件
……
双中台
业务中台
用户中心
项目管理
报告管理
功能设置
数据中心
系统配置
引擎中台
AI漏洞检测伴侣
复杂代码结构分析
漏洞代码解读
AI修复方案伴侣
详细说明修复建议
……
AI代码生成平台
代码语义理解分析
IDE安全扫描插件
服务层
负载均衡
文件服务
消息队列
日志服务
操作系统
数据库
中间件

产品形态

从漏洞检测、代码审计到漏洞修复一站式代码安全管理赋能，多种产品形态满足不同的应用场景。


李涛
湖北 武汉
扫一扫上面的二维码图案，加我为朋友。