ProphetFuzz: Fully Automated Prediction and Fuzzing of High-Risk Option Combinations with Only Documentation via Large Language Model

会议： CCS 2024

作者：

Dawei Wang
Zhongguancun Laboratory
Beijing, China
wangdw@zgclab.edu.cn

Geng Zhou
Zhongguancun Laboratory
Beijing, China
zhougeng@zgclab.edu.cn

Li Chen*
Zhongguancun Laboratory
Beijing, China
lichen@zgclab.edu.cn

Dan Li
Tsinghua University
Beijing, China
tolidan@tsinghua.edu.cn

Yukai Miao
Zhongguancun Laboratory
Beijing, China
miaoyk@zgclab.edu.cn

开源： https://github.com/NASP-THU/ProphetFuzz

主要贡献

1. 开发了ProphetFuzz，这是一种基于大型语言模型(LLM)的完全自动化工具，专门用于预测高风险选项组合并进行模糊测试。
2. ProphetFuzz成功地预测了覆盖52个程序的数据集上的1748个高风险选项组合，发现364个唯一漏洞，此外，ProphetFuzz 确定了这些程序最新版本中 140 个零日或半日漏洞，开发人员确认 93 个，获 CVE 编号 21 个。

具体思路

总体框架

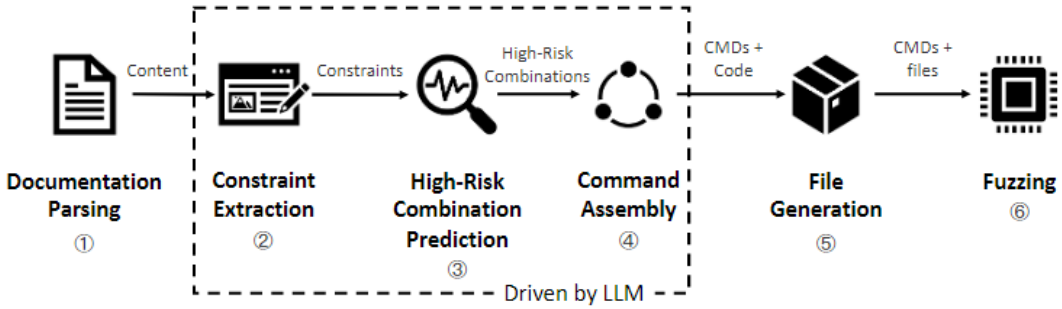


Figure 1: Overview of ProphetFuzz.

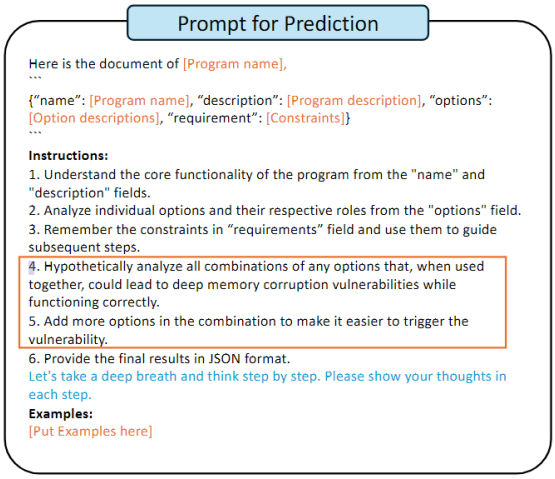
Constraint Extraction

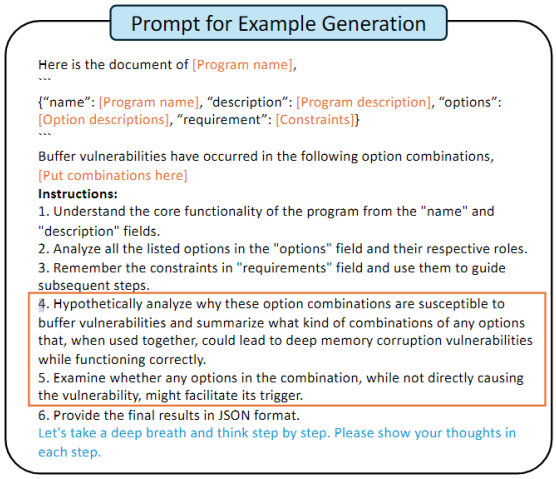
- 考虑到不合适的参数之间会让程序提前终止，we utilize LLM to extract constraints between options from the program documentation before predicting high-risk combinations
- provide the LLM with the program description and the option descriptions found in the program documentation
- Prompt
  - "Please separate the options and create individual descriptions for each option based on the original description"
  - "Please find any options that are mutually exclusive or logically conflicting when selected together and find any options that have dependencies on other options."
  - adjusting the temperature setting
  - instruct the LLM to make multiple inferences for each program and retain the union of the results of each inference
- 考虑到llm固有的幻觉 -> self-check approach based on bidirectional reasoning 基于双向推理的self-check方法，见Table1，即针对每一对的冲突和依赖分别考虑验证和反例。为了进一步验证
  - 降低温度 and 在新的会话中重新评估

总结：该阶段主要用于提取参数之前的约束，交由LLM实现。具体实现包括：提取参数之间的约束，调整温度、多次实验取并集；利用self-check对约束进行验证和举反例，同时调整温度并在不同的session中评估。

High-Risk Combination Prediction

- 关键方法：chain of thought
- six steps in prompts
  1. 基于程序名和功能描述，引导LLM对该程序有宏观的了解
  2. LLM分析选项（参数）
  3. 提醒LLM考虑前一步提取出来的选项之间的约束
  4. LLM检查所有组合并识别可能构成漏洞的组合
  5. 让LLM大胆假设并识别会造成深度内存破坏的组合 - “when used together, they could lead to vulnerabilities in deep memory corruption while functioning correctly”。同时，一旦LLM识别了能触发漏洞的选项，会进一步要求LLM尝试添加其他有助于触发漏洞的选项
  6. 以JSON格式输出结果
- present examples，并且参考vulnerability reports in GitHub Issues.并且修改prompts中的step4和step5





总结：该阶段是利用前面对程序文档和选项限制的分析，大胆预测什么样的选项组合会导致内存漏洞。并且给LLM提供了样例，样例是手动从github上查找的

Command Assembly

- 利用LLM自动化地组装命令
- 步骤：

  将高风险选项组合转换为实际的命令行指令 -> 据选项描述和概要，识别出需要值的选项，并为这些选项生成或分配值 -> 生成初步的可执行命令 -> 文件占位符，并为文件占位符生成可以在沙盒中生成对应文件的python脚本

File Generation and Fuzzing

- fuzz之前，利用python代码生成所需的文件

实验

- GPT-4 Turbo (gpt-4-1106-preview)
- using docker to generate file
- Comparison benchmark : CarpetFuzz
- Each fuzzing instance is run for 72 hours and repeated five times

RQ

- ProphetFuzz identifies 1.33 times more than that of CarpetFuzz under the same conditions
- ProphetFuzz在约束提取的精确度和召回率方面均优于CarpetFuzz，这得益于ProphetFuzz精心设计的提示和LLM的文本理解与推理能力。自检方法对于提高精确度至关重要，而LLM的强文本理解和推理能力在没有自检方法的情况下无法保证正确的约束提取结果
- 通过消融实验证明了自检机制和少量样本学习方法的有效性
- 通过消融实验证明了Command Assembly模块的重要性
- 实验证明能够有效发现零日漏洞

可进一步完善的方向

- 随着LLM的发展，实验精准度会提升
- 如果文档不够准确或者完善，实验精准度会受到影响。未来可以通过让ProphetFuzz分析源码来进一步提高它的能力
- 未来可以让ProphetFuzz预测更多类型的漏洞，而不仅仅是内存漏洞
- 计划在未来版本中允许LLM自动选择最适合变异的目标
- 计划让安全专家进行更深入的分析，以产生更高质量的语料库，提高LLM学习预测高风险选项组合的效果