Obfuscation Techniques and Their Impact on LLM Vulnerability Detection Performance

Assignee Research

June 4, 2026

Abstract

This report synthesises findings from 13 peer-reviewed papers addressing the following research question: What is the impact of variable renaming and control flow flattening on the F1 scores of Llama3 versus Codestral when evaluated on the Big-Vul dataset. 15 claims were extracted from source literature; 3 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 5.5/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: A Systematic Study of Code Obfuscation Against LLM-based Vulnerability Detection. Research question: What is the impact of variable renaming and control flow flattening on the F1 scores of Llama3 versus Codestral when evaluated on the Big-Vul dataset?.

2 Methodology

Systematic literature search across multiple databases yielded 13 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 5.5/10.

3 Results

13 papers retrieved. 15 claims extracted; 3 independently verified. Quality review score: 5.5/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
Code obfuscation transformations can unexpectedly improve vulnerability detection accuracy by removing misleading surfac	×	0.09
Control-flow virtualization and mixed-programming-language transformations have the strongest degrading effect on LLM-ba	✓	0.18
Models smaller than 8B parameters show pronounced instability under code obfuscation.	×	0.05
Models larger than 8B parameters maintain higher resilience to code obfuscation, though additional scaling yields dimini	×	0.04
Reasoning-augmented models perform better on unobfuscated code but are more sensitive to obfuscation than non-reasoning	×	0.05
Vulnerability types involving pointer safety, reentrancy, and access control show the largest fluctuations in detection	×	0.07
Coding agents exhibit higher detection success rates than general-purpose LLMs on unobfuscated code.	×	0.07
Coding agents experience both downgrade and upgrade effects in detection accuracy under code obfuscation.	×	0.10
Coding agents are particularly susceptible to performance degradation from inline assembly and virtualization obfuscatio	×	0.07
Hot-plugging a new model into an agent framework can reduce the effectiveness of transferring vulnerability-detection kn	×	0.06
The study categorizes existing obfuscation techniques into three major classes: layout, data flow, and control flow.	✓	0.21
The obfuscation taxonomy covers 11 subcategories and 19 concrete methods.	×	0.13
The evaluation framework implements transformations across four programming languages: Solidity, C, C++, and Python.	×	0.14
The study evaluates the impact of obfuscation on 15 LLMs spanning four model families: DeepSeek, OpenAI, Qwen, and LLaMA	✓	0.23
The study evaluates two coding agents: GitHub Copilot and Codex.	×	0.14

References

- http://arxiv.org/abs/2106.16020v1
- http://arxiv.org/abs/2512.16538v1
- http://arxiv.org/abs/2508.19294v2