



GraphLedger: Repository-Level Vulnerability Detection via Graph-Based Compression and Assumption Validation

Thi-Hong-Cuc Le^{1,2} Hoang-Quoc-Bao Hua^{1,2} Xuan-Bach Le^{1,2}

¹Faculty of Computer Science and Engineering,
Ho Chi Minh City University of Technology (HCMUT)

²Vietnam National University Ho Chi Minh City, Vietnam

RAISE Lab

Presenter: Thi-Hong-Cuc Le





Xuan-Bach Le

lexuanbach.github.io

Advisor / Corr. author

RAISE Lab • Lab Head

Advisor profile

Lecturer at HCMUT working on program verification, formal logic, RAG/LLMs, program synthesis and reliable software engineering. **Specialty:** Safe LLM agents



Thi-Hong-Cuc Le

nausmind.com

First author / Presenter

RAISE Lab • Core Member

Research focus

Trustworthy AI for software engineering, including explainable AI and LLM-assisted vulnerability detection.



Hoang-Quoc-Bao Hua

hhqbao.sdh241@hcmut.edu.vn

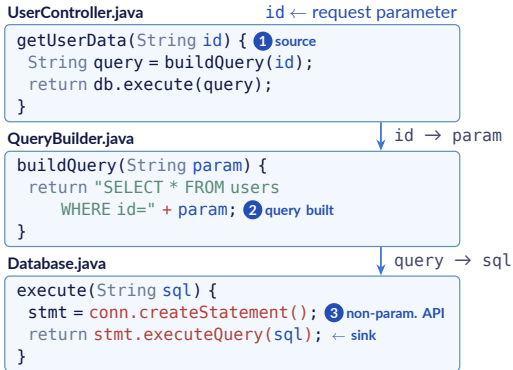
Co-author / M.Sc. student

Computer Science • HCMUT

Research focus

LLM agents for static analysis.

Motivation and Hypothesis



The Problem

Find security bugs where untrusted input crosses files before reaching a dangerous operation.

Why It Is Hard

- ⦿ Whole repositories overwhelm LLMs.
- ⦿ Short excerpts may hide the path or a defense.
- ⦿ Automated warnings may be false alarms.

Hypothesis: keeping the code needed to check a suspected bug can cut false alarms without missing many real bugs.

Related Work

Context selection

e.g. LLMxCPG¹ (USENIX Sec'25), SnapVuln² (FSE'23), VulnSC³ (ISSTA'25)

APPROACH

slice or abstract the code

LIMITATION

compression is a separate step from later checks

Hypothesis validation

e.g. VulAgent⁴ (ACL Findings'26)

APPROACH

an agent validates LLM hypotheses

LIMITATION

only as good as the context it is given

GraphLedger

compression co-designed with verification

APPROACH

AACC retention rules match ledger checks

DIFFERENCE

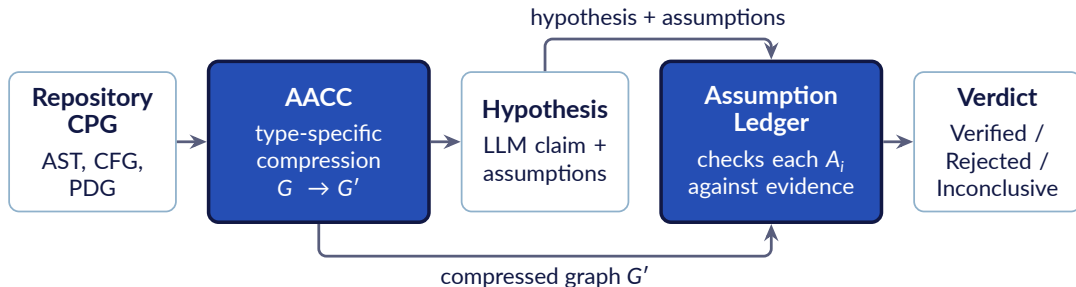
API evidence for A_3 survives compression

¹ Lekssays et al., LLMxCPG: Context-Aware Vulnerability Detection Through Code Property Graph-Guided Large Language Models, USENIX Security 2025.

² Wu et al., Learning Program Semantics for Vulnerability Detection via Vulnerability-Specific Inter-Procedural Slicing, ESEC/FSE 2023.

³ Wu et al., Enhancing Vulnerability Detection via Inter-Procedural Semantic Completion, ISSTA 2025.

⁴ Wang et al., VulAgent: Hypothesis-Validation Driven Multi-Agent Architecture for Vulnerability Detection, Findings of ACL 2026.

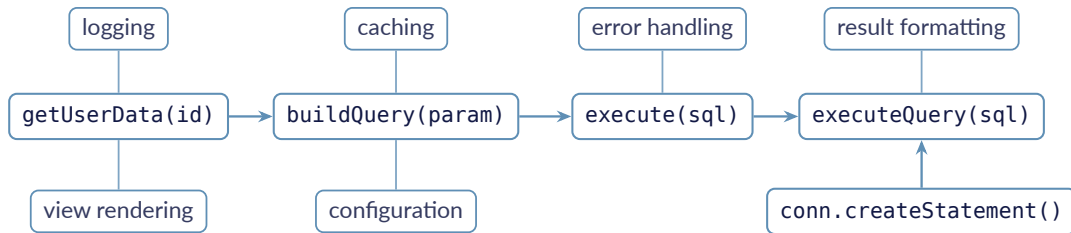


Ledger-Oriented Retention

AACC (Adaptive Assumption-Corroborating Compression) uses retention predicates designed to preserve evidence for downstream assumption checks.

CPG: code property graph (syntax tree, control flow and program dependencies in one graph).

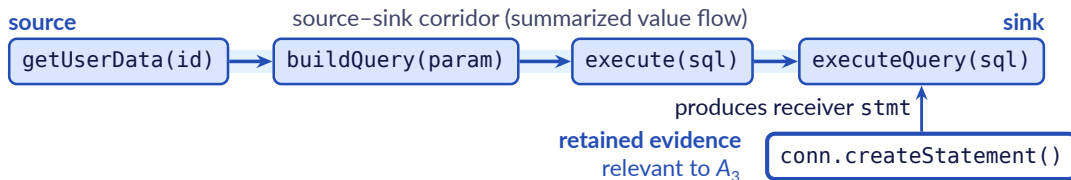
Graph Compression



Before: every node reachable from the path looks alike.

Keep: dependency paths and API evidence needed for ledger checks.

Graph Compression



After: six peripheral roles pruned, corridor and A_3 evidence kept.

Keep: dependency paths and API evidence needed for ledger checks.

Assumption Ledger

Assumption	Repository evidence	State
① A_1 id is user-controlled	id bound to a request parameter	validated
② A_2 no sanitization on the path	flow <code>id→param→sql</code> reaches the sink, no sanitizer on it	validated
③ A_3 non-parameterized sink	<code>conn.createStatement()</code> <code>→ stmt.executeQuery(sql)</code>	validated

Verdict: **Verified** — a bounded witness path exists and all three assumptions are validated.

Verified = bounded path + all validated · **Rejected** = any undermined · **Inconclusive** = otherwise.

No evidence $\Rightarrow$ **pending**, never validated.

Illustrative ledger. Search bound: paths of length ≤ 20 , at most 8 paths per source–sink pair.

Assumption Ledger

Hypothetical parameterized variant, evaluated separately

Assumption	Repository evidence	State
① A_1 id is user-controlled	id bound to a request parameter	validated
② A_2 no sanitization on the path	id reaches the query only via <code>setString(1, id)</code> , no sanitizer on it	validated
③ A_3 non-parameterized sink	<code>conn.prepareStatement("... WHERE id=?")</code>	undermined

Verdict: **Rejected** — the parameterized API contradicts A_3 , so A_3 is undermined.

Verified = bounded path + all validated · **Rejected** = any undermined · **Inconclusive** = otherwise.

No evidence $\Rightarrow$ **pending**, never validated.

Illustrative ledger. Search bound: paths of length ≤ 20 , at most 8 paths per source-sink pair.

Experimental Setup

Datasets

CWE-Bench-Java (from IRIS, ICLR 2025)

120 Java repositories with manually checked vulnerabilities in four classes: path traversal, XSS, code injection and command injection. Each case identifies a vulnerable project version and its affected files.

CVEfixes (PROMISE 2021)

CVEs linked to fixing commits, giving code before and after a fix. We study 100 Java cases from 32 projects.

Research Questions

RQ1 Detection accuracy

RQ2 Compression quality

RQ3 Ledger decisions

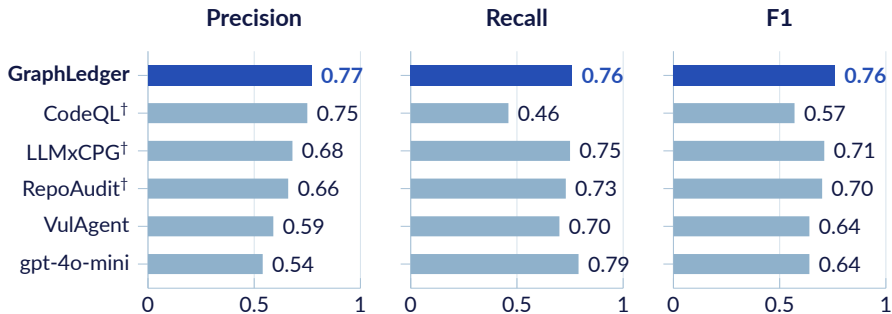
RQ4 Components and runtime

Protocol

CWE-Bench-Java: 100 repositories for detection, 20 for sensitivity.

Ten baselines use the same scorer. Inconclusive counts as negative.

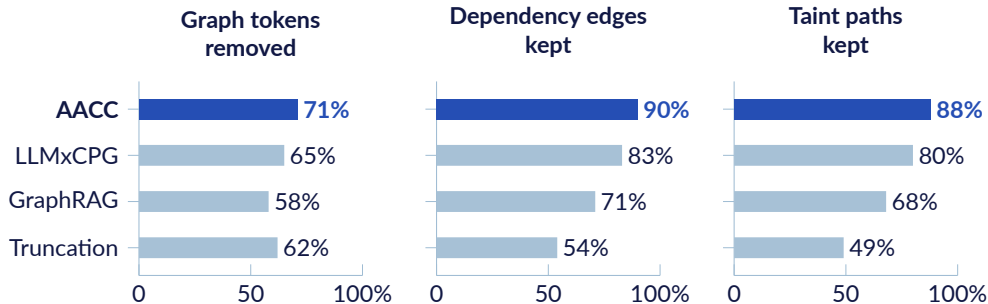
RQ1: Detection Performance



CWE-Bench-Java, 100 repos, $n=200$. [†]Offline adapter. FDR (false discovery rate) = $1 - \text{precision}$.

■ Highest F1 (**0.76** vs. 0.71 for LLMxCPG) and lowest FDR (**0.23** vs. 0.32).
gpt-4o-mini has higher recall (0.79) but an FDR of 0.46.

RQ2: Compression Quality



Measured over all 120 repositories (not only the detection partition). Three quantities, each with its own denominator.

Tokens = graph serialization.

AACC leads on all three measures, yet about 12% of measured taint paths are still lost.

RQ3: Ledger Behavior

Metric	What it counts	CWE-Bench-Java	CVEfixes
Invalidation rate	assumptions contradicted by evidence	18%	36%
Evidence coverage	assumptions supported or contradicted	86%	82%
Hypothesis stability	support retained as evidence grows	89%	90%
Inconclusive rate	unresolved hypotheses	8%	23%

CWE-Bench-Java: 120 repositories. CVEfixes: 100 Java cases. Metrics have different denominators.

■ Evidence covers 82–86% of assumptions. Invalidation reaches 36% on CVEfixes and 18% on CWE-Bench-Java.

RQ4: Component Ablation

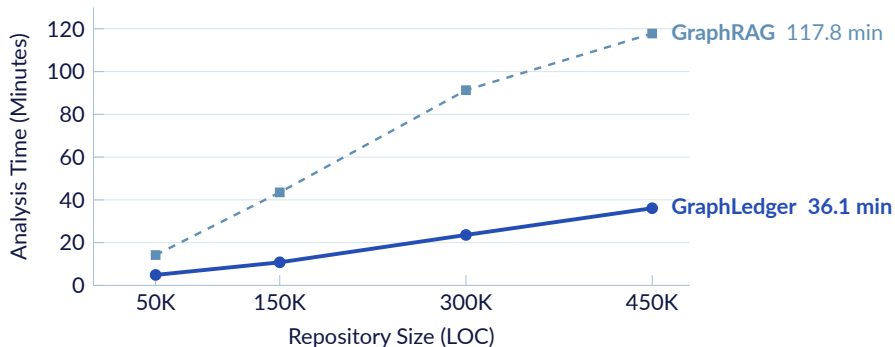
Configuration	Precision	Recall	F1	McNemar p
Neither	0.60	0.72	0.66	0.0081
Compression only	0.67	0.68	0.67	0.0377
Ledger only	0.70	0.73	0.71	0.2070
Full	0.77	0.76	0.76	—

CWE-Bench-Java, 100-repository partition, $n=200$. McNemar p values compare each row with Full.

Neither still uses the full CPG.

- Full obtains the highest reported P/R/F1.
- McNemar is significant vs. Neither and Compression only, not vs. Ledger only.

RQ4: Runtime and Scope



Analysis time per repository at four sizes. **Scope:** paths ≤ 20 steps, ≤ 8 per pair, Java source-sink classes.

Lower reported runtime at all four sizes: 36.1 vs. 117.8 min at 450K LOC.

Conclusions

Observed result, reported 100-repository partition

F1 **0.76** vs. 0.71 for LLMxCPG | precision **0.77**

- Contribution** Evidence-preserving compression and explicit assumption checks.
- Limits** Java evaluation, bounded paths and hypothesis quality.
- Future work** Improve framework models and adaptive search, then evaluate beyond Java.

Thank you · Q&A

lthcuc.sdh242@hcmut.edu.vn

Artifact (code + data)

doi.org/10.5281/zenodo.19253935



Backup Slides

Baseline results	precision, recall, F1, FDR for every method
Baseline integration	inline, rules-only, harness, offline adapters
Evaluation configuration	fixed settings, tuning provenance, data splits
Evidence rules	which evidence validates or undermines an assumption
Formal properties	what is proved, under which condition, what is not
Inconclusive outcomes	how often, why, and how they are scored
CVEfixes ablation	component analysis on the second dataset

[← Back to Q&A](#)

Method	Integration	Prec.	Rec.	F1	FDR
CodeQL	offline adapter	0.75	0.46	0.57	0.25
IRIS	offline adapter	0.58	0.60	0.59	0.42
gpt-4o-mini	controlled harness	0.54	0.79	0.64	0.46
GraphRAG	inline reimpl.	0.56	0.70	0.62	0.44
Vul-RAG	inline reimpl.	0.53	0.71	0.61	0.47
LLMAO	offline adapter	0.64	0.67	0.66	0.36
RepoAudit	offline adapter	0.66	0.73	0.70	0.34
SemTaint	inline reimpl.	0.62	0.49	0.55	0.38
VulAgent	inline reimpl.	0.59	0.70	0.64	0.41
LLMxCPG	offline adapter	0.68	0.75	0.71	0.32
GraphLedger	—	0.77	0.76	0.76	0.23

Bold = best per column (lowest FDR). CWE-Bench-Java, 100-repository partition, $n=200$.

Mode	Methods	Shared with GraphLedger / note
Inline	GraphRAG, Vul-RAG, VulAgent	checkout, repository representation, gpt-4o-mini backend
Inline, rules	SemTaint	checkout and representation, deterministic rules, max. 20 hops, no LLM call
Harness	gpt-4o-mini	full-CPG context, no compression, no ledger, temperature 0
Offline adapter	CodeQL, IRIS, LLMAO, RepoAudit, LLMxCPG	external outputs converted to the same per-case detector schema

Measures performance under one common protocol, not each method's best in its own optimized environment.

Item	Value	Limit
Configuration	<i>balanced</i> : dependency distance $d=10$, similarity threshold $\tau=0.3$, combine=any, path length $\ell=20$, $m=8$ paths per pair	archived and fixed, not reconstructed as a selection outcome
Tuning	separate 20-repository subset	identifiers and selection records not archived
Reporting split	seed 42, repository-disjoint: 100 evaluation / 20 sensitivity	nothing selected on it
All-120 analysis	complementary analysis	not strictly held out

Assumption class	Sanitizer	Flow	None
<i>Absence</i> (“no sanitization”)	undermined	validated	pending
<i>Presence</i> (“sanitizer exists”)	validated	undermined	pending
<i>Flow</i> (“taint reaches sink”)	—	validated	pending
<i>Default</i>	undermined	validated	pending

— : no rule is listed for this pair.

Interpretation

Assumptions are invalidated only by explicit contradicting evidence. With no evidence they stay pending. Sanitizer markers are CWE-specific and listed in the artifact.

Property	Condition	Limit
Monotonic rejection (proved)	ledger update rule: <i>undermined</i> is never reverted	about verdicts, not end-to-end detection soundness
Path correspondence (conditional)	every summary edge is witness-backed	condition not formally verified for the implementation
Taint paths kept, 88%	empirical measurement	does not show summaries are witness-backed
Heuristic parts	summary extraction, CHA call graph, evidence recovery, LLM hypotheses	no formal claim. “Validated on the witness path” is an engineering invariant

How Often

Inconclusive rate **8%** on CWE-Bench-Java, **23%** on CVEfixes.
Hardest to resolve: authorization bypass and SSRF. Bounded search can leave hypotheses unresolved, e.g. paths longer than 20 steps.

How Scored

Reported metrics count **Inconclusive** as negative. If these cases are instead surfaced as deferred findings for review, precision stays **0.77** and F1 moves **0.76 → 0.77**.

■ The convention decides whether unresolved cases count as misses or as deferred reviews. F1 changes by 0.01.

Configuration	Precision	Recall	F1	FDR
Neither	0.61	0.69	0.65	0.39
Compression only	0.67	0.69	0.68	0.33
Ledger only	0.70	0.73	0.71	0.30
Full	0.74	0.71	0.72	0.26

Bold = best in each column. Archived CVEfixes evaluation set: 100 labeled cases from 32 Java projects, similarity threshold $\tau=0.0$.

Interpretation

Full vs. Neither: $p = 0.0162$. The other paired comparisons are not significant.