Automated vs. Human Security Patching Patterns in Pull Requests: Evidence from the AIDev Dataset

10/28/2025

Felix Wang, Brian Do, Jacie Jermier

UNIVERSITY OF WATERLOO

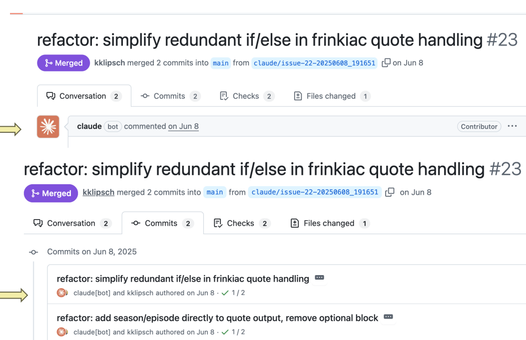
Motivating Question

- Among *successfully* merged pull requests (PRs), what are the patterns of CWE instances detected in codes generated by (1) fully autonomous agents, (2) agent–human co-authored, and (3) fully human-authored?
  - CWE: Common Weakness Enumeration - community-developed dictionary of software and hardware weaknesses that have the potential to lead to security vulnerabilities.
    - Example Label: CWE-79: Improper Neutralization of Input During Web Page Generation

Fully Automated PR

Pull Requests created by agents

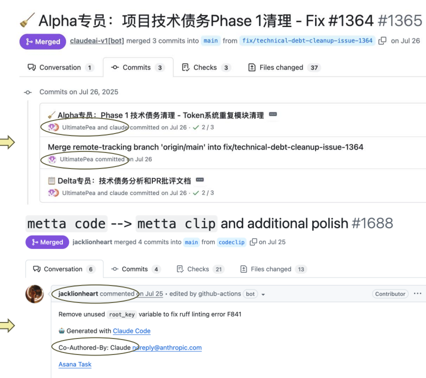
Fully agent-generated code



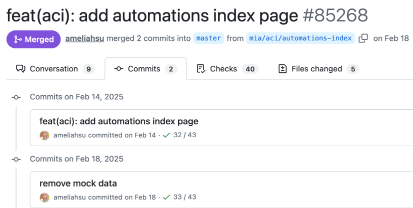
Semi-Automated PR

Pull Requests that humans and agents co-author together.

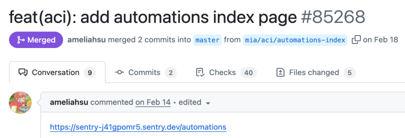
Humans created pull requests that are co-authored by agents.



Fully Human PR



Humans created PRs and human-written code.



Research Questions

- RQ1: What is the proportion of PRs for each PR type that are about security vulnerabilities?
- RQ2: What is the distribution of CWE types among security vulnerability PRs with and without regard to each PR type?
- RQ3: How do different coding agents vary in distributions of CWE types among security vulnerability PRs?

Prior Research

1. Agent-created Pull Requests
2. Security-related Pull Requests
3. Categorizing security patches

Agent Created Pull Requests

- Why are people loving agents?
  - Easy to integrate into GitHub Projects.
  - Monitor vulnerability databases in real-time.
  - Agent-generated configuration files are stable most of the time.
- Why are people hating agents?
  - Excessive notifications.
  - Developers don't know how to communicate with agents.
  - Sometimes take wrong actions.

Security-related Pull Requests

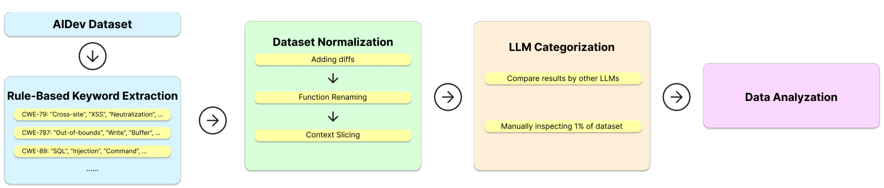
- Polarizing trend:
  - The vast majority are merged within one day after creation.
  - ~35% of security PRs remain open and unattended for long periods.
- Average vulnerability exposure lifetime: 512 days.
- Main reasons why PRs are not merged:
  - PR is superseded by newer PRs.
  - Dependency is already updated or removed, or not updatable.
  - PR has high complexity.

Categorizing Security Patches

- Prior work demonstrated the applicability of incorporating LLM to categorize security patches on a scope of memory-related vulnerabilities.
- GraphSPD -> identifying security patches by binary outputs.
- TREEVUL -> classifying CWEs based on a dataset drawn from the CVE database with CWE annotations.

Methodology

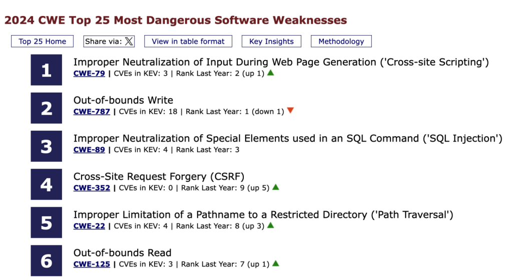
We first need to decide how we will detect which PRs are dealing with security tasks. And then decide how we are going to categorize them.



The pipeline of the dataset security patch identification

Justifications – Keyword Extraction

1. Derive keywords from "2024 CWE Top 25 Most Dangerous Software Weaknesses" -> CWE specific keyword list.
2. Combine with generic patterns: ".*priva.*", ".*vulnerab.*", ".*secur.*". -> Generic keyword list.



Justifications – Dataset Normalization & Data Augmentation

- Fetching diffs information from: https://patch-diff.githubusercontent.com/raw/[user_name]/[repository_name]/pull/[pr_number].patch.

```
From 29dfe3d895ec02aed4c5194a90536b485558139b Mon Sep 17 00:00:00 2001
From: Draco <byoung3310@gmail.com>
Date: Fri, 25 Jul 2025 09:47:21 -0600
Subject: [PATCH 1/8] fix(ci): replace black/isort with ruff commands

CRITICAL: Restore CI functionality by replacing missing tools:
- Replace 'black --check' with 'ruff format --check'
- Replace 'isort --check-only' with 'ruff check --select I'

This aligns CI with local development setup which already uses ruff.
Black and isort are not installed in the project dependencies.
---
 .github/workflows/ci.yml   |   8 +-
 Cleanup_Git_Guide.md       | 151 -------------------
 ERA001_Action_Plan.md      | 300 -------------------------------------
 ERA001_PR_Description.md   | 100 -------------
 ERA001_Phase1_Summary.md   |  73 ---------
 ERA001_Phase2_Plan.md      | 182 ----------------------
 GitHub_Auth_Setup_Guide.md | 136 -----------------
 Ruff Analysis v1.md        | 148 ------------------
 8 files changed, 4 insertions(+), 1094 deletions(-)
 delete mode 100644 Cleanup_Git_Guide.md
 delete mode 100644 ERA001_Action_Plan.md
 delete mode 100644 ERA001_PR_Description.md
 delete mode 100644 ERA001_Phase1_Summary.md
 delete mode 100644 ERA001_Phase2_Plan.md
 delete mode 100644 GitHub_Auth_Setup_Guide.md
 delete mode 100644 Ruff Analysis v1.md

diff --git a/.github/workflows/ci.yml b/.github/workflows/ci.yml
index 9c134a2..a5e4430 100644
--- a/.github/workflows/ci.yml
+++ b/.github/workflows/ci.yml
@@ -39,13 +39,13 @@ jobs:
       run: |
         ruff check gal_friday tests

-    - name: Check formatting with black
+    - name: Check formatting with ruff
       run: |
-        black --check gal_friday tests
+        ruff format --check gal_friday tests
```


Justifications – Dataset Normalization & Data Augmentation

- Rename functions: Avoid project-specific function naming conventions; Add more context for LLM categorization.
- Example: The function name "install()" is ambiguous and lack context.
- Apply LLM to change to "install_codeclip_tool()" for better context.

```python
def install(self) -> None:
    """Install codeclip as an editable uv tool."""
    codeclip_dir = self.repo_root / "metta" / "setup" / "tools" / "codeclip"

    if not codeclip_dir.exists():
        warning(f"Codeclip directory not found at {codeclip_dir}")
        return

    info("Installing codeclip tool...")

    # Install as editable package using uv
    try:
        # Use --force to update if already installed
        self.run_command(["uv", "tool", "install", "--force", "-e", str(codeclip_dir)])
        success("Codeclip tool installed successfully!")
        info("You can now use 'metta clip' or 'codeclip' commands")
    except subprocess.CalledProcessError as e:
        warning(f"Failed to install codeclip: {e}")
        warning("You can manually install it with:")
        warning(f"  cd {codeclip_dir} && uv tool install --force -e .")
```


Justifications – Dataset Normalization & Data Augmentation

- Context slicing: Slice ±3 lines of the changed line for enough contexts.
- Original:

```
+        codeclip_dir = self.repo_root / "metta" / "setup" / "tools" / "codeclip"
```

- Content slicing for better context:

```python

    def install(self) -> None:
        """Install codeclip as an editable uv tool."""
        codeclip_dir = self.repo_root / "metta" / "setup" / "tools" / "codeclip"

        if not codeclip_dir.exists():
            warning(f"Codeclip directory not found at {codeclip_dir}")
```


Justification – LLM Categorization

- Model Selection Criteria:
  - Reasoning capability
  - Open-source accessibility
- 2 best-performing open-sourced reasoning models on GPQA benchmark:
  - GPT-OSS-120B (OpenAI)
  - Nemotron-Ultra-253B (Nvidia)
- Cross-validate and manually inspect 1% of the dataset.

Prompt we will use:

```
You are a professional software developer and system
     security expert.
Decide if the following description of a pull request
     is likely a security patch, then decide its
     related type of Common Weakness Enumeration id.
No explanation is needed, result should be in form of
     {is_security_patch: (boolean), cwes: [{cwe_id,
     cwe_title}]}
Description:
{text}
Code Diff and Context:
{diff_context}
Results:
```

Justification – Why don't we use a recent related model?

- We are aware that a pipeline of GraphSPD + TREEVUL (by prior work) can also identify security patches and map them to CWEs.
- Reason why we don't do it this way:
  1. Labelling a training/testing dataset out of AIDev is a non-trivial task.
  2. Their training set is domain-shifted. AI's coding style and (especially) description style are drastically different.
  3. Their accuracy individually are not high enough -> Errors will propagate.
- We will use it as a baseline for comparison if we have enough time.

GraphSPD: Identify security patch

TREEVUL: map to CWE categories

Methodology Reasoning

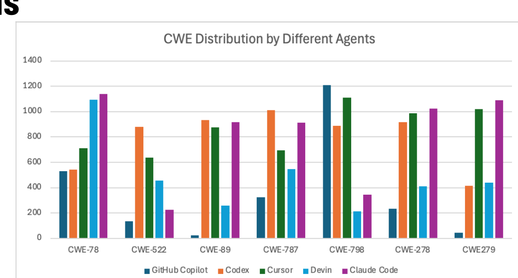
LLM Based Approach

- Multi-Language & Multi-Repo
- Semantic Understanding – meaning of text versus keywords
- Understanding AI Generated Code – Agents write fixes that humans phrase differently
- Computer heavy – batch processing / GPU
- May over/misclassify without fine tuning.
- Validation becomes difficult without a benchmark.

Rule Based Approach

- Fast – can run very quickly
- Fully transparent – we create the keywords and the logic
- Low compute cost
- Potential for high false negatives
- Language specific / repo specific – key words failing over different languages as well as different coding languages
- No context awareness

Hypothesis



Hypothesis #1

H1: Agents have higher CWE density/distribution. If agents show more CWEs or a greater distribution of CWEs then this could reflect differences in actual code quality.

Hypothesis #2

H2: Autonomous vs. co-authored differ. Differences due to potential human oversight.

Hypothesis #3

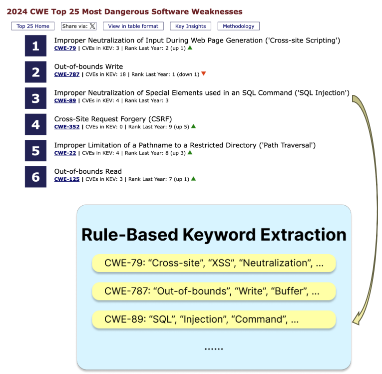
H3: Different AI agents (Copilot, Claude, Codex, etc.) show distinct fingerprints in security PRs as well as consistent tool misses.

5-Week Milestone Plan: From 1M PRs to CWE Insights

- **Week 1: Keyword Extraction**
  - Extract keywords from 2024 CWE Top 25
  - Rule-based matching on PRs
- **Week 2: Dataset Normalization**
  - Fetch diffs via GitHub API
- **Week 3: LLM Categorization**
  - Run **GPT-OSS-120B** & **Nemotron-Ultra-253B** in parallel
  - Prep analysis scripts
- **Week 4: Analysis & Data Visualization**
  - Inspect 1% of data → validate LLM outputs
  - Run final analysis (RQ1–RQ3)
- **Week 5: Conclusions & Paper Finalizations**
  - Complete findings and analysis
  - Update methodology and limitations where necessary

Internal Validity - Keyword Selection Bias

- Keywords derived from the “2024 CWE Top 25 Most Dangerous Software Weaknesses” combined with generic vulnerability terms.
- This may:
  - Over-represent certain CWEs
  - Miss subtle cases.



Internal Validity - Context Slicing Bias

- We extract ±3 lines around the modified line to preserve contextual richness for LLM’s categorization.
- A fixed slicing window cannot fully capture the data flow and control flow and give enough context to the LLM, possibly leading to some misclassifications.

Internal Validity - Manual Inspection Bias

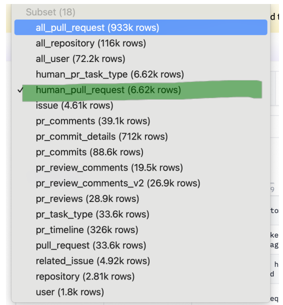
- We manually verify 1% of the dataset.
- It could be affected by the annotator's expertise and subjective judgment.

Internal Validity - LLM Categorization Bias

- We are using LLMs to categorize PRs into CWEs.
- LLMs could make systematic underlying misclassifications, so we try to mitigate this by cross-validating between two LLMs and manual inspection, but such risks might still exist.

External Validity – Dataset bias

- AIDev - Significant class imbalance.
- 87.3% are attributed to OpenAI Codex.
- Fully human PRs only ~6.6k.



External Validity – Temporal Limitations

- The security patching practices evolve rapidly. Therefore, this dataset may not fully reflect future developments in agent-assisted coding workflow.
- Our findings are limited to data from December 2024 to July 2025.

Discussion

Thank you for listening!

UNIVERSITY OF WATERLOO