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

12/2/2025

Felix Wang, Brian Do, Jacie Jermier

UNIVERSITY OF WATERLOO

Motivations

- The rapid emergence of autonomous coding agents
- Productivity gains vs. Trustworthiness
- Security patches are difficult to produce and review due to the specialized domain knowledge

=> How do security patches generated by AI coding agents differ from those authored by human developers?

Prior Research

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

Prior Research

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.

Prior Research

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 pull requests (PRs) are not merged:
  - PR is superseded by newer PRs.
  - Dependency is already updated or removed, or not updatable.
  - PR has high complexity.

Prior Research

Categorizing Security Patches

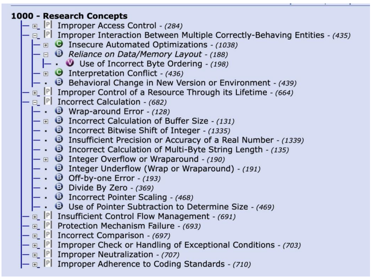
- Prior work demonstrated the applicability of incorporating LLM to categorize security patches on a scope of memory-related vulnerabilities.
- GraphSPD -> identifying security patches by binary outputs.
- TREEVUL -> classifying CWEs based on a dataset drawn from the CVE database with CWE annotations.

Research Gap

- Understanding of the characteristic of security patches at the level of vulnerability types:
  - Whether agent-generated patches address different categories of weaknesses than human-authored patches.
  - Whether certain vulnerability classes are disproportionately represented in agent-generated pull requests.

Terminologies

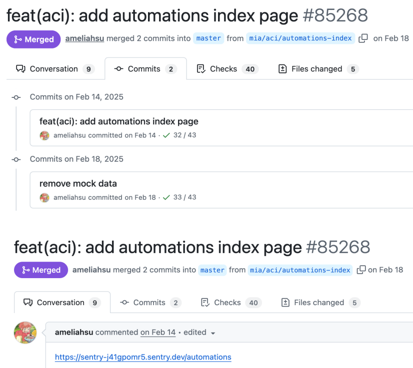
- CWE: Common Weakness Enumeration - community-developed dictionary of software and hardware weaknesses that have the potential to lead to security vulnerabilities.
- CWE abstraction levels:
  - Pillar-level
  - Class-level
  - Base-level
  - Variant-level
  - Chain-level



Terminologies

Human-authored PRs

Humans created PRs and human-written code.



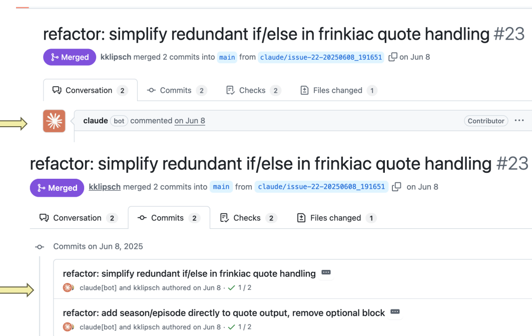
UNIVERSITY OF WATERLOO

Terminologies

Agent-generated PRs

Pull Requests created by agents

Fully agent-generated code



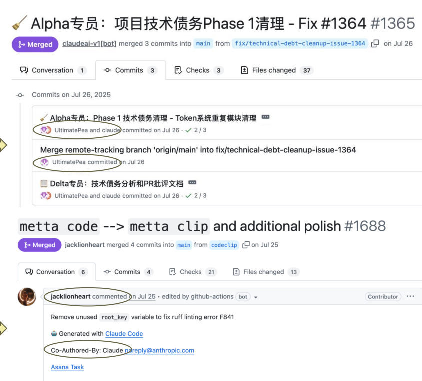
UNIVERSITY OF WATERLOO

Terminologies

Agent-generated PRs

Pull Requests that humans and agents co-author together.

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



UNIVERSITY OF WATERLOO

Research Objectives

UNIVERSITY OF WATERLOO

Research Question #1

- How do **human-authored** and **agent-generated security-related PRs** differ in their **size** and **proportion**?

- Current Research:
  - Improving LLM-based coding agents' trustworthiness itself.
  - Focusing on code-snippets, not real-world large-scale Pull Requests.
  - Only on subdomains of LLM coding agents' vulnerabilities (Dependencies, Memory-related, etc.).
  - A **holistic** comparison between human-authored and agent-generated security patches remains a research gap.

Research Question #2

- What are the **most common CWE types** that occur in human-authored and agent-generated PRs?

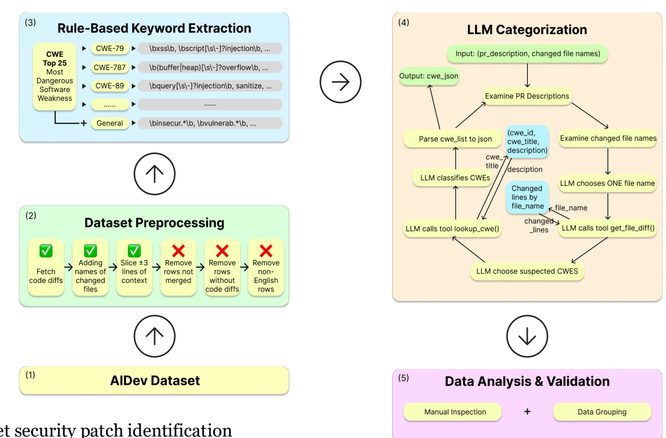
- Current Research:
  - Different coding agents tend to address different categories of security vulnerabilities.
  - We can determine whether agents disproportionately introduce or modify particular categories of vulnerabilities.

Research Question #3

- How do human-authored and agent-generated security-related PRs differ in the **distribution of abstract CWE type grouping** they address?

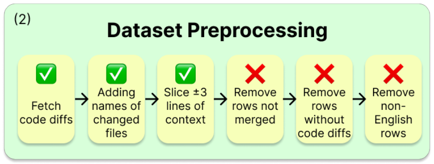
- Current Research:
  - Specific CWE labels are different from broader security behavioral groupings.
  - Group them into different security concern categories.
  - Discover security reasoning patterns.

Methodology



The pipeline of the dataset security patch identification

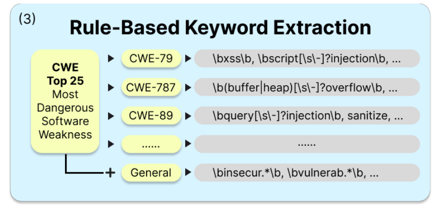
Step 1 – Dataset Preprocessing

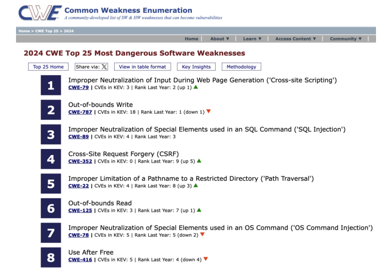


- First fetch from AIDev.pr_commit_details table.
- GitHub API Endpoint: https://patch-diff.githubusercontent.com/raw/[org_name]/[repo_name]/pull/[pr_number].patch
- 20.76% of human-authored PRs were excluded.
  8.69% of agent-generated PRs were excluded

files_changed	code_diff
["a/dev-packages/node-integration-tests/package.json b/dev-packages/node-integration-tests/package.json"]	[""" === CHUNK a/dev-packages/node-integration-tests/package.json b/dev-packages/node-integration-tests/package.json === "dependencies": { "@aws-sdk/client-s3": "^3.552.0", "@hapi/hapi": "^21.3.10", - "@nestjs/common":"10.4.6", + "@nestjs/common": "11.0.16", "@nestjs/core": "10.4.6", "@nestjs/platform-express": "10.4.6", "@sentry/aws-serverless": "9.12.0", """]

Step 2 – Rule-Based Keyword Extraction



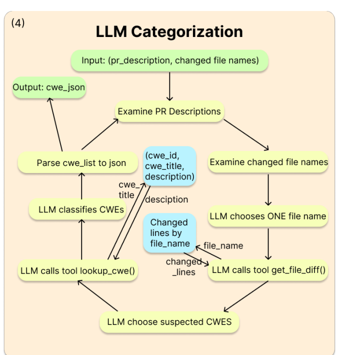


➕

General Vulnerability-related Keywords

- Retrieves a broad set of potentially security-related PRs with a high recall.
- 11.76% of human-authored PRs retrieved
  10.42% of agent-generated PRs were retrieved

Step 3 – LLM Categorization



- LLM Selection: GPT-OSS-120B
  Contextual reasoning ability and classification ability into classes with subtle differences
- Input token limit: 30,000
  Total token limit: 40,000
  Completion token limit: 1000
- 0.85% of human PRs exceed token limit
  0.48% of agent PRs exceed token limit
- Return template:

```
{   pr_id: string,
    is_security_patch: bool,
    cwe_lst: [
        {    cwe_id: string,
             cwe_title: string }]
}
```


Step 4 – Manual Inspection

- Manually security patch classification result and the CWE classification result.
- Inspected 7 (Human PRs) + 99 (Agent PRs).
- For the security patch classification result, we ask 2 independent voters: Is this a security patch?
- For the CWE classification result, we ask 2 independent voters: Is the LLM's classification accurate?

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Results

Classification Reliability

- Fleiss's kappa (
  is_security_patch_voter_1,
  is_security_patch_voter_2,
  is_security_patch_llm_pred
)

- Cohen's kappa (
  is_llm_cwe_lst_correct_voter_1,
  is_llm_cwe_lst_correct_voter_2
)

Dataset	N	κ	Interp. (n)
(A) Security Classification			
Human Pull Requests	7	0.6912	Subst. (3)
Agentic Pull Requests	99	0.7472	Subst. (3)
(B) CWE Classification			
Human Pull Requests	7	0.5882	Moderate (2)
Agentic Pull Requests	99	0.2137	Fair (2)

Classification Reliability

- **Substantial Agreement** for Security PR Classification:
  - LLM's security/non-security labels largely align with human judgments.
- Only **Moderate** and **Fair** Agreement for CWE Classification:
  - We are only moderately or fairly sure that the human PRs and Agentic PRs are correctly classified.

Dataset	N	κ	Interp. (n)
(A) Security Classification			
Human Pull Requests	7	0.6912	Subst. (3)
Agentic Pull Requests	99	0.7472	Subst. (3)
(B) CWE Classification			
Human Pull Requests	7	0.5882	Moderate (2)
Agentic Pull Requests	99	0.2137	Fair (2)

Fleiss Kappa	Interpretation
<0.00	Poor agreement
0.00 to 0.20	Slight agreement
0.21 to 0.40	Fair agreement
0.41 to 0.60	Moderate agreement
0.61 to 0.80	Substantial agreement
0.81 to 1.00	Almost perfect

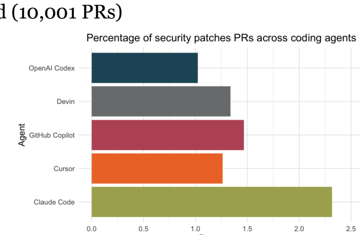
Cohen's Kappa	Interpretation
0	No agreement
0.10 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 0.99	Near perfect agreement
1	Perfect agreement

Source 1: https://www.statology.org/cohens-kappa-statistic/
Source 2: https://www.researchgate.net/figure/Fleiss-Kappa-and-Inter-rater-agreement-interpretation-24_tbl3_281652142

Results

RQ1: Proportion & size of Security-related PRs

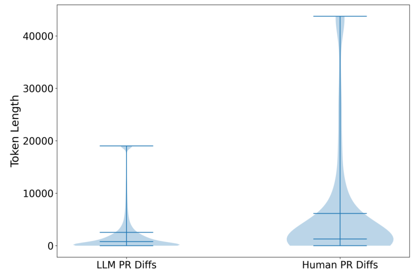
- Agent-generated security-related PRs are roughly **one-third** of those authored by human:
  - 3.26% of **human-authored** PRs are security-related (169 PRs)
  - 1.18% of **agent-generated** PRs are security-related (10,001 PRs)
- Across individual agents,
  - 2.32% of PRs by **Claude Code** are security-related
  - 1–1.5% of PRs by other agents are security-related



Results

RQ1: Proportion & size of Security-related PRs

- Human-authored security-related PRs contain **more code changes** and **more descriptive context**
  - ~3261 code tokens and ~1217 description tokens per human-authored PRs
  - ~2480 code tokens and ~182 description tokens per agent-generated PRs



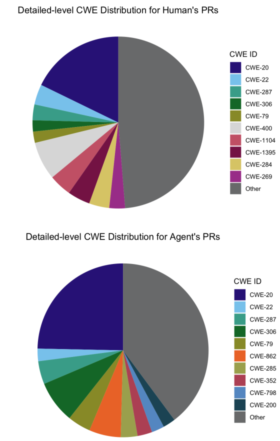
Distribution of token length of code diffs (trimmed at the 95th percentile)

UNIVERSITY OF WATERLOO

Results

RQ2: Most commonly occurred CWE types

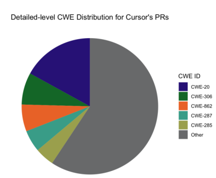
- **CWE-20 (Improper Input Validation)** appeared most frequently in both human-authored and agent-generated PRs.
- Human-authored PRs emphasize **resource management** (CWE-400, 1104, 1395)
- Agent-generated PRs target **authentication and authorization** (CWE-306, 862, 287)

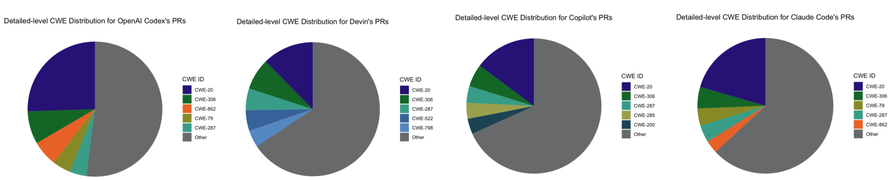


Results

RQ2: Most commonly occurred CWE types

- **CWE-20 and CWE-306 (Missing Authentication for Critical Function)** appear most frequently for all agents.



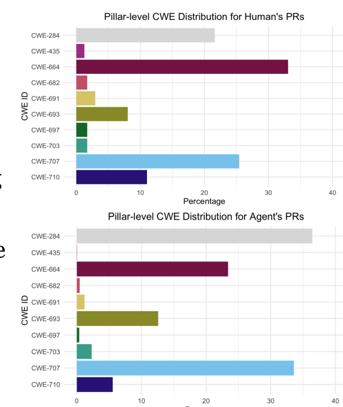


UNIVERSITY OF WATERLOO

Results

RQ3: Distribution of abstract CWE type groupings

- Human-authored PRs focus more on **resource management throughout the resource lifetime (CWE-664)** and **adherence to coding standards (CWE-710)**.
- Agent-generated PRs disproportionately emphasize **access control (CWE-284)** and **input neutralization (CWE-707)**.



UNIVERSITY OF WATERLOO

Discussion

Discussion: What did we actually learn?

- Agents touch CWE Top-25 code ~3× less often than humans
- When they do touch it they have nearly identical CWE distribution (same top 4: CWE-20, -79, -306, -22)
- *Current real-world AI coding agents are not flooding repositories with insecure changes — they are actually avoiding or being steered away from the most dangerous weakness classes that humans manage.*

https://en.wikipedia.org/wiki/Agent_Smith

Discussion: How do agents handle CWE-20?

- 81 % of agent fixes use structured, template-style validation (allow-lists, schema validation, required fields, regex/format checks) i.e. "textbook" patterns are often easier to review and maintain.
- Humans often contain more ad-hoc/localized fixes (custom logic that fits only that exact spot).
- *Agents favor systematic, reusable patterns that are easier to audit and scale*

Discussion: Why so many human false positives?

- 47.6 % of human "CWE-20" PRs were actually dependency / lock-file bumps (hashes, version strings trigger keywords + LLM).
- Only 4 % of agent "CWE-20" PRs had the same problem.
- The measured human security-touch rate (3.26 %) is inflated.
  - *The real ratio is probably closer to 3–4 : 1*

Discussion: What do we do with this information?

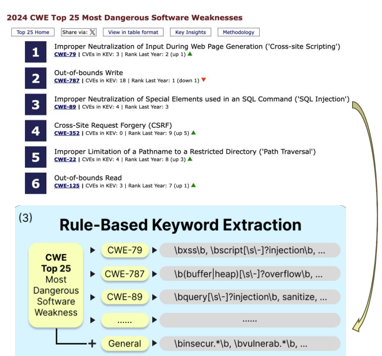
- Routine tasks (features, refactors, dependency updates) can safely be delegated to agents.
- High-risk CWE changes (CWE-20, -79, -306, -22) should be kept as human in the loop or extra review.
- There doesn't seem to be a need to treat agent & human PRs differently in security review (no panic button needs to be pressed).
- Companies can adopt AI agent integration for non-critical code without the fear or paranoia of an increase in high-severity security risk.
- Use our open-source pipeline to automatically flag Top-25 touches.
- *Current real-world agents are more conservative than humans on the most dangerous vulnerabilities.*

Future Work

1. Extend to other languages (C/C++, Rust, Go, Java) where memory-safety issues dominate.
2. Temporal study – track whether agents become more (or less) security-active as models mature. Do vulnerabilities become more evident or remain mostly the same?
3. Distinguish fixes vs. introductions – pair AIDev with vulnerability databases (i.e., CVE-linked commits, GitHub Advisories, NVD, OSS-Fuzz).
4. Cross-model validation – combine heavy reasoning LLM (GPT-OSS-120B) with code-specialized models (Qwen-Coder, DeepSeek-Coder).

Internal Validity - Keyword Selection Bias

- Keywords derived from the "2024 CWE Top 25 Most Dangerous Software Weakne sses" combined with generic vulnerability terms.
- This may:
  - Over-represent certain CWEs.
  - Miss subtle case.
  - Sequential design - propagating errors through rest of pipeline.
- Manual Validation showed:
  - Security-related: κ = 0.69–0.75
    - Despite keyword selection bias binary classification is reliable
    - Reported proportions (3.26 % human vs. 1.18 % agent) are trustworthy.



Internal Validity - Context Slicing Bias

- We extract ±3 lines around the modif ied line to preserve contextual richnes s 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.
  - Cannot capture distant sanitizers, auth checks, macro definitions etc...
- Many false positives in human PRs were dependency / package-lock bumps that contained CWE-related keywords. Showing context slicing wasn't the primary driver of misclassifications.

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

    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}")
```

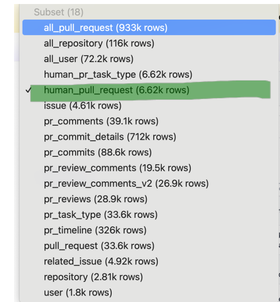
```
6  typespec/package-lock.json
@@ -62,9 +62,9 @@
        }
      },
      "node_modules/@babel/runtime": {
-       "version": "7.25.6",
-       "resolved": "https://registry.npmjs.org/@babel/runtime/-/runtime-7.25.6.tgz",
-       "integrity": "sha512-VBj9MYyDb9tuLq7yzqjgzt6Q+IBQLrGZfdjOekyEirZPHxXWoTSGUTMrpsfi58Up73d13NfYLv8HT9vmznjzhQ==",
+       "version": "7.26.10",
+       "resolved": "https://registry.npmjs.org/@babel/runtime/-/runtime-7.26.10.tgz",
+       "integrity": "sha512-2WJMeRQPHKSPemqk/awGrAiuFfzBmOIPXKizAsVhWH9YJqLZ0H+HS4c8loHGgW6utJ3E/ejXQUsiGaQy2NZ9Fw==",
        "license": "MIT",
        "dependencies": {
          "regenerator-runtime": "^0.14.0"
```

Internal Validity - LLM Categorization Bias

- We are using LLMs to categorize PRs into CWEs.
- Even with identical prompt + code, an LLM can sometimes output different CWE labels on separate runs.
- Causes: internal randomness, silent model updates, tokenization quirks, JSON formatting differences.
- Manual validation showed:
  - Exact CWE label: κ = 0.21–0.59 (worse on agents)
    - Fine-grained CWE distribution has more noise but does not affect our core finding(agents touch high-risk code far less often).

External Validity - Dataset bias

- AIDev - Significant class imbalance.
- Fully human PRs only ~6.6k.
  - After filtering: 617 rows (11.76%)
- Fully Agent PRs only ~932k.
  - After filtering: 91,694 rows (10.42%)



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.

Questions

Thank you for listening!

UNIVERSITY OF
WATERLOO