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

Felix Wang*
felix.wang@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Brian Do*
brian.do@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Jacie Jermier*
jacie.jermier@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Abstract

AI coding agents have rapidly been introduced into real-world software development pipelines, raising concerns about their impact on code security. This paper presents an empirical study analyzing the number and characteristics of human-authored and agent-generated security-related pull requests (PRs) in the AIDev dataset. We developed an LLM-based pipeline to classify PRs into detailed-level Common Weakness Enumeration (CWE) types, enabling us to uncover clear patterns in the frequency and types of security-related PRs made by both humans and agents. Our results show that agent-generated security-related PRs occur less frequently than those authored by humans. However, the distribution of CWE types among agent-generated PRs closely mirrored that of human-authored PRs, with improper input validation (CWE-20) and missing authentication (CWE-306) being the dominant weaknesses in both groups. Additionally, different coding agents exhibit similar distributions of CWE types in their security-related PRs. To support ongoing monitoring of agent evolution, we have made our LLM-based classification tool openly available.

ACM Reference Format:
Felix Wang, Brian Do, and Jacie Jermier. 2025. Automated vs. Human Security Patching Patterns in Pull Requests: Evidence from the AIDev Dataset. In *Proceedings of MSR '26: Proceedings of the 23rd International Conference on Mining Software Repositories (MSR 2026).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The rapid emergence of autonomous coding agents - such as Codex, Cursor, Copilot, and Claude Code - marks a transition toward Software 3.0, an era which Large Language Models (LLMs) participate directly in software development by proposing changes, generating patches, and authoring pull requests (PRs) [7, 19]. These systems promise substantial productivity gains, yet their increasing autonomy raises important concerns about trustworthiness [8, 14, 16], particularly in security-related contexts. Security patches are historically difficult to produce and review due to the specialized domain knowledge they require. As LLM-generated code becomes more pervasive, a fundamental question arises: *How do security patches produced by AI coding agents differ from those authored by human developers?*

Although the body of research evaluating agent-generated PRs is rapidly expanding, prior work has primarily examined review behaviors, acceptance rates, and developer perceptions. This leaves a critical gap in understanding: the characteristic of security patches at the level of vulnerability types. In particular, it remains unclear whether agent-generated patches address different categories of weaknesses than human-authored patches and whether certain vulnerability classes are disproportionately represented in agent-generated PRs.

To address this gap, our study conducts the first systematic, vulnerability-level comparison of security-related PRs authored exclusively by human contributors (Human's PRs) and those authored by coding agents with or without human involvement (Agent's PRs). Using the AIDev dataset [11], we identify security-related patches through keyword extraction and LLM-based Common Weakness Enumeration (CWE) classification, mapping each PR to its corresponding CWE type within the broader hierarchy [5]. Through this analysis, we aim to characterize the security strengths, limitations, and behavioral patterns of AI coding agents in real-world software repositories, ultimately contributing empirical insights that support safer and more transparent integration of autonomous coding systems into modern development workflows.

2 Related Works

Prior research on AI-assisted software development has highlighted both the promise and limitations of autonomous coding agents. Li et al. [11] show that LLM-based agents now function as genuine collaborators, capable of rapid development, thorough documentation, and competitive performance on feature and bug-fix tasks, although their PR acceptance rates remain significantly lower than those of human developers. Other empirical studies echo these observations, reporting substantial differences between human-authored and agent-generated PRs - particularly in review time and merge success [19, 22].

Security-focused investigations present a similarly mixed picture. While some work suggests that agent-generated security-related PRs can be merged quickly - often within a single day - large proportions of such PRs remain open or unattended for extended periods [1], contributing to vulnerability lifetimes exceeding 500 days in some ecosystems [15]. Researchers have attributed these delays to factors such as project complexity, a lack of transparency around agent-generated patches, and maintainers' reluctance to trust automated changes [1, 19–21]. Nevertheless, automated agents retain notable advantages: they can monitor vulnerability databases

*All three authors contributed equally to this research.

MSR 2026, Rio de Janeiro, Brazil

ACM ISBN 978-1-4503-XXXX-X/2025/06
https://doi.org/XXXXXXX.XXXXXXX

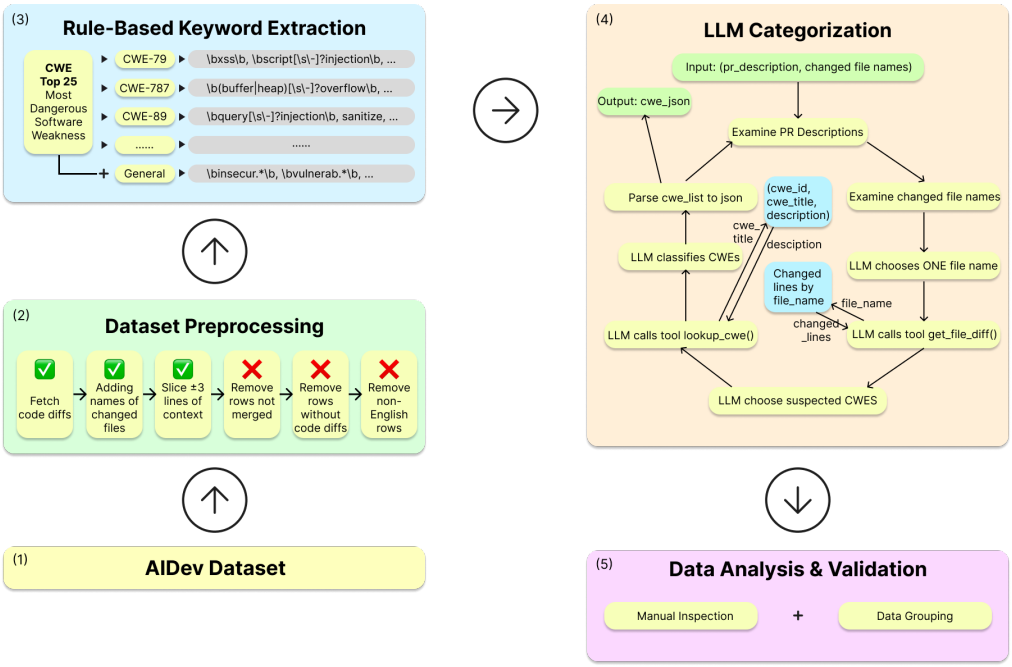

Figure 1: The pipeline for identifying security-related PRs in the dataset.

continuously and proactively propose security fixes, reducing the burden on human oversight [15].

A number of tools and datasets support the automated identification and classification of security vulnerabilities. GraphSPD [18] detects security patches within PRs using structural properties of code changes, while TREEVUL [13] provides CWE-labeled human-authored commits linked to CVEs [4]. More recently, Li et al. [12] demonstrated that LLMs can assist in categorizing security patches, particularly in domains involving memory-related vulnerabilities.

However, existing work has not examined agent-generated security-related PRs at scale nor compared them directly against human-authored security-related PRs at the CWE category level. Our study extends this literature by conducting the first systematic, vulnerability-level analysis of security-related PRs authored by human contributors and PRs partially or fully authored by coding agents, providing empirical evidence about how coding agents behave in security patching tasks.

3 Study Design

This section introduces the research questions we aim to answer (Section 3.1), provides an overview of the dataset and task (Section 3.2), and describe the process of identifying security-related PRs and mapping to CWE types (Section 3.3).

3.1 Research Questions

RQ1: How do human-authored and agent-generated security-related PRs differ in their size and proportion?

Motivation:

Prior research has focused on improving the trustworthiness of LLM-based coding agents [2, 8, 17]. However, these studies predominantly focused on small, representative code snippets, rather than large, real-world PRs. Existing analyses tend to limit their research scope to a subdomain of weaknesses - such as memory-related vulnerabilities [12] - leaving broader categories of software weaknesses underexplored. To date, a holistic and systematic comparison between human-authored and agent-generated security patches remains a research gap.

RQ2: What are the most common CWE types that occur in human-authored and agent-generated PRs?

Motivation: Human developers and different coding agents tend to address different categories of security vulnerabilities. By classifying human-authored and agent-generated PRs across the full CWE taxonomy, we can determine whether agents disproportionately introduce or modify particular categories of vulnerabilities. Understanding these differences helps determine where agents can be safely deployed and where human oversight remains essential.

RQ3: How do human-authored and agent-generated security-related PRs differ in the distribution of abstract CWE type grouping they address?

Motivation: While specific CWEs provide detailed vulnerability labels, they correspond to broader behavioral grouping in the first layer of CWE-1000 (Research Concepts) [6]. These CWE type grouping (e.g. “Improper Access Control”, “Incorrect Comparison”) offer insight into the abstract security concerns. By grouping CWE leaf nodes, we can assess whether human-authored and agent-generated PRs exhibit similar or different security reasoning patterns.

3.2 Dataset & Task Overview

AIDev is a dataset introduced by Li et al. [11] that capture the activities of autonomous coding agents within real-world open-source software repositories. The dataset consists of 932,791 PRs mined from GitHub that are associated with five widely recognized

autonomous coding agents - OpenAI Codex, Devin, GitHub Copilot, Cursor, and Claude Code. It also includes 6,618 PRs created by humans for comparison.

The goal of this study is to identify the security-related PRs among all PRs and map them into specific types of CWE types. The AIDev dataset provides rich contextual information for this, including PR titles, descriptions, timestamps, commit messages, and changed lines. However, the dataset also presents several limitations. First, it is multilingual, containing PRs in Chinese, Japanese, Korean, and other languages, making it unsuitable for traditional rule-based keyword extraction. Second, many PRs contain a empty or missing title or descriptions, making it difficult to determine intent reliably and limiting the effectiveness of heuristics such as keyword matching. Finally, some PRs lack information about changed lines, preventing us from fully contextualizing patches during the LLM-assisted security inference process. Due to these limitations, we performed preprocessing steps before applying our LLM-based classification pipeline.

3.3 Data Processing Pipeline

3.3.1 Dataset Preprocessing.

We began preprocessing by retrieving code differences for each PR using the "pr_commit_details" table in the AIDev dataset. Because this table does not reliably include all commits or changed lines, we implemented a fallback mechanism: if the dataset did not contain the necessary diff information, we fetched the patch directly from the GitHub API (https://patch-diff.githubusercontent.com/raw/[org_name]/[repo_name]/pull/[pr_number].patch). Prs for which diffs were unavailable from both sources were excluded from our analysis. During preprocessing, we also recorded the names of all modified files and extracted ±3 lines of surrounding context for every changed code chunk to provide the LLM with sufficient semantic information for the classification.

We further restricted our scope to merged English-language PRs with valid title and description. Based on these criteria, 20.76% of human-authored PRs and 8.69% of agent-generated PRs were excluded during preprocessing.

3.3.2 Rule-Based Keyword Extraction.

We next applied a rule-based extraction step by using keyword-matching heuristics. Our keyword list consists of carefully designed regular expressions targeting vocabulary likely to appear in security-related titles or descriptions. The list combines two categories of signals. The first category contains terms associated with specific CWE types, derived from the "CWE Top 25 Most Dangerous Software Weaknesses" [3]. This includes representative lexical variants - such as "`\bsql[\s-]injection\b`" or "`\bxss\b`" - along with semantically related security terminology frequently found in security-related PRs. The second category is composed of general security-related keywords intended to capture broader vulnerability-mitigating actions. Examples include "`\binsecure\b`", "`\bvulnerab.\b`", and "`\binvalid.\b`".

This hybrid keyword approach retrieves a broad set of potentially security-related PRs with a high recall. In total, 11.76% of human-authored PRs and 10.42% of agent-generated PRs were retrieved through keyword matching. However, keyword heuristics alone are insufficient, as many generic terms frequently appear in non-security contexts. Thus, we treat this step as a recall-oriented initial filter rather than a standalone classifier.

3.3.3 LLM Categorization.

To achieve both high recall and high accuracy, we incorporated a structured prompting framework and tool-assisted reasoning to guide the LLM toward reliable vulnerability classifications. Model selection was based on two criteria: strong reasoning capabilities and open-source availability.

Security classification requires contextual reasoning about control flow, dependencies, and subtle security implications—not merely pattern recognition. Open-source LLMs facilitate transparency, reproducibility, and fine-grained behavior analysis. Guided by these considerations, we selected *GPT-OSS-120B*, which demonstrates strong performance on complex classification tasks. For each query, we set limits of 30,000 input tokens, 40,000 total tokens, and 1,000 completion tokens. Approximately 0.85% of human PRs and 0.48% of agent PRs exceeded these limits and were skipped.

We designed an autonomous loop in which the LLM iteratively classifies each PR into a JSON object with the following schema:

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

The categorization prompts are provided in Appendix A. The LLM first inspects the PR description and the list of modified files, then selects the file most likely to contain decisive security-relevant evidence. Our tool returns all changed lines in that file along with ±3 lines of surrounding context.

During preliminary experiments, we observed that the LLM occasionally struggled to distinguish between semantically similar CWE categories - particularly when titles were ambiguous or conceptual boundaries overlapped. To mitigate this, we introduced a lookup_cwe() tool. Once the LLM identifies a PR as security-related and proposes candidate CWE titles, our tool returns the full definitions and descriptions for each CWE, enabling more precise categorization.

After the LLM produces a classification, we parse the output into JSON. If the output is invalid or the tools were improperly used, the LLM is prompted to make another attempt (Appendix A.2).

3.3.4 Data Validation.

To evaluate the reliability of the LLM-generated labels, we manually inspected 106 human and agent-generated PRs. Depending on the number of annotators involved, we measured inter-rater agreement using Fleiss's and Cohen's κ coefficient:

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

where p_o denotes the observed agreement among annotators and p_e represents the expected agreement by chance.

4 Results

This section presents the empirical findings from our analysis of human-authored and agent-generated PRs in the AIDev dataset, addressing RQ1–RQ3. We report extraction statistics, the proportions of security-related PRs, and distributions of CWE types at both the pillar and detailed levels.

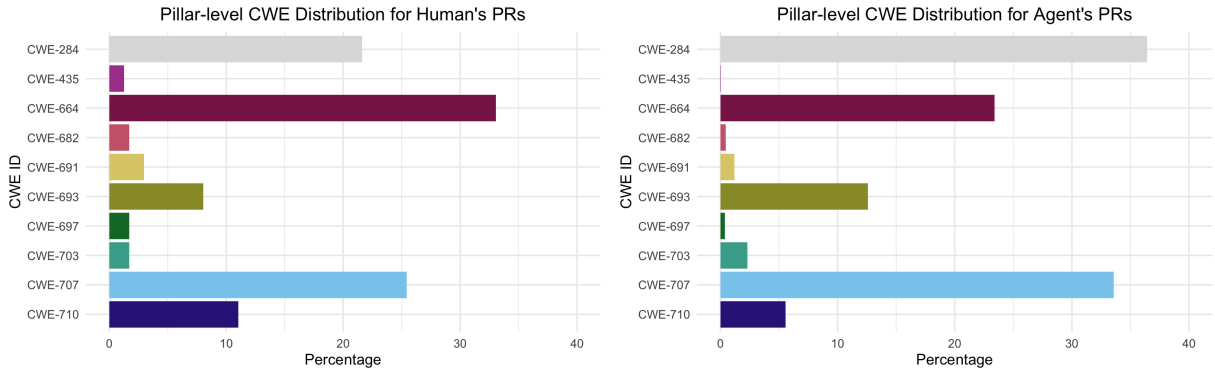

Figure 2: Distribution of CWE groupings in Human and Agent PRs.

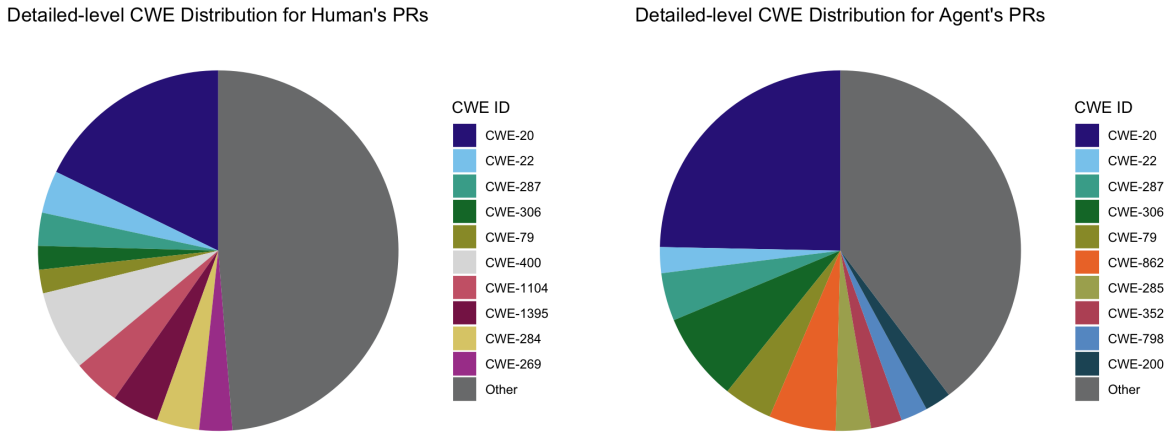

Figure 3: Detailed-level CWEs distribution in Human and Agent PRs.

4.1 Classification Reliability

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)

Table 1: Inter-rater agreement for both tasks. *N* denotes the number of records inspected; *n* denotes the number of voters.

To evaluate the reliability of our LLM-classified security labels and subsequent CWE mappings, two independent human annotators manually inspected a subset of PRs: 7 human-authored and 99 agent-generated. Agreement was measured using Cohen's κ and Fleiss's κ coefficients, following standard practice for assessing categorical labeling consistency in empirical software engineering. Table 1 summarizes the observed agreement levels.

For the binary classification of whether a PR is security-related, Fleiss's κ was calculated among the two human annotators and the LLM. We observed a *substantial level of agreement* for both human and agentic subsets, indicating that the LLM's security/non-security labels largely align with human judgments. This suggests that the classification results are stable and reliable for downstream CWE mapping.

For the more fine-grained CWE classification, annotators independently assessed whether each PR was correctly linked to the set of CWEs predicted by the LLM. This task requires deeper semantic evaluation of code changes and vulnerability types, making it inherently more challenging. Annotators reached a *moderate level of agreement* for human-authored PRs, which decreased to a *fair level* for agent-generated PRs. Despite this, given the complexity and ambiguity of vulnerability attribution, these levels were deemed acceptable for subsequent analysis.

4.2 Results for Research Questions

RQ1: Proportion & Size of Security-Related Pull Requests

The AIDev dataset contains 932,791 agent-generated PRs and 6,618 human-authored PRs. After preprocessing - removing unmerged PRs, records without code diffs, non-English PRs, and those

exceeding token limits - 21.62% of human PRs and 9.18% of agent-generated PRs were excluded. The dataset after this step comprises 5,188 human PRs and 847,162 agent-generated PRs.

Rule-based keyword extraction initially identified 11.76% of human PRs and 10.42% of agent-generated PRs as potentially security-related. After LLM categorization, 169 human-authored PRs were confirmed as security-related (3.26% of the total), while 10,001 agent-generated PRs were labeled as security-related (1.18% of the non-skipped records).

Across individual agents, 2.32% of PRs by Claude Code are security-related, the highest among all agents and closest to the human-authored rate. Other agents fall between 1–1.5%: Copilot (1.47%), Devin (1.34%), Cursor (1.26%), and Codex (1.02%).

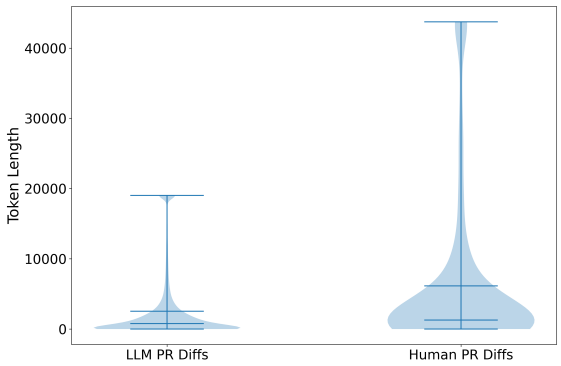

Figure 4: Distribution of token length of code diffs (trimmed at the 95th percentile)

We measured PR size using the number of tokens as processed by the GPT-OSS-120B tokenizer (Figure 4). Overall, agent-generated PRs tend to be larger than human-authored PRs. Using the keyword-extracted subsets as representative samples, human PRs average 24,044.87 tokens per PR (range: 6–7,636,892), while agent-generated PRs average 28,162.03 tokens (range: 6–75,978,686).

Focusing on security-related PRs (excluding those exceeding the 30,000-token input limit), human-authored PRs average 3,261.56 code tokens and 1,217.43 description tokens. Agent-generated security PRs average 2,479.96 code tokens, but only 181.53 description tokens per PR.

Findings #1
Agent-generated security-related PRs are roughly one-third as numerous as human-authored security PRs. While agent-generated PRs are generally larger, security-related human-authored PRs contain more code changes and substantially more descriptive context.

RQ2: Most Commonly Fixed CWE Types

During the LLM categorization step, each security-related PR is linked to a set of specific CWE types. Because the CWE hierarchy is a Directed Acyclic Graph (DAG), child CWE nodes may semantically overlap with their parent nodes. To avoid double-counting, we cleaned the LLM outputs by removing ancestor nodes, retaining only the deepest distinct CWE nodes. Our analysis focuses on all child nodes of *CWE-1000 Research Concepts* [6], which organizes vulnerabilities by abstract behavioral categories to facilitate research. In our results, three human PRs and seven agent PRs were classified as security-related but returned empty CWE lists, indicating that their vulnerabilities fall outside our research scope; these PRs were excluded from subsequent analyses. Figure 3 illustrates the detailed CWE distribution.

The distribution of CWE types highlights notable differences between human and agent-authored security PRs. For human PRs, the most frequently addressed vulnerabilities include CWE-20 (Improper Input Validation) at 17.8%, followed by CWE-400 (Uncontrolled Resource Consumption) at 7.2%, and CWE-1104 (Use of Unmaintained Third-Party Components) and CWE-1395 (Dependency on Vulnerable Third-Party Components), each at 4.24%. Other common categories include CWE-22 (Path Traversal) and CWE-284 (Improper Access Control), both at 3.81%.

For agent-generated PRs, CWE-20 also appears most frequently, but at a higher rate of 24.68%, followed by CWE-306 (Missing Authentication for Critical Function) at 7.93% and CWE-862 (Missing Authorization) at 5.88%. Additional frequent categories include CWE-79 (Cross-site Scripting) at 4.36%, CWE-287 (Improper Authentication) at 4.29%, and CWE-285 (Improper Authorization) at 3.15%.

These patterns indicate contrasting priorities: both humans and agents focus on input validation, but humans are more likely to address resource management issues, whereas agents more often target authentication and authorization vulnerabilities.

When examining CWE distributions by individual agents, we observed that all agents most frequently fix CWE-20 and CWE-306. Beyond these shared priorities, distinct patterns emerges:

- *OpenAI Codex*: CWE-862, CWE-79, CWE-287
- *Devin*: CWE-287, CWE-522 (Insufficiently Protected Credentials), CWE-798 (Hard-coded Credentials)
- *GitHub Copilot*: CWE-287, CWE-285, CWE-200 (Exposure of Sensitive Information)
- *Cursor*: CWE-862, CWE-287, CWE-285
- *Claude Code*: CWE-79, CWE-287, CWE-862

Despite having the largest proportion of security-related PRs, Claude Code exhibits the narrowest CWE coverage, spanning only 63 distinct types. In contrast, Codex, with the lowest security-related PR rate, addresses the widest range of vulnerabilities (193 CWE types), followed by Copilot (126), Cursor (88), and Devin (87).

Findings #2
Both human-authored and agent-generated PRs most commonly address input validation. Human-authored PRs emphasize resource management, whereas agent-generated PRs more frequently target authentication and authorization. Among coding agents, Claude Code has the narrowest CWE coverage, while Codex exhibits the broadest.

RQ3: Distribution of Abstract Security Behaviours

The first layer of CWE-1000 (Research Concepts) is designed to organize vulnerabilities "according to abstractions of behaviours instead of how they can be detected, where they appear in code, or

when they are introduced in the development life cycle." [6]. We leverage these abstract categories to compare the security-relevant behaviours of human and agent PRs. For each CWE leaf node identified by our LLM, we trace it to its corresponding highest-level parent node within the CWE-1000 hierarchy.

Because CWE is a DAG, there is no single canonical path from the top-level to the leaf nodes. Each leaf CWE may belong to multiple parent nodes simultaneously, reflecting valid semantic groupings. Consequently, we count each leaf-to-top-layer relationship once, which may result in overall percentages exceeding 100% due to overlapping.

For human-authored PRs, the most common behavioral categories are CWE-664 (Improper Control of a Resource Through its Lifetime) at 33.05%, CWE-707 (Improper Neutralization) at 25.42%, and CWE-284 (Improper Access Control) at 21.61%. Less frequent but notable categories include CWE-710 (Improper Adherence to Coding Standards) at 11.02% and CWE-693 (Protection Mechanism Failure) at 8.05%, with all remaining categories under 3%.

In contrast, agent-generated PRs prioritize different behaviors. The most prevalent categories are CWE-284 (Improper Access Control) at 36.43%, CWE-707 (Improper Neutralization) at 33.58%, and CWE-664 (Improper Control of a Resource Through its Lifetime) at 23.41%. Agents also address CWE-693 (Protection Mechanism Failure) at 12.57% and CWE-710 (Improper Adherence to Coding Standards) at 5.56% more frequently than humans, while other categories appear minimally.

Findings #3
Compared to human-authored PRs, agent-generated PRs disproportionately emphasize access control and input neutralization, whereas humans focus more on resource management throughout the resource lifetime and adherence to coding standards.

5 Discussion

Our analysis of over 930,000 PRs from the AIDev dataset reveals notable patterns in how human developers and AI coding agents engage with security-related changes. In this section, we interpret these findings, assess their robustness in light of methodological limitations, and consider the broader implications for software engineering teams and organizations.

5.1 Main Causes for Misclassifications

Despite careful data preprocessing and structured LLM prompting, our manual inspection revealed several systematic sources of misclassification. These issues arise not solely from the inherent non-determinism of LLMs but also from the complexities of distinguishing security-related patches from other PRs in real-world repositories.

Security Logic Embedded in Feature PRs. Some PRs introduce entirely new features or substantial refactoring while incorporating minor security-related logic (e.g., input validation or access control). These PRs are technically not security patches, as our study focuses on modifications that fix existing vulnerabilities rather than feature evolution. This overlap occasionally leads to misclassification.

Overly Large PRs. Agent-generated PRs occasionally reach extreme sizes. While agent-generated PRs are generally larger than human-authored PRs (~4000 tokens on average), some outliers in our pre-categorization dataset contain over 75 million tokens - orders of magnitude larger than the largest human PRs. Such ultra-large PRs exceed the effective context window of most LLMs, even models claiming 128k+ token capacities, which typically perform best below 32k tokens [9]. In our dataset, 6.99% of human PRs and 3.36% of agent PRs exceed this practical limit. To mitigate this, we introduced the "get_file_diff()" tool, which retrieves code changes for the files most likely to contain security-related modifications. However, file-level truncation is imperfect, as filenames do not always reflect security relevance, contributing to misclassification.

Dependency Metadata Perturbation. To capture security patches involving dependency updates in response to known CWEs, we retained dependency metadata files such as package-lock.json and pnpm-lock.yaml. These files, however, often include numerous cryptographic hashes, which occasionally misled the LLM into interpreting them as cryptographic validation logic or signature-checking, negatively impacting certain CWE classifications.

Limited Code Comprehension Capability. While GPT-OSS:120B demonstrates strong reasoning and classification capabilities for subtle CWE distinctions, its code comprehension is not specialized for software engineering tasks compared to dedicated code models [10]. Future work could explore cross-validation between a heavy reasoning model like GPT-OSS-120B and a lightweight, code-optimized model such as Qwen3-Coder-30B to improve classification reliability.

5.2 Subtype Analysis of CWE-20 Validation Behaviors

As highlighted in our results, CWE-20 (Improper Input Validation) is the most frequently observed CWE type for both human-authored and agent-generated PRs, and it ranks as the top CWE type for each coding agent. To explore this in more detail, we manually examined 42 human-authored CWE-20 PRs (the maximum available) and 50 agent-generated CWE-20 PRs. Full results are presented in Appendix B.

We observed that misclassification in human CWE-20 PRs is largely influenced by the "Dependency Metadata Perturbation" issue discussed earlier. Specifically, 20 of 42 human-authored PRs (47.6%) were misclassified as CWE-20 despite containing no actual data validation logic, whereas only 2 of 50 agent-generated PRs (4%) were similarly affected. This indicates a dataset asymmetry: single-file dependency lockfile changes constitue a disproportionately larger portion of human-authored PRs.

Agent-generated PRs tend to rely heavily on explicit validation constructs. Of the 48 agentic patches analyzed, 39 (81.25%) fall into four structured validation categories: Allowlist-based Validation & Sanitization, Required-field Validation, Format Validation, and Schema Validation. By contrast, 17 of 22 of human CWE-20 PRs (77.3%) fall into these same categories, suggesting that coding agents favor systematic, template-driven approaches, whereas human PRs often contain localized, ad hoc fixes.

Additionally, agent-generated PRs exhibit finer-grained patterns, including JWT validation and signature validation. While representing a smaller portion of patches, these cases demonstrate the breadth and systematic nature of agent-generated input validation.

5.3 Practical Applicability

Our findings, derived from the top 25 most dangerous software weaknesses, reflect highest-risk vulnerability patterns recognized in both academic research and industry practice. These CWEs dominate real-world exploitation reports, supporting the broad applicability of our results.

For researchers, the findings indicate that, despite similarities in overall CWE types distribution and abstract behaviors, human-authored and agent-generated security-related PRs differ in subtle but meaningful ways. Future work should explore whether current code review practices and heuristics can be adapted to assess agent-generated PRs effectively.

For development teams, our results suggest that routine tasks - such as feature enhancements or general refactoring - can safely be delegated to AI agents, while modifications affecting high-risk CWEs should remain under human oversight. The structured nature of agent PRs also offers opportunities for developers to leverage AI to identify niche or lower-frequency vulnerabilities that may be overlooked by humans.

From a team workflow perspective, the similarities in CWE distributions imply that PR review processes can largely treat human and agent contributions similarly. Teams could also implement monitoring tools to flag PRs involving Top 25 CWEs, potentially uncovering hidden inefficiencies when human and agent contributions are pooled.

Finally, for organizations, these results support strategic integration of AI agents into software development workflows. Key benefits include reduced risk for low-criticality vulnerabilities, potential cost savings through automation, and more informed, data-driven decisions that enhance safe human-agent collaboration.

6 Future Work

Our study provides an initial understanding of how agent-generated security-related PRs differ from those authored by humans, but several opportunities remain for extending this line of research.

First, our manual inspection concentrated primarily on CWE-20 due to its prevalence across both human-authored and agent-generated PRs. A natural next step is to conduct manual analysis on additional CWE subtypes. Since the LLM categorization process is inherently a classification problem, systematically evaluating the precision, recall, and F1-score across major CWE categories would offer a more rigorous assessment of classification quality.

Second, our current classification pipeline depends on GPT-OSS:120B for identifying security-related PRs and assigning CWE subtypes. Future work should examine cross-validation using high-capacity reasoning models and specialized coding models like Qwen3-Coder:30B.

Finally, our findings highlight the inherent difficulty of classifying security patches when PR diffs exceed an LLM's effective context window. Even with agent-driven file selection, models occasionally fail to retrieve the relevant lines of code. Future work should explore more advanced retrieval or summarization mechanisms that can reliably handle ultra-large PRs without sacrificing accuracy.

7 Threats to Validity

As with any empirical study on a large-scale which relies on an LLM for classification of code changes, our work contains several threats to internal and external validity. While we have thoughtfully designed and validated our pipeline to mitigate risks where possible, below we discuss the limitations involved.

7.1 Internal Validity

Pipeline and Selection Biases. Our pipeline relies on sequential filtering and keyword-based selection, initially targeting the 2024 Top 25 CWEs along with generic security-related keywords. This approach may over-represent certain vulnerability types while excluding subtle or emerging ones. Errors introduced at early stages - such as false positives or false negatives in PR titles, filenames, or code diffs - can propagate through the pipeline, potentially removing true security patches from the final dataset.

LLM Non-determinism. Due to the stochastic nature of LLMs, identical prompts can produce different CWE labels. Variability can arise from tokenization, floating-point operations, model versioning, or output formatting constraints, limiting perfect reproducibility.

Manual Inspection Bias. We manually verified a random subset of 106 PRs with double annotation. While this step increases confidence in the labels, it remains subject to annotator expertise and may overlook systematic misclassifications, particularly for rare CWE types.

7.2 External Validity

Dataset Representativeness. The AIDev dataset is heavily skewed: 87.3% of agent-involved PRs originate from OpenAI Codex, and only 6,618 PRs are human-authored, compared to 932,791 agent-generated PRs. Additionally, the dataset primarily contains Python and JavaScript/TypeScript repositories, limiting generalizability to other agents, programming languages, or closed-source projects.

Temporal Limitations. The dataset spans only December 2024 to July 2025. As security practices and AI agent capabilities evolve, our findings may not fully reflect trends in future agent-assisted coding workflows.

8 Conclusion

This paper presented a large-scale empirical comparison of security-related pull requests authored by human developers and AI coding agents. Using an LLM-based pipeline to identify and categorize vulnerabilities at the CWE level, we found that when agents produce security-related changes, their vulnerability profiles closely mirror those of humans. Dominant categories such as CWE-20 and CWE-306 appear consistently across both groups. Overall, however, humans authored a higher proportion of security-related PRs.

These results suggest that AI-generated PRs are not inherently more security-sensitive and may generally be treated as lower-risk within routine development workflows. Software teams can continue delegating repetitive or low-impact tasks to coding agents while reserving high-stakes or semantically complex modifications

for human oversight. Our released classification pipeline provides a reproducible basis for monitoring these trends over time.

Future work should investigate additional programming languages, longer temporal coverage, and whether agents differ from humans in their likelihood of introducing exploitable weaknesses - an open challenge for the research community.

Data and Code Availability

Data: https://huggingface.co/datasets/felixwangg/AIDev-CWE-Classification.
Code: https://github.com/U70-TK/MSR-2026-challenge.

References

[1] Mahmoud Alfadel, Diego Elias Costa, Emad Shihab, and Mouafak Mkhallalati. 2021. On the Use of Dependabot Security Pull Requests. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 254–265. doi:10.1109/MSR52588.2021.00037

[2] Owura Asare, Meiyappan Nagappan, and N. Asokan. 2024. A User-centered Security Evaluation of Copilot. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 158, 11 pages. doi:10.1145/3597503.3639154

[3] The Mitre Corporation. 2025. 2024 CWE Top 25 Most Dangerous Software Weaknesses. https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html

[4] The Mitre Corporation. 2025. CVE Program Mission. https://www.cve.org/

[5] The Mitre Corporation. 2025. CWE- Common Weakness Enumeration. https://cwe.mitre.org/

[6] The Mitre Corporation. 2025. CWE VIEW: Research Concepts. https://cwe.mitre.org/data/definitions/1000.html

[7] Ahmed E Hassan, Gustavo A Oliva, Dayi Lin, Boyuan Chen, Zhen Ming, et al. 2024. Towards AI-native software engineering (SE 3.0): A vision and a challenge roadmap. *arXiv preprint arXiv:2410.06107* (2024). https://doi.org/10.48550/arXiv.2410.06107

[8] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Trans. Softw. Eng. Methodol.* 33, 8, Article 220 (Dec. 2024), 79 pages. doi:10.1145/3695988

[9] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models?. In *First Conference on Language Modeling*. https://openreview.net/forum?id=kIoBbc76Sy

[10] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-world Github Issues?. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=VTF8yNQM66

[11] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. arXiv:2507.15003 [cs.SE] https://arxiv.org/abs/2507.15003

[12] Xingyu Li, Juefei Pu, Yifan Wu, Xiaochen Zou, Shitong Zhu, Qiushi Wu, Zheng Zhang, Joshua Hsu, Yue Dong, Zhiyun Qian, et al. 2025. What Do They Fix? LLM-Aided Categorization of Security Patches for Critical Memory Bugs. *arXiv preprint arXiv:2509.22796* (2025). https://doi.org/10.48550/arXiv.2509.22796

[13] Shengyi Pan, Lingfeng Bao, Xin Xia, David Lo, and Shanping Li. 2023. Fine-grained Commit-level Vulnerability Type Prediction by CWE Tree Structure. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 957–969. doi:10.1109/ICSE48619.2023.00088

[14] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2025. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. *Commun. ACM* 68, 2 (Jan. 2025), 96–105. doi:10.1145/3610721

[15] Hocine Rebatchi, Tégawendé F Bissyandé, and Naouel Moha. 2024. Dependabot and security pull requests: large empirical study. *Empirical Software Engineering* 29, 5 (2024), 128.

[16] Mark Vero, Niels Mündler, Victor Chibotaru, Veselin Raychev, Maximilian Baader, Nikola Jovanović, Jingxuan He, and Martin Vechev. 2025. BaxBench: Can LLMs Generate Correct and Secure Backends?. In *Forty-second International Conference on Machine Learning*. https://openreview.net/forum?id=il3KRr4H9u

[17] Mark Vero, Niels Mündler, Victor Chibotaru, Veselin Raychev, Maximilian Baader, Nikola Jovanović, Jingxuan He, and Martin Vechev. 2025. BaxBench: Can LLMs Generate Correct and Secure Backends? (2025). arXiv:2502.11844

[18] Shu Wang, Xinda Wang, Kun Sun, Sushil Jajodia, Haining Wang, and Qi Li. 2023. GraphSPD: Graph-Based Security Patch Detection with Enriched Code Semantics. In *2023 IEEE Symposium on Security and Privacy (SP)*. 2409–2426. doi:10.1109/SP46215.2023.10179479

[19] Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E Hassan. 2025. On the use of agentic coding: An empirical study of pull requests on github. *arXiv preprint arXiv:2509.14745* (2025). https://doi.org/10.48550/arXiv.2509.14745

[20] Mairieli Wessel. 2020. Enhancing developers' support on pull requests activities with software bots. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1674–1677. doi:10.1145/3368089.3418539

[21] Mairieli Wessel and Igor Steinmacher. 2020. The Inconvenient Side of Software Bots on Pull Requests. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 51–55. doi:10.1145/3387940.3391504

[22] Marvin Wyrich, Raoul Ghit, Tobias Haller, and Christian Müller. 2021. Bots Don't Mind Waiting, Do They? Comparing the Interaction With Automatically and Manually Created Pull Requests. In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*. 6–10. doi:10.1109/BotSE52550.2021.00009

A Prompts For LLM Categorization

A.1 Initial Prompt

You are a professional software developer and security engineer.

You will classify whether this pull request is a SECURITY PATCH and determine the correct CWE categories.

VERY IMPORTANT:

1. First, examine the PR description and the list of changed files.

2. Then choose EXACTLY ONE file that is most relevant for deciding whether this PR fixes a security issue.

3. Call get_file_diff(filename=…) ONCE using a filename from the list.

4. Carefully read the returned diff to understand the actual code change.

5. If you suspect this **is** a security patch:
– Think of several possible CWE titles that might apply.
– For each suspected CWE, call: lookup_cwe(title=…)
– Read the CWE description returned by the tool.
– ONLY keep CWEs whose description truly matches the PR context.
– Discard any CWE that is only loosely related.

6. Once you have fully used the tools and made your decision, respond ONLY with JSON matching this schema (no markdown, no code fences):

{*ResponseSchema.model_json_schema()*}

Do NOT include explanations, comments, or extra text.

Pull Request Description

{*body*}

Files Changed (you MUST choose one of these for get_file_diff):

{*files_block*}

A.2 Response Error Prompt

Your previous output was not a valid JSON or a valid tool call.
Please strictly follow the JSON format or produce a valid tool call.

Table 2: CWE-20 Subtype Distribution in 42 Security-Related Human-authored PRs

Category	Count	PR IDs
Misclassification	20	2382067611, 2389918400, 2438539377, 2339752897, 2287603793, 2458036182, 2416870587, 2425765290, 2386305443, 2420290502, 2534253365, 2394268453, 2430909406, 2502403389, 2625803728, 2315795950, 2386412881, 2347480106, 2491989694, 2344238227
Allowlist-based Validation & Sanitization	9	2337822294, 2478411367, 2270707038, 2528604644, 2501937539, 2315827314, 2563706628, 2399445785, 2361911239
Schema Validation	3	2601011916, 2463457386, 2300488657
Format Validation	3	2552633641, 2582168338, 2526683398
Dependency Change of Validation Libraries	3	2588670843, 2436364208, 2620656205
Required-field Validation	2	2285999586, 2554422043
Length Validation	1	2424210775
SQL Injection Sanitization	1	2368306114
Total	**42**	

Table 3: CWE-20 Subtype Distribution in 50 Security-Related Agent-generated PRs

Category	Count	PR IDs
Allowlist-based Validation & Sanitization	12	3248833818, 3115582224, 3217349748, 3082895543, 3240706877, 3165077339, 3141078763, 3265729225, 3275408870, 3275477167, 3176321997, 3275396931
Required-field Validation	8	3147329333, 3274811584, 3128828228, 3225195377, 3265384778, 3191162279, 3124129557, 3245592436
Format Validation	7	3217638579, 3202658979, 3183647662, 3141495378, 3197527688, 3087269819, 3224590283
Schema Validation	6	3210518727, 3206018407, 3134759475, 3074589866, 3225221997, 3225252416
Length/Range/Boundary Validation	6	3114818986, 3114932140, 3144403926, 3206077611, 3122694492, 3100768367
SQL Injection Sanitization	4	3137127768, 3144744310, 3144761945, 3176474851
Misclassification	2	3230700736, 3239548369
Unicode sanitization	1	3180583914
Cryptographic Signature Validation	1	3224680884
Precondition validation	1	3124764187
Option/Flag Validation	1	3188688714
JWT Validation	1	3232968512
Total	**50**	

B CWE-20 Manual Inspection Result

B.1 Human Pull Requests

CWE-20 related Human PRs are in Table 2.

B.2 Agent Pull Requests

CWE-20 related Agentic PRs are in Table 3.