MulitaMiner: A Multi-Version Evaluation of LLM-Based Vulnerability Report Extraction

Beatriz Machado[1], Douglas Lautert[1], Cristhian Kapelinski[1]
Diego Kreutz[1], Isadora Garcia Ferrão[2], Alessandro Bof[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
[2]Université de Bretagne Occidentale (UBO)
{beatrizmachado,douglaslautert,cristhianavila}.aluno@unipampa.edu.br
{diegokreutz,alessandrooliveira}@unipampa.edu.br
isadora.garciaferrao@univ-brest.fr

Abstract. *Security teams face 131 new CVEs per day, yet the National Vulnerability Database fails to enrich 44% of new submissions, leaving practitioners without the structured metadata required to prioritize remediation. MulitaMiner closes this gap: an LLM-based pipeline for structured vulnerability extraction from heterogeneous scanner PDFs, hardened across three successive versions (V1–V3) that evolve from a functional baseline, to scanner-aware segmentation, and finally to per-LLM tuning with few-shot prompting. We evaluate five LLMs (DeepSeek, GPT-4, GPT-5, LLaMA 3, LLaMA 4) on three manually curated OpenVAS baselines across 450 independent runs. Without changing the underlying models, Exact Record Match rises from 37.5% to 90.4%, vulnerability-level omission falls by an order of magnitude (*$20.5\% \rightarrow 1.7\%$*), and cross-model variance in field-level omission collapses from 41.8 to 2.9 percentage points, all with non-overlapping 95% bootstrap confidence intervals. These results establish that pipeline engineering, not model substitution, is the primary lever for extraction quality: within the scope evaluated here, a well-engineered pipeline turns LLM choice into a question of cost and latency rather than accuracy.*

1. Introduction

In 2025, more than 48,000 *Common Vulnerabilities and Exposures* (CVEs) were published [NIST 2026], approximately 131 new entries per day, and submissions grew 263% between 2020 and 2025. The *National Vulnerability Database* (NVD) has been unable to fully enrich nearly 44% of newly submitted CVEs, depriving security teams of the metadata needed to prioritize remediation [The Sequence 2025]. These figures converge on a single structural problem: security teams are overwhelmed not only by the volume of vulnerabilities, but by the inability to extract, structure, and act on the information that describes them.

Much of the relevant knowledge remains scattered across unstructured sources, including PDF reports, Computer Security Incident Response Team (CSIRT) bulletins, such as those from the Brazilian national ecosystem coordinated by RNP[1], and vendor security advisories. Structured extraction from technical documents has been explored across multiple domains using LLMs [Dunn et al. 2024, Klusty et al. 2024], yet the security domain presents unique challenges: highly specialized terminology, heterogeneous

[1]https://www.rnp.br/

formats across scanner generations, and extraction errors with direct operational consequences, as a misclassified severity or an omitted CVE identifier can delay remediation of a critical exposure [Ponce et al. 2024].

Vulnerability management relies on tools such as OpenVAS[2] and Tenable WAS[3], which are widely used to identify vulnerabilities in network infrastructures and web applications [Foreman 2019]. Despite their adoption, their reports differ significantly in structure, field naming, and severity classification, making unified automated processing an open research challenge. This heterogeneity compromises the quality of analytical models [Dabholkar 2023, Mitchell et al. 2025, Croft et al. 2023, Guo et al. 2024] and keeps manual processes widely employed [Rafati 2025]. Although LLM-based approaches have advanced structured extraction of security data, they still lack systematic evaluations, adaptive segmentation strategies, and reproducible benchmarks at production scale [Hu et al. 2024, Ferrag et al. 2025, Brokman et al. 2024].

MulitaMiner is a pipeline for structured vulnerability extraction from heterogeneous PDF reports, hardened across three successive versions (V1, V2, V3), each closing the failure modes of its predecessor: from a functional baseline with fixed chunking, to scanner-aware segmentation, and finally to per-LLM tuning with few-shot prompting. The central finding is that systematic pipeline refinement yields more return than substituting one capable LLM for another. The pipeline ingests reports from more than one scanner, but the evaluation reported here is confined to OpenVAS, for the reasons discussed in Section 4.7; the heterogeneity that motivates the design is therefore addressed architecturally and not yet measured across scanners.

The claim we defend is empirical rather than methodological. None of the techniques MulitaMiner combines is new in isolation: marker-aware segmentation, per-LLM token configuration, few-shot prompting, and schema-anchored recovery all appear in prior work. What is missing from the literature is a quantification of how much their systematic integration matters relative to the choice of model, which is the decision practitioners actually face. Across 450 independent runs, pipeline refinement alone raises Exact Record Match from 37.5% to 90.4% and reduces vulnerability-level omission from 20.5% to 1.7% without changing the underlying LLMs, a margin that no model substitution in our study approaches.

MulitaMiner[4] makes four contributions: (i) we quantify, across 450 independent runs, the extent to which pipeline refinement rather than model substitution drives extraction quality, under the ERM and omission figures above; (ii) we release a reproducible pipeline with adaptive scanner-aware chunking, per-LLM token configuration, and configurable deduplication; (iii) we establish that cross-model variance in field-level omission collapses from 41.8 to 2.9 pp under V3 on the OpenVAS targets evaluated, leaving cost and latency, rather than quality, as the dominant criterion for choosing among the models we evaluated; and (iv) we publish source code, three manually curated baselines, and all per-run artifacts.

The remainder of the paper covers related work (Section 2), the MulitaMiner ar-

[2]`https://www.openvas.org`
[3]`https://docs.tenable.com/web-app-scanning.htm`
[4]`https://github.com/AnonShield/MulitaMiner/tree/V3`

chitecture (Section 3), experimental results (Section 4), and conclusions (Section 5).

2. Related Work

The structured extraction of information from unstructured technical documents has been explored across multiple domains using natural language processing and large language models. Table 1 compares representative works along dimensions relevant to our contributions, of which the last two, adaptive segmentation and public dataset release, are the most critical for scientific rigor and reproducibility.

General Structured Extraction with LLMs. In the business and enterprise domain, DocLLM [Wang et al. 2024] extends language models with layout-aware spatial attention to handle visually rich documents such as forms and invoices. TableLLM [Zhang et al. 2024] targets tabular data embedded in PDFs and spreadsheets, applying LLaMA 3.1 for table-specific reasoning and evaluating with accuracy and F1 metrics. LangchainIQ [Ghane et al. 2024] employs retrieval-augmented generation (RAG) for heterogeneous corporate documents, while ChatDoc [Lin 2024] applies enhanced PDF structure recognition for technical manual retrieval. Although these systems produce structured outputs, they rely on *fixed-size* text segmentation and lack adaptive mechanisms to handle semantically heterogeneous documents. Evaluation is often limited to benchmark accuracy or qualitative assessment, and no public dataset is released for the specific extraction task. In the health sciences domain, [Garcia et al. 2024] demonstrate zero-shot extraction from clinical reports using GPT-3.5 and LLaMA 2, reporting accuracy and ROUGE metrics, establishing a methodological precedent for quantitative assessment of LLM-based extraction quality.

Cybersecurity-Specific Approaches. Within the cybersecurity domain, structured extraction from heterogeneous technical documents has motivated diverse approaches. Gao et al. [Gao et al. 2022] proposed ThreatKG, an NLP pipeline for entity and relation extraction from threat intelligence reports to construct knowledge graphs, quantitatively evaluated using precision and recall. Park and Lee [Park and Lee 2022] introduced a supervised full-stack information extraction system for cybersecurity intelligence, achieving strong F1 scores, but requiring labeled training data and not addressing heterogeneous PDF inputs. Both works predate the widespread adoption of capable LLMs and do not target the processing of reports generated by vulnerability scanners.

More recent LLM-based approaches include PDFInspect [Sharmila 2026], which combines RAG and LLaMA 2 for threat modeling from technical design documents, and Dabholkar [Dabholkar 2023], who integrates the OpenAI API to aggregate and summarize outputs from open-source vulnerability scanners into readable reports. While these works establish the feasibility of LLM-based processing in security contexts, both rely on fixed-size segmentation and qualitative evaluation, the same combination that, in MulitaMiner V1, produced 20.5% vulnerability-level omission (Section 4), a failure mode that qualitative evaluation can spot anecdotally but cannot quantify or characterize across scale. CHATIOT [Dong et al. 2025] introduces a RAG-based LLM assistant for IoT security intelligence queries, but focuses on interactive question-answering rather than scalable, bulk structured extraction from heterogeneous PDF reports.

A recent comparative analysis of open-source LLM-based vulnerability scanners [Brokman et al. 2024] identified three critical gaps common to all evaluated systems:

the absence of adaptive chunking strategies, insufficient quantitative evaluation methodology, and the lack of reproducible benchmarks. These limitations directly motivate the design of MulitaMiner.

Most closely related to our work, a prior modular pipeline for OpenVAS and Tenable WAS extraction [Machado et al. 2025] employs fixed-size segmentation rather than adaptive, scanner-aware chunking; does not incorporate configurable semantic deduplication; and evaluates extraction quality only through lexical similarity (ROUGE-L), without measuring vulnerability-level coverage, field-level exactness, or severity classification. MulitaMiner extends this line of work with adaptive per-LLM chunking, a configurable deduplication layer, and a metric battery that quantifies the failure modes (omission, hallucination, ERM) that lexical metrics alone cannot detect.

Research Gaps. As Table 1 makes explicit, no existing work combines adaptive scanner-aware segmentation, multi-LLM evaluation beyond similarity metrics, configurable deduplication, and the public release of a reproducible vulnerability dataset. MulitaMiner bridges the four gaps within a single pipeline.

Table 1. Comparative analysis of structured extraction approaches from unstructured documents

Domain	Reference	Input Document	Approach	Models	Evaluation	Adapt. Segm.	Public Dataset
Business & Enterprise	[Wang et al. 2024]	Business docs (forms, invoices, reports)	Layout-aware visual and textual attention (DocLLM)	LLaMA 2 / Falcon	Benchmark accuracy (WTQ, TAT-QA)	No	No
	[Zhang et al. 2024]	Tabular content in PDFs and spreadsheets	Tabular data-specific reasoning (TableLLM)	LLaMA 3.1 8B	Accuracy / F1	No	No
	[Ghane et al. 2024]	Corporate docs (text + tables)	RAG with document integration and response generation	Mistral 7B	Qualitative	No	No
	[Lin 2024]	Technical manuals and PDFs	RAG with enhanced PDF structure recognition	GPT-3.5 Turbo	Accuracy	No	No
Health Sciences	[Garcia et al. 2024]	Unstructured clinical reports and medical records	Zero-shot LLM extraction to JSON	GPT-3.5 / LLaMA 2	Accuracy, ROUGE	No	No
Cybersecurity	[Gao et al. 2022]	Threat intelligence reports (unstructured text)	Entity and relation extraction for Knowledge Graph construction	NLP + DL (BERT-based)	Precision / Recall	No	No
	[Park and Lee 2022]	Technical security reports	Supervised full-stack IE (NER + relation extraction)	Supervised ML	F1, Precision, Recall	No	No
	[Sharmila 2026]	Technical design docs and specs (PDF)	RAG + LLM for threat modeling with CVE identification	LLaMA 2 / KeyBERT	Qualitative	No	No
	[Dabholkar 2023]	Vulnerability scanner outputs (open-source tools)	OpenAI API integration for automated report aggregation	GPT-3 / GPT-3.5	Qualitative	No	No
	[Dong et al. 2025]	IoT threat intelligence documents	RAG + LLM for interactive security Q&A (CHATIOT)	GPT-4 family	Response quality	No	No
	[Machado et al. 2025]	OpenVAS / Tenable WAS reports	Modular pipeline with field identification and unified schema mapping	GPT-4/4.1, DeepSeek, LLaMA3/4	ROUGE-L	No	Yes
	MulitaMiner (this work)	**Heterogeneous vulnerability reports (PDF)**	**Scanner-aware adaptive chunking with per-LLM tuning, few-shot prompting, and deduplication**	**GPT-4/5, DeepSeek, LLaMA 3/4**	**BERTScore, ROUGE-L, Token-F1, Field-F1, Coverage, Severity**	**Yes**	**Yes**

3. Architecture

The MulitaMiner architecture was designed addressing specific challenges in the cybersecurity domain, particularly the heterogeneity, structural inconsistency, and high semantic variability of vulnerability reports generated by different scanners. Existing solutions frequently assume pre-structured inputs or rely on fixed segmentation and inflexible pipelines, which compromises context preservation, hinders consolidation of large data volumes, and introduces redundancies and inconsistencies in the results.

As illustrated in Figure 1, MulitaMiner organizes its processing into six stages (**input data, text extraction, preprocessing, LLMs, output, and analysis**), structured to ensure modularity, low coupling, and reproducibility. This organization explicitly addresses common limitations in existing approaches, such as the absence of adaptive semantic segmentation, ineffective deduplication, and lack of systematic evaluation mechanisms.

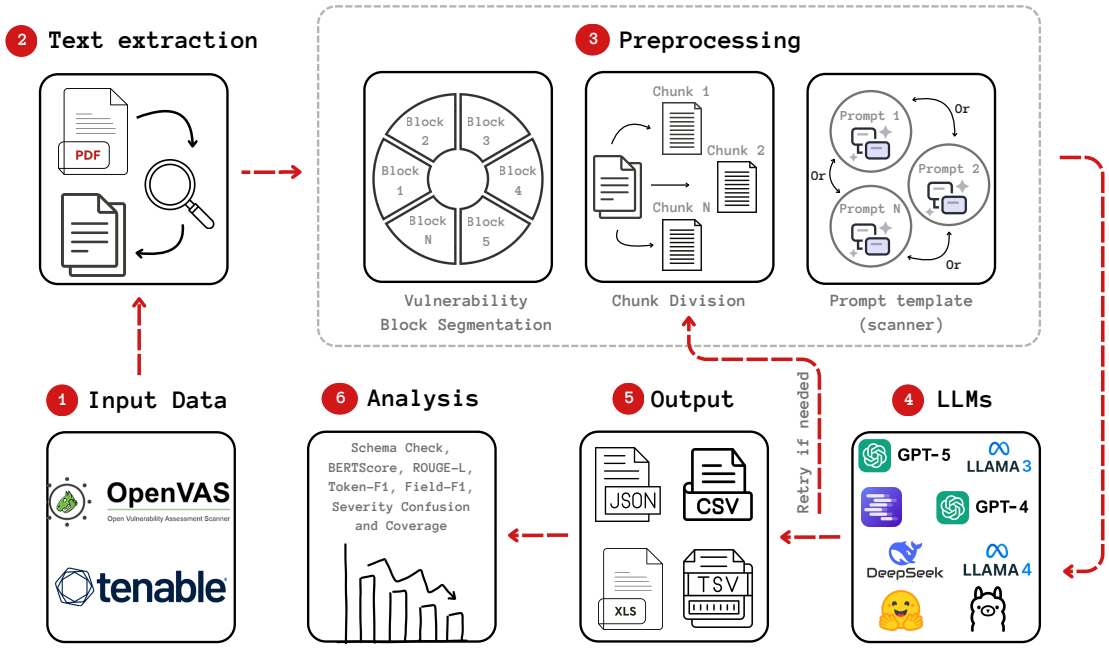

Figure 1. MulitaMiner pipeline

Input and extraction layer: handles the ingestion of PDF reports from different scanners (e.g., OpenVAS, Tenable WAS), preserving relevant semantic structures such as titles, identifiers, and technical sections. Unlike simplified approaches based on linear text extraction, this layer incorporates Unicode normalization, encoding handling, and structural validation, reducing noise introduced by formatting artifacts typical of these reports.

Preprocessing: directly addresses a recurring limitation in existing solutions that use fixed-size segmentation and disregard the semantic organization of reports. MulitaMiner implements adaptive scanner-oriented segmentation (Vulnerability Block Segmentation), token-controlled chunking (Chunk Division), and scanner-specific prompt templates, reducing context loss and improving extraction quality. Parameterizable templates allow rapid adaptation to new report formats, while the generation of specialized prompts directs models toward consistent extraction of vulnerability attributes. Failures in JSON

parsing or response truncation first invoke a recovery step that salvages malformed outputs; only when recovery fails does the chunk enter the adaptive retry mechanism that recursively subdivides and reprocesses it.

Large Language Models (LLMs): this layer dispatches each prompt to multiple models, including commercial APIs (GPT-4/5, LLaMA3/4, DeepSeek) and local execution backends (Ollama, Hugging Face, LM Studio), enabling multi-LLM execution under heterogeneous deployment scenarios. Unlike monolithic pipelines, this layer enables comparative analysis between models under different configurations. A validated JSON schema standardizes the outputs, with fallback mechanisms handling inconsistencies and reducing structural variations commonly observed in LLM-based applications.

Output: consolidates results into structured formats including *JavaScript Object Notation* (JSON), *Comma-Separated Values* (CSV), *Excel Spreadsheet (Office Open XML)* (XLSX), and *Tab-Separated Values* (TSV), with automatic schema validation. To mitigate redundancies, a recurring problem in vulnerability extraction, MulitaMiner applies configurable deduplication strategies based on vulnerability name, textual similarity, and identifiers. This mechanism allows balancing duplicate elimination while preserving legitimate occurrences, contributing to structured data quality.

Analysis: integrates an optional quantitative evaluation module, a capability absent in most solutions for this domain. The module scores every extraction against expert-defined baselines along four dimensions, detailed in Section 4: structural conformance, textual similarity, field-level exactness, and vulnerability-level coverage. This enables systematic comparison between models and processing strategies.

Together, these six stages overcome the fixed-segmentation and qualitative-evaluation limitations of prior work, as Section 4 demonstrates quantitatively across 450 runs and five LLMs.

3.1. Architectural progression

Table 2 summarizes the architectural differences between the three iterations of the MulitaMiner pipeline. Each version evolved in response to limitations observed in the previous one.

The progression reflects a shift from one-size-fits-all heuristics toward configuration that adapts to the LLM and the input. V1 treated scanner markers as a soft hint, so vulnerabilities exceeding the token budget were cut mid-block; V2 hardened the boundary into a strict rule but applied the token and character caps in separate stages; V3 unifies both in a single pass and derives the character ceiling from the observed character-to-token density. The same principle extends to the remaining dimensions of Table 2.

Source code, prompt templates, and per-version configuration files for V1 and V2 are publicly available in the corresponding repositories[5], while those for V3 are available in the repository referenced in Footnote 4.

[5]V1: `https://github.com/AnonShield/Vulnerability_Extractor/tree/dev`; V2: `https://github.com/AnonShield/MulitaMiner/tree/main`.

Table 2. Architectural differences across MulitaMiner pipeline versions.

Dimension	V1	V2	V3
Chunk size	Read from JSON, identical across LLMs (∼2600 tokens; 1000 reserved for response).	Hardcoded, identical across LLMs (∼3000 tokens; 1000 reserved).	Tuned per-LLM, configurable via JSON.
Chunking algorithm	Single-pass; accumulates lines up to the token budget before the next scanner marker.	Two-step: marker-aware char split (8000), then token slice (∼3000).	Marker-aware split with dual cap: token (from JSON) and char (derived from token density, with safety margin).
Tokenizer	`cl100k_base`	`cl100k_base`	Per-LLM, configurable via JSON.
Retry on failure	Up to 3 attempts; failed chunk split in half at nearest line break and reprocessed.	Up to 3 attempts; failed chunk redivided respecting scanner markers.	Library-based JSON repair recovers malformed outputs before falling back to up to 3 error-aware redivision attempts respecting scanner markers. Fatal API errors abort the run.
LLM providers	ChatOpenAI.	ChatOpenAI.	ChatOpenAI, Ollama, Hugging Face, LM Studio (local execution).
Metrics	None.	BERTScore, ROUGE-L.	Schema check, BERTScore, ROUGE-L, Token-F1, Field-F1, Severity confusion, Coverage.

4. Results

4.1. Experimental Setup

The evaluation uses three manually curated baselines extracted from OpenVAS PDF reports of Docker containers with known vulnerability profiles: Artifactory OSS 5.11.0 (116 vulnerabilities), OWASP Juice Shop (34), and Buggy Bank Web Application (bBWA, 58). These targets were selected to cover distinct application categories (artifact repository, deliberately vulnerable e-commerce application, and intentionally insecure banking web application) and three points along the vulnerability density spectrum (34, 58, and 116 records), enabling the evaluation to expose scale-dependent behavior in both pipeline and models.

Each baseline was constructed in a two-stage process: a primary annotator extracted vulnerabilities from the source PDF, and a second specialist independently reviewed the resulting baseline against the original PDF. The review was field-by-field over the canonical schema: each record was checked for the presence of a corresponding entry in the report, for the exact value of the deterministic fields (`name`, `port`, `protocol`, `severity`, `cvss`), and for the faithfulness of the free-text fields to the corresponding section of the report. Entries present in the report but absent from the baseline were added, and divergences were resolved against the source PDF, which was treated as the sole authority. This sequential extract-then-review protocol prioritizes consistency over inter-annotator agreement metrics: by design, it produces a single converged baseline rather than two parallel annotations, so standard agreement statistics (Cohen's κ, raw agreement) are not applicable.

Five LLMs were evaluated under each pipeline version: DeepSeek-Coder, GPT-

4o mini (2024-07-18), GPT-5 mini (2025-08-07), Llama 3.3 70B Versatile, and Llama 4 Scout 17B 16E Instruct. All models were accessed through their official cloud APIs (OpenAI for the GPT family, the native DeepSeek API, and Groq for the Llama models), ensuring that the observed differences reflect model behavior rather than deployment artifacts.

Each (version, model, baseline) combination was evaluated in 10 independent runs to capture the run-to-run variability introduced by stochastic LLM decoding, resulting in a total of 450 executions (3 pipeline versions $\times$ 5 models $\times$ 3 baselines $\times$ 10 runs). To comply with API rate limits, models were grouped by provider (OpenAI for the GPT family, the native DeepSeek API, and Groq for the Llama models), and each provider group was executed sequentially.

4.2. Metrics Battery

All metrics follow the same two-stage aggregation. First, the metric is computed independently for each run of each (model, baseline) pair. Second, per-run values are aggregated for visualization: scalar metrics (e.g., hallucination rate, omission rate) are averaged across runs and baselines; categorical metrics (e.g., similarity categories) aggregate the per-run category counts and report their mean share. This two-stage design treats each run as an independent sample of stochastic LLM behavior, preserving run-to-run variability that would be masked by pooling raw scores or averaging at the field level before categorization. The metric battery covers four complementary dimensions:

- **Vulnerability-level alignment:** Extracted records are aligned with baseline records via a three-stage hierarchy: (1) composite-key match on a scanner-aware tuple (`name|port|protocol` for OpenVAS). Positions absent in either record are treated as wildcards and match any value, with a reduced contribution to the match score so that fully-populated keys are preferred over wildcard matches; (2) exact normalized-name match for records whose composite key fails; (3) fuzzy name match (rapidfuzz ratio ≥ 0.85) as final fallback. Each baseline record is consumed at most once. From this alignment, *hallucination rate* measures the fraction of extracted records without a baseline match, and *omission rate* measures the fraction of baseline records without a corresponding extraction. The mean number of successfully aligned pairs per run, averaged across the three baselines, is reported as *matched pairs*.
- **Field-level quality (within matched pairs):** exact comparison is well-defined only for deterministic fields, evaluated separately from free-text content. *Exact Record Match* (ERM) is the fraction of matched pairs in which all deterministic fields (`cvss`, `port`, `protocol`, `severity`, `source`) agree exactly with the baseline. *Field-level hallucination* counts fields filled in the extraction where the baseline is empty; *field-level omission* counts the inverse. Precision, recall, and F1 are also reported per field via exact comparison.
- **Textual similarity (free-text fields):** for free-text fields such as `description`, `impact`, `detection_method`, and `references`, three scores are reported between extracted and baseline content: *BERTScore F1* (semantic, embedding-based), *ROUGE-L* (lexical, longest common subsequence), and *Token-F1* (token overlap). Each per-field score is also categorized discretely as *Highly Similar* ($\geq$

0.7), *Moderately Similar* (0.6–0.7), *Slightly Similar* (0.4–0.6), *Divergent* (< 0.4), or *Absent* (extraction empty when baseline is not).
- **Severity classification.** *Macro F1* averages per-class F1 across the five severity levels (LOG / LOW / MEDIUM / HIGH / CRITICAL). *Coverage-aware Macro F1* weights each per-class F1 by the fraction of baseline records of that class actually recovered, penalizing pipelines that classify well only on what they happen to extract.

V1 is not unfairly penalized for requesting only 16 of the 18 canonical fields: coverage metrics count a baseline-side cell toward omission only when the baseline populates it, and the two missing fields (`plugin_details`, `instances`) have a baseline-side population rate of zero across all curated targets, so restricting omission to V1's 16 fields leaves the V1→V3 gap essentially unchanged.

For each headline metric we report 95% bootstrap confidence intervals (10,000 resamples, percentile method) over the $N{=}150$ observations per version (5 models $\times$ 3 baselines $\times$ 10 runs) that compose the aggregate mean. We treat non-overlapping CIs between two versions as evidence that the observed difference exceeds run-level noise. Non-overlap is a sufficient but not a necessary condition for significance at the 5% level, so the criterion is conservative rather than permissive.

The complete run-level metric tables, per-model per-field breakdowns, and interactive comparison artifacts are available in the repository[1].

4.3. Aggregate Performance Across Versions

Table 3 consolidates the aggregate metrics across the three pipeline versions, grouped by three evaluation dimensions of the metric battery: vulnerability-level coverage, field-level quality, and severity classification.

The V1→V2 transition is dominated by gains in coverage. Matched pairs per run rise from 54.0 to 63.7 ($+18\%$), vulnerability-level omission drops from 20.5% to 8.1% (a $2.5\times$ reduction), and ERM more than doubles ($37.5\% \rightarrow 88.0\%$). The non-overlapping CIs (e.g., ERM: V1 [34.3, 40.7] vs V2 [86.9, 89.2]) confirm that these are not noise. Field-level omission halves in the same transition ($25.1\% \rightarrow 12.5\%$) and field-level hallucination drops by roughly 28% ($6.8\% \rightarrow 4.9\%$). These gains track three architectural changes introduced in V2: the marker boundary in the initial chunker becomes a strict rule (in V1 it was a soft hint that only triggered when the token budget was near exhaustion), the retry path becomes marker-aware (in V1, retried chunks were split blindly at line breaks), and the extraction prompt becomes more directive.

The V1→V3 omission improvement ($20.5\% \rightarrow 1.7\%$, a $12\times$ reduction) reflects two distinct effects. Across the four non-pathological model families (DeepSeek, GPT-4, LLaMA 3, LLaMA 4), V3 reduces vulnerability-level omission by roughly $4\times$ ($7.5\% \rightarrow 1.8\%$), attributable to the marker-aware chunking and retry path introduced in V2 plus the redesigned, schema-anchored extraction prompt of V3 (Section 3). The remaining factor of $\sim$3 comes from V3 stabilizing GPT-5, whose V1 failure stemmed from execution parameters uniformly calibrated for the broader model pool that could not accommodate its reasoning overhead: outputs truncated before producing valid JSON, triggering a cascade of parse failures. V3's per-LLM tunable configuration (Table 2) gives

Table 3. Aggregate metrics across pipeline versions. Each cell shows the mean over N=150 (model, baseline, run) observations; values in brackets are 95% bootstrap confidence intervals

Dimension	Metric	Dir.	V1	V2	V3
Vuln.-level coverage	Matched pairs / run	↑	54.0 [49.1, 59.1]	63.7 [58.4, 68.8]	**67.8** [62.8, 73.2]
	Hallucination rate (vuln)	↓	7.1% [5.2, 9.0]	**5.7%** [5.4, 6.0]	6.8% [6.3, 7.4]
	Omission rate (vuln)	↓	20.5% [16.2, 25.2]	8.1% [6.5, 9.9]	**1.7%** [1.4, 2.0]
Field-level quality	Exact Record Match (ERM, 5 fields)	↑	37.5% [34.3, 40.7]	88.0% [86.9, 89.2]	**90.4%** [89.3, 91.5]
	Hallucination rate (field)	↓	6.8% [5.6, 8.2]	4.9% [4.6, 5.2]	**4.5%** [4.3, 4.8]
	Omission rate (field)	↓	25.1% [22.5, 27.7]	12.5% [11.3, 13.9]	**8.3%** [7.4, 9.2]
Severity classification	Macro F1	↑	94.8% [94.2, 95.4]	**97.7%** [97.2, 98.3]	97.0% [96.5, 97.5]
	Coverage-aware Macro F1	↑	79.6% [76.6, 82.5]	90.4% [89.4, 91.6]	**91.3%** [90.5, 92.1]

GPT-5 the response headroom and native tokenizer it requires, and library-based JSON repair salvages residual truncations, together eliminating the cascade and recovering an architecturally fragile model that prior versions could not handle.

4.4. Hallucination × Omission Trade-off

Figure 2 plots vulnerability-level hallucination against omission for every (model, version, baseline) configuration, and the three versions occupy visibly distinct regions of the plane. V1 (red) spreads across the chart, with outliers in both directions: GPT-5 fails systematically with omission above 50% on all baselines, while three other configurations exceed 20% hallucination. V2 (yellow) consolidates most configurations into a moderate-omission, low-hallucination band. V3 (blue) collapses all configurations into the low-omission, low-hallucination region, with every pair below 10% omission and only two above 10% hallucination (bounded at 15%).

4.5. Field-Level Omission Patterns

Figure 3 disaggregates omission by model, version, and field (darker cells indicate higher omission), revealing two patterns that the aggregate numbers obscure: a catastrophic per-model failure in V1 and progressive cross-model convergence in V3.

First, GPT-5 under V1 collapses: nearly the entire row is red, with field-level omission $\geq 50\%$ across most fields and across all three baselines. V2 partially recovers GPT-5 (the row turns predominantly yellow-green), and V3 fully recovers it, aligning the row with the other models. The recovery is consistent with two V3 changes acting in combination: adaptive per-LLM chunking that prevents oversized inputs (to which GPT-5 was particularly sensitive in V1), and a redesigned extraction prompt that is roughly 40% shorter than V2's and structurally clearer, with an explicit output contract, a JSON schema, and two end-to-end few-shot examples covering common edge cases.

Second, the best-to-worst spread in field-level omission rate collapses from 41.8% in V1 (12.1%–53.9%) to 14.7% in V2 (7.4%–22.1%) to only 2.9% in V3 (7.3%–10.2%). The convergence is asymmetric: models that suffered most in V1/V2 (GPT-5, and to a

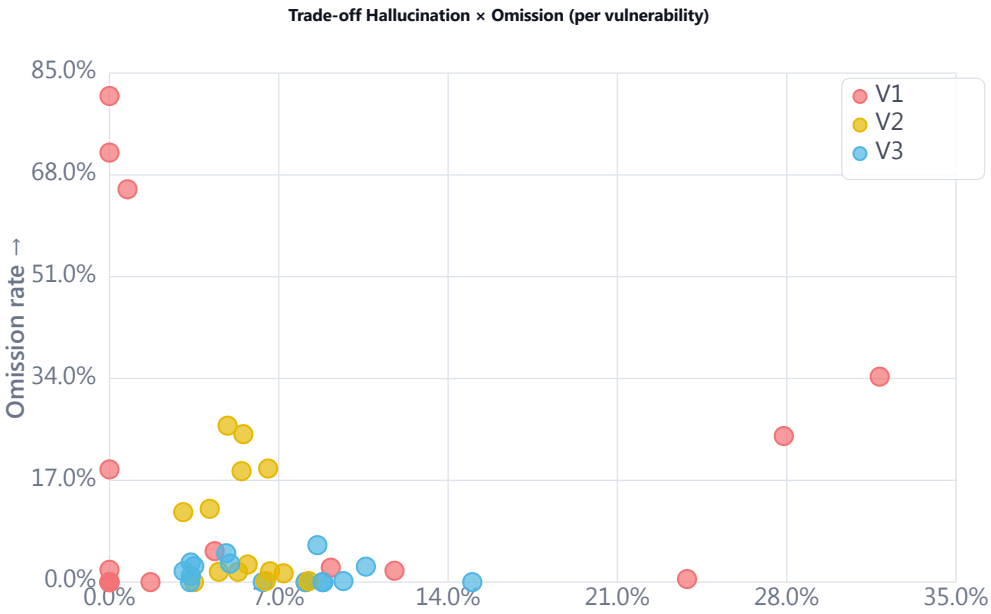

Figure 2. Hallucination versus omission rates at the vulnerability level

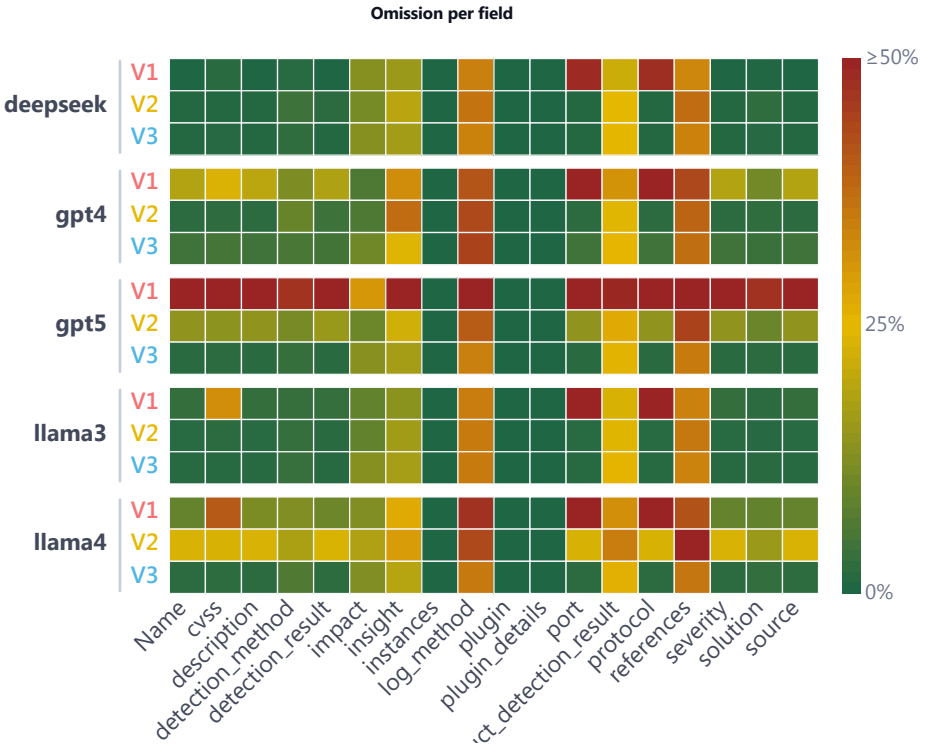

Figure 3. Field-level omission rate by (model, version, field)

lesser extent LLaMA 4) are precisely those that improve most in V3, while models that already performed well (DeepSeek, LLaMA 3) remain near their V2 levels. This pattern is consistent with the V3 prompt's few-shot examples acting as a leveler: weaker models benefit disproportionately from being shown explicit input-output patterns.

4.6. Similarity Distribution

Table 4 summarizes the BERTScore F1 similarity distributions (Highly Similar and Absent categories) across three baselines of increasing size: JuiceShop (34 vulnerabilities), bBWA (58), and Artifactory (116). Figures 4 and 5 show the full five-category breakdown for the smallest and largest baselines; the bBWA distribution (intermediate size) follows the same convergence pattern.

Table 4. BERTScore F1 similarity distribution: fraction of matched records in Highly Similar (≥ 0.7) and Absent (vulnerability omitted), averaged across the five LLMs.

Category	Baseline	Dir.	V1	V2	V3
Highly Similar	JuiceShop (34)	↑	65.1%	90.3%	**98.6%**
	bBWA (58)		79.5%	84.1%	**96.5%**
	Artifactory (116)		59.1%	61.9%	**92.2%**
Absent	JuiceShop (34)	↓	20.4%	9.5%	**1.2%**
	bBWA (58)		15.3%	6.3%	**0.8%**
	Artifactory (116)		25.9%	8.6%	**3.1%**

V1: coverage failure dominates. V1 is bimodal across all three baselines: most matched pairs reach Highly Similar while a large fraction is Absent (Table 4). The bulk of the Absent segment is driven by GPT-5, which omits 81%, 72%, and 66% of baseline vulnerabilities on JuiceShop, bBWA, and Artifactory respectively. The remaining models behave inconsistently across targets: GPT-4 and LLaMA 4 show elevated Absent rates on Artifactory (34% and 24%), and LLaMA 3 on JuiceShop produces a Divergent share of 25% despite extracting most records. V1's dominant failure mode is nevertheless recall: across all (model, baseline) combinations, Absent contributes substantially more than Divergent to the non-Highly-Similar fraction.

V2: coverage recovered, quality partially. V2 collapses the Absent segment across all targets, confirming that the marker-aware chunking and retry changes recover most previously-missed vulnerabilities. On the smaller baselines (JuiceShop, bBWA), similarity rises sharply. On Artifactory, however, Highly Similar barely moves while the Moderately Similar band grows to 21.4% (visible in Figure 5): V2 recovers the missing vulnerabilities but the additional content is extracted with mid-tier quality, exposing a coverage-quality tradeoff that the aggregate ERM metric obscures.

V3: convergence across baselines. V3 closes both gaps simultaneously. The improvement on Artifactory is the most informative: Highly Similar rises by 30 percentage points while the Moderately Similar band collapses from 21.4% to 4.4% (Figure 5), showing that the V3 chunking and prompt changes recover not just *more* vulnerabilities but also *better-extracted* ones. The convergence holds across all five LLMs and is consistent across baselines: by V3, all three reach Highly Similar $\geq 92\%$ with Absent $\leq 3\%$ (Table 4).

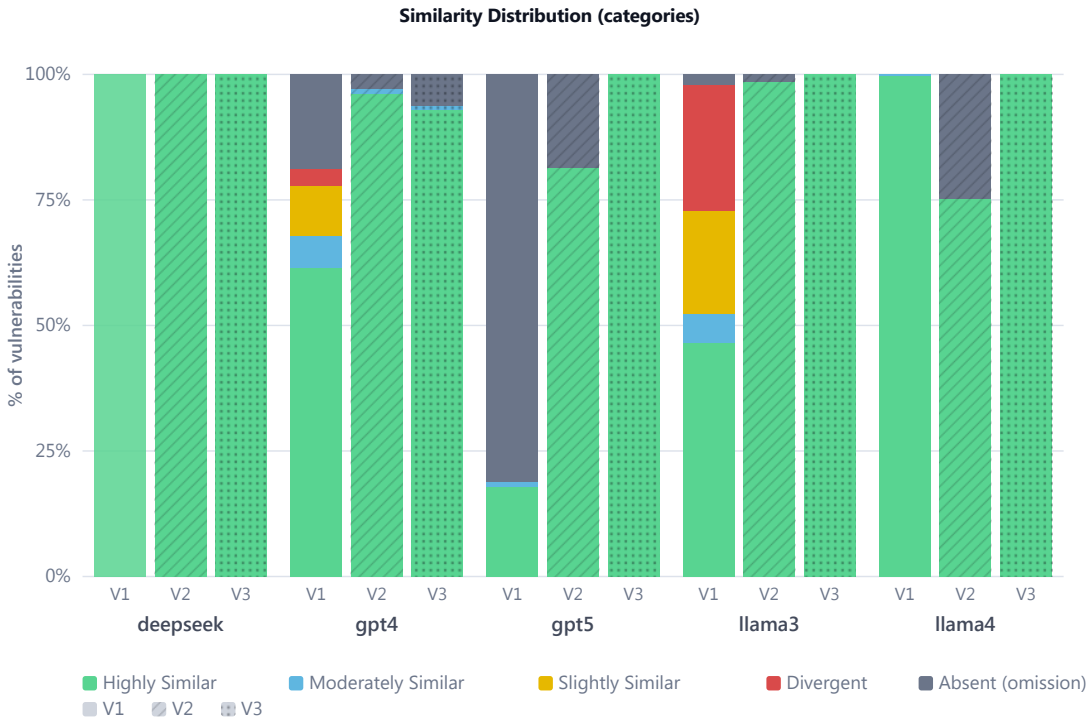

Figure 4. BERTScore F1 similarity distribution (JuiceShop)

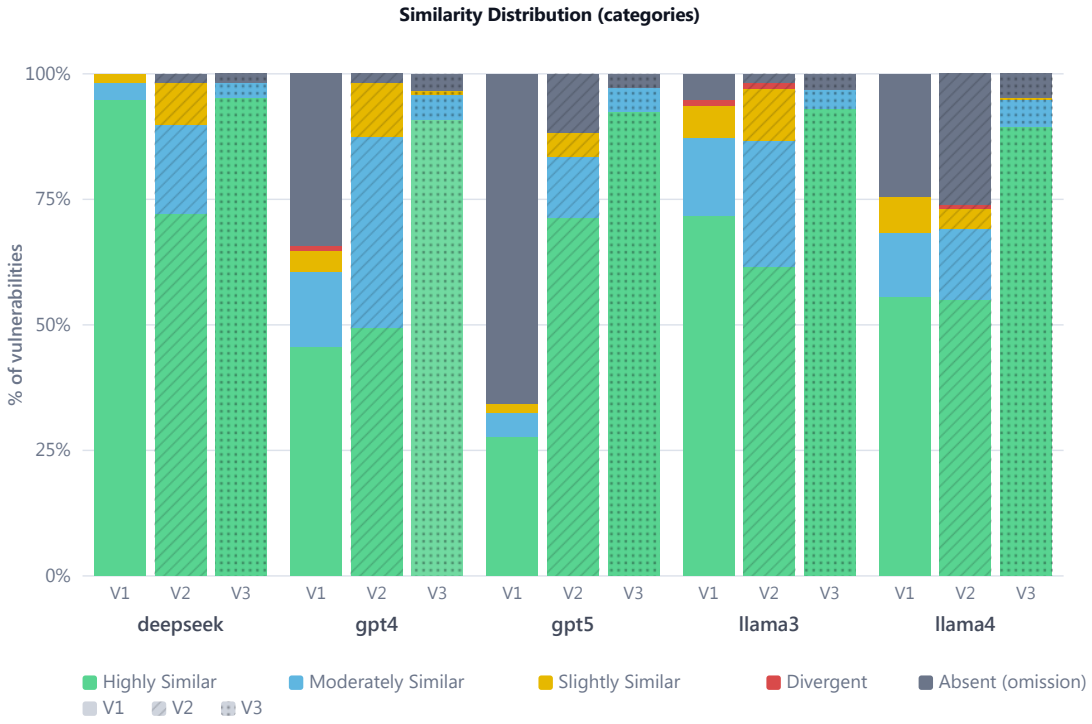

Figure 5. BERTScore F1 similarity distribution (Artifactory)

4.7. Limitations

Single scanner. All baselines are generated by OpenVAS. MulitaMiner supports Tenable WAS and the pipeline has been validated against it, but Tenable WAS is a paid commercial product with restricted access, while OpenVAS is open-source. The volume of Tenable WAS test data obtainable for this study was therefore substantially smaller, and including it in the main evaluation would have produced statistically asymmetric comparisons.

Number of baselines. The evaluation uses three baselines (Artifactory, JuiceShop, bBWA) totaling 208 vulnerabilities. Although they span three application categories and density points (34, 58, 116 vulnerabilities), three baselines limit cross-target generalization; we mitigate this with 95% bootstrap confidence intervals over $N{=}150$ per-version observations, treating non-overlapping CIs as evidence of signal above run-level noise. Each baseline was curated following the two-stage extract-then-review protocol described in Section 4.1, which ensures ground-truth consistency but remains subject to human error.

Ground-truth reliability. The extract-then-review protocol described in Section 4.1 produces a single converged baseline rather than two parallel annotations, so we report no inter-annotator agreement statistic and the reliability of the ground truth is not quantified. The protocol also carries an anchoring risk: the reviewing specialist starts from the extraction produced by the first annotator, and a vulnerability that both overlook is not recoverable by the procedure. Reviewing against the source PDF rather than against the extraction alone mitigates this risk but does not eliminate it. A parallel double annotation on at least one target, with the agreement statistic reported, would place the baselines on firmer ground and is left for future work.

Cloud LLMs only. The five evaluated backends are cloud-hosted commercial models executable in parallel. Local execution via Ollama, Hugging Face, and LM Studio is supported by V3 and was validated end-to-end on representative inputs, confirming pipeline portability; full benchmark runs (150 per model) are deferred due to sequential-execution constraints on the available hardware.

Persistent fields. The heatmap (Figure 3) shows that fields such as detection method, log method, product detection result, and references continue to exhibit non-trivial omission rates (10–25%) across most models, even under V3. Although V3 substantially reduces omissions overall, these fields remain comparatively difficult to extract, suggesting that further improvements will require additional mechanisms beyond those investigated in this work.

5. Conclusion and Future Work

Across 450 independent runs spanning three pipeline iterations and five LLMs, the evidence is consistent in one direction: pipeline engineering, not model substitution, is the primary lever for extraction quality on the targets evaluated. The most striking case is GPT-5, catastrophic under V1 (66–81% omission across baselines) and fully recovered under V3 through adaptive per-LLM chunking, a redesigned extraction prompt, and a specialized tokenizer.

BERTScore and the hallucination-omission scatter corroborate this at complementary granularities: by V3, all five models converge into a compact low-omission,

low-hallucination region regardless of report size. Severity classification was the least pipeline-sensitive dimension (Macro F1 stable at 94.8–97.7% across versions). For security teams facing 131 new CVEs per day, the practical implication is direct: once the pipeline absorbs most of the variance across models, the dominant basis for selection shifts to operational factors such as cost, latency, and data-residency policy. This conclusion rests on three OpenVAS baselines and five cloud models, and its extension to other scanners, to larger report populations, and to locally executed models remains to be tested.

Future work targets the limitations of Section 4.7: extending the evaluation to Tenable WAS and to further scanners (Nessus, Nuclei); enlarging the set of curated baselines in number and in application category; completing the 150-run protocol for locally deployed models, a prerequisite for air-gapped deployments; a V4 iteration addressing chronic omission through field-aware re-extraction; and reporting processing time and token cost per model.

Acknowledgments. This work was partially supported by Capes, CNPq (Grant No. 409743/2025-9), and FAPERGS (Grant Nos. 24/2551-0001368-7 and 24/2551-0000726-1). The authors also acknowledge the use of Generative AI as a supporting tool for reviewing and improving the code, manuscript, and figures.

References

Brokman, J. et al. (2024). Insights and current gaps in open-source LLM vulnerability scanners: A comparative analysis. Affiliated with the Institute of Cybersecurity and Artificial Intelligence, Ben-Gurion University of the Negev, Israel.

Croft, R., Babar, M. A., and Kholoosi, M. M. (2023). Data quality for software vulnerability datasets. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 121–133. IEEE.

Dabholkar, C. S. (2023). Integrating open-source vulnerability scanning tools reports with openai api for automated report generation. Master's thesis, National College of Ireland, Dublin, Ireland.

Dong, Y., Chattopadhyay, S., Aung, Y. L., and Zhou, J. (2025). Chatiot: Large language model-based security assistant for internet of things with retrieval-augmented generation. *arXiv preprint arXiv:2502.09896*.

Dunn, A., Dagdelen, J., Walker, N., Lee, S., Rosen, A. S., Ceder, G., Persson, K. A., and Jain, A. (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, 15:1418.

Ferrag, M. A., Al-Hawawreh, M., Battah, A., et al. (2025). Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities. arXiv preprint arXiv:2405.12750.

Foreman, P. (2019). *Vulnerability Management*. Auerbach Publications.

Gao, P., Liu, X., Choi, E., Ma, S., Yang, X., and Song, D. (2022). Threatkg: An ai-powered system for automated open-source cyber threat intelligence gathering and management. *arXiv preprint arXiv:2212.10388*.

Garcia, A. M., Beunza, J.-J., et al. (2024). Enhanced medical data extraction: Leveraging llms for accurate retrieval of patient information from medical reports. *Preprints.org*.

Ghane, S., Sawant, R., et al. (2024). Langchainiq: Intelligent content and query processing. *International Journal of Management, Technology, and Social Sciences (IJMTS)*, 9(3):34–43.

Guo, Y., Bettaieb, S., and Casino, F. (2024). A comprehensive analysis on software vulnerability detection datasets: trends, challenges, and road ahead. *International Journal of Information Security*, 23(5):3311–3327.

Hu, Y., Sun, D., and Liu, P. (2024). Adapting generative large language models for information extraction. *Journal of Data Intelligence*.

Klusty, M. A., Solie, E. C., Leach, C. N., and Logan, W. V. (2024). Leveraging LLMs for structured data extraction from unstructured patient notes. arXiv preprint arXiv:2512.13700.

Lin, D. (2024). Revolutionizing retrieval-augmented generation with enhanced pdf structure recognition. *chatdoc.com Technical Report*.

Machado, B., Lautert, D., Kapelinski, C., and Kreutz, D. (2025). Structured extraction of vulnerabilities in OpenVAS and Tenable WAS reports using LLMs. In *Anais da XXII Escola Regional de Redes de Computadores (ERRC)*, pages 144–150, Porto Alegre. Sociedade Brasileira de Computação.

Mitchell, E., Are, E. B., Colijn, C., and Earn, D. J. D. (2025). Using artificial intelligence tools to automate data extraction for living evidence syntheses. *PLoS One*, 20(4):e0320151.

NIST (2026). NIST updates NVD operations to address record CVE growth. `https://www.nist.gov/news-events/news/2026/04/nist-updates-nvd-operations-address-record-cve-growth`.

Park, Y. and Lee, T. (2022). Full-stack information extraction system for cybersecurity intelligence. In *Proceedings of EMNLP 2022 Industry Track*, pages 531–539.

Ponce, L. M., Ribeiro, I., Oliveira, E., Ítalo Cunha, Hoepers, C., Steding-Jessen, K., Chaves, M. H. P. C., Guedes, D., and Jr., W. M. (2024). Identificação de serviços e dispositivos em dados de motores de busca para o enriquecimento de análise de vulnerabilidades. In *Anais do XXIV SBSeg*, pages 367–382. SBC.

Rafati, R. (2025). Why relying solely on automated vulnerability scanners is risky for CVE detection. Online. Discusses limitations of automated scanners and the need for manual validation and expert review.

Sharmila, S. P. (2026). PDFInspect: A unified feature extraction framework for malicious document detection.

The Sequence (2025). The vulnerability data crisis: Why you can't trust your security tools (and what to do about it). `https://the-sequence.com/the-vulnerability-data-crisis`. Accessed: 2025.

Wang, B. et al. (2024). Docllm: A lightweight extension for visually rich document understanding. *Proceedings of the 62nd ACL 2024*.

Zhang, X., Luo, S., et al. (2024). Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *arXiv preprint arXiv:2403.19318v3*.