Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Secure Domain Adaptation and Retrieval Augmented Generation for Compact Language Models Using Archived Cybersecurity Intelligence

Syrus Redhouse
Texas A&M University – San Antonio
Izzat Alsmadi
ialsmadi@tamusa.edu

Texas A&M University – San Antonio
Anas AlSobeh
Utah Valley University

Research Article

Keywords: Large language models, Retrieval-augmented generation, Domain-adaptive pretraining, Cyber-threat intelligence, Malware detection, Empirical evaluation

Posted Date: November 4th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11209286/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Secure Domain Adaptation and Retrieval Augmented Generation for Compact Language Models Using Archived Cybersecurity Intelligence

Syrus Redhouse[1], Izzat Alsmadi[1*], Anas AlSobeh[2]

[1]Department of Computing and Cyber Security, Texas A&M University–San Antonio, One University Way, San Antonio, 78224, TX, USA.
[2]Department of Information Systems and Technology, Utah Valley University, 800 W University Parkway, Orem, 84058, UT, USA.

*Corresponding author(s). E-mail(s): ialsmadi@tamusa.edu;
Contributing authors: sredhouse@tamusa.edu; anas.alsobeh@uvu.edu;

Abstract

Security analysts must reconcile vulnerability, exploit and malware records that are scattered across public repositories, yet it is unclear whether compact, locally deployable language models can reason over such archives. We present SecureLLM, an open framework that normalizes 161,190 records from the National Vulnerability Database, the Known Exploited Vulnerabilities catalog, ExploitDB and MalwareBazaar into one corpus, which we use both to continue pretraining Qwen2.5-1.5B-Instruct and to build a dense retrieval index. We compare the adapted model with its base and three other compact models on 1,000 stratified security prompts, with and without retrieval (10,000 responses), using paired non-parametric tests. Continued pretraining lowered validation loss by 12% during training but left cybersecurity-term coverage in closed-book answers statistically unchanged (Holm-adjusted p = 0.062). With retrieval, the adapted model's answers agreed less with the retrieved context than those of its base (cosine similarity 0.41 versus 0.46; adjusted p ¡ 0.001). On a balanced 10,000-sample EMBER malware test set, gradient-boosted trees reached 0.856 accuracy, whereas the zero-shot language models reached 0.646 (base) and 0.630 (adapted), with precision near 0.91 but recall of only 0.32 and 0.29. In a 100-sample pilot, adding retrieved-neighbor statistics to the prompt made the model label every sample benign. We trace these outcomes to four causes: a mismatch between raw-text pretraining and instruction-style use, a mismatch between the prompt set and the corpus, encoder truncation, and uninformative retrieval neighborhoods. We release the pipeline and propose a leakage-controlled evaluation protocol for cyber-intelligence language models.

Keywords: Large language models, Retrieval-augmented generation, Domain-adaptive pretraining, Cyber-threat intelligence, Malware detection, Empirical evaluation

1 Introduction

Cybersecurity knowledge is spread across public repositories that were never designed to work together. The National Vulnerability Database (NVD) publishes Common Vulnerabilities and Exposures (CVE) records with severity scores and weakness classes [1]. The Cybersecurity and

Infrastructure Security Agency (CISA) maintains the Known Exploited Vulnerabilities (KEV) catalog of CVEs that are exploited in the wild [2]. ExploitDB archives public proof-of-concept exploits [3], and MalwareBazaar shares malware samples together with their metadata [4]. Each source uses its own schema, identifiers and update cadence, so an analyst who asks whether a weakness is exploited, has public exploit code or is associated with a malware family must cross-reference several sources by hand.

Large language models (LLMs) are an obvious candidate for this integration task. They generalize across tasks from few examples [5], and security applications now span vulnerability detection, penetration testing and threat-intelligence analysis [6–8]. Three obstacles remain, however. First, identifier-dense security records are rare in general pretraining corpora, so general models handle CVE identifiers, exploit metadata and malware naming conventions poorly. Second, security data are often sensitive, which favors compact models that can run on premises over hosted frontier models. Third, fluent but unsupported output (hallucination) is particularly costly in security decisions [9], and even strong models reason unreliably about vulnerabilities [10]. The two standard remedies are domain-adaptive pretraining, which continues training on in-domain text [11, 12], and retrieval-augmented generation (RAG), which conditions generation on documents retrieved from an external index [13, 14].

Whether these remedies help *compact* models (1–4 billion parameters) over *archived, multi-source* intelligence has not been established. Recent security-specialized models adapt 7–8B-parameter models using curated corpora, instruction data or reasoning distillation [15–17], and most evaluations score multiple-choice knowledge benchmarks [8, 18, 19]. Few studies release a pipeline that unifies vulnerability, exploit and malware records. Few compare an LLM with a conventional detector on the same labeled data, and few report null or negative outcomes with the paired statistics that the security machine-learning literature recommends [20].

This paper addresses that gap with *SecureLLM*, an open framework and empirical study. We normalize 161,190 records from four public sources into one corpus. We use the corpus twice: to continue pretraining Qwen2.5-1.5B-Instruct [21], and to build a dense retrieval index. We then evaluate the resulting models against a conventional malware classifier and other compact models. Four research questions (RQs) guide the study:

RQ1 Does continued causal language-model pretraining on archived cybersecurity intelligence change how a compact instruction-tuned model answers security questions?

RQ2 Does retrieval augmentation over the same archive increase agreement between generated answers and archived evidence?

RQ3 Can heterogeneous public intelligence sources be unified into a single corpus and index that supports both adaptation and retrieval at scale?

RQ4 How do the base and adapted models compare with a gradient-boosted classifier on downstream malware classification, and does retrieved evidence help?

The paper makes four contributions:

1. **A reproducible multi-source pipeline.** We release the crawlers, normalization schema, corpus builder, chunker, index builder and training script that turn four public sources into a 161,190-record corpus and a 164,419-vector index. All parameters are documented (RQ3).
2. **A controlled, statistically tested comparison.** We evaluate the adapted model, its base and three other compact open models on 1,000 stratified prompts, in closed-book and retrieval-augmented modes (10,000 responses). We use precisely defined automated metrics, paired Wilcoxon and McNemar tests, and Holm correction (RQ1, RQ2).
3. **A downstream security evaluation.** We compare XGBoost with the base and adapted models on a balanced 10,000-sample EMBER test set and report full confusion matrices and bootstrap confidence intervals. We complement this with a retrieval case study over 1,039,867 executables and a pilot of retrieval-augmented classification (RQ4).
4. **A diagnosis and an evaluation protocol.** We identify four mechanisms behind the negative results. From them we derive design recommendations and a leakage-controlled,

per-source protocol for evaluating cyber-intelligence language models against established tools.

The results are largely negative, and we report them as such. Continued pretraining lowered validation loss by 12% but did not change closed-book answers measurably. Retrieval did not raise answer–evidence agreement and lowered it for the adapted model. The language models trailed XGBoost by 21–23 accuracy points on EMBER. Retrieved neighbor evidence caused the pilot classifier to label every sample benign. These outcomes are informative, because they show where naive domain adaptation and retrieval fail for compact models over security archives, and why.

The paper is organized as follows. Section 2 reviews related work. Section 3 describes the framework, and Section 4 the experimental design. Section 5 reports the results, and Section 6 discusses them. Section 7 analyzes threats to validity, Section 8 addresses ethical considerations, and Section 9 concludes.

2 Background and related work

2.1 Language models for cybersecurity

Domain-specific language models for security began with encoders. SecureBERT continued masked-language-model pretraining of RoBERTa on security text and improved downstream security text-understanding tasks [12]. Decoder models followed the same recipe at larger scale. Foundation-Sec-8B continues pretraining Llama 3.1 8B on a curated security web corpus [15]. Primus releases open pretraining, instruction and reasoning datasets for security and applies them to Llama-3.1-8B-Instruct [16]. SEvenLLM builds an instruction corpus for threat-intelligence tasks [22]. Evaluation has moved in parallel: CTIBench [8], CyberMetric [18] and CyberSecEval 2 [19] measure knowledge, reasoning and misuse risk, and a systematic review maps the rapidly growing set of security applications of LLMs [6]. Studies of vulnerability reasoning caution that LLM answers are often non-deterministic and unfaithful [10], whereas agentic systems such as PentestGPT show value when a model is embedded in a tool loop [7].

For source code, MultiCode learns joint syntactic and dependency representations for code analysis [23]. VulnLLM-R distills reasoning traces from large teacher models into a 7B-parameter Qwen2.5-7B-Instruct model and wraps it in an agent that retrieves program context with CodeQL; it outperforms static analyzers on function- and project-level vulnerability detection [17]. VulnLLM-R shares SecureLLM's model family but differs in task (code rather than intelligence records), in adaptation (supervised reasoning distillation rather than raw continued pretraining) and in the knowledge it retrieves (project-local code rather than archived multi-source intelligence).

Our group's earlier work examined customized LLMs and contextual embeddings for spam detection [24, 25], transformer ensembles for security-relevant text classification [26], and the security risks of LLM-based systems, including role-based prompt injection, dependency hallucination and toxic flows in agentic pipelines [27, 28]. These risks motivate outputs that are grounded in verifiable sources, which is the premise that this study tests.

2.2 Domain-adaptive and continual pretraining

Continued pretraining on in-domain text improves downstream performance even for large pretrained models [11]. It has costs, however. Continual pretraining without learning-rate re-warming or replay of general data can degrade previously acquired capabilities [29], and continual fine-tuning is prone to catastrophic forgetting [30]. For an instruction-tuned starting point, raw-text continued pretraining can therefore dilute instruction-following behavior, even while in-domain perplexity improves. Our institutional domain-adaptation studies combined supervised fine-tuning on instruction-formatted data with RAG, which sidesteps this risk [31, 32]. SecureLLM deliberately tests the simpler, cheaper raw-text variant. Evaluations must also control for contamination, because benchmark or corpus content may already appear in a model's pretraining data [33].

2.3 Retrieval-augmented generation and dense retrieval

RAG combines the parametric memory of a generator with a non-parametric index [13]. REALM trains the retriever and language model jointly [14], and augmented language models generalize this idea to tools [34]. Dense passage retrieval outperforms BM25 [35] on open-domain question answering [36]. Sentence-BERT [37] and distilled encoders such as MiniLM [38] make dense retrieval inexpensive, and FAISS supports exact and approximate search at scale [39]. Surveys document the many design choices in RAG pipelines [40]. Models also use long contexts unevenly [41], and reference-free RAG metrics measure faithfulness and context relevance only indirectly [42]. Retrieval reduces hallucination only when the retrieved evidence is relevant, which is the condition that Section 6.2 examines.

2.4 Machine learning for static malware detection

EMBER provides static features of Windows portable executable (PE) files together with a gradient-boosted baseline [43]. Deep models over engineered features [44] or raw bytes [45] followed, and gradient-boosted trees such as XGBoost [46] remain strong baselines. Methodological guidance warns against pitfalls such as sampling bias, spurious correlations and inappropriate baselines [20], and against temporal and spatial bias in malware evaluation [47]. Time-aware learning addresses concept drift in Android malware and cloud intrusion detection [48, 49], and supervised contrastive embeddings illustrate how task-specific representation learning can outperform generic encoders [50]. We adopt these principles by comparing LLMs with a conventional detector on identical data, a comparison that is rarely reported for LLM-based malware triage.

2.5 Positioning

Table 1 contrasts SecureLLM with the closest systems. Unlike prior security LLMs, SecureLLM combines (i) a compact 1.5B-parameter model that is practical on premises, (ii) continued pretraining and dense retrieval over the *same* multi-source archive of vulnerability, exploit and malware records, and (iii) a head-to-head comparison with a conventional malware classifier. The study's value lies less in a new state-of-the-art result than in a transparent, statistically tested account of what this common and inexpensive recipe does and does not deliver.

3 The SecureLLM framework

Figure 1 summarizes the framework. Per-source crawlers acquire and clean records (lane 1) and a normalizer merges them into one corpus (lane 2). The corpus then feeds two independent branches: continued pretraining produces the SecureLLM checkpoint (lane 3a), and chunking and embedding produce a FAISS index (lane 3b). At inference time, a query is answered either closed-book or with retrieved chunks prepended to the prompt (lane 3c). Four experiments evaluate the outputs (lane 4). Each component is a stand-alone script, so any stage can be replaced without changing the others [51].

3.1 Data acquisition and normalization

Table 2 lists the four sources. NVD records were retrieved through the CVE API 2.0 [1] by publication date, with a page size of 200. Retrieval covered three blocks: 1 January–29 April 2024, 1 January–30 April 2025 and 1 January–23 March 2026. Each block corresponds to one query window of at most 120 days, the maximum range that the API accepts. The NVD component is therefore a recent, seasonally restricted slice, not a census of the catalog (Section 7). CISA KEV entries were collected through the catalog feed [2], the ExploitDB repository was cloned locally [3], and MalwareBazaar metadata were read from its full CSV export [4]. The MalwareBazaar component comprises the 89,090 rows processed from that export, a subset of the export, and the processed row range is recorded in the ingestion manifests. No malware binaries were downloaded.

Each crawler writes a cleaned text body and a JSON metadata file per record. The corpus builder then emits a uniform document consisting of a structured header followed by the cleaned content.

Table 1 Positioning of SecureLLM against representative security-specialized language models

System	Base model (size)	Adaptation	Retrieval	Security knowledge	Evaluation focus
SecureBERT [12]	RoBERTa (125M)	Masked-LM continued pretraining	None	Security reports and articles	Security text classification, entity recognition
Foundation-Sec-8B [15]	Llama 3.1 (8B)	Continued pretraining	None	Curated security web corpus	Security knowledge and CTI benchmarks
Primus [16]	Llama-3.1-Instruct (8B)	Continued pretraining, SFT, reasoning distillation	None	Open security corpora	Security knowledge benchmarks
VulnLLM-R [17]	Qwen2.5-Instruct (7B)	Reasoning distillation (SFT)	CodeQL code context (agent)	Labeled vulnerable and benign functions	Code vulnerability detection vs. static analyzers
SecureLLM (ours)	Qwen2.5-Instruct (1.5B)	Full-parameter continued causal-LM pretraining	Dense top-3 (MiniLM + FAISS)	NVD, CISA KEV, ExploitDB, MalwareBazaar records	Five-model benchmark with paired tests; EMBER vs. XGBoost

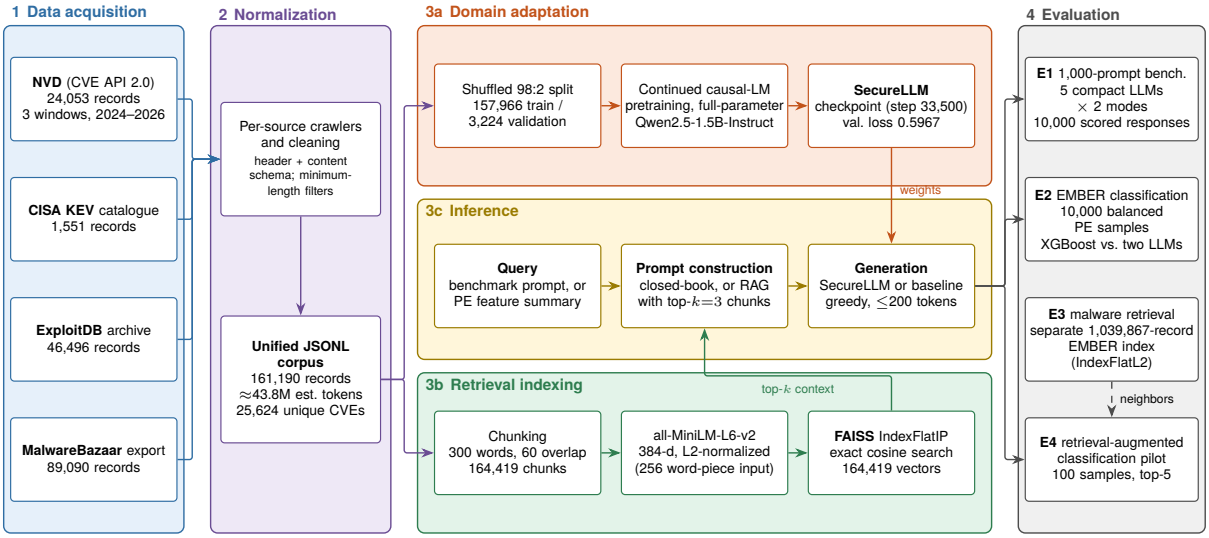


Fig. 1 SecureLLM architecture. Four public sources are crawled and normalized into a unified JSONL corpus (lanes 1–2). The corpus is used twice: for full-parameter continued pretraining of Qwen2.5-1.5B-Instruct (lane 3a) and for a 164,419-vector FAISS index built with a 384-dimensional MiniLM encoder (lane 3b). Queries are answered closed-book or with the top-3 retrieved chunks prepended (lane 3c). Experiments E1–E4 evaluate the system (lane 4). E3 and E4 use a separate index of 1,039,867 EMBER feature summaries rather than the corpus index.

The header contains the source, record identifier, title, dates, tags and source-specific fields: CVE identifier, severity, Common Vulnerability Scoring System (CVSS) score and Common Weakness Enumeration (CWE) class for NVD and KEV; exploit identifier and platform for ExploitDB; and hashes, file type and signature for MalwareBazaar. Records whose cleaned body fell below a per-source minimum (20 words for NVD, 8 for the other sources) were discarded. ExploitDB bodies were capped at 20,000 characters, and the CVE identifiers that they mention were extracted by regular expression from the title, path and the first 8,000 characters of the exploit text. These identifiers appear in the text of each ExploitDB document, but the released builder does not copy them into a structured `cve_id` field. The corpus therefore contains the identifiers but lacks a join key for programmatic ExploitDB-to-CVE linkage; the ingestion code has since been corrected, but the corrected mapping is not used in the reported results.

The final corpus contains 161,190 records, approximately 43.8 million estimated tokens (mean 271.8 tokens per record), about 1.09 million distinct regular-expression word forms and 25,624 unique CVE identifiers. MalwareBazaar and ExploitDB metadata account for 84% of the records (Table 2). This skew toward identifier-dense text matters for both adaptation and retrieval (Section 6).

3.2 Domain-adaptive continued pretraining

We continued pretraining Qwen2.5-1.5B-Instruct [21] on the corpus with a causal (next-token) language-modeling objective, using the Hugging Face Trainer [52]. Adaptation was full-parameter; no parameter-efficient adapters were used. The corpus was shuffled with seed 42 and split 98:2 into 157,966 training and 3,224 validation records. Table 3 lists the remaining settings. Because the objective is next-token prediction on raw records rather than on instruction–response pairs, this procedure is continued pretraining, not instruction tuning. Section 6.1 discusses the consequences of this choice.

Training ran on one NVIDIA A100-PCIE-40GB GPU for approximately 10.4 h. Early stopping (patience of two evaluations, one evaluation every 500 steps) ended training at step 34,500, that is, after 1.75 of the 3 planned epochs. The checkpoint with the lowest validation loss was restored. Figure 2 shows that validation loss fell monotonically from 0.680 at the first evaluation to a minimum of 0.5967 at step 33,500; this minimum corresponds to a validation perplexity of 1.82 and determined the selected checkpoint. The trainer-reported mean training loss over the run was 0.695, and the last logged training loss was 0.672. The base model's validation loss before adaptation was not logged. The curve therefore shows that the model learned the corpus distribution during training; it neither quantifies the gain over the base model nor implies better downstream behavior.

3.3 Retrieval index and inference

For retrieval, each corpus document was split into 300-word windows with a 60-word overlap. Most documents fit in one window, so 161,190 records yielded 164,419 chunks. Each chunk was embedded with the sentence-transformers model all-MiniLM-L6-v2 [37, 38], which produces 384-dimensional, L2-normalized vectors. The vectors were stored in a FAISS `IndexFlatIP` index [39], so the search is exact cosine search rather than an approximate one. This encoder truncates inputs at 256 word pieces, whereas a full 300-word chunk tokenizes to more than 256 word pieces. For such chunks the tail is not represented in the embedding, and the leading header fields dominate the vector. We return to this limitation in Sections 6.2 and 7.

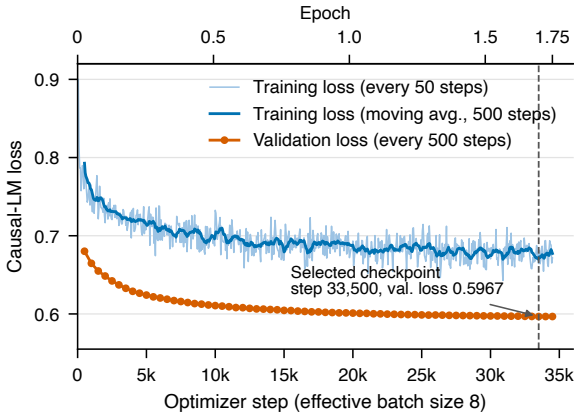


Fig. 2 Training and validation loss during continued pretraining. The dashed line marks the selected checkpoint (step 33,500; validation loss 0.5967). The upper axis converts steps to epochs (19,746 steps per epoch at an effective batch size of 8).

At inference, the prompt is embedded with the same encoder, and the top $k = 3$ chunks are retrieved and prepended to the question under a system instruction to answer only from the context and to state uncertainty when the context is insufficient. In closed-book mode, the system instruction asks for a clear, direct answer without invented facts. Appendix A gives both instructions. All models used the same decoding configuration (Table 3). The repetition penalty was raised from 1.1 to 1.3, and 4-gram blocking was added, after pilot runs in which one baseline produced looping output. Section 7 discusses the consequences of this change.

4 Experimental design

Four experiments address the research questions. E1 is a 1,000-prompt benchmark for RQ1 and

Table 2 Sources of the SecureLLM corpus

Source	Records	Share (%)	Temporal coverage	Min. words	Normalized content
NVD	24,053	14.9	Three windows: Jan–Apr 2024, Jan–Apr 2025, Jan–Mar 2026	20	CVE ID, description, CVSS severity and score, CWE, references
CISA KEV	1,551	1.0	Full catalog at collection	8	CVE ID, vendor, product, vulnerability name, required action
ExploitDB	46,496	28.8	Full repository at collection	8	Exploit title, platform, type, CVE mentions, PoC text (cap 20,000 chars)
MalwareBazaar	89,090	55.3	Rows processed from the full CSV export	8	SHA-256, MD5, imphash, ssdeep and TLSH hashes; file type; signature; tags
Total	**161,190**	100.0			43.8M estimated tokens; 25,624 unique CVE IDs

Table 3 Configuration of adaptation, retrieval and generation

Setting	Value
Continued pretraining	
Base model	Qwen2.5-1.5B-Instruct (1,543,714,304 parameters)
Objective, method	Causal LM; full-parameter
Sequence length	1,024 tokens (padded; padding masked from the loss)
Batch size	1 per device × 8 accumulation steps = 8
Learning rate, schedule	2×10^{-5}; linear decay, no warm-up
Precision, clipping	bfloat16; gradient norm 1.0
Early stopping	Validation loss every 500 steps; patience 2
Seed, split	42; 98:2 train/validation
Retrieval index	
Chunking	300-word windows, 60-word overlap (164,419 chunks)
Encoder	all-MiniLM-L6-v2; 384-d; L2-normalized; 256 word-piece input limit
Index, k	FAISS `IndexFlatIP` (exact cosine); $k = 3$
Generation (all models, both modes)	
Decoding	Greedy; at most 200 new tokens
Repetition control	Repetition penalty 1.3; no repeated 4-grams
Precision	bfloat16 on one A100 GPU

RQ2. E2 is a malware classification experiment on EMBER, and E3 is a malware retrieval case study; both address RQ4. E4 is a pilot of retrieval-augmented classification that addresses RQ2 and RQ4. RQ3 is answered by the construction and statistics reported in Section 3.

4.1 E1: benchmark prompts and models

The benchmark contains 1,000 natural-language security questions (mean length 12.3 words) in ten equal strata of 100: fundamentals; vulnerability management; malware analysis; network security; incident response; threat intelligence; cloud and identity; web security; security operations; and advanced mixed reasoning. The released prompt file contains exactly 1,000 non-empty prompts with no exact or normalized duplicates. All prompts are conceptual or scenario questions, for example "Why do malware authors keep creating new variants of existing malware families?". None contains a CVE identifier, file hash or other corpus record key. Appendix A lists one example per stratum.

We compared SecureLLM with its base, Qwen2.5-1.5B-Instruct [21], and with three widely used compact instruction-tuned models: Llama-3.2-1B-Instruct (1.2B parameters) [53], Gemma-2-2B-it (2.6B) [54] and Phi-3.5-mini-instruct (3.8B) [55]. Each model answered every prompt in closed-book and in RAG mode, which gives $5 \times 2 \times 1{,}000 = 10{,}000$ scored responses. The primary comparison is SecureLLM versus its own base, which isolates the effect of continued pretraining. The other three models indicate how much compact models vary among themselves.

4.2 Automated metrics

Let a be a generated answer, $\ell(a)$ its lowercased text, $|a|$ its number of whitespace-separated words, and $C = c_1 \oplus c_2 \oplus c_3$ the concatenation of the retrieved chunks. Let ϕ be the normalized MiniLM embedding. The benchmark computes

five signals:

$$\mathrm{Cov}(a) = \textstyle\sum_{t\in\mathcal{T}} \mathbb{1}[\,t \sqsubseteq \ell(a)\,], \quad |\mathcal{T}| = 15, \tag{1}$$

$$\mathrm{Dens}(a) = \mathrm{Cov}(a)/|a|, \tag{2}$$

$$\mathrm{Sim}(a) = \cos\big(\phi(a), \phi(C)\big), \tag{3}$$

$$\mathrm{Flag}(a) = \mathbb{1}[\,\mathrm{Sim}(a) < 0.5 \wedge \neg\mathrm{Ref}(a)\,], \tag{4}$$

$$\mathrm{Unc}(a) = \textstyle\sum_{u\in\mathcal{U}} \mathbb{1}[\,u \sqsubseteq \ell(a)\,], \quad |\mathcal{U}| = 6. \tag{5}$$

Here $t \sqsubseteq s$ denotes substring containment. $\mathcal{T}$ is a 15-term cybersecurity lexicon, $\mathcal{U}$ a six-term hedging lexicon, and $\mathrm{Ref}(a)$ is true if a contains one of eight explicit refusal phrases (Appendix A). Three properties of these definitions matter for interpretation. First, *cyber-term coverage* (Eq. 1) counts *distinct* lexicon terms, bounded by 15, rather than occurrences, so it measures topical breadth rather than intensity; unanchored matching also counts, for example, "patch" inside "dispatch". Second, *grounding similarity* (Eq. 3) measures embedding-level agreement between an answer and the retrieved evidence; it does not measure factual correctness. Third, the *hallucination-risk flag* (Eq. 4) is a similarity floor at an uncalibrated threshold of 0.5 that exempts only canonical refusals, so we report it as a *risk flag*, not as a hallucination rate. Human relevance judgments were not collected, so we do not report Recall@k or mean reciprocal rank. All five signals are proxies, and we interpret them as relative comparisons between models under identical conditions.

4.3 Statistical analysis

Because all models answered the same prompts, every comparison is paired by prompt. For each metric and mode, all ten model pairs were compared with two-sided Wilcoxon signed-rank tests [56] for continuous metrics and exact McNemar tests [57] for the binary flag. p-values were adjusted with the Holm procedure [58] across the ten comparisons, at $\alpha = 0.05$. Effect sizes are reported as Cohen's d_z, the mean paired difference divided by the standard deviation of the differences [59]. Model-level 95% confidence intervals (CIs) are t-based for means and Wilson score intervals for proportions [60]. We report the primary comparison in full; the other nine pairs entered the correction but are described only qualitatively.

4.4 E2–E4: downstream malware experiments

E2, classification. From an EMBER-derived dataset of static PE features [43], we drew a balanced training set of 10,000 samples and a balanced test set of 10,000 samples (5,000 benign and 5,000 malicious). An XGBoost classifier [46] was trained on the structured static features, which include entropy, import-table, section and byte-histogram statistics. The base and adapted language models were used zero-shot. Each received the same prompt, which contained five scalar features (entropy, file size, import count, section count and packer count) and asked for a JSON label (0 for benign, 1 for malicious). Generation was greedy with at most 32 new tokens in half precision. Labels that could not be parsed would have defaulted to benign; neither model produced a parse failure in 10,000 predictions. All three systems were scored on the same samples in the same order. We report accuracy, precision, recall, F_1, specificity and the Matthews correlation coefficient (MCC), with 95% nonparametric bootstrap CIs [61]. Balanced accuracy equals accuracy on this balanced set and is not repeated.

E3, malware retrieval. To test whether generic dense retrieval captures maliciousness, we embedded compact text summaries of 1,039,867 EMBER records (519,929 benign and 519,938 malicious; no unlabeled records) with the same MiniLM encoder and stored them in a FAISS `IndexFlatL2` index. A hand-specified query describing a packed, high-entropy executable (entropy 7.9; 220,000 bytes; 120 imports; 8 sections; one packer indicator) retrieved its five nearest neighbors. Because the encoder output is unit-normalized, a squared Euclidean distance d^2 corresponds to a cosine similarity of $1 - d^2/2$.

E4, retrieval-augmented classification pilot. For each of 100 held-out EMBER samples (63 benign, 37 malicious), the top-5 neighbors were retrieved from the E3 index and summarized as the malicious-neighbor ratio, the mean neighbor entropy, the mean neighbor import count and the mean similarity. These statistics were appended to the query features. Qwen2.5-1.5B-Instruct was asked for a JSON label, a confidence value and a short justification. The same 100 samples had earlier been classified with the feature-only prompt, which yields a paired comparison.

5 Results

5.1 E1: closed-book and retrieval-augmented benchmark (RQ1, RQ2)

Table 4 reports all metrics with 95% CIs, Table 5 gives the paired tests for the primary comparison, and Figure 3 visualizes the main quantities.

RQ1: closed-book behavior.

In closed-book mode, SecureLLM covered 2.114 distinct cybersecurity terms per response, compared with 2.216 for its base. The difference was small ($|d_z| = 0.073$) and not significant after correction ($p_{\text{Holm}} = 0.062$). The same holds for technical density (1.46 versus 1.52 per 100 words; $p_{\text{Holm}} = 0.127$) and for hedging (1.058 versus 1.053 terms). Continued pretraining therefore did not change closed-book answers measurably on these proxies, although validation loss on the corpus fell by 12%. Among the other models, Gemma-2-2B had the highest density (1.63) and the least hedging (0.897), and Llama-3.2-1B was the fastest (1.99 s). The spread between models (coverage 1.84–2.22) exceeds the effect of adaptation, so the choice of base model mattered more than domain adaptation.

RQ2: retrieval augmentation.

With retrieval, no model reached the 0.5 similarity threshold: mean grounding similarity ranged from 0.4065 to 0.4666, and 62–75% of responses were flagged (Figure 3b,c). Retrieval affected the models differently. Cyber-term coverage rose for Llama-3.2-1B (1.84 to 2.37) and Gemma-2-2B (2.17 to 2.36), with non-overlapping CIs, rose slightly for Qwen2.5 (2.22 to 2.31), and fell for Phi-3.5-mini (1.84 to 1.63) and SecureLLM (2.11 to 1.93). In the paired comparison (Table 5), the adapted model had lower coverage (-0.381; $p_{\text{Holm}} = 5.2 \times 10^{-13}$), lower grounding similarity (-0.0515; $p_{\text{Holm}} = 2.1 \times 10^{-25}$, $|d_z| = 0.373$) and a higher flag rate (75.4% versus 62.2%; $p_{\text{Holm}} = 3.4 \times 10^{-13}$) than its base. Under RAG, domain adaptation thus had a small-to-moderate *negative* effect. Because none of the 1,000 prompts is answerable from a specific corpus record (Section 4.1), these results show how the models used *topically related* evidence; they do not show whether retrieval improves factual accuracy. Section 6.2 analyzes this limitation.

Latency.

Retrieval added at most 0.3 s per response for the models run with fused attention. SecureLLM and its base, which share one architecture, had similar latencies (3.59 and 3.69 s closed-book). The Phi-3.5-mini RAG latency (24.0 s) reflects eager attention combined with long retrieved contexts, not a property of the model (Table 4).

5.2 E2: malware classification on EMBER (RQ4)

Table 6 and Figure 4 show a clear gap between the conventional classifier and the zero-shot language models. XGBoost reached 0.856 accuracy, 0.853 F_1, 0.713 MCC and a ROC-AUC of 0.918. The language models reached 0.646 (base) and 0.630 (adapted) accuracy and F_1 of 0.477 and 0.437. Both language models were conservative. They labeled only 1,766 and 1,573 of 10,000 samples as malicious, and they achieved high specificity (0.970 and 0.973) and precision (0.914 for both) but recall of only 0.323 and 0.287, so 68–71% of malicious samples were missed. Domain adaptation made this behavior slightly more pronounced: SecureLLM's recall was lower than its base's, with non-overlapping CIs (0.287 [0.275, 0.300] versus 0.323 [0.310, 0.336]), while precision was unchanged.

The LLMs' precision exceeds XGBoost's (0.870) with non-overlapping CIs. This comparison, however, contrasts operating points: XGBoost is evaluated at its default threshold of 0.5, where it trades precision for recall, and its ROC-AUC of 0.918 implies that raising the threshold would also raise its precision. The LLMs emit hard labels, so no comparable curve is available for them. The comparison is also asymmetric in information, because the LLMs saw five scalar features and XGBoost saw the full static representation. We therefore do *not* interpret the precision gap as evidence that the language models are better detectors (Section 6.3).

Table 4 Benchmark results on 1,000 prompts per model and mode (means, with 95% CIs below)

Model	Params	Coverage	Density ($\times 10^2$)	Uncertainty	Latency (s)
(A) Closed-book					
Qwen2.5-1.5B-Instruct	1.5B	**2.216** [2.139, 2.293]	1.52 [1.46, 1.57]	1.053 [1.002, 1.104]	3.69 [3.65, 3.72]
SecureLLM (adapted)	1.5B	2.114 [2.034, 2.195]	1.46 [1.40, 1.52]	1.058 [1.008, 1.108]	3.59 [3.54, 3.63]
Llama-3.2-1B-Instruct	1.2B	1.842 [1.758, 1.926]	1.36 [1.30, 1.42]	1.137 [1.084, 1.190]	**1.99** [1.96, 2.02]
Gemma-2-2B-it	2.6B	2.169 [2.088, 2.250]	**1.63** [1.55, 1.71]	**0.897** [0.846, 0.948]	4.75 [4.69, 4.81]
Phi-3.5-mini-instruct	3.8B	1.842 [1.766, 1.918]	1.29 [1.24, 1.35]	1.096 [1.047, 1.145]	5.27 [5.27, 5.27]
Model	**Params**	**Coverage**	**Grounding sim.**	**Risk flag (%)**	**Latency (s)**
(B) Retrieval-augmented, $k = 3$					
Qwen2.5-1.5B-Instruct	1.5B	2.312 [2.225, 2.399]	0.4580 [0.4501, 0.4658]	**62.2** [59.2, 65.2]	3.98 [3.93, 4.02]
SecureLLM (adapted)	1.5B	1.931 [1.843, 2.019]	0.4065 [0.3983, 0.4147]	75.4 [72.6, 78.0]	3.68 [3.62, 3.74]
Llama-3.2-1B-Instruct	1.2B	**2.368** [2.278, 2.458]	0.4570 [0.4498, 0.4642]	65.0 [62.0, 67.9]	**2.23** [2.20, 2.25]
Gemma-2-2B-it	2.6B	2.356 [2.273, 2.439]	**0.4666** [0.4602, 0.4731]	62.5 [59.5, 65.4]	4.72 [4.65, 4.79]
Phi-3.5-mini-instruct	3.8B	1.628 [1.557, 1.699]	0.4277 [0.4212, 0.4343]	75.4 [72.6, 78.0]	23.99* [20.90, 27.08]

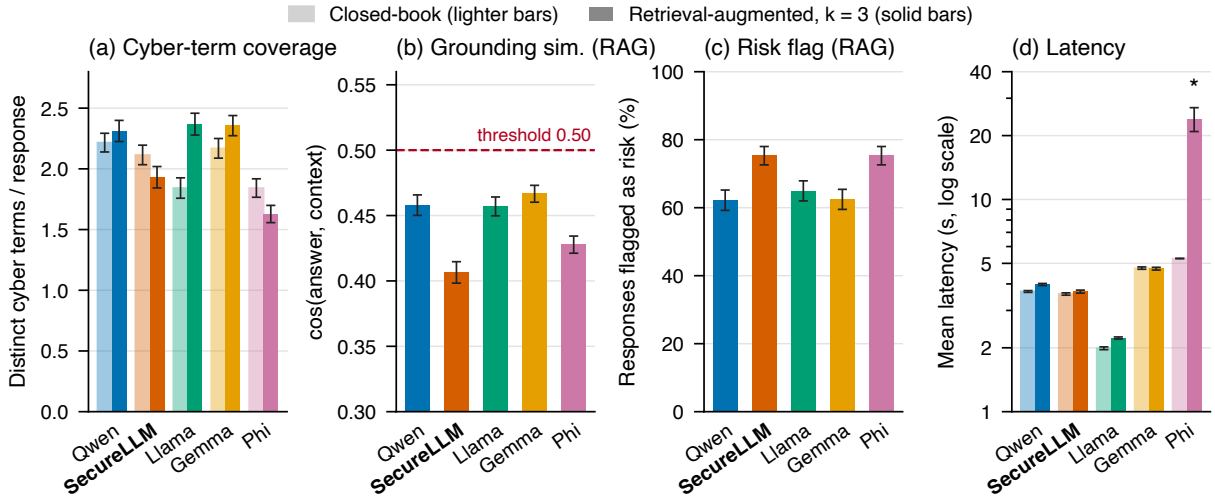


Fig. 3 Benchmark results for five compact models on 1,000 prompts per mode, with error bars showing 95% CIs. (a) Cyber-term coverage in closed-book (light) and retrieval-augmented (solid) modes. (b) Grounding similarity between answers and retrieved context; the dashed line is the 0.5 flag threshold. (c) Share of responses flagged as a hallucination risk. (d) Mean latency on a logarithmic scale. *Phi-3.5-mini used eager attention, so its latency is not comparable with the other models. SecureLLM is shown in vermillion and its Qwen2.5 base in blue.

5.3 E3: nearest-neighbor retrieval over one million executables (RQ4)

Exact search over 1,039,867 vectors returned the five neighbors in Table 7. Their squared distances (0.199–0.205) correspond to cosine similarities of about 0.90, so the neighbors are close in the embedding space. All five, however, are benign executables with no antivirus detections (0 of 77 or 0 of 73 engines), even though the query describes a

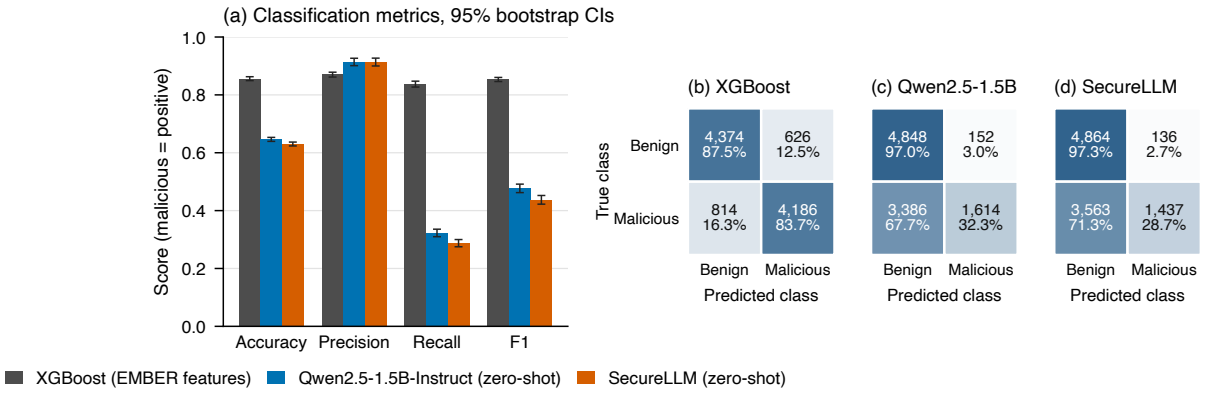


Fig. 4 E2 on the balanced EMBER test set. (a) Accuracy, precision, recall and F_1 with 95% bootstrap CIs. (b)–(d) Confusion matrices; percentages are relative to the 5,000 samples in each true class. Both zero-shot language models are conservative: fewer than 3.1% false positives, but more than two-thirds of malicious samples missed.

Table 5 Primary paired comparison: SecureLLM versus its Qwen2.5-1.5B-Instruct base ($n = 1{,}000$ prompts)

Metric	Base	Adapt.	Δ	p_{Holm}	$\lvert d_z \rvert$
Closed-book					
Coverage	2.216	2.114	−0.102	0.062	0.073
Density[a]	1.52	1.46	−0.06	0.127	0.058
Retrieval-augmented					
Coverage	2.312	1.931	−0.381	5×10^{-13}	0.233
Grounding	0.4580	0.4065	−0.0515	2×10^{-25}	0.373
Risk flag (%)	62.2	75.4	+13.2	3×10^{-13}	–

Wilcoxon signed-rank tests for continuous metrics and an exact McNemar test for the risk flag; Holm correction across ten model pairs. Red cells: significant differences unfavorable to SecureLLM. [a]Per 100 words.

Table 6 E2: malware classification on the balanced EMBER test set (5,000 benign, 5,000 malicious; 95% bootstrap CIs below)

Metric	XGBoost	Qwen2.5	SecureLLM
Accuracy	**0.856** [0.849, 0.863]	0.646 [0.639, 0.653]	0.630 [0.624, 0.637]
Precision	0.870 [0.862, 0.878]	**0.914** [0.901, 0.927]	**0.914** [0.900, 0.927]
Recall	**0.837** [0.827, 0.847]	0.323 [0.310, 0.336]	0.287 [0.275, 0.300]
F_1	**0.853** [0.846, 0.860]	0.477 [0.462, 0.492]	0.437 [0.422, 0.452]
Specificity	0.875	**0.970**	**0.973**
MCC	**0.713**	0.383	0.357
ROC-AUC	**0.918** [0.912, 0.923]	–	–
TP / FN	4,186 / 814	1,614 / 3,386	1,437 / 3,563
FP / TN	626 / 4,374	152 / 4,848	136 / 4,864
Pred. malicious	4,812	1,766	1,573

Malicious is the positive class. Specificity and MCC are computed from the confusion matrices. ROC-AUC requires scores and is available only for XGBoost. Both language models produced 10,000 parseable labels.

packed, high-entropy file. The five entries also represent only three distinct files: the index contains 520,000 unique SHA-256 values among 1,039,867 records, so most files appear twice. Proximity in a generic sentence-embedding space of numeric feature strings therefore reflects surface similarity rather than maliciousness. This single-query case study shows that exact search over a million records is feasible. It is not an estimate of retrieval quality; that would require many deduplicated queries.

5.4 E4: retrieval-augmented classification pilot (RQ2, RQ4)

Adding retrieved-neighbor statistics to the prompt did not improve classification; the model predicted *benign* for all 100 samples (Table 8). Its accuracy of 0.63 equals the benign base rate, and recall, precision and F_1 are zero. On the same samples, the feature-only prompt had produced 11 true and 1 false positive. Because the retrieval-augmented run predicted benign everywhere, the paired outcome is fully determined: the feature-only prompt was correct and the retrieval-augmented prompt wrong on 11 samples, and the reverse held for one sample. The exact McNemar test gives $p = 0.0063$. On this pilot, retrieved evidence made the classifier significantly worse. The model predicted the majority class, which is consistent with

Table 7 E3: five nearest neighbors of a packed, high-entropy query in the 1,039,867-record index

Rank	d^2	cos	Label	AV det.	File
1	0.1987	0.901	Benign	0/77	A
2	0.1987	0.901	Benign	0/77	A
3	0.2009	0.900	Benign	0/73	B
4	0.2009	0.900	Benign	0/73	B
5	0.2045	0.898	Benign	0/77	C

d^2: squared Euclidean distance from `IndexFlatL2`; cos $= 1 - d^2/2$ for unit vectors. Ranks with the same letter share a SHA-256 hash.

Table 8 E4: feature-only versus retrieval-augmented prompting on the same 100 EMBER samples (63 benign, 37 malicious; Qwen2.5-1.5B-Instruct)

Metric	Feature-only	+ Neighbors
TP / FN	11 / 26	0 / 37
FP / TN	1 / 62	0 / 63
Accuracy	**0.730**	0.630
Balanced accuracy	**0.641**	0.500
Precision	**0.917**	undefined
Recall	**0.297**	0.000
F_1	**0.449**	0.000
Exact McNemar, 11 vs. 1 discordant: $p = 0.0063$		

Precision is undefined without positive predictions. With 12 positive predictions, the feature-only precision has a 95% Wilson interval of [0.646, 0.985]; the 10,000-sample results in Table 6 supersede these pilot estimates.

evidence that pointed toward benign neighbors (Section 5.3).

6 Discussion

Table 9 summarizes the answer to each research question, the mechanism we consider most likely and the corresponding design recommendation. The following subsections explain the reasoning.

6.1 Why continued pretraining did not help

The falling validation loss shows that the model learned the corpus. Four factors explain why this knowledge did not surface in answers. First, the objective does not match the use. The adaptation data are raw records, 84% of which are MalwareBazaar and ExploitDB metadata dominated by hashes, signatures and templated fields. Much of the gradient signal is therefore spent on predicting identifier strings and record templates rather than explanatory prose, while the benchmark asks for explanations. Second, the starting point was already instruction-tuned, and raw-text continued pretraining without instruction data or replay can erode instruction-following [29, 30]. The null closed-book result, followed by significant degradation under the longer and more constrained RAG prompt, is consistent with such erosion. We cannot confirm the mechanism without instruction-following probes. Third, coverage counts distinct terms and saturates quickly, so it is insensitive to deeper changes in content. Fourth, the decoding configuration interacts with adaptation. A repetition penalty of 1.3 with 4-gram blocking penalizes repeated domain tokens, and presumably penalizes a model trained on templated text more. An earlier 100-prompt run with a penalty of 1.1 had shown the opposite sign (2.41 versus 2.31 terms for the adapted and base models). The prompt set also changed between the two runs, so the two effects cannot be separated. The practical lesson agrees with our institutional studies [31]: domain knowledge should be injected in the format in which it will be used, for example as question–answer pairs derived from records, mixed with general instruction data.

6.2 Why retrieval did not raise grounding

Retrieval helps when the retrieved evidence contains the answer. The E1 design largely excludes this case. The questions are conceptual (Section 4.1), whereas the corpus consists of individual records, so the top-3 chunks for a question such as "what is a CVE?" are CVE records that share vocabulary but not the answer. The cosine similarity between an explanatory answer and identifier-dense record text is bounded well below 1 even when the answer is correct. With a fixed threshold of 0.5, the flag rate therefore largely restates this topical mismatch rather than measuring unsupported generation. Encoder truncation compounds the problem: any chunk longer than 256 word pieces is embedded only up to its header-heavy prefix, which weakens both retrieval and

Table 9 Summary of findings, diagnosed causes and recommendations

RQ	Answer	Most likely causes	Recommendation
RQ1	No measurable closed-book change; worse under RAG	Raw-record objective vs. instruction use; corpus dominated by hashes and exploit metadata; possible erosion of instruction following; decoding penalizes templated text	Mix instruction-formatted and replay data into adaptation; format records as question–answer pairs; evaluate with task-level correctness
RQ2	Retrieval did not raise answer–evidence agreement; it hurt the adapted model and the E4 classifier	Prompts not answerable from records; 256-word-piece truncation; generic encoder; uninformative neighbors	Corpus-grounded questions with extractable answers; long-context or security-tuned encoders; hybrid lexical–dense retrieval for identifiers; retrieval metrics with relevance labels
RQ3	Yes: 161,190 records from four sources in one schema and index	Structured-field gap for ExploitDB CVE links; skew toward malware metadata	Keep structured join keys; rebalance or stratify sources; record per-source attrition
RQ4	XGBoost far ahead (0.856 vs. 0.630–0.646 accuracy); LLMs conservative	Five-feature prompt; no labeled supervision; hard labels only	Compare at matched operating points; analyze error overlap; supervise the LLM on labeled tasks before hybrid use

the similarity metric. Finally, the RAG instruction to "use only the context" is counterproductive when the context is off-target. Models that complied closely may have produced answers that were topically narrower or more evasive; this would explain lower coverage for SecureLLM and Phi-3.5-mini, but we did not measure it directly. E3 and E4 show the analogous failure for malware: numeric feature strings embedded with a general sentence encoder yield neighbors that are close but not label-informative, and a model that faithfully uses such evidence is misled by it. The lesson is that retriever quality, not the presence of retrieval, determines whether RAG helps. Security retrieval needs (i) questions whose answers exist in the index, (ii) encoders that fit the chunk length or are trained with task-specific objectives, for example with contrastive supervision [50], and (iii) lexical or structured lookup for exact identifiers such as CVE IDs and hashes, where BM25-style matching is strong [35].

6.3 Implications for malware triage

Zero-shot compact language models are not competitive with a trained gradient-boosted classifier on static features: they miss 68–71% of malicious samples, and domain adaptation on intelligence text does not change this. Their high specificity suggests a possible role as a secondary, conservative reviewer in a hybrid pipeline, where a trained detector screens samples and the language model explains or confirms high-risk cases. We present this role as a hypothesis, not a finding. Supporting it would require (i) comparing precision at matched recall, using XGBoost's score threshold; (ii) measuring whether the two systems' errors overlap; (iii) supervising the language model on labeled examples; and (iv) accounting for the cost gap between seconds per sample for the language model and microseconds for the trees. The result mirrors broader findings that LLMs are strongest when embedded in tool loops and weakest as stand-alone classifiers of numeric data [7, 10].

6.4 Toward leakage-controlled, tool-comparable evaluation

Our long-term goal is to use archived intelligence for security decisions and to compare the resulting systems with established tools rather than with other LLMs. The present study shows that this requires correctness-based labels. We therefore propose decomposing "malicious or not" into per-source binary tasks, each labeled from authoritative metadata: (T1) malware versus benign, using MalwareBazaar positives against a benign control set such as EMBER's; (T2) known-exploited versus not, using KEV membership over NVD records; and (T3) public exploit available versus not, using the repaired ExploitDB-to-CVE join key. Each task should be evaluated only on records published after the training cutoffs of all compared models, with time-aware splits [47, 48], and every system should be scored on identical examples. The systems should include SecureLLM, its base, larger LLMs and conventional detectors, and

the metrics should be precision, recall and F_1 with CIs. A human-annotated subset (for example, 200 responses rated by two annotators, with Cohen's κ reported) should be used to validate any automated proxy. An expanded NVD collection of 338,899 records, close to the full catalog at collection time, has been assembled for this purpose; post-cutoff evaluation splits must be carved out of it explicitly by publication date.

7 Threats to validity

Construct validity.

All E1 metrics are automated proxies that no human has validated. Coverage and uncertainty use unanchored substring matching over small lexicons. Grounding similarity measures embedding overlap, not correctness. The risk flag uses an uncalibrated threshold and exempts only eight canonical refusal strings, so hedged but unsupported answers are flagged and canned refusals are not. Retrieval metrics such as Recall@k could not be computed without relevance labels. We therefore make only relative, same-condition comparisons and avoid absolute claims about factuality.

Internal validity.

Decoding settings (repetition penalty 1.3, 4-gram blocking, a 200-token limit) apply to all models but may affect them unequally, and the sign of the closed-book adaptation effect differed under an earlier configuration (Section 6.1). The comparison models differ in size (1.2–3.8B parameters) and training data. Phi-3.5-mini's latency reflects eager attention. Adaptation was run once, with one seed and without learning-rate warmup; sequences were padded to 1,024 tokens, which affects efficiency but not the loss. All models received the same system prompt, which introduces the assistant as "SecureLLM".

External validity.

The NVD component covers only January–April of 2024 and 2025 and January–March of 2026 and is filtered by length, so it does not represent the whole catalog. The corpus is dominated by MalwareBazaar metadata, and the MalwareBazaar component is a subset of the export rather than a time-stratified sample. E2 uses static EMBER features, and the language models received only five of them. E3 uses a single hand-specified query over an index with substantial duplication. E4 is a 100-sample pilot run with the base model only. Results may differ for larger models, other sources or dynamic-analysis features.

Data leakage.

The 2024 NVD block precedes the release of Qwen2.5, so part of it may be present in the base model's pretraining data. No decontamination analysis [33] was performed, so base-versus-adapted comparisons on content from before 2025 are not guaranteed to be leakage-free. The protocol in Section 6.4 addresses this.

Conclusion validity.

Paired tests and Holm correction control the family-wise error within each metric and mode but not across metrics. Effect sizes are small to moderate ($|d_z| \leq 0.373$), and some differences, such as the closed-book coverage difference, are not significant. We therefore describe them as the absence of evidence for an effect, not as evidence of equivalence.

Reproducibility.

The released code covers ingestion, corpus construction, chunking, indexing, training and both benchmark harnesses, together with the 1,000 prompts [51]. Scripts for E2–E4 are available from the corresponding author. The model weights are not publicly released (Section 8).

8 Ethical considerations

This study used only publicly available intelligence and executed no exploit code. No malware binaries were obtained, stored or executed; MalwareBazaar data were used only as hashes and static metadata. ExploitDB records contain proof-of-concept exploit text (capped at 20,000 characters per record), which is dual-use. For this reason we release the ingestion code and normalization schema rather than the assembled corpus, and we do not publish the adapted weights; researchers can regenerate the corpus from the original sources under those sources' terms. A retrieval-grounded assistant over exploit archives could lower the effort needed to find weaponizable material as well as the effort needed to defend against it, so we

frame SecureLLM as an analyst-facing research tool subject to human review. Given its measured recall of 0.29 on EMBER, it must not be deployed as a stand-alone detector. No human participants or personal data were involved.

9 Conclusion and future work

We presented SecureLLM, an open framework that unifies 161,190 vulnerability, exploit and malware records from four public sources into one corpus and index, and we used it to test two common recipes for compact language models. Continued pretraining of a 1.5B-parameter instruction-tuned model lowered in-domain loss but did not change its closed-book answers measurably, and it degraded retrieval-augmented answers. Retrieval did not raise answer–evidence agreement under a conceptual prompt set, and in a malware pilot it misled the model into labeling every sample benign. On a balanced 10,000-sample EMBER test set, the zero-shot language models were conservative, with specificity of 0.97 and recall of about 0.3, and remained far behind a gradient-boosted baseline. We traced these outcomes to four mechanisms: objective mismatch, benchmark–corpus mismatch, encoder truncation and uninformative retrieval neighborhoods.

Future work will follow directly from this diagnosis. We plan to (i) adapt with instruction-formatted records mixed with general instruction data; (ii) rebuild the index with chunking that fits the encoder, or with long-context and security-tuned encoders and hybrid lexical–dense retrieval; (iii) evaluate on corpus-grounded questions with extractable answers and human-validated metrics; (iv) run the leakage-controlled per-source protocol of Section 6.4 against conventional tools at matched operating points; and (v) study whether chain-of-thought supervision [62] makes each decision auditable. We hope that the released pipeline and these documented negative results help others avoid the same pitfalls.

Statements and Declarations

Funding. This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Competing interests. The authors have no relevant financial or non-financial interests to disclose.

Ethics approval and consent to participate. Not applicable. The study involved no human participants, human data or animals.

Consent for publication. Not applicable.

Data availability. All source data are publicly available from the NVD CVE API (https://nvd.nist.gov/developers/vulnerabilities), the CISA KEV catalog (https://www.cisa.gov/known-exploited-vulnerabilities-catalog), the ExploitDB repository (https://gitlab.com/exploit-database/exploitdb), the MalwareBazaar export (https://bazaar.abuse.ch/export/) and the EMBER dataset [43]. The 1,000 benchmark prompts are included in the code repository. The assembled corpus is not redistributed because it contains dual-use exploit text (Section 8); it can be regenerated with the released scripts. Benchmark outputs and per-prompt scores are available from the corresponding author on reasonable request.

Materials availability. Not applicable.

Code availability. The ingestion, corpus construction, chunking, indexing, training and benchmarking code is available at https://github.com/sredh01/SecureLLM-Master [51]. Scripts for the EMBER experiments (E2–E4) and the adapted model weights are available from the corresponding author on reasonable request for research purposes.

Author contributions. Syrus Redhouse: software, data curation, investigation, formal analysis, visualization, writing – original draft. Izzat Alsmadi: conceptualization, methodology, supervision, resources, writing – review and editing. Anas AlSobeh: methodology (evaluation design and statistical analysis), validation, writing – review and editing. All authors read and approved the final manuscript.

Use of generative AI. The authors used an AI writing assistant (Codex, OpenAI) to help restructure and copy-edit the manuscript and to check bibliographic metadata and create figure.

Appendix A Benchmark prompts, instructions and lexicons

System instructions. Closed-book: "You are SecureLLM, a cybersecurity focused language model. Answer the question clearly and directly. Do not invent facts. If you are unsure, say that you are unsure." RAG: "You are SecureLLM, a cybersecurity assistant. Use only the context below to answer the question. If the context does not support the answer, say you are unsure." In RAG mode the user turn contains the three retrieved chunks, followed by the question. The same instructions were given to all five models through each model's chat template.

Lexicons. Cybersecurity terms $\mathcal{T}$ (15): *cve, cvss, cwe, exploit, vulnerability, malware, hash, ransomware, privilege, authentication, patch, threat, ioc, command and control, remote code execution.* Hedging terms $\mathcal{U}$ (6): *not sure, unsure, uncertain, may, might, possibly.* Refusal phrases (8): *i am unsure, i'm unsure, not enough context, context does not support, the context does not support, i do not know, i don't know, not supported by the context.*

Table A1 Benchmark strata with one example prompt each (100 prompts per stratum)

IDs	Stratum	Example prompt
1–100	Fundamentals	People throw around the term CWE a lot, so where does that fit compared to CVE?
101–200	Vulnerability management	Someone told me exploitability matters more than severity. Is that true?
201–300	Malware analysis	Someone told me a hash can identify malware. What are the limits of that approach?
301–400	Network security	A server started communicating with an unfamiliar external address. How concerning is that?
401–500	Incident response	Someone reported ransomware on a single workstation. How can we tell whether the problem extends beyond that machine?
501–600	Threat intelligence	Someone mentioned indicators of compromise during a briefing. What are they actually used for?
601–700	Cloud and identity	Could a cloud environment be compromised even if no malware is present?
701–800	Web security	Could a web application be compromised even if the underlying server is fully patched?
801–900	Security operations	Could an important security incident be hidden among thousands of routine alerts?
901–1000	Advanced mixed reasoning	A vulnerability affects thousands of organizations, but only a handful have been compromised. What factors could explain that?

References

[1] National Institute of Standards and Technology: National Vulnerability Database: CVE API 2.0. https://nvd.nist.gov/developers/vulnerabilities (2026)

[2] Cybersecurity and Infrastructure Security Agency: Known Exploited Vulnerabilities catalog. https://www.cisa.gov/known-exploited-vulnerabilities-catalog (2026)

[3] OffSec: Exploit Database (ExploitDB) repository. https://gitlab.com/exploit-database/exploitdb (2026)

[4] abuse.ch: MalwareBazaar full data export. https://bazaar.abuse.ch/export/ (2026)

[5] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., *et al.*: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020)

[6] Xu, H., Wang, S., Li, N., Wang, K., Zhao, Y., Chen, K., Yu, T., Liu, Y., Wang, H.: Large language models for cyber security: A systematic literature review. arXiv:2405.04760 (2024). https://doi.org/10.48550/arXiv.2405.04760

[7] Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., Rass, S.: PentestGPT: Evaluating and

harnessing large language models for automated penetration testing. In: 33rd USENIX Security Symposium (USENIX Security 24), pp. 847–864 (2024)

[8] Alam, M.T., Bhusal, D., Nguyen, L., Rastogi, N.: CTIBench: A benchmark for evaluating LLMs in cyber threat intelligence. In: Advances in Neural Information Processing Systems 37 (Datasets and Benchmarks Track) (2024). https://doi.org/10.48550/arXiv.2406.07599

[9] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Computing Surveys **55**(12), 248–124838 (2023) https://doi.org/10.1145/3571730

[10] Ullah, S., Han, M., Pujar, S., Pearce, H., Coskun, A., Stringhini, G.: LLMs cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks. In: 2024 IEEE Symposium on Security and Privacy (SP), pp. 862–880 (2024). https://doi.org/10.1109/SP54263.2024.00210

[11] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., Smith, N.A.: Don't stop pretraining: Adapt language models to domains and tasks. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8342–8360 (2020). https://doi.org/10.18653/v1/2020.acl-main.740

[12] Aghaei, E., Niu, X., Shadid, W., Al-Shaer, E.: SecureBERT: A domain-specific language model for cybersecurity. In: Security and Privacy in Communication Networks (SecureComm 2022). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 462, pp. 39–56. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-25538-0_3

[13] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474 (2020)

[14] Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.-W.: REALM: Retrieval-augmented language model pre-training. In: Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 3929–3938 (2020)

[15] Kassianik, P., Saglam, B., Chen, A., Nelson, B., Vellore, A., Aufiero, M., et al.: Llama-3.1-FoundationAI-SecurityLLM-Base-8B technical report. arXiv:2504.21039 (2025). https://doi.org/10.48550/arXiv.2504.21039

[16] Yu, Y.-C., Chiang, T.-H., Tsai, C.-W., Huang, C.-M., Tsao, W.-K.: Primus: A pioneering collection of open-source datasets for cybersecurity LLM training. arXiv:2502.11191 (2025). https://doi.org/10.48550/arXiv.2502.11191

[17] Nie, Y., Li, H., Guo, C., Jiang, R., Wang, Z., Li, B., Song, D., Guo, W.: VulnLLM-R: Specialized reasoning LLM with agent scaffold for vulnerability detection. arXiv:2512.07533 (2025). https://doi.org/10.48550/arXiv.2512.07533

[18] Tihanyi, N., Ferrag, M.A., Jain, R., Bisztray, T., Debbah, M.: CyberMetric: A benchmark dataset based on retrieval-augmented generation for evaluating LLMs in cybersecurity knowledge. In: 2024 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 296–302 (2024). https://doi.org/10.1109/CSR61664.2024.10679494

[19] Bhatt, M., Chennabasappa, S., Li, Y., Nikolaidis, C., Song, D., Wan, S., et al.: CyberSecEval 2: A wide-ranging cybersecurity evaluation suite for large language models. arXiv:2404.13161 (2024). https://doi.org/10.48550/arXiv.2404.13161

[20] Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of

machine learning in computer security. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 3971–3988 (2022)

[21] Qwen Team, Yang, A., Yang, B., Zhang, B., Hui, B., et al.: Qwen2.5 technical report. arXiv:2412.15115 (2024). https://doi.org/10.48550/arXiv.2412.15115

[22] Ji, H., Yang, J., Chai, L., Wei, C., Yang, L., Duan, Y., et al.: SEvenLLM: Benchmarking, eliciting, and enhancing abilities of large language models in cyber threat intelligence. arXiv:2405.03446 (2024). https://doi.org/10.48550/arXiv.2405.03446

[23] Duan, X., Wu, J., Du, M., Luo, T., Yang, M., Wu, Y.: MultiCode: A unified code analysis framework based on multi-type and multi-granularity semantic learning. In: 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 359–364 (2021). https://doi.org/10.1109/ISSREW53611.2021.00102

[24] Shatnawi, A., AlSobeh, A., Alsmadi, I., Al-Ahmad, B.: Tailored large language models for spam detection: From model customization to benchmarking effectiveness. In: 2025 5th Intelligent Cybersecurity Conference (ICSC), pp. 264–271 (2025). https://doi.org/10.1109/ICSC65596.2025.11140025

[25] Alshattnawi, S., Shatnawi, A., AlSobeh, A.M.R., Magableh, A.A.: Beyond word-based model embeddings: Contextualized representations for enhanced social media spam detection. Applied Sciences **14**(6), 2254 (2024) https://doi.org/10.3390/app14062254

[26] Al-Shawakfa, E.M., Alsobeh, A.M.R., Omari, S., Shatnawi, A.: RADAR#: An ensemble approach for radicalization detection in Arabic social media using hybrid deep learning and transformer models. Information **16**(7), 522 (2025) https://doi.org/10.3390/info16070522

[27] AlSobeh, A., Gwarzo, Z., Shatnawi, A.: ShadowPlay: Engineering defenses against role-based prompt injection and dependency hallucination in LLM-powered development. In: 2025 International Conference on Cybersecurity and AI-Based Systems (Cyber-AI), pp. 317–325 (2025). https://doi.org/10.1109/Cyber-AI66431.2025.11233258

[28] Shatnawi, A., AlSobeh, A., Khamaiseh, S., Al-Abdullah, M.: Secure-by-design framework for agentic AI: Mitigating toxic flows and adversarial exploits in multi-agent ecosystems. In: 2026 Intermountain Engineering, Technology and Computing (IETC), pp. 1–6 (2026). https://doi.org/10.1109/IETC69527.2026.11568651

[29] Ibrahim, A., Thérien, B., Gupta, K., Richter, M.L., Anthony, Q., Lesort, T., Belilovsky, E., Rish, I.: Simple and scalable strategies to continually pre-train large language models. arXiv:2403.08763 (2024). https://doi.org/10.48550/arXiv.2403.08763

[30] Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., Zhang, Y.: An empirical study of catastrophic forgetting in large language models during continual fine-tuning. arXiv:2308.08747 (2023). https://doi.org/10.48550/arXiv.2308.08747

[31] Alsmadi, I., Alsobeh, A., Redhouse, S.: A reproducible framework for institutional domain adaptation of LLMs via supervised fine-tuning and retrieval-augmented generation. In: 2026 2nd International Conference on Artificial Intelligence Systems (AIS), pp. 496–501 (2026). https://doi.org/10.1109/AIS69919.2026.11621883

[32] Alsmadi, I., Alsobeh, A.: TAMUSA-Chat: A domain-adapted large language model conversational system for research and responsible deployment. arXiv:2603.09992 (2026). https://doi.org/10.48550/arXiv.2603.09992

[33] Sainz, O., Campos, J., García-Ferrero, I., Etxaniz, J., Lopez de Lacalle, O., Agirre, E.: NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 10776–10787 (2023). https://doi.org/10.18653/v1/2023.findings-emnlp.722

[34] Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., Scialom, T.: Augmented language models: A survey. Transactions on Machine Learning Research (2023)

[35] Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval **3**(4), 333–389 (2009) https://doi.org/10.1561/1500000019

[36] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.-t.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pp. 6769–6781 (2020). https://doi.org/10.18653/v1/2020.emnlp-main.550

[37] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 3982–3992 (2019). https://doi.org/10.18653/v1/D19-1410

[38] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: Advances in Neural Information Processing Systems, vol. 33, pp. 5776–5788 (2020)

[39] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2021) https://doi.org/10.1109/TBDATA.2019.2921572

[40] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997 (2023). https://doi.org/10.48550/arXiv.2312.10997

[41] Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics **12**, 157–173 (2024) https://doi.org/10.1162/tacl_a_00638

[42] Es, S., James, J., Espinosa Anke, L., Schockaert, S.: RAGAs: Automated evaluation of retrieval augmented generation. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pp. 150–158 (2024). https://doi.org/10.18653/v1/2024.eacl-demo.16

[43] Anderson, H.S., Roth, P.: EMBER: An open dataset for training static PE malware machine learning models. arXiv:1804.04637 (2018). https://doi.org/10.48550/arXiv.1804.04637

[44] Saxe, J., Berlin, K.: Deep neural network based malware detection using two dimensional binary program features. In: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), pp. 11–20 (2015). https://doi.org/10.1109/MALWARE.2015.7413680

[45] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.: Malware detection by eating a whole EXE. In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence (2018). https://doi.org/10.48550/arXiv.1710.09435

[46] Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016). https://doi.org/10.1145/2939672.2939785

[47] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: TESSERACT: Eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium (USENIX Security 19), pp. 729–746 (2019)

[48] AlSobeh, A.M.R., Gaber, K., Hammad, M.M., Nuser, M., Shatnawi, A.: Android

malware detection using time-aware machine learning approach. Cluster Computing **27**(9), 12627–12648 (2024) https://doi.org/10.1007/s10586-024-04484-6

[49] Terawi, N., Ashqar, H.I., Darwish, O., Alsobeh, A., Zahariev, P., Tashtoush, Y.: Enhanced detection of intrusion detection system in cloud networks using time-aware and deep learning techniques. Computers **14**(7), 282 (2025) https://doi.org/10.3390/computers14070282

[50] AlSobeh, A., Khamaiseh, S., Middendorf, D., Dziubinski, J.: CodeLLM-Auth: Multimodal authorship attribution and verification via supervised contrastive code embeddings. In: 2026 IEEE 50th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 732–737 (2026). https://doi.org/10.1109/COMPSAC69091.2026.00100

[51] Redhouse, S., Alsmadi, I.: SecureLLM: Source code for corpus construction, domain adaptation, retrieval and benchmarking. https://github.com/sredh01/SecureLLM-Master. Commit 7591dd0 (2026)

[52] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., *et al.*: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45 (2020). https://doi.org/10.18653/v1/2020.emnlp-demos.6

[53] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., et al.: The Llama 3 herd of models. arXiv:2407.21783 (2024). https://doi.org/10.48550/arXiv.2407.21783

[54] Gemma Team, Riviere, M., Pathak, S., Sessa, P.G., Hardin, C., et al.: Gemma 2: Improving open language models at a practical size. arXiv:2408.00118 (2024). https://doi.org/10.48550/arXiv.2408.00118

[55] Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A.A., et al.: Phi-3 technical report: A highly capable language model locally on your phone. arXiv:2404.14219 (2024). https://doi.org/10.48550/arXiv.2404.14219

[56] Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945) https://doi.org/10.2307/3001968

[57] McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika **12**(2), 153–157 (1947) https://doi.org/10.1007/BF02295996

[58] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**(2), 65–70 (1979)

[59] Cohen, J.: Statistical Power Analysis for the Behavioral Sciences, 2nd edn. Lawrence Erlbaum Associates, Hillsdale, NJ (1988)

[60] Wilson, E.B.: Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association **22**(158), 209–212 (1927) https://doi.org/10.1080/01621459.1927.10502953

[61] Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall/CRC, New York (1994). https://doi.org/10.1201/9780429246593

[62] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837 (2022)