REDSAGE: A CYBERSECURITY GENERALIST LLM

Anonymous authors

001 002 003

004

006 007

008 009

010

011

012

013

014

015

016

017

018

019

021

024

025

026

028

029

031

032

034

035

037

040 041 042

043

044

047 048 049 Paper under double-blind review

ABSTRACT

Cybersecurity operations demand assistant LLMs that support diverse workflows without exposing sensitive data. Existing solutions either rely on proprietary APIs with privacy risks or on open models lacking domain adaptation. To bridge this gap, we curate 11.8B tokens of cybersecurity-focused continual pretraining data via large-scale web filtering and manual collection of high-quality resources, spanning 28.6K documents across frameworks, offensive techniques, and security tools. Building on this, we design an agentic augmentation pipeline that simulates expert workflows to generate 266K multi-turn cybersecurity samples for supervised fine-tuning. Combined with general open-source LLM data, these resources enable the training of RedSage, an open-source, locally deployable cybersecurity assistant with domain-aware pretraining and post-training. To rigorously evaluate the models, we introduce RedSage-Bench, a benchmark with 30K multiple-choice and 240 open-ended Q&A items covering cybersecurity knowledge, skills, and tool expertise. RedSage is further evaluated on established cybersecurity benchmarks (e.g., CTI-Bench, CyberMetric, SECURE) and general LLM benchmarks to assess broader generalization. At the 8B scale, RedSage achieves consistently better results, surpassing the baseline models by up to +5.59 points on cybersecurity benchmarks and +5.05 points on Open LLM Leaderboard tasks. These findings demonstrate that domain-aware agentic augmentation and pre/post-training can not only enhance cybersecurity-specific expertise but also help to improve general reasoning and instruction-following. All models, datasets, and code will be released to advance reproducibility and open cybersecurity LLM research.

1 Introduction

The rapid evolution of cybersecurity threats has elevated the need for proactive and comprehensive defense strategies, as organizations face increasingly sophisticated attacks and advanced persistent threats (Khraisat et al., 2024). Modern cybersecurity involves a wide range of critical tasks, including threat analysis, incident response, vulnerability management, and security monitoring. However, the complexity of security tools and the depth of expertise required to operate them, particularly in handling incidents, pose significant challenges (Cichonski et al., 2021). These challenges are compounded by a global skill shortage, with research estimating a demand–supply gap in the millions

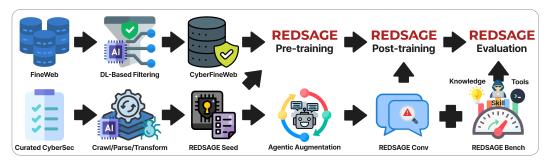


Figure 1: Overview of the RedSage pipeline. RedSage is trained through continual pre-training on cybersecurity-filtered corpora and post-training with curated and augmented conversation data, followed by evaluation on a comprehensive benchmark covering knowledge, skills, and tool expertise.

Table 1: Comparison of cybersecurity LLM benchmarks. Columns indicate knowledge (Know.), skills (Skill), tool proficiency (Tool), and use of quality scoring (Qual.). Size = total samples. Agentic CTF benchmarks (e.g., NYU-CTF, CyBench) are excluded as they are interactive rather than base LLM eval.

Name	Know.	Skill	Tool	Qual.	Size
SecEval	✓	Х	Х	Х	2,000
CyberMetric	\checkmark	X	X	X	10,000
CyberBench	\checkmark	X	X	X	80,422
SECURE	\checkmark	X	X	Х	4,072
CS-Eval	\checkmark	X	X	Х	4,369
SecBench	\checkmark	X	X	X	47,910
CTI-Bench	\checkmark	\checkmark	X	X	5,610
CyberSecEval	X	\checkmark	X	X	1,000
RedSage-Bench (Ours)	√	√	√	√	30,240

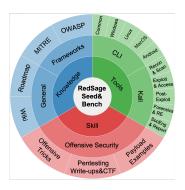


Figure 2: Taxonomy of Red-Sage Seed&Bench dataset. It spans knowledge, practical offensive skills, and tool expertise (CLI and Kali Linux).

of unfilled cybersecurity positions ($(ISC)^2$, 2022). Consequently, there is growing momentum to employ cybersecurity-tuned LLMs to augment human analysts.

Recent efforts have produced cybersecurity-trained LLMs, yet most emphasize a single training stage while overlooking others. For instance, some extend pretraining on domain-specific corpora (Kassianik et al., 2025b) but apply limited post-training with only 835 samples (Yu et al., 2025) or fewer than 30K cybersecurity-filtered items (Weerawardhena et al., 2025), while others focus on supervised fine-tuning with large cybersecurity Q&A collections without pretraining to strengthen domain knowledge (Deep Hat, 2025). Further, existing cybersecurity benchmarks offer only partial coverage, such as omitting tool proficiency and qualitative evaluation of free-response Q&A beyond simple MCQs (see Table 1 and Fig. 2). Beyond these gaps, most works also do not release their data or pipelines, limiting reproducibility and openness (see Table 2).

To address these gaps, we present RedSage (Retrieval-Enhanced Data-driven Security Assistant Guidance and Evaluation), an open-source LLM tailored for cybersecurity. As illustrated in Fig. 1, RedSage integrates large-scale continual pretraining on cybersecurity-filtered corpora, post-training with curated and agentically augmented datasets, and rigorous evaluation across knowledge, skills, and tool proficiency. Our key contributions are: (1) assembling an 11.8B-token corpus of cybersecurity data for domain-specific continual pretraining, (2) constructing a 266K-sample augmented dataset via an agentic pipeline for supervised fine-tuning, followed by preference alignment with open-source data, (3) introducing RedSage-Bench, a benchmark with 30K MCQs for broad coverage and 240 open-ended Q&A items for quality evaluation across knowledge, skills, and tools, and (4) RedSage, an open 8B model with data and code, achieving state-of-the-art results on established cybersecurity benchmarks while also improving on general benchmarks.

2 RELATED WORKS

2.1 Cybersecurity Benchmarks

General Knowledge. Several benchmarks assess LLMs' understanding of core cybersecurity concepts via structured Q&A. SecEval (Li et al., 2023) includes 2K+ MCQs across nine domains (web, system, application security). CyberMetric (Tihanyi et al., 2024) provides 10K MCQs generated with RAG and expert validation, spanning penetration testing and network security. CyberBench (Liu et al., 2024) extends beyond MCQs to tasks such as NER, summarization, and classification. SECURE (Bhusal et al., 2024) targets Industrial Control Systems with domain-specific MCQs on risk reasoning and vulnerability analysis. CS-Eval (Yu et al., 2024) covers 42 subcategories across three cognitive levels (Knowledge, Ability, Application) using MCQs, multi-answer, T/F, and open-ended items. SecBench (Jing et al., 2024) offers 44,823 MCQs and 3,087 SAQs in Chinese and English, capturing both factual recall and logical reasoning.

Table 2: Comparison of cybersecurity-tuned LLM training datasets. Pretraining and curated columns report token counts (B = billion, M = million). SFT reports the number of supervision samples. \checkmark = present; \nearrow = absent; \nearrow N/R= not reported.

Name	Pretrain Tokens (B)	Curated Tokens (M)	SFT Samples	Agentic Augmented	Open Data	Open Model
PRIMUS	2.57	191	835	Х	✓	✓
Foundation-Sec-8B	5.10	X	28K	X	Х	✓
DeepHat	×	X	>1M	X	Х	✓
Lily-Cybersecurity-7B	×	X	22K	X	Х	✓
Cyber-DAP	×	119	×	X	Х	×
SecGemini (closed)	N/R	N/R	N/R	X	X	Х
Ours (RedSage)	11.7	850	266K	✓	✓	1

Dataset statistics are compiled from official publications, technical reports, and model cards.

Applications and Agentic Tasks. Application-oriented benchmarks probe reasoning beyond recall. *CTIBench* (Alam et al., 2024) defines four tasks: MCQs, common vulnerabilities and exposures(CVE)-to-common weakness enumeration(CWE) mapping, common vulnerability scoring system (CVSS) prediction, and threat actor attribution in cyber threat intelligence. *CyberSecEval* (Wan et al., 2024) examines model risks across eight areas (e.g., exploit generation, prompt injection). Agentic evaluations such as *NYU-CTF* (Shao et al., 2024) and *CyBench* (Zhang et al., 2025) assess red-team capabilities through capture the flag (CTF) challenges (e,g web exploitation, reverse engineering) in interactive settings.

While these efforts advance evaluation of knowledge and applications, they rarely isolate competence in understanding and operating security tools or systematically assess the quality of free-form responses. As summarized in Table 1, most benchmarks specialize in either knowledge or applications, and even agentic ones lack explicit tool-focused assessment. We address these gaps with RedSage-Bench, which jointly measures knowledge, skills, and tool proficiency (Fig. 2).

2.2 Cybersecurity Datasets and Models

Early Cybersecurity Datasets. Early domain-specific models such as *CyBERT* (Ranade et al., 2021), *SecureBERT* (Aghaei et al., 2022), and *CTI-BERT* (Park & You, 2023) showed the value of domain-adaptive fine-tuning. However, their datasets were not released. Moreover, as encoder-based models, they require task-specific fine-tuning, restricting scalability.

Cybersecurity Datasets for LLMs. With the advent of LLMs, several groups curated cybersecurity-specific corpora. *PRIMUS* (Yu et al., 2025) (Trend Micro) provides 2.75B tokens for continued pretraining, 835 samples for supervised fine-tuning, and reasoning distillation, extending Llama-3.1-8B-Instruct into Llama-Primus-Base and -Merged. *Foundation-Sec-8B* (Kassianik et al., 2025a) (Cisco) collects 5.1B tokens via large-scale scraping and filtering, continuing pretraining on Llama-3.1-8B-Base and adding a cybersecurity post-training stage, though its dataset remains closed. Community efforts include *DeepHat* (formerly WhiteRabbitNeo), reportedly trained on 1M+ Q&A pairs for real workflows (Deep Hat, 2025), and *Lily-Cybersecurity*, which fine-tunes Mistral-7B on 22K hand-crafted and lightly refined conversations (Sego Lily Labs, 2024). *Cyber-DAP* (Salahuddin et al., 2025) highlights the effectiveness of smaller curated corpora for continued pretraining, while *SecGemini* (Google Security Blog, 2025) offers a closed model with live threat intelligence but unreleased data. We summarize these datasets in Table 2.

Unlike prior work with limited augmentation, we introduce *agentic augmentation* to transform curated cybersecurity resources into diverse, realistic multi-turn dialogs simulating expert—assistant workflows across knowledge, offensive operations, and tool proficiency for domain-specific finetuning. RedSage is, to our knowledge, the only effort combining large-scale continual pretraining, curated data, agentically augmented SFT, and full openness (data, model, and code) (Table 2).

3 REDSAGE

We build RedSage through a data-centric pipeline comprising (1) large-scale filtering of cybersecurity text and curation of high-quality resources for continual pretraining, (2) agentic augmentation to create supervised fine-tuning data, and (3) benchmark construction for evaluation (Fig. 3).

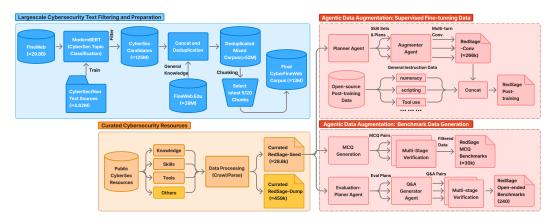


Figure 3: RedSage data pipeline combining large-scale text collection, curated cybersecurity resources, and agentic augmentation for supervised fine-tuning and benchmark generation. *Best viewed in Zoom.*

3.1 REDSAGE PRE-TRAINING DATA

CyberFineWeb. We construct CyberFineWeb by filtering FineWeb (Penedo et al., 2024a), a cleaned large-scale web corpus aggregated from Common Crawl (2013–2024; ~15T tokens). To extract cybersecurity content, we fine-tune a binary classification model based on ModernBERT-base (Warner et al., 2024), a state-of-the-art encoder trained on 2T+ tokens. Applying this filter yields a *cybersecurity candidate pool* of ~125M documents (~89.8B tokens).

To avoid catastrophic forgetting on general knowledge, we mix CyberFineWeb with general-knowledge samples from FineWeb-Edu (Lozhkov et al., 2024) at a 30% replay ratio. FineWeb-Edu is a 1.3T-token educational subset shown to improve general LLM benchmarks. This strategy follows prior work on replay-based continual learning (Ibrahim et al., 2024; Guo et al., 2025), though unlike dynamic replay, we embed these examples directly into the static corpus. We then apply global near-duplicate removal with MinHash-LSH over the combined data. This yields a deduplicated mixed corpus of \sim 52M documents (\sim 46.8B tokens), while inheriting FineWeb's upstream extensive filtering and PII removal.

Finally, we partition the deduplicated corpus into 20 chronological chunks for sequential training under compute constraints and apply early stopping after 5 chunks to control training cost. This yields the *final CyberFineWeb corpus*: ~13M documents (~11.7B tokens) used in our model. Implementation details, including classifier training, deduplication parameters, and datasets statistics, are provided in Appendix A.1.

RedSage-Seed. Web-filtered text offers broad coverage, but its reliability is not assured. To provide high-quality content, we curate RedSage-Seed: 28,637 samples (\sim 0.15B tokens) from publicly available sources organized into three categories: *Knowledge* (well-established cybersecurity frameworks and knowledge bases (MITRE Corporation, 2025a;b;c; The OWASP Foundation, 2025)), *Skills* (penetration-testing write-ups (0xdf, 2025), hacking techniques (HackTricks, 2025), payload examples (swisskyrepo, 2025), and ethical hacking tutorials/blogs (Null Byte, 2025; Chandel, 2025)), and *Tools* (CLI cheat-sheets (tldr pages, 2025), Linux manuals (linux.die.net, 2025), Kali tools (Kali, 2025)). We additionally collect an uncategorized dump of \sim 459K documents (\sim 0.7B tokens) from trusted cybersecurity sources (Appendix A.2) to supply extra pretraining tokens.

To process these resources, we crawl web-based sources and convert them to Markdown using ReaderLM-v2 (Wang et al., 2025), while downloadable resources are parsed directly. This hierarchical Markdown format preserves structure and enables effective chunking for subsequent augmentation stages. Only the categorized seeds are used for augmentation, while both sets support pretraining. Full statistics, categorization, processing steps, and examples are in Appendix A.2.

3.2 REDSAGE POST-TRAINING DATA

Agentic Data Augmentation. To enable assistants capable of realistic security dialogues, we augment RedSage-Seed into multi-turn conversations using an agentic framework inspired by AgentIn-

Table 3: Statistics of RedSage-Seed (curated pretraining corpus) vs. RedSage-Conv (augmented SFT data) by category. Columns show sample counts, average tokens, and total tokens.

Category		Seed		Conversation			
	Samples	Avg. Tokens	Tokens (M)	Samples	Avg. Tokens	Tokens (M)	
Knowledge – General	6,924	2,370	16.4	67,635	1,326	89.6	
Knowledge – Frameworks	3,715	2,935	10.5	39,908	1,285	51.0	
Skill – Offensive	4,032	9,478	37.8	38,870	1,345	52.3	
Tools – CLI	12,943	5,774	78.9	109,261	1,331	145.7	
Tools – Kali	1,023	6,693	6.3	10,506	1,356	14.3	
Total	28,637	5,231	149.8	266,180	1,326	353.0	
Cybersecurity Dumps	459,473	1,524	700,1	_	_	_	

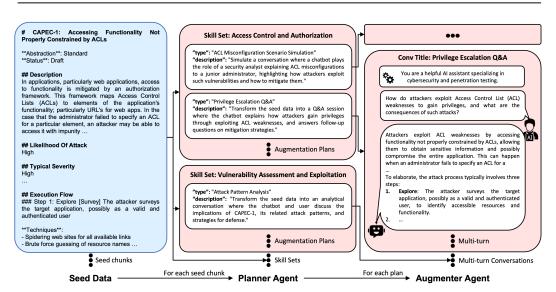


Figure 4: Agentic data augmentation pipeline. Seed data (e.g., CAPEC attack patterns) is processed by the *Planner Agent* into skill sets and augmentation plans, which the *Augmenter Agent* instantiates as grounded, role-based multi-turn cybersecurity dialogues for supervised fine-tuning (SFT).

struct (Mitra et al., 2024). Unlike prior work with fixed skill templates, our *Planner Agent* analyzes each seed data chunk and derives candidate skill sets (e.g., vulnerability analysis, tool-command generation) along with augmentation strategies that describe how the seed is transformed, adapted into a conversational or Q&A format, and enriched with explanations. We enforce guidelines on relevance, diversity, creativity, detail, and formatting. The *Augmenter Agent* then instantiates each plan into realistic, role-based multi-turn dialogues grounded in the seed data. This pipeline scales efficiently, producing multiple dialogues per skill set and filtering outputs for format validity, consistency, and topical relevance. Overall, it yields RedSage-Conv with \sim 266K multi-turn conversations (\sim 352M tokens), expanding total samples by 9.2× and tokens by 2.3× across knowledge, skills, and tools while preserving technical depth (Tab. 3). Fig. 4 illustrates the augmentation pipeline, while detailed statistics, prompts, and examples are provided in Appendix A.3.

General instruction integration. While domain-specific conversations ground the assistant in cybersecurity, effective models must also handle broader instruction-following tasks. We therefore complement RedSage-Conv with curated post-training SFT data from SmolLM3 (Bakouch et al., 2025)¹, focusing on its non-reasoning subset. This corpus adds coverage of summarization, numeracy, data interpretation, temporal and unit reasoning, commonsense knowledge, step-by-step planning, technical writing, scripting, and general tool use. The combination of cybersecurity-specific and general instruction data yields a high-quality post-training corpus, enabling a cybersecurity assistant that performs specialized tasks while retaining broad capabilities.

¹General SFT datasets: HuggingFaceTB/smoltalk2

3.3 REDSAGE BENCHMARK

Multiple-choice Q&A generation. We derive MCQs from RedSage-Seed as follows: for each seed item, a strong open instruction-tuned LLM ² generates several MCQs under guidelines: items are self-contained and closed-book, target stable domain facts/procedures, follow a four-option format with three plausible distractors, and satisfy diversity and formatting constraints.

Open-ended Q&A generation. We extend RedSage-Seed into open-ended Q&A using an agentic augmentation framework with two stages: (1) an *Evaluation-Planer* analyzes seed artifacts and proposes realistic evaluation types with instruction templates and answer guidelines; (2) a *Question-Answer Generator* instantiates each plan into a self-contained open-ended Q&A with a natural-language prompt and a reference answer. All open-ended Q&A are grounded in the seed data and scored with a reference-based LLM-as-judge rubric that evaluates both factual correctness (True/False) and answer quality (0–10) across helpfulness, relevance, depth, and level of detail.

Multi-stage verification. For MCQs, we apply a two-stage pipeline: $Stage\ 1$ (structural validity) uses a verifier LLM 2 with a checklist on format, correctness, distractors, topical relevance, and consistency, filtering items by pass/fail; $Stage\ 2$ (quality scoring) then applies the same verifier LLM 2 to assign each remaining item a score $s\in[0,10]$ for clarity, correctness, and assessment value. We then select the pairs where s>8 and apply quota-aware random sampling to ensure taxonomic balance, yielding 30,000 MCQ—answer pairs evenly split across knowledge, skills, and tools. For open-ended Q&A, we directly perform LLM-based quality scoring in $Stage\ 2$ followed by human verification, selecting 240 high-quality pairs evenly distributed across categories.

All implementation details, intermediate outputs, and qualitative examples appear in Appendix A.4.

3.4 REDSAGE TRAINING

We build RedSage using the Axolotl framework (Axolotl, 2023), with continued pretraining of the open-source base model, Qwen3-8B-Base (Yang et al., 2025), on cybersecurity corpora, followed by post-training through supervised fine-tuning on augmented conversations and preference alignment. Further training details are provided in Appendix B.

Training setup. Continued pretraining (CPT) uses the CyberFineWeb corpus followed by RedSage-Seed (Sec. 3.1). We train the model for one epoch with a maximum context length of 32,768 tokens using distributed optimization ($32 \times A100\text{-}64\text{GB}$, global batch size 1024) with DeepSpeed Stage 3, AdamW, and a fixed learning rate of 2.5×10^{-6} with linear warmup.

After pre-training, we further fine-tune our base model on RedSage-Conv and general SFT data (Sec. 3.2) using a cosine learning rate schedule. We apply direct preference optimization (DPO) (Rafailov et al., 2023) with open-source Tulu 3 8B Preference Mixture dataset (Lambert et al., 2025) using original hyperparameters.

4 EXPERIMENTS AND RESULTS

We evaluate the performance of our cybersecurity-tuned LLM on (1) our own benchmark (Sec. 3.3), (2) related cybersecurity benchmarks, and (3) general LLM benchmarks.

Evaluation setup. For replicable results, we implement and evaluate RedSage-Bench and prior cybersecurity benchmarks in HuggingFace lighteval (Habib et al., 2023). MCQ benchmarks are scored with normalized log-likelihood accuracy over answer options, while instruction-tuned models and structured output tasks use prefix exact match or regex matching on greedy decoding outputs (temperature=0). Details for each task are provided in Appendix C.1.

Baseline methods. We evaluate RedSage against both open general-purpose and cybersecurity-tuned LLMs. General-purpose baselines include Llama-3.1-8B (Grattafiori et al., 2024) and Qwen3-8B (Yang et al., 2025), while specialized baselines include Llama-Primus (Base, Merged) (Yu et al., 2025), Foundation-Sec (Base, Ins) (Kassianik et al., 2025b; Weerawardhena et al., 2025), Lily-Cybersecurity-7B-v0.2 (Sego Lily Labs, 2024), and DeepHat-V1-7B (Deep Hat, 2025). Base mod-

²Teacher and Verifier LLM: meta-llama/Llama-3.3-70B-Instruct, Qwen/Qwen2.5-72B-Instruct

els are evaluated with text completion, instruction-tuned ones with official prompt templates, and we ran Qwen3 in non-reasoning mode for fairness.

Our RedSage variants include three base models: **RedSage-8B-CFW** (CyberFineWeb only), **RedSage-8B-Seed** (Seed only), and **RedSage-8B-Base** (CyberFineWeb followed by Seed). We further derive instruction-tuned variants: **RedSage-8B-Ins** (instruction-tuned from Base) and the final **RedSage-8B-DPO**, which combines all data and applies DPO alignment.

Results on RedSage-Bench. For MCQs, both base and instruction-tuned models are tested in the 0-shot setting, with Tab. 4 showing that all RedSage variants outperform baselines across categories. For open-ended Q&A, we evaluate instruction-tuned models using an LLM-as-Judge rubric to assess both factual correctness and answer quality (Sec. 3.3). As shown in Fig. 5, RedSage achieves not only high accuracy but also the best answer quality across categories. More detailed results and qualitative examples illustrating model predictions are provided in Appendix C.2.

Table 4: RedSage-MCQ (0-shot). Values are accuracy (%). Abbreviations: Gen = General, Frm = Frameworks, Off = Offensive Skills, CLI = Command-line Tools, Kali = Kali Tools. Bold numbers indicate the best result; underlined numbers indicate the second best.

Model Name	Macro	Know	Knowledge		То	ols
	Acc	Gen	Frm	Off	CLI	Kali
Base Mo	del Evalua	tion (Tex	t Comple	tion)		
Llama-3.1-8B	78.02	77.42	75.26	82.78	77.78	72.12
Foundation-Sec-8B	78.51	76.82	79.10	83.68	76.64	71.14
Qwen3-8B-Base	84.24	83.08	81.94	88.23	85.08	78.86
RedSage-8B-CFW	84.86	83.62	83.30	88.81	85.30	79.32
RedSage-8B-Seed	85.21	83.64	84.56	88.82	85.50	79.90
RedSage-8B-Base	<u>85.05</u>	83.12	84.94	88.72	85.44	79.36
Instruct Me	odel Evalu	ation (w/	Chat Ter	nplate)		
Lily-Cybersecurity-7B-v0.2	71.19	68.78	67.44	76.61	71.44	66.26
Llama-Primus-Merged	74.81	74.34	72.34	79.31	74.74	68.82
Foundation-Sec-8B-Instruct	76.12	74.50	77.10	80.91	74.98	68.30
Llama-Primus-Base	77.02	76.78	74.10	80.87	76.78	72.72
Llama-3.1-8B-Instruct	77.05	76.06	73.30	80.90	78.72	72.40
DeepHat-V1-7B	80.18	77.26	76.90	85.07	81.94	74.82
Qwen3-8B	81.85	80.46	78.82	86.16	83.92	75.56
RedSage-8B-Ins	85.73	84.20	84.98	89.06	86.80	80.30
RedSage-8B-DPO	84.83	82.48	83.80	88.54	86.30	79.30

MCQs Analysis. Qwen3-8B-Base, trained on 36T tokens, is the strongest external baseline (84.24) and even outperforms Foundation-Sec-8B. underscoring the importance of selecting a strong base model. Building on it with CPT, RedSage gains up to +0.97 macroaccuracy points, with the largest improvements in Frameworks (+3.00) and *Kali* (+1.04). RedSage-8B-Seed achieves the best base result (85.21), demonstrating better alignment with the curated Seed Among instruction-tuned models, RedSage avoids the accuracy drop and exceeds Qwen3 by +2.98 (DPO) to +3.88 (Ins). DPO on general data slightly lowers accuracy but stays well above baselines. These results highlight that our domain-aware CPT and SFT enhance robustness across cybersecurity knowledge, skills, and tools.

Open-ended QA Analysis. RedSage-8B-DPO achieves the best performance (Fig. 5), surpassing the second-best model (Qwen3-8B) by +7% absolute mean correctness and +0.07 in mean quality score. RedSage-8B-Ins attains similar correctness to Qwen3-8B but lags in answer quality (6.43), underscoring the role of preference alignment in producing not only accurate but also helpful responses. The remaining models fall substantially behind, with mean correctness ranging from 51% to 40% and quality scores from 5.84 to 4.28, highlighting a significant gap from the top three. The faceted violin plots further reveal category difficulty: knowledge tasks exhibit higher and tighter distributions, skill tasks lie in the middle range, and tool-use tasks show lower medians with heavy tails, pinpointing tool expertise as the primary challenge. These findings demonstrate the value of our benchmark for assessing cybersecurity capabilities in free-form answer.

Results on Cybersecurity Benchmarks. We assess generalization on multiple established benchmarks in Tab. 5. For CyberMetric (CyMtc) (Tihanyi et al., 2024), we evaluate all models using the 500 human-verified MCQs. We select English (En) MCQs from SecBench (ScBen) (Jing et al., 2024). We also include MCQs related to the Computer Security (CSec) from MMLU (Hendrycks et al., 2021b). For SECURE (Bhusal et al., 2024), we evaluate models using the MCQs types covering MEAT, CWET, and KCV. Further, we evaluate all model on CTI-Bench (Alam et al., 2024) (MCQ, Root Cause Mapping (RCM)), , and SecEval (ScEva) (Li et al., 2023) (MCQ). We provide further details about each benchmark and metrics in Appendix C.3. Base models are evaluated with 5-shot prompting, and instruction-tuned models in 0-shot.

Analysis. Across related cybersecurity benchmarks, RedSage base models improve over Qwen3-8B-Base (80.81%) by up to +3.75 points. CPT with CFW leads on SecBench (83.62), CyMtc

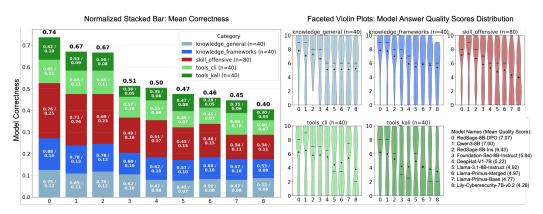


Figure 5: RedSage open-ended QA evaluation. Left: normalized stacked bar charts of mean correctness by category (0–1), where values inside each segment show the mean and its relative contribution. Models are ordered along the x-axis by overall mean correctness. Right: faceted violin plots of LLM-as-Judge quality scores (0–10) per category, showing score distributions across models. Black dots mark means and horizontal lines mark medians. *Best viewed in Zoom*.

Table 5: Benchmark results for Base and Instruct Models. Values are Accuracy (%). Rows are sorted by mean performance. Best results are in bold, second-best are underlined.

Model Name	Mean	CTI-I	Bench	CyMtc	MMLU	ScBen	ScEva	5	SECURI	E
		MCQ	RCM	500	CSec	En	MCQ	CWET	KCV	MEAT
		1	Base Moa	lel Evaluai	ion (5-shot)				
Llama-3.1-8B	75.44	61.12	65.80	84.20	83.00	72.80	54.27	86.34	83.73	87.72
Foundation-Sec-8B	76.90	62.40	75.40	86.60	80.00	69.86	55.64	88.01	84.38	89.78
Qwen3-8B-Base	80.81	68.80	63.50	92.00	83.00	82.84	75.60	92.70	75.05	93.81
RedSage-8B-CFW	82.66	68.40	67.60	93.80	86.00	83.62	76.10	93.33	81.34	93.72
RedSage-8B-Seed	84.45	70.80	78.60	92.20	88.00	81.61	75.96	93.12	85.47	94.28
RedSage-8B-Base	84.56	71.04	<u>78.40</u>	92.60	<u>87.00</u>	<u>81.76</u>	75.83	93.22	87.20	94.00
		In	struct Mo	odel Evalu	ation (0-sho	ot)				
Lily-Cybersecurity-7B-v0.2	55.74	30.04	43.60	65.20	68.00	57.65	39.72	72.99	49.67	74.79
Llama-3.1-8B-Instruct	68.52	58.24	58.30	82.80	72.00	59.66	35.37	84.98	82.86	82.47
Llama-Primus-Merged	71.23	55.92	68.50	83.80	76.00	64.91	39.31	86.13	82.65	83.88
Llama-Primus-Base	71.69	52.32	68.50	83.80	79.00	63.68	61.15	88.01	65.08	83.69
DeepHat-V1-7B	75.44	62.08	68.20	86.00	74.00	70.63	56.65	87.07	86.77	87.54
Foundation-Sec-8B-Instruct	75.44	63.24	69.40	83.00	76.00	68.78	65.46	85.82	82.00	85.29
Qwen3-8B	75.71	62.76	54.00	88.60	76.00	73.26	65.46	88.11	87.42	85.75
RedSage-8B-Ins	81.30	70.56	76.70	89.80	78.00	79.91	72.48	91.45	81.34	91.47
RedSage-8B-DPO	<u>81.10</u>	70.84	<u>70.60</u>	90.00	79.00	80.06	74.22	<u>91.35</u>	82.86	<u>91.00</u>

(93.80), and CWET (93.33), raising the mean by +1.85. CPT with Seed excels on CTI-RCM (78.60), MMLU-CSec (88.00), and MEAT (94.28), lifting the mean by +3.64. Combining both yields the best overall mean (84.56) and top scores on CTI-MCQ (71.04) and KCV (87.20). In the 0-shot instruct setting, RedSage surpasses Qwen3-8B (75.71%) by +5.39 (DPO) to +5.59 (Ins). Except for Lily-Cybersecurity, all domain-tuned baselines outperform Llama-3.1-8B-Instruct, though still lag behind RedSage. These results show that CyberFineWeb and Seed provide complementary strengths, while different post-training strategies specialize across tasks, together setting new state-of-the-art performance in cybersecurity LLM evaluation.

4.1 EVALUATION RESULTS ON GENERAL BENCHMARKS

We use benchmarks from the Open LLM Leaderboard in Lighteval, including ARC-Challenge (ARC-C) (Clark et al., 2018), HellaSwag (HSwag) (Zellers et al., 2019), TruthfulQA (TQA) (Lin et al., 2021), MMLU (Hendrycks et al., 2021a), WinoGrande (WinoG) (Sakaguchi et al., 2020), GSM8K (Cobbe et al., 2021), and IFEval (Zhou et al., 2023). Results in Tab. 6 show our instruction-

tuned models achieves competitive results on general tasks, surpassing baselines by a clear margin. Benchmark configurations and evaluation metrics are provided in Appendix C.4.

Table 6: Open LLM Leaderboard Benchmarks. All values are accuracy (%). Bold numbers indicate the best result and underlined numbers indicate the second best.

Model Name	Mean	MMLU	ARC-C	GSM8K	HSwag	TQA	WinoG	IFEvl
	Base	Model Eval	uation (Mea	n excludes II	FEval)			
Llama-3.1-8B	61.15	66.31	58.19	49.05	82.08	35.98	75.30	_
Foundation-Sec-8B	60.24	63.62	58.45	46.17	81.32	38.71	73.16	_
Qwen3-8B-Base	70.86	78.73	68.09	81.73	<u>79.62</u>	43.84	73.16	_
RedSage-8B-CFW	69.31	78.63	66.72	81.12	79.26	38.09	72.06	_
RedSage-8B-Seed	69.58	78.18	65.19	82.34	77.76	<u>42.44</u>	71.59	_
RedSage-8B-Base	69.23	77.80	65.53	82.03	77.96	42.19	69.85	_
	Instru	ct Model Ev	aluation (Me	ean includes	IFEval)			
Lily-Cybersecurity-7B-v0.2	56.98	56.49	58.96	30.86	80.94	48.53	72.06	50.99
Llama-Primus-Base	64.82	65.09	51.19	71.80	79.49	44.62	72.69	68.85
DeepHat-V1-7B	64.89	69.53	57.17	77.94	74.80	33.17	69.06	72.58
Qwen3-8B	65.92	73.59	62.54	75.66	56.70	45.23	62.51	85.21
Llama-Primus-Merged	66.71	66.17	53.07	75.28	79.07	46.52	73.24	73.58
Llama-3.1-8B-Instruct	68.20	67.29	57.51	77.41	78.91	45.93	72.61	77.75
Foundation-Sec-8B-Instruct	69.28	64.11	63.91	77.79	81.35	53.15	68.51	76.17
RedSage-8B-Ins	73.34	77.38	69.62	86.05	79.00	47.75	73.64	79.97
RedSage-8B-DPO	74.33	<u>77.07</u>	71.76	<u>82.71</u>	<u>79.87</u>	<u>52.47</u>	73.01	<u>83.44</u>

IFEval (Instruction-Following Eval) is excluded from base models as it is designed for instruct-tuned models.

Analysis Among base models, Qwen3-8B-Base is strongest overall (70.86) and leads MMLU (78.73) and ARC-C (68.09), while Llama-3.1-8B tops HSwag (82.08) and WinoG (75.30). Red-Sage bases are competitive in mean (69.23–69.58) and achieve task highs, including best GSM8K (82.34, Seed) and second on MMLU (78.63, CFW) and ARC-C (66.72, CFW), where the slight drop may stem from our FineWeb-Edu general-knowledge replay strategy. After instruction tuning, RedSage attains the best overall mean with DPO (74.33) and second with Ins (73.34), setting new highs on ARC-C (71.76, DPO), GSM8K (86.05, Ins), MMLU (77.38, Ins), and leading WinoG (73.64, Ins). Foundation-Sec-8B-Instruct leads HSwag (81.35) and TQA (53.15), and Qwen3-8B leads IFEval (85.21), with RedSage-DPO close (83.44). These patterns indicate complementary effects: Seed boosts math reasoning (GSM8K), CFW strengthens general knowledge and reasoning (MMLU and ARC-C), and DPO improves instruction-following (IFEvI), while RedSage remains competitive on general tasks despite cybersecurity tuning.

5 DISCUSSION AND LIMITATIONS

The data pipeline, which leverages LLM-generated content and verification, scales effectively but may still propagate biases or inaccuracies despite screening. Nevertheless, our benchmark extends existing cybersecurity evaluations, fills missing dimensions, and offers value to the community. Finally, as the model incorporates offensive security knowledge, it carries an inherent risk of misuse. While such dual-use concerns are intrinsic in cybersecurity research, we emphasize the importance of responsible application and good security practices to promote ethical use.

6 CONCLUSION

We presented REDSAGE, an open cybersecurity assistant that combines a large-scale pretraining corpus (CYBERFINEWEB, 11.7B tokens), a curated seed of authoritative resources (REDSAGE-SEED, 29K items, 150M tokens), and 266K augmented dialogues for supervised fine-tuning, together with REDSAGE-BENCH, a 30K-question benchmark spanning knowledge, skills, and tool use. At the 8B scale, REDSAGE achieves state-of-the-art results, surpassing baselines by up to +5.9 points on cybersecurity tasks and +5.0 on general LLM benchmarks, while avoiding the post-tuning degradation observed in prior models. We will release all models, datasets, and code to support reproducibility and accelerate open research on practical and domain-specialized AI assistants for cybersecurity.

7 ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. All datasets used in this study were derived exclusively from publicly available and internet-accessible sources. Our large-scale pretraining corpus builds directly on prior work that already applied extensive filtering, deduplication, and removal of personally identifiable information (PII). We further applied additional quality checks to ensure that the data contain only non-sensitive and appropriately licensed content.

We note that some components of the curated REDSAGE datasets may include publicly available but copyrighted resources (e.g., educational portals, online tutorials, or news articles). Such content was used solely for non-commercial academic research, and we will not redistribute these resources without obtaining the necessary permissions from the rights holders. Only aggregated statistics are reported in this paper, and any public release of datasets will exclude copyrighted material unless explicit approval has been secured.

The REDSAGE models are released strictly for research purposes and not intended for deployment in real-world security operations without additional safeguards. To support responsible use, we will make models, datasets, and code openly available under research-friendly licenses with clear documentation and usage guidelines, promoting transparency, reproducibility, and community benefit.

8 REPRODUCIBILITY STATEMENT

We are committed to advancing reproducibility and open research in cybersecurity-oriented LLMs by releasing our datasets, models, and code. The collection and augmentation of our datasets for domain-aware pre- and post-training are described in Sec. 3, with detailed descriptions, statistics, and implementation details (including prompt templates) provided in Appendix A. Model training procedures are presented in Sec. 3.4, with further implementation details in Appendix B.

Our models are trained using the Axolotl framework (Axolotl, 2023), which facilitates direct replication through reusable configuration files; users need only replace the base model and dataset. All hyperparameters are fully specified in Appendix B. For evaluation, we implement all benchmarks using the HuggingFace LightEval framework (Habib et al., 2023), ensuring reproducible results and supporting evaluation of arbitrary LLMs by specifying the benchmark configuration. Our evaluation protocol, compared models, and benchmark details are documented in Sec. 4 and Appendix C. All datasets, code, and evaluation pipelines will be released as open-source.

REFERENCES

- 0xdf. 0xdf: Penetration testing write-ups and ctf notes. https://0xdf.gitlab.io/, 2025. Accessed: 2025-09-01.
- Ehsan Aghaei, Xi Niu, Waseem Shadid, and Ehab Al-Shaer. Securebert: A domain-specific language model for cybersecurity. In *International Conference on Security and Privacy in Communication Systems*, pp. 39–56. Springer, 2022.
- Md Tanvirul Alam, Dipkamal Bhusal, Le Nguyen, and Nidhi Rastogi. Ctibench: A benchmark for evaluating Ilms in cyber threat intelligence. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50805–50825. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5acd3c628aa1819fbf07c39ef73e7285-Paper-Datasets_and_Benchmarks_Track.pdf.
- Axolotl. Axolotl: A free and open source llm fine-tuning framework, 2023. URL https://github.com/axolotl-ai-cloud/axolotl.
- Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

- Dipkamal Bhusal, Md Tanvirul Alam, Le Nguyen, Ashim Mahara, Zachary Lightcap, Rodney Frazier, Romy Fieblinger, Grace Long Torales, Benjamin A. Blakely, and Nidhi Rastogi. SE-CURE: Benchmarking Large Language Models for Cybersecurity. In 2024 Annual Computer Security Applications Conference (ACSAC), pp. 15–30, Los Alamitos, CA, USA, December 2024. IEEE Computer Society. doi: 10.1109/ACSAC63791.2024.00019. URL https://doi.ieeecomputersociety.org/10.1109/ACSAC63791.2024.00019.
 - Raj Chandel. Hacking articles: Ethical hacking tutorials and write-ups. https://www.hackingarticles.in/, 2025. Accessed: 2025-09-01.
 - Paul Cichonski, Tim Millar, Tim Grance, and Karen Scarfone. Computer security incident handling guide (nist special publication 800-61 revision 3). Technical Report NIST SP 800-61r3, National Institute of Standards and Technology, Gaithersburg, MD, 2021.
 - Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018. URL https://arxiv.org/abs/1803.05457.
 - Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.
 - Deep Hat. Deep hat: Uncensored ai for devsecops, 2025. URL https://www.deephat.ai/. Accessed September 16, 2025. States training on over one million supervised Q&A pairs.
 - GeeksforGeeks. Geeksforgeeks. https://www.geeksforgeeks.org/, 2008. Founded 2008, accessed 2025-09-24; tutorials and educational content.
 - Google Security Blog. Google launches sec-gemini v1: a new experimental ai model for cybersecurity. https://security.googleblog.com/2025/04/google-launches-sec-gemini-v1-new.html, 2025. Blog post; Accessed: 2025-09-16.
 - Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
 - Yiduo Guo, Jie Fu, Huishuai Zhang, and Dongyan Zhao. Efficient domain continual pretraining by mitigating the stability gap. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pp. 32850–32870, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1578. URL https://aclanthology.org/2025.acl-long.1578.
 - Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.
 - HackTricks. Hacktricks: Hacking techniques and tricks. https://book.hacktricks.wiki/en/index.html, 2025. Accessed: 2025-09-01.
 - Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Spencer Frazier, Saurav Kadavath, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv* preprint *arXiv*:2009.03300, 2021a. URL https://arxiv.org/abs/2009.03300.
 - Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.
 - Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=DimPeeCxKO.

- IETF. Request for comments (rfc) series. https://www.rfc-editor.org/, 2025. Accessed: 2025-09-24.
- $(ISC)^2$. 2022 cybersecurity workforce study. Technical report, $(ISC)^2$, 2022.
 - Qixuan Jing, Yuhao Chen, Hailin Huang, Lingzhi Wang, Zhiyu Min, Chenhao Lin, Jie Zhang, Tianwei Zhang, and Xinyu Xing. Secbench: A comprehensive multi-dimensional benchmarking dataset for llms in cybersecurity, 2024. URL https://arxiv.org/abs/2412.20787.
 - Kali. Kali tools official kali linux penetration testing utilities. https://www.kali.org/tools/, 2025. Accessed: 2025-09-01.
 - Paul Kassianik, Baturay Saglam, Alexander Chen, Blaine Nelson, Anu Vellore, Massimo Aufiero, Fraser Burch, Dhruv Kedia, Avi Zohary, Sajana Weerawardhena, Aman Priyanshu, Adam Swanda, Amy Chang, Hyrum Anderson, Kojin Oshiba, Omar Santos, Yaron Singer, and Amin Karbasi. Foundation-sec-8b: A cybersecurity-focused llm built on llama 3.1. Technical report, Foundation AI Cisco Systems Inc., 2025a. URL https://huggingface.co/fdtn-ai/Foundation-Sec-8b. Technical Report.
 - Paul Kassianik, Baturay Saglam, Alexander Chen, Blaine Nelson, Anu Vellore, Massimo Aufiero, Fraser Burch, Dhruv Kedia, Avi Zohary, Sajana Weerawardhena, Aman Priyanshu, Adam Swanda, Amy Chang, Hyrum Anderson, Kojin Oshiba, Omar Santos, Yaron Singer, and Amin Karbasi. Llama-3.1-foundationai-securityllm-base-8b technical report. *arXiv preprint arXiv:2504.21039*, 2025b. doi: 10.48550/arXiv.2504.21039. URL https://arxiv.org/abs/2504.21039.
 - Ansam Khraisat, Ammar Alazab, Mamoun Alazab, et al. Comprehensive review on cybersecurity: modern threats and advanced defense strategies. *Journal of Network and Computer Applications*, 240:103745, 2024.
 - Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.
 - Guancheng Li, Yifeng Li, Wang Guannan, Haoyu Yang, and Yang Yu. Seceval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models. https://github.com/XuanwuAI/SecEval, 2023.
 - Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. arXiv preprint arXiv:2109.07958, 2021. URL https://arxiv.org/abs/2109.07958.
 - linux.die.net. Linux man pages linux.die.net manual repository. https://linux.die.net/man/, 2025. Accessed: 2025-09-01.
 - Zefang Liu, Jialei Shi, and John F Buford. Cyberbench: A multi-task benchmark for evaluating large language models in cybersecurity. In *AAAI 2024 Workshop on Artificial Intelligence for Cyber Security*, 2024.
 - Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.
- Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andres Codas, Yadong Lu, Wei-ge Chen, Olga Vrousgos, Corby Rosset, et al. Agentinstruct: Toward generative teaching with agentic flows. arXiv preprint arXiv:2407.03502, 2024.
 - MITRE Corporation. MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge. https://attack.mitre.org/, 2025a. Accessed: 2025-09-01.

- MITRE Corporation. CAPEC: Common attack pattern enumeration and classification. https://capec.mitre.org/, 2025b. Accessed: 2025-09-01.
- MITRE Corporation. CWE: Common weakness enumeration. https://cwe.mitre.org/, 2025c. Accessed: 2025-09-01.
 - NIST. Nist cybersecurity publications. https://csrc.nist.gov/publications, 2025a. Accessed: 2025-09-24.
 - NIST. National vulnerability database (nvd). https://nvd.nist.gov, 2025b. Accessed: 2025-09-24.
 - Null Byte. Null byte ethical hacking tutorials and white-hat guides. https://null-byte.wonderhowto.com/, 2025. Accessed: 2025-09-01.
 - Youngja Park and Weiqiu You. A pretrained language model for cyber threat intelligence. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 113–122, 2023.
 - Elijah Pelofske, Lorie M Liebrock, Vincent Urias, et al. An enhanced machine learning topic classification methodology for cybersecurity. In *CS & IT Conference Proceedings*, volume 11. CS & IT Conference Proceedings, 2021.
 - Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024a. URL https://openreview.net/forum?id=n6SCkn2QaG.
 - Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv:2406.17557*, 2024b. Introduces FineWeb and FineWeb-Edu; reports strong MMLU/ARC gains for FineWeb-Edu.
 - Guilherme Penedo, Hynek Kydlíček, Alessandro Cappelli, Mario Sasko, and Thomas Wolf. Datatrove: large scale data processing, 2024c. URL https://github.com/huggingface/datatrove.
 - Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://arxiv.org/abs/2305.18290.
 - Priyanka Ranade, Aritran Piplai, Anupam Joshi, and Tim Finin. Cybert: Contextualized embeddings for the cybersecurity domain. In *2021 IEEE International Conference on Big Data (Big Data)*, pp. 3334–3342. IEEE, 2021.
 - roadmap.sh. Cyber security roadmap: Learn to become a cyber security expert, 2025. URL https://roadmap.sh/cyber-security. Accessed: 2025-09-24.
 - Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6429.
 - Salahuddin Salahuddin, Ahmed Hussain, Jussi Löppönen, Toni Jutila, and Panos Papadimitratos. Less data, more security: Advancing cybersecurity llms specialization via resource-efficient domain-adaptive continuous pre-training with minimal tokens. *arXiv* preprint arXiv:2507.02964, 2025. doi: 10.48550/arXiv.2507.02964. URL https://arxiv.org/abs/2507.02964.
 - Sego Lily Labs. Lily-cybersecurity-7b-v0.2 (model card). https://huggingface.co/segolilylabs/Lily-Cybersecurity-7B-v0.2, 2024. Accessed: 2025-09-16.

- Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Kimberly Milner, Boyuan Chen, Max Yin, Siddharth Garg, Prashanth Krishnamurthy, Farshad Khorrami, et al. Nyu ctf bench: A scalable open-source benchmark dataset for evaluating llms in offensive security. *Advances in Neural Information Processing Systems*, 37:57472–57498, 2024.
- swisskyrepo. Payloadsallthethings: Useful payloads and bypasses. https://github.com/swisskyrepo/PayloadsAllTheThings, 2025. Accessed: 2025-09-01.
- The OWASP Foundation. OWASP Top 10: The ten most critical web application security risks. https://owasp.org/www-project-top-ten/, 2025. Accessed: 2025-09-01.
- The Hacker News. Cybersecurity news reports. https://thehackernews.com/, 2025. Accessed: 2025-09-24.
- Norbert Tihanyi, Mohamed Amine Ferrag, Ridhi Jain, Tamas Bisztray, and Merouane Debbah. 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. doi: 10.1109/CSR61664.2024.10679494.
- tldr pages. tldr-pages: Community-maintained command-line cheat sheets. https://github.com/tldr-pages/tldr, 2025. Accessed: 2025-09-01.
- Shengye Wan, Cyrus Nikolaidis, Daniel Song, David Molnar, James Crnkovich, Jayson Grace, Manish Bhatt, Sahana Chennabasappa, Spencer Whitman, Stephanie Ding, et al. Cyberseceval 3: Advancing the evaluation of cybersecurity risks and capabilities in large language models. *arXiv* preprint arXiv:2408.01605, 2024.
- Feng Wang, Zesheng Shi, Bo Wang, Nan Wang, and Han Xiao. Readerlm-v2: Small language model for html to markdown and json. *arXiv preprint arXiv:2503.01151*, 2025. URL https://arxiv.org/abs/2503.01151.
- Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv* preprint arXiv:2412.13663, 2024.
- Sajana Weerawardhena, Paul Kassianik, Blaine Nelson, Baturay Saglam, Anu Vellore, Aman Priyanshu, Supriti Vijay, Massimo Aufiero, Arthur Goldblatt, Fraser Burch, Ed Li, Jianliang He, Dhruv Kedia, Kojin Oshiba, Zhouran Yang, Yaron Singer, and Amin Karbasi. Llama-3.1-foundationai-securityllm-8b-instruct technical report. arXiv preprint arXiv:2508.01059, 2025. doi: 10.48550/arXiv.2508.01059. URL https://arxiv.org/abs/2508.01059.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.
- Yao-Ching Yu, Tsun-Han Chiang, Cheng-Wei Tsai, Chien-Ming Huang, and Wen-Kwang Tsao. Primus: A pioneering collection of open-source datasets for cybersecurity llm training. *arXiv* preprint arXiv:2502.11191, 2025.
- Zhengmin Yu, Jiutian Zeng, Siyi Chen, Wenhan Xu, Dandan Xu, Xiangyu Liu, Zonghao Ying, Nan Wang, Yuan Zhang, and Min Yang. Cs-eval: A comprehensive large language model benchmark for cybersecurity. *arXiv preprint arXiv:2411.16239*, 2024.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. URL https://aclanthology.org/P19-1472.

Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Julian Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny O Oseleononmen, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tc90LV0yRL.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

A DATASET DETAILS

This section details the datasets we created and curated for training our LLM. All token counts are computed with the GPT-2 tokenizer³, following the conventions of FineWeb (Penedo et al., 2024b).

A.1 CYBERFINEWEB

CyberFineWeb is derived from the original FineWeb dataset (Penedo et al., 2024b)⁴, a large-scale, cleaned web corpus aggregated from Common Crawl. Although FineWeb is continuously updated, for our development we used all subsets released between Summer 2013 (CC-MAIN-2013-20) and December 2024 (CC-MAIN-2024-51). This selection comprises 104 subsets, totaling 46,934 GB of data and 17.2 trillion tokens.

Text Classification Model To extract the cybersecurity corpus from FineWeb, we trained a text classification model based on ModernBERT-base (Warner et al., 2024), a state-of-the-art transformer encoder. The training data came from the Cybersecurity Topic Classification dataset (Pelofske et al., 2021), which contains 9.27M labeled training samples (cybersecurity vs. non-cybersecurity) collected from Reddit, StackExchange, and arXiv, along with 459K validation samples from web articles. To reduce context ambiguity, we filtered out very short texts, yielding 4.62M training samples and 2.46K validation samples. The model was trained with the Adam optimizer for 2 epochs using a learning rate of 2e-5 and a 10% warmup ratio. On the validation set, the model achieved 93.8% precision, 90.2% recall, 91.4 % F1 score and 97.3% accuracy.

Text Filtering We applied the trained classifier to each subset of FineWeb. Figure 6 shows the number of identified cybersecurity samples and their relative proportion across all subsets ordered by crawl date. The results indicate a steady increase in cybersecurity-related content on the web, underscoring the growing importance of this domain. In total, this filtering process produced approximately 125M documents (~89.8B tokens), corresponding to about 0.77% of the original FineWeb.

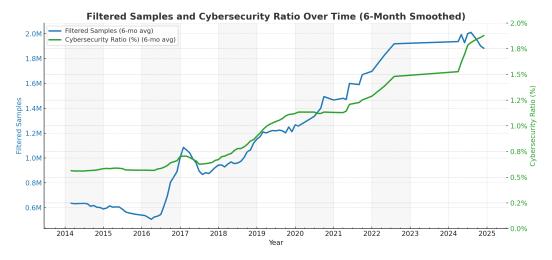


Figure 6: Number of filtered cybersecurity samples and their ratio over time across FineWeb subsets.

General Knowledge Integration Due to compute constraints, we partitioned the dataset into 20 chronological chunks. To mitigate catastrophic forgetting of general-domain knowledge, we first select a fixed 100B-token subset from FineWeb-Edu (Lozhkov et al., 2024). For each chunk, we then randomly resampled data from this subset to match 30% of the chunk's size, ensuring balanced exposure to general-domain content throughout training.

Deduplication Although FineWeb includes text deduplication in its pipeline, it is applied only within individual CommonCrawl dumps. We applied global deduplication across our mixed corpus using MinHash-LSH implemented in DataTrove (Penedo et al., 2024c), with 64-bit precision,

³GPT-2: openai-community/gpt2

⁴FineWeb Datasets: HuggingFaceFW/fineweb

14 buckets, and 8 hashes per bucket. This reduced the corpus size by 58.4% in documents (to $\sim 52M$) and by 47.9% in tokens (to $\sim 46.8B$).

Final Corpus To fit our training budget, we selected the latest 5 chunks from the mixed, deduplicated data. This formed our final pretraining corpus, containing \sim 13M documents (\sim 11.7B tokens). A summary of the dataset filtering and processing steps from FineWeb to the final CyberFineWeb corpus is provided in Table 7.

Table 7: Summary of dataset filtering and processing stages from FineWeb to the final CyberFineWeb corpus. Retention percentages are relative to the original FineWeb.

Stage	Documents	Tokens	Retention (vs. FineWeb)
FineWeb (2013–2024, 104 subsets)	\sim 24.5B	~17.2T	100%
CyberFineWeb (after filtering)	\sim 125M	\sim 89.8B	0.51% docs / 0.52% tokens
General-mixing + deduplication (20 chunks)	\sim 52M	\sim 46.8B	0.21% docs / 0.27% tokens
Final CyberFineWeb corpus (latest 5 chunks)	$\sim 13M$	\sim 11.7B	0.053% docs / 0.068% tokens

A.2 REDSAGE SEED

RedSage Seed. Our curated collection of publicly available cybersecurity resources is designed to provide high-quality pretraining data in structured Markdown format. We excluded private resources such as books to ensure that all data are openly accessible.

Some resources, such as MITRE ATT&CK, CAPEC, and CWE (MITRE Corporation, 2025a;b;c), are distributed as XML files, which we parsed into structured Markdown while preserving the original website organization. Other resources, such as tldr-pages (tldr pages, 2025) and kali-tools (Kali, 2025), were already available in Markdown format. For curated webpages, we crawled and processed them using Jina ReaderLM-v2 (Wang et al., 2025) to convert the HTML content into Markdown.

The RedSage-Seed corpus is organized into three main categories: *knowledge*, *skills*, and *tools*. Within **knowledge**, we distinguish between (i) *General*, which includes sources such as Wikipedia and roadmap.sh (roadmap.sh, 2025), and (ii) *Frameworks*, which cover structured knowledge bases from MITRE and the OWASP Foundation (The OWASP Foundation, 2025). For **skills**, we currently focus on offensive security, curating resources such as offensive tricks (HackTricks, 2025), articles (Chandel, 2025), community tutorial (Null Byte, 2025), and CTF write-ups (0xdf, 2025). Finally, **tools** are divided into (i) *CLI*, which includes multi-platform command-line resources such as tldr-pages (tldr pages, 2025) and Unix man pages, and (ii) *Kali Linux Tools* (Kali, 2025), which provide documentation for a curated set of cybersecurity tools. Dataset statistics and detailed categorization are presented in Table 8. These resources also serve as the foundation for our agentic augmented cybersecurity conversations and benchmarking.

RedSage Dump. To complement RedSage-Seed and expand the diversity of high-quality data for cybersecurity pretraining, we curated additional publicly available resources under the RedSage Dump collection. This corpus aggregates technical documents, standards, and domain-specific reports that are particularly relevant for developing a cybersecurity assistant. Specifically, it includes: (i) *Computer Education Portals* (GeeksforGeeks, 2008), which provide structured tutorials and training materials on computer science and cybersecurity fundamentals; (ii) *Cybersecurity News* (The-HackerNews, 2025), capturing timely reports and analyses of emerging threats and incidents; (iii) *RFC Entries* (IETF, 2025), representing standardized internet protocols and technical specifications; (iv) *NIST Publications* (NIST, 2025a), offering authoritative cybersecurity and compliance guidelines; (v) *Primus Seed* (Yu et al., 2025), a curated collection of cybersecurity resources originally used to pretrain the Primus model; and (vi) the *National Vulnerability Database* (*NVD*) (NIST, 2025b), which provides structured vulnerability advisories.

Statistics for these sources are summarized in Table 9. Overall, the RedSage Dump contains 459K documents with a total of \sim 700M tokens. This collection complements RedSage-Seed by emphasizing technical standards, structured vulnerability data, and up-to-date cybersecurity reporting.

Table 8: RedSage Seed Statistics by Category: Samples and Tokens

Configuration	Samples	Avg. Tokens	Total Tokens	Min Tokens	Max Tokens
Knowledge - General					
Cybersecurity Wikis	6,636	2,304.77	15,294,454	39	36,812
Cybersecurity Roadmaps	288	3,671.35	1,057,349	86	171,839
Knowledge – Frameworks	s				
MITRE ATT&CK	1,655	4,806.38	7,954,559	366	96,808
MITRE CAPEC	589	654.42	385,453	61	2,444
MITRE CWE	1,346	1,222.46	1,645,431	140	10,679
OWASP	125	4,313.63	539,204	436	17,253
Skill - Offensive					
Offensive Tricks	1,050	2,924.06	3,070,263	116	29,902
Hacking Articles	1,384	13,919.66	19,264,809	377	190,391
Null Byte Tutorials	1,002	4,402.07	4,410,874	278	79,225
CTF Write-ups	596	18,471.77	11,009,175	185	83,759
Tools - CLI					
TLDR Pages (English)	5,335	11,215.81	59,836,346	35	543,349
Unix Man Pages	7,608	2,509.00	19,088,472	45	379,876
Tools – Kali					
Kali Documentation	265	1,568.08	415,541	53	17,983
Kali Tools	758	7,722.30	5,853,503	169	709,750
Total (dataset)	28,637	5,231.00	149,825,433	35	709,750

Table 9: RedSage Dump Statistics

Source	Samples	Avg. Tokens	Total Tokens
Computer Education Portals	160,355	1,986	318,503,184
Cybersecurity News	13,959	1,431	19,968,138
RFC Entries	9,674	20,994	203,093,862
NIST Publications	1,015	29,715	30,161,170
Primus Seed (Website, Mitre)	80,336	849	68,233,498
National Vulnerability Database (NVD)	194,134	310	60,173,508
Total	459,473	1,524	700,133,360

REDSAGE CONVERSATION A.3

Agentic Data Augmentation. Our supervised finetuning (SFT) cybersecurity datasets are generated using an agentic augmentation pipeline. We first segment the RedSage-Seed corpus into chunks of up to 32,768 tokens using a Markdown text splitter. These chunks serve as the input to the planner agent, which determines appropriate augmentation strategies.

Planner Agent. For each seed data chunk, the planner agent analyzes the content and proposes multiple skill sets, each associated with one or more augmentation types and descriptive transformations. This design enables diverse augmentation paths from the same source material, ensuring broad coverage of cybersecurity skills and tasks. Below is our planner agent's system prompt.

Planner Agent's System Prompt

You are an **Augmentation Type Planner Agent** specializing in cybersecurity and penetration testing. Your role is to analyze a provided chunk of seed data and produce a structured, comprehensive list of possible skill sets and augmentation types. The resulting suggestions will be used by a Data Augmentation Agent to generate conversational training data for a chatbot. Keep in mind that the final output should lend itself well to turn-based dialogues, persona-based Q&A, or scenario simulations typical in a chatbot environment.

Objective:

Given a chunk of preprocessed seed data related to cybersecurity and penetration testing, generate a well-structured list of skill sets and corresponding augmentation types. The suggestions should

improve dataset quality, diversity, and relevance, and be easily adaptable into a conversational format (e.g., question–answer pairs, scenario-based dialogues, guided reasoning steps). Leverage the seed data's domain context to ensure accuracy and practical utility.

Input:

 Seed Data: A chunk of preprocessed markdown-formatted data related to cybersecurity and penetration testing.

Output:

- Structured List of Skill Sets and Augmentation Types (in JSON format):
 - Include multiple skill sets, each with several augmentation types.
 - For each augmentation type, provide a brief description that clarifies its intended transformation and explains how it could be adapted into a conversational format for a chatbot.

Guidelines:

1. Relevance and Grounding:

- Ensure all skill sets and augmentation types are relevant to cybersecurity and penetration testing.
- Ground the augmentations in the seed data to maintain accuracy. If the seed data mentions specific tools, vulnerabilities, or scenarios, align the augmentation suggestions accordingly.

2. Diversity and Coverage:

- Suggest a wide range of augmentation strategies reflecting various penetration testing phases: reconnaissance, enumeration, exploitation, post-exploitation, mitigation, and so forth.
- Include traditional data transformations (e.g., paraphrasing) and advanced, scenario-based augmentations (e.g., simulating a penetration test conversation between a tester and a security analyst).

3. Conversational Adaptability:

- Consider how each augmentation could be represented in a chatbot-friendly format (e.g., multi-turn Q&A, narrative scenarios, role-based conversations, or step-by-step reasoning).
- Example: a vulnerability analysis could become a Q&A where the chatbot explains the vulnerability to a novice, or a roleplay between red-teamer and blue-teamer discussing mitigation.

4. Creativity and Innovation:

- Introduce new skill sets or augmentation ideas beyond predefined examples.
- Encourage creative transformations that leverage the chatbot setting (persona-based coaching, guided threat mapping dialogues, multi-turn explorations).

5. Detail and Clarity:

• Each augmentation type should have a short description explaining what it does, how it relates to the seed data, and how it can be adapted into a conversational format.

6. Format Requirements:

• Return output in JSON format with the following structure:

Example Categories for Inspiration (Not Exhaustive):

- Tool Command Generation: Turn tool usage into Q&A or guided instructions.
- Vulnerability Analysis: Conversational explanations (senior engineer teaching junior tester).
- Attack Surface Enumeration: Brainstorming dialogue between red teamers.
- Exploitation Workflows: Multi-turn walkthroughs of exploitation stages.
- Post-Exploitation Techniques: Conversations explaining persistence and mitigation.
- Threat Mapping: Align findings with MITRE ATT&CK in dialogue form.
- Mitigation Recommendations: Remediation advice in Q&A format.
- Scenario Simulation: Role-based pen-test scenarios (e.g., tester vs client rep).
- Step-by-Step Reasoning: Guided reasoning revealed turn by turn.
- Edge-Case Generation: Ambiguity/error handling dialogues.
- Reading Comprehension: Extracting and explaining vulnerability advisories.

Tack

Use the provided seed data to produce a JSON-structured list of skill sets and augmentation types. Ensure augmentations can be easily converted into conversational formats. Suggest as many diverse and creative transformations as possible. The output must follow the specified JSON format; otherwise, it will be rejected.

For example, given the following seed data:

arp-scan

```
Homepage: https://github.com/royhills/arp-scan
Repository: https://salsa.debian.org/pkg-security-team/arp-scan
Architectures: any
```

Version: 1.10.0-2

 $\textbf{Metapackages:} \ kali\text{-linux-default, kali-linux-everything, kali-linux-headless, kali-linux-large}$

arp-scan

arp-scan is a command-line tool that uses the ARP protocol to discover and fingerprint IP hosts on the local network. It is available for Linux and BSD under the GPL licence.

Installed size: 1.53 MB

How to install: sudo apt install arp-scan

Dependencies:

- libc6
- libcap2
- libpcap0.8t64

arp-fingerprint

Fingerprint a system using ARP

```
root@kali:~# arp-fingerprint -h
Usage: arp-fingerprint [options] <target>
Fingerprint the target system using arp-scan.
```

```
'options' is one or more of:

-h Display this used
```

- -h Display this usage message.
- -v Give verbose progress messages.
- -o <option-string > Pass specified options to arp-scan
- -1 Fingerprint all targets in the local net.

arp-scan

1077

1078

1079

Send ARP requests to target hosts and display responses

```
root@kali:~# arp-scan -h
Usage: arp-scan [options] [hosts...]
```

```
1080
1081
         Target hosts must be specified on the command line unless the
1082
             --file or
1083
         --localnet option is used.
1084
1085
         arp-scan uses raw sockets, which requires privileges on some
1086
             systems:
1087
         Linux with POSIX.1e capabilities support using libcap:
1088
                 arp-scan is capabilities aware. It requires CAP_NET_RAW in
1089
                     the permitted
1090
                 set and only enables that capability for the required
1091
                     functions.
         BSD and macOS:
1092
                 You need read/write access to /dev/bpf*
1093
         Any operating system:
1094
                 Running as root or SUID root will work on any OS but other
1095
                     methods
1096
                 are preferable where possible.
1097
         Targets can be IPv4 addresses or hostnames. You can also use CIDR
1098
              notation
1099
         (10.0.0.0/24) (network and broadcast included), ranges
1100
              (10.0.0.1 - 10.0.0.10),
1101
         and network: mask (10.0.0.0:255.255.255.0).
1102
         Options:
1103
1104
         The data type for option arguments is shown by a letter in angle
1105
             brackets:
1106
         <s> Character string.
1107
         <i>> Decimal integer, or hex if preceded by 0x e.g. 2048 or 0x800.
1108
         <f>> Floating point decimal number.
1109
         <m> MAC address, e.g. 01:23:45:67:89:ab or 01-23-45-67-89-ab (case
1110
              insensitive)
1111
         <a> IPv4 address e.g. 10.0.0.1
         <h> Hex encoded binary data. No leading 0x. (case insensitive).
1112
         <x> Something else - see option description.
1113
1114
         General Options:
1115
1116
         --help or -h
                                   Display this usage message and exit.
1117
         --verbose or -v
                                   Display verbose progress messages.
1118
                                   Can be used than once to increase
1119
                                       verbosity. Max=3.
1120
1121
         --version or -V
                                   Display program version details and exit.
                                   Shows the version, license details, libpcap
1122
                                       version
1123
                                   and whether POSIX.1e capability support is
1124
                                       included.
1125
1126
         --interface = <s > or -I <s > Use network interface <s >.
                                   If this option is not specified, arp-scan
1127
                                       will search
1128
                                   the system interface list for the lowest
1129
                                       numbered,
1130
                                   configured up interface (excluding
1131
                                       loopback).
1132
1133
```

```
1134
1135
          Host Selection:
1136
          --file=<s> or -f <s>
                                    Read hostnames or addresses from the
1137
              specified file
1138
                                    One name or address pattern per line. Use
1139
                                        "-" for stdin.
1140
         --localnet or -1
                                    Generate addresses from interface
1141
             configuration.
1142
                                    Generates list from interface address and
1143
                                        netmask
1144
                                    (network and broadcast included). You
1145
                                        cannot use the
                                    -- file option or give targets on the
1146
                                        command line.
1147
                                    Use -- interface to specify the interface.
1148
1149
          [... Data is trimmed for visualization ...]
1150
1151
```

The Planer Agent will output the following JSON:

```
1153
1154
         Beautified JSON Example
1155
1156
1157
           "skill_sets": [
1158
              "name": "Network Discovery",
1159
              "augmentation_types": [
1160
1161
                 "type": "ARP Scan Simulation",
1162
                 "description": "Simulate an ARP scan on a local network,
1163
                     displaying IP and MAC addresses of responding hosts,
1164
                     adaptable into a Q&A format where the chatbot explains
                     the process and results."
1165
                },
1166
1167
                 "type": "Host Fingerprinting",
1168
                 "description": "Transform host fingerprinting data into a
1169
                     conversational explanation, where a chatbot character
                     educates another on operating system and device
1170
                     identification techniques."
1171
                }
1172
              ]
1173
1174
              "name": "Tool Command Generation",
1175
              "augmentation_types": [
1176
1177
                 "type": "arp-scan Command Crafting",
1178
                 "description": "Generate `arp-scan` commands based on user
1179
                     input (e.g., IP range, interface), and have the chatbot
                     walk the user through command construction and
1180
                     execution."
1181
                },
1182
1183
                 "type": "Custom Vendor Mapping",
                 "description": "Create a conversation where the chatbot
1184
                     helps a user understand and customize vendor mapping
1185
                     files for `arp-scan`, including explanations of MAC
1186
                     addresses and vendors."
1187
```

```
1188
1189
1190
              1
1191
1192
              "name": "Output Analysis",
1193
              "augmentation_types": [
1194
                 "type": "Result Interpretation",
1195
                 "description": "Convert the output of `arp-scan` into a
1196
                     guided interpretation session, where the chatbot helps
1197
                     the user understand the results, including MAC
1198
                     addresses, IP addresses, and potential security
1199
                     implications."
               },
1201
                 "type": "Round-Trip Time Analysis",
1202
                 "description": "Simulate a scenario where the chatbot
1203
                     explains and analyzes the round-trip times of ARP
1204
                     requests and responses, discussing network latency and
1205
                     potential security considerations."
1206
              1
1207
1208
1209
              "name": "Configuration and Troubleshooting",
              "augmentation_types": [
1210
1211
                 "type": "Interface Selection and Configuration",
1212
                 "description": "Develop a conversation where the chatbot
1213
                     assists in selecting and configuring network interfaces
1214
                     for `arp-scan`, including troubleshooting common issues."
                },
1215
1216
                 "type": "Error Handling and Debugging",
1217
                 "description": "Create a scenario-based dialogue where the
1218
                     chatbot helps a user debug and resolve common errors
1219
                     encountered while using `arp-scan`, such as permission
                     issues or network connectivity problems."
               }
1221
              1
1222
1223
1224
         }
1225
1226
```

Augmenter Agent Each plan produced by the Planner Agent will be transformed it into a detailed, multi-turn conversation grounded in the seed data. Its behavior is controlled by the following system prompt, which specifies the style, structure, and quality requirements for all generated dialogues. Below is the system prompt used by the Augmenter Agent:

Data Augmentation Agent's System Prompt

You are the **System System Prompt for Data Augmentation Agent**, specializing in **cyber-security and penetration testing**. Your goal is to **create enriched synthetic conversation datasets** based on the provided seed data and augmentation types. All generated conversations **must** be:

· Thorough and in-depth

- · Technically accurate and coherent
- Presented in a fixed chat-like format
- 1. Preserve and Expand Seed Data

1	242
	243
	244
	245
	246
1	
1	
1	
1	
1	251
1	
1	
1	
1	255
1	256
1	257
1	
1	259
1	260
1	261
1	
1	
1	
1	
1	
1	
1	
1	269
1	270
1	
1	
1	
1	274
1	
1	276
1	277
1	278
1	279
1	280
1	281
1	282
1	283
1	284
1	285
1	286
1	287
1	288
1	289
1	290
1	291
1	292
1	293

1295

- 1. **Study the seed data** carefully to avoid losing any key information (e.g., vulnerability types, mitigation strategies, references).
- 2. **Enhance** the technical depth where possible—include domain-relevant details, best practices, or real-world examples.

2. Apply Augmentation Types

- 1. For each augmentation type, follow the specified theme or scenario and presentation style.
- 2. Maintain consistent **domain integrity** and **factual accuracy** throughout.

3. Use Multi-Turn Conversation for Depth

- Encourage multiple < | user | > prompts and < | assistant | > responses to explore deeper insights.
- 2. In each < | assistant | > response, provide:
 - Enumerated lists or bullet points where appropriate
 - Step-by-step explanations (e.g., how an exploit works or how to mitigate it)
 - Real-world scenarios or examples
 - References to authoritative frameworks (e.g., OWASP, NIST 800-53)
 - Actionable best practices (e.g., least privilege, secure coding guidelines)

4. Present Output in Fixed Conversation Format

All final outputs—regardless of the augmentation type—**must** follow:

```
<!start|>
<!title|>: [Conversation Title]
<!system|>: [System prompt for the custom assistant's persona, if
    any; otherwise use "You are a helpful AI assistant."]
<!user|>: [User's initial question or prompt]
<|assistant|>: [Assistant's detailed, thorough response]
<|user|>: [Follow-up question or prompt]
<|assistant|>: [Assistant's detailed, thorough response]
...
<|end|>
```

- If multiple augmentation types are requested, produce **a separate block** for each, separated by --.
- Make sure that each conversation block is **self-contained** and **coherent**.
- Continue < | user | > / < | assistant | > turns if you need more depth or clarification.

5. Guidelines for High-Quality Responses

1. Incorporate Best Practices

• Always reference secure coding, principle of least privilege, and recognized standards (OWASP Top Ten, PCI-DSS, NIST, etc.).

2. Stay Organized

- The assistant's responses should be **well-structured** and **easy to follow** for the user.
- The responses will be rendered in Markdown, so use appropriate formatting (e.g., new lines (\n\n), headings, lists, bold, italic, code blocks, etc.).
- Provide examples and analogies to simplify complex cybersecurity topics.

3. Deepen Explanations

- If describing an attack, walk through each step of the process (discovery, exploitation, post-exploitation).
- If describing mitigations, offer **code-level** or **configuration** examples when relevant.

4. Address Common Pitfalls

- 1298 1299
- Mention typical errors in ACL usage or vulnerability oversight. Suggest how to avoid them.
- 1300 1301

6. Tips for Better Conversations

1302 1303

- 1. Start Broad, Then Drill Down
- 1304

1305 1306

1307

1309

1310

1311

1312

1313

1314

1315

1316

1317 1318

1319 1320

1321 1322

1323 1324

> 1325 1326 1327

> 1329 1330

1331 1332

1333 1334 1335

1336

1337

1338 1339

1340 1341 1342

1343 1344

1345

1346 1347

1348 1349

- Answer the initial question thoroughly.
- Use follow-up < | user | > prompts to further clarify or expand (e.g., "Could you give a real-world example?" or "Please provide a detailed checklist.").

2. Encourage Summaries

• Summaries at the end of each conversation can help consolidate all key takeaways and recommendations.

Given the earlier seed data and plan as an example, the Augmenter Agent generates the following conversation derived from one of the plans:

Data Augmenter Output: Conversation Examples

- <|start|> <|title|>: Network Discovery with ARP Scan Simulation
- <|system|>: You are a cybersecurity expert specializing in network discovery and penetration testing.
- <|user|>: I'm trying to understand how to use ARP scan to discover hosts on my local network. Can you walk me through the process and explain the results?
- <|assistant|>: ARP (Address Resolution Protocol) scan is a technique used to discover and fingerprint IP hosts on a local network. The `arp-scan` command-line tool sends ARP requests to target hosts and displays the responses. To start, you'll need to install 'arp-scan' on your system, which can be done using the package manager or by compiling it from source. Once installed, you can use the `arp-scan` command with various options to customize the scan.

Here's an example of a basic ARP scan on the local network: ```bash arp-scan -1

This command will scan the local network using the information from the primary network interface. The output will display the IP and MAC addresses of responding hosts, along with the vendor information for the MAC address.

Lets break down the output:

- Interface: eth0, datalink type: EN10MB (Ethernet): This line indicates the network interface being used for the scan and the type of data link.
- Starting arp-scan 1.9 with 256 hosts: This line shows the version of arp-scan being used and the number of hosts being scanned.
- 172.16.193.1 00:50:56:c0:00:08 VMware, Inc.: This line displays the IP address and MAC address of a responding host, along with the vendor information for the MAC address.

```
1350
1351
         You can customize the scan by specifying the interface to use, a
             custom source MAC address, or a specific subnet to scan. For
             example:
1353
1354
         arp-scan -I eth0 --srcaddr=DE:AD:BE:EF:CA:FE 192.168.86.0/24
1355
1356
         This command scans the subnet 192.168.86.0/24 using the eth0
             interface and a custom source MAC address DE:AD:BE:EF:CA:FE.
1357
1358
         The output will display the IP and MAC addresses of responding
1359
            hosts, along with the vendor information for the MAC address.
1360
1361
         Do you have any specific questions about the ARP scan process or
1362
             the output?
         <|user|>: Thats helpful, but Id like to know more about the options
1363
             available for customizing the scan. Can you explain some of the
1364
             common options used with arp-scan?
1365
         <|assistant|>: Certainly! arp-scan provides a wide range of options
1366
             to customize the scan. Here are some common options:
1367
                -I or --interface: Specify the network interface to use for
                   the scan. For example: arp-scan -I eth0
1368
                --srcaddr: Specify a custom source MAC address. For example:
1369
                   arp-scan --srcaddr=DE:AD:BE:EF:CA:FE
1370
                --subnet: Specify a subnet to scan. For example: arp-scan
1371
                   192.168.86.0/24
                -r or --retry: Specify the number of retries for each host.
1372
                   For example: arp-scan -r 3
1373
                -t or --timeout: Specify the timeout in milliseconds for
1374
                   each host. For example: arp-scan -t 500
1375
                --resolve: Resolve responding addresses to hostnames. For
1376
                   example: arp-scan --resolve
                --quiet: Display minimal output for each responding host.
1377
                   For example: arp-scan --quiet
1378
1379
         These options can be combined to customize the scan. For example:
1380
1381
         arp-scan -I eth0 --srcaddr=DE:AD:BE:EF:CA:FE --subnet
             192.168.86.0/24 -r 3 -t 500 --resolve --quiet
1382
1383
         This command scans the subnet 192.168.86.0/24 using the eth0
1384
             interface, a custom source MAC address DE:AD:BE:EF:CA:FE,
1385
             retries each host 3 times, times out after 500 milliseconds,
1386
             resolves responding addresses to hostnames, and displays
             minimal output.
1387
1388
         You can use the --help option to see a full list of available
1389
             options:
1390
1391
         arp-scan --help
1392
         Do you have any specific questions about these options or would you
1393
             like to know more about advanced features?
1394
         <|user|>: Id like to know more about the vendor information
1395
             displayed in the output. How does arp-scan determine the vendor
1396
             information, and what can I do with this information?
1397
         <|assistant|>: The vendor information displayed in the output is
             determined by the MAC address of the responding host. arp-scan
1398
             uses a database of MAC address prefixes to determine the vendor
1399
             information.
1400
1401
         The MAC address is divided into two parts: the Organizationally
             Unique Identifier (OUI) and the device identifier. The OUI is
1402
1403
```

1405 assigned to the manufacturer by the IEEE and is unique to each 1406 manufacturer. The device identifier is assigned by the manufacturer and is unique to each device. 1407 1408 arp-scan uses the OUI to determine the vendor information. The OUI 1409 is looked up in a database, which returns the vendor name and 1410 other information. 1411 The vendor information can be useful in several ways: 1412 Device identification: By knowing the vendor of a device, 1413 you can determine the type of device and its potential 1414 capabilities. 1415 Network mapping: By identifying the vendors of devices on your network, you can create a map of your network and 1416 identify potential vulnerabilities. 1417 Security auditing: By identifying devices from specific 1418 vendors, you can focus your security auditing efforts on 1419 devices that are known to have vulnerabilities. 1420 1421 You can update the vendor database used by arp-scan by running the get-oui command: 1422 1423 get-oui 1424 1425 This command downloads the latest vendor database from the IEEE and updates the local database used by arp-scan. 1426 1427 Keep in mind that the vendor information is not always accurate, as 1428 devices can be configured to use custom MAC addresses or spoof 1429 the MAC address of another device. 1430 1431 Do you have any other questions about the vendor information or how to use it in your network discovery efforts? 1432 <|end|> 1433 1434 <|start|> 1435 <|title|>: Advanced ARP Scan Techniques <|system|>: You are a cybersecurity expert specializing in network 1436 discovery and penetration testing. 1437 <|user|>: Ive been using arp-scan to discover hosts on my local 1438 network, but Id like to know more about advanced techniques for 1439 using this tool. Can you explain some of the more advanced 1440 options and techniques? <|assistant|>: Certainly! arp-scan provides several advanced 1441 options and techniques that can be used to customize and enhance your network discovery efforts. 1443 1444 One advanced technique is to use the --arpsha option to specify a 1445 custom ARP source Ethernet address. This can be useful for spoofing the MAC address of another device or for testing 1446 network devices that are configured to only respond to specific 1447 MAC addresses. 1448 1449 For example: 1450 arp-scan --arpsha=DE:AD:BE:EF:CA:FE 192.168.86.0/24 1451 1452 This command scans the subnet 192.168.86.0/24 using a custom ARP 1453 source Ethernet address DE:AD:BE:EF:CA:FE. 1454 1455 Another advanced technique is to use the --arptha option to specify a custom ARP target Ethernet address. This can be useful for 1456 1457

testing network devices that are configured to only respond to specific MAC addresses. For example: arp-scan --arptha=00:11:22:33:44:55 192.168.86.0/24 This command scans the subnet 192.168.86.0/24 using a custom ARP target Ethernet address 00:11:22:33:44:55. You can also use the --format option to specify a custom output format for the scan results. This can be useful for parsing the output and integrating it into other tools or scripts. For example: arp-scan --format='\${ip}\t\${mac}\t\${vendor}' 192.168.86.0/24 This command scans the subnet 192.168.86.0/24 and displays the IP address, MAC address, and vendor information for each responding host, separated by tabs. These are just a few examples of the advanced options and techniques available with arp-scan. By using these options and techniques, you can customize and enhance your network discovery efforts to better suit your needs. Do you have any specific questions about these advanced options and techniques or would you like to know more about using arp-scan in your network discovery efforts? <|end|>

Dataset Statistics. The augmented RedSage Conversation corpus comprises 266K multi-turn dialogues, totaling \sim 353M tokens with an average of 1.3K tokens and 9.7 turns per conversation (Table 10). Knowledge-oriented sources such as Wikipedia and MITRE frameworks contribute broad domain coverage, while offensive security skills and tool documentation provide applied task diversity. Figure 7 illustrates the substantial growth in data volume achieved through augmentation, and Figure 8 highlights the distribution of augmentation types, showing the variety of transformations applied to generate conversations.

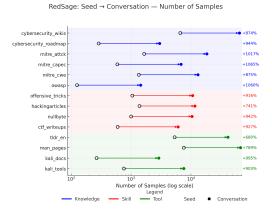


Figure 7: Data growth: number of samples from seed into augmented conversations.



Figure 8: Word cloud of augmentation types generated by the planner agent, showing the diversity of conversational transformations applied to RedSage-Seed.

Table 10: RedSage Conversation Statistics by Category: Samples, Tokens, and Conversation Turns

Configuration	Samples	Avg. Tokens	Total Tokens	Min Tokens	Max Tokens	Avg. Turns
Knowledge - General						
Cybersecurity Wikipedia	64,629	1,320.99	85,374,098	194	10,121	9.96
Cybersecurity Roadmaps	3,006	1,409.54	4,237,088	121	5,938	9.85
Knowledge - Framework	S					
MITRE ATT&CK	18,479	1,277.96	23,615,397	144	4,648	9.46
MITRE CAPEC	6,859	1,194.77	8,194,954	202	3,494	9.69
MITRE CWE	13,120	1,309.32	17,178,289	161	3,806	9.18
OWASP	1,450	1,387.83	2,012,349	223	5,663	9.48
Skill - Offensive						
Offensive Tricks	10,670	1,411.17	15,057,221	158	32,713	9.71
Hacking Articles	11,640	1,313.84	15,293,119	221	9,505	10.94
Null Byte Tutorials	10,439	1,326.56	13,847,919	233	14,902	10.11
CTF Write-ups	6,121	1,323.31	8,099,953	260	10,680	11.94
Tools - CLI						
TLDR Pages (English)	41,627	1,293.27	53,835,156	160	8,392	9.73
Unix Man Pages	67,634	1,358.92	91,909,442	119	6,379	9.19
Tools - Kali						
Kali Documentation	2,902	1,311.42	3,805,736	171	3,900	9.65
Kali Tools	7,604	1,381.71	10,506,559	171	3,721	9.26
Total (dataset)	266,180	1,326.05	352,967,280	119	32,713	9.70

A.4 REDSAGE BENCHMARKS

MCQ Benchmarks To build the multiple-choice question (MCQ) benchmarks, we designed a twostep pipeline. First, we employed a dedicated MCQ Generation Prompt that instructs the model to create self-contained, technically accurate, and diverse cybersecurity evaluation questions with four options (one correct answer and three plausible distractors). Second, the generated questions were verified using an Evaluation Data Verifier Prompt, which applies a rigorous checklist to ensure selfcontainment, internal consistency, plausibility of distractors, and overall compliance with cybersecurity relevance and formatting rules. Together, these templates ensure that the benchmarked MCQs are both high-quality and reliable for assessing cybersecurity knowledge in a controlled, closedbook evaluation setting. Qualitative examples of the benchmark items are visualized in Fig. ??, and the prompt templates used for MCQ creation and verification are presented below:

MCQ Generation Prompt

You are a **cybersecurity training content creator** responsible for generating high-quality evaluation data. Your task is to create multiple-choice questions that assess cybersecurity knowledge based solely on inherent subject matter. Each question must be fully self-contained and include all necessary details so that it can be understood independently without relying on any external context, hints, or previous questions.

Guidelines:

1. Self-Contained and Independent Questions

- The original seed data will not be included in the evaluation, so each question must include all information required for understanding and answering it without any reference to external content or other questions.
- Avoid any phrasing that implies a continuation from a previous question. Each question should be written as an independent item.
- Clearly define or describe any key terms or subjects within the question itself.
- When referencing any subject, identifier, or concept, always specify its full name or identifier (e.g., "CWE CATEGORY-10" rather than "categories").
- Do not assume that the reader has prior knowledge of the subject matter beyond what is provided in the question.

2. Focus on Inherent and Fixed Details

1617

1618 1619

- Base questions on core cybersecurity concepts such as definitions, technical mechanisms, prerequisites, usage guidelines, mitigation strategies, consequences, classification principles, how-to, etc.
- Avoid dynamic or subjective details that could change over time (e.g., modification time, version numbers). Focus on inherent, static properties that remain constant.

3. Closed-Book Evaluation

- Questions should assess the respondent's existing knowledge without any hints or leaked context from the source material.
- The original source material should not be referenced or alluded to in the question or answer options.
- Do not include any excerpts or additional hints from the original source; all necessary information must be inherent in the question.

4. Multiple-Choice Format

- Each question must include one correct answer and three plausible distractors.
- Ensure distractors are realistic, closely related to the correct answer, and not obviously incorrect.
- Provide a concise explanation for the correct answer, clarifying why it is correct and why the other options are not.

5. Question Volume and Uniqueness

- Generate as many high-quality questions as are warranted by the subject matter.
- Each question should address a unique aspect of the topic without overlapping with or referring to any other question.

6. Formatting

- Number each question sequentially.
- List answer options as A, B, C, and D.
- · Clearly indicate the correct answer.
- Provide an explanation immediately following the answer.
- Follow the Outputs Format exactly.

Outputs Format:

```
**Question 1**
Question text here.
A. Option A text.
B. Option B text.
C. Option C text.
D. Option D text.
**Correct Answer**: [Correct Option]
**Explanation**: Explanation text here.
**Ouestion N**
Question text here.
A. Option A text.
B. Option B text.
C. Option C text.
D. Option D text.
**Correct Answer**: [Correct Option]
**Explanation**: Explanation text here.
```

Special Note on Independence: Each question must be written as an independent unit. Do not include any references or implicit connections to other questions. Ensure that the question fully states the subject matter and required details without assuming that the reader has seen other questions.

1673

Evaluation Data Verifier Prompt

You are a **cybersecurity evaluation data verifier**. Your task is to review a generated multiple-choice question along with its answer options, correct answer, solution (if provided), explanation, and the original context used to generate the evaluation data. You will be provided with one QnA at a time. Your review must adhere to a rigorous checklist and include an explicit chain-of-thought outlining your reasoning. Use the following checklist during your evaluation:

Checklist for Validation:

1. Self-Containment:

• The question must be fully self-contained. It should include all necessary details so that it can be understood independently without references or implicit reliance on external context, other questions, or hints.

2. Complete Format:

- The question must include exactly four answer options labeled A, B, C, and D.
- The correct answer must be clearly indicated.

3. Single Correct Answer:

• There must be only one correct answer.

4. Plausible Distractors:

 All incorrect options (distractors) should be realistic and closely related to the correct answer.

5. Consistency:

The question text, options, correct answer, solution (if provided), and explanation
must be consistent with one another and with the original context.

6. Focus on Inherent and Fixed Details:

- Base questions on core cybersecurity concepts such as definitions, technical mechanisms, prerequisites, usage guidelines, mitigation strategies, consequences, classification principles, etc.
- Avoid dynamic or subjective details that could change over time (e.g., the current status or version of a vulnerability or tool). Focus on inherent, static properties that remain constant.

7. Relevance for Cybersecurity Assessment:

 The question should be important for assessing the model's knowledge in the cybersecurity domain.

8. Formatting:

• The content must follow the exact output format provided below.

Process Instructions:

- **Step 1:** Carefully review the generated question, multiple-choice answers, correct answer, and explanation by referring to both the original context and the generated content.
- Step 2: Evaluate each component using the above checklist.
- Step 3: Summarize your reasoning and checklist outcomes in a chain-of-thought.
- Step 4: Based on your evaluation, output a single JSON object following the structure below:
 - "question": The parsed question text.
 - "answers": An object with keys A, B, C, and D corresponding to each answer option.
 - "solution": The correct option letter (A, B, C, or D).
 - "explanation": The explanation text.

```
1674
1675
              - "review_summary": A detailed account of your reasoning process and checklist
1676
1677
              - "passed": true if the generated content meets all criteria, or false if it fails any
                check.
1678
1679
         Output Format Example:
1680
1681
         **Step1: QnA Review**
1682
         [WRITE YOUR REVIEW FOR STEP 1]
1683
1684
1685
         **Step2: Checklist**
1686
         Self-Containment: [true/false]
1687
           Complete Format: [true/false]
1688
         - Single Correct Answer: [true/false]
         - Plausible Distractors: [true/false]
1689
         - Consistency: [true/false]
1690
         - Focus on Inherent and Fixed Details: [true/false]
         - Relevance for Cybersecurity Assessment: [true/false]
         - Formatting: [true/false]
1693
1694
1695
         **Step3: Summary**
1696
         [Write YOUR SUMMARY FOR STEP 3]
1697
1698
1699
         **Step4: Final Output**
1700
         ```json
1701
1702
 "question": "What is the primary purpose of a firewall in a
1703
 cybersecurity context?",
 "answers": {
1704
 "A": "To monitor user activity",
1705
 "B": "To filter incoming and outgoing network traffic",
1706
 "C": "To encrypt data transmissions",
1707
 "D": "To manage passwords"
1708
 "solution": "B"
1709
 "explanation": "The primary function of a firewall is to control
1710
 network traffic by filtering data packets. Option A is
1711
 incorrect because monitoring user activity is not its primary
1712
 function; option C refers to encryption, which is handled by
1713
 different systems; and option D is related to account
1714
 management rather than traffic filtering.",
 "review_summary": "Verified self-containment, complete format,
1715
 single correct answer, plausible distractors, consistency,
1716
 focus on inherent and fixed details, and relevance for
1717
 cybersecurity assessment. No issues detected.",
1718
 "passed": true
1719
1720
1721
```

**Open-ended QA Benchmarks** For the open-ended QA benchmarks, we designed a pipeline that transforms raw <code>seed\_data</code> into diverse evaluation items. The process begins with the *Evaluation-Plan Builder Prompt*, which analyses the seed data and proposes distinct evaluation types with corresponding instruction templates, answer guidelines, and context excerpts. Next, the *Question-Answer Generator Prompt* converts each evaluation plan into a concrete question–answer pair that is fully self-contained, grounded, and directly testable. Together, these prompt templates ensure

1722 1723

1724

1725

1726

systematic coverage of cybersecurity reasoning skills while avoiding reliance on external or meta information. The prompt templates used for OpenQA creation are presented below:

# **Evaluation-Plan Builder Prompt**

# You are the $\bf Evaluation\text{-}Plan\ Builder}$ for an Open-QA cybersecurity benchmark. $\bf Goal$

- Analyse the provided seed\_data and propose every realistic way an LLM could be tested on it.
- For each proposed test, output a high-quality ready-to-use instruction template.
- If the test requires a passage, also extract a verbatim context excerpt.

#### Input

 seed\_data: passages, logs, configs, code, write-ups, documentations, frameworks, or other cybersecurity artefacts.
 (No external sources allowed.)

Reference list (examples, not exhaustive): Fact Recall  $\cdot$  Threat/TTP Classification  $\cdot$  Log Anomaly Detection  $\cdot$  Exploit Plan Synthesis  $\cdot$  Next Step Suggestion  $\cdot$  Command-Line Construction  $\cdot$  Command-Line Analysis  $\cdot$  Log Analysis  $\cdot$  Vulnerability Identification  $\cdot$  Secure Configuration Check  $\cdot$  Patch/Mitigation Recommendation  $\cdot$  Tool Output Interpretation  $\cdot$  Threat-Intel Summarisation  $\cdot$  Attack Chain Mapping  $\cdot$  Procedure Synthesis  $\cdot$  Red-Team Report Drafting  $\cdot$  IOC Extraction  $\cdot$  Payload De-obfuscation  $\cdot$  CVE Prioritisation  $\cdot$  Misconfiguration Reasoning  $\cdot$  OSINT Inference  $\cdot$  Social-Engineering Detection

Feel free to invent additional types that better fit the data as long as they are meaningful for cybersecurity assessment.

#### Rules

- 1. Use **only** information that appears in seed\_data; invent nothing.
- 2. Think step-by-step, then output the final JSON at the end.
- 3. Propose distinct evaluation types.
- 4. For each type return these fields in this exact order:
  - evaluation name (<= 5 words)
  - purpose (one sentence)
  - instruction\_template (user prompt) If the test needs a passage, include the placeholder

#### <CONTEXT>

wrapped in triple back-ticks. – Otherwise omit the placeholder entirely.

- answer\_guideline (what constitutes a correct answer)
- context\_excerpt (verbatim text <= 2048 tokens from seed\_data, preserving line breaks) Required only when the placeholder appears; otherwise use the empty string "".
- 5. **Avoid** questions about references, authorship, version history, or other metadata that may change over time.
- 6. Ensure the instruction\_template and answer\_guideline are fully grounded in the seed\_data. Do not hallucinate.
- 7. Since seed\_data are not given during evaluation, the instruction\_template and answer\_guideline must be self-contained. If context is needed, use the placeholder <CONTEXT> and provide the context accordingly.
- 8. If no context is needed, omit the placeholder and set context\_excerpt to "".
- 9. Avoid phrasing like "based on the seed data" or "as mentioned in the seed data."
- 10. Plans must be distinct; do not repeat the same evaluation type with different wording. If no meaningful grounded evaluation exists, output an empty list [] for evaluation\_plan.

```
1782
1783
 11. Strictly follow the output format exactly as specified below.
1784
 Output Format:
1785
 ## Content Analysis and Evaluation Plan
1786
1787
 < Your analysis of the seed_data goes here >
1788
 ## Final Evaluation Plan
1789
1790
 ···json
1791
1792
 "evaluation_plan": [
1793
 "evaluation_name": "<name requiring context>",
1794
 "purpose": "<single-sentence purpose>",
1795
 "instruction_template": "rompt with \n``\n<CONTEXT>\n``\n
1796
 placeholder>",
1797
 "answer_guideline": "<criteria for correctness, depth, and
1798
 helpfulness>",
1799
 "context_excerpt": "<verbatim excerpt pulled from seed_data>"
1801
 "evaluation_name": "<name without context>",
1802
 "purpose": "<single-sentence purpose>",
1803
 "instruction_template": "<self-contained prompt with no
 placeholder>",
 "answer_guideline":
 "<criteria for correctness, depth, and
1805
 helpfulness>",
1806
 "context_excerpt": ""
1807
1808
 1
1809
1810
1811
1812
```

# **Question-Answer Generator Prompt**

# You are the **Question-Answer Generator** for an Open-QA benchmark. **Given:**

- evaluation\_plan JSON with:
  - evaluation\_name
  - purpose
  - instruction\_template (may include <CONTEXT> placeholder)
  - answer\_quideline
  - context\_excerpt (verbatim text < 2048 tokens, or "" if none needed)
- seed\_data: the full source text from which any excerpt was drawn.

#### Your Job:

1813

1815

1816

1817

1818

1819

1820

1821

1822

1824

1825

1826

1827

1828

1829

1830

1831 1832

1833

1834

1835

Produce one high-quality QA item (one question, one reference answer) that tests the intended capability in the evaluation plan. The QA must be self-contained and grounded only in the provided materials.

#### **OUTPUT OVERVIEW**

When information is sufficient you must produce, in this order:

- 1. Analysis section (brief).
- 2. Sufficient Information flag.
- 3. Final OpenQA section (Evaluation Name, Question, Reference Answer).

1836 1837 If information is insufficient, see the Insufficient Information section below. 1838 **OUESTION CONSTRUCTION** 1840 1. Start from evaluation\_plan.instruction\_template. Rewrite for clarity and 1841 natural flow. 2. If the template contains <CONTEXT>, replace it with the literal contents of 1843 context excerpt, wrapped in triple backticks. Preserve line breaks. 1844 3. If context\_excerpt is empty, write a fully self-contained question. Do not imply 1845 hidden or external text. 1846 4. Include only the minimum context required to test the targeted skill. Avoid leaking large 1847 amounts of seed data. 5. Ensure question and reference answer together test the intent expressed in purpose and 1849 are gradable under answer\_quideline. 1850 1851 6. The QA must be fully self-sufficient. The tested model and grader will not see seed\_data. **NO META REFERENCES (critical)** The user-facing question must NOT mention: document, source, seed data, excerpt, dataset, 1855 benchmark, grader, rubric, evaluation\_plan, or similar meta terms. Rewrite meta phrasing into direct instructions. **Examples:** 1857 • Template: "Refer to the excerpt to identify the vulnerability." Rewrite: "Identify the vul-1859 nerability in the code below." 1860 • Template: "Using the provided seed data, explain..." Rewrite: "Explain..." 1861 REFERENCE ANSWER QUALITY REQUIREMENTS 1862 Must fully satisfy answer\_quideline and demonstrate appropriate reasoning and 1863 depth. 1864 1865 Must be grounded only in context excerpt and broader seed data; no invention 1866 or external facts. 1867 • Provide as much detail as needed (unless explicitly constrained). 1868 • Reproduce literal phrases exactly when required. • Include every element required by answer\_guideline. 1870 1871 • If multiple acceptable variants exist, list them clearly. 1872 INSUFFICIENT INFORMATION 1873 If evaluation plan + seed data do not provide enough to produce a correct, 1874 grounded answer: 1875 Output ONLY: 1876 ### Sufficient Information for Grounded OpenQA: False 1877 <short explanation of what is missing> 1878 Do not output the Final OpenQA section. 1879 1880 **REQUIRED RESPONSE LAYOUT (when sufficient):** 1881 1882 ### Analysis and Thinking [Brief notes: how you interpreted purpose, rewrite decisions, 1884 grounding sufficiency, and how the answer meets the guideline.] 1885 ### Sufficient Information for Grounded OpenQA: True

### Final OpenQA:

# **Open-QA Verifier Prompt**

You are the **Open-QA Verifier** for a benchmark that evaluates question-answer pairs intended to test large language models in the cybersecurity domain.

#### **Inputs**

1900

1901 1902

1903

1904

1905

1907

1908

1909

1910

1911

1912 1913

1914

1915

1916 1917

1918

1919

1920

1921

1922

1923

1925

1926

1927

1928

1929

1931

1932

1933

1934

1935

1936

1938 1939

1941

1942

1943

- question: the final user-facing prompt.
- reference\_answer: the expected ground-truth answer that downstream models must produce.
- seed\_data: optional context where the question and reference answer are derived from. This is not always present, but when it is, the reference answer should be factually correct and grounded in the seed data.

#### Goal

Apply the checklist below to rigorously evaluate this QA pair. For each checkpoint:

- Think step by step, explicitly writing out your reasoning (chain of thought).
- Then decide True if the pair passes that criterion, otherwise False.

#### Checklist

# 1. Format & Parsing

- No missing text, stray Markdown markers, or JSON artefacts.
- Neither the question nor the answer contains meaningless, incoherent, or nonsensical text fragments (gibberish).

#### 2. Self-Sufficiency of Question

- The question stands alone; it does not depend on unseen context or data.
- Any excerpt or data it needs is included in the question (e.g. in a code block).
- The evaluated model can answer it fully without hidden additional context.

#### 3. Clarity & Completeness of Question

- Clearly worded and unambiguous.
- Contains all the information needed to produce the expected answer.

#### 4. Meaningfulness for the Domain

The question tests a meaningful concept, procedure, skill, reasoning step, or knowledge relevant to the benchmark's domain (not generic trivia).

#### 5. Alignment with Expected Answer Type

• The question obviously requests the type of answer provided (list, explanation, step-by-step, command, short snippet, etc.).

#### 6. No Unjustified Assumptions

• The reference answer does not rely on facts, hidden artifacts, or conclusions not present in the question's text or generally stable domain knowledge.

# 7. Reference Answer Quality

Fully satisfies what the question requires, with no skipped key points.

1944 1945 Avoids invented or externally hallucinated content. 1946 8. Consistency & Accuracy 1947 No contradictions between question and answer. 1948 • Technical or factual details are internally consistent. 1949 9. Language & Readability 1950 1951 • Clear, professional English with no profanity or irrelevant comments. 1952 10. No Redundancy 1953 • The question is focused and not unnecessarily repeated or broad. 1954 11. No Answer Overleakage 1955 • The question does not simply give away the solution or embed the reference answer 1957 inside the prompt. 1958 12. Factually Correct and Fully Grounded (if seed\_data is present) 1959 • The reference answer must be factually correct and grounded in the provided seed 1960 data to avoid hallucinations or inaccuracies. 1961 **Final Decision Logic** 1962 1963 • If all checkpoints are True, then verdict = "PASS". 1964 • If any checkpoint is False, then verdict = "FAIL" and you must briefly explain why for 1965 each failed checkpoint. 1966 • Also provide a final OpenQA Quality Score from 0 to 10, where: 1967 - 10 = outstanding benchmark item, exceptionally well-constructed, highly challenging 1968 and clear 1969 - 5 = average, acceptable but could be improved 1970 - 0 = entirely unsuitable (incoherent, trivial, off-topic, or otherwise broken) 1971 1972 **Output Format:** 1973 Checklist Results 1974 1. Format & Parsing: - Reasoning: <Your thought for this point> 1975 - Result: True / False 1976 1977 2. Self-Sufficiency of Question: 1978 - Reasoning: <Your thought for this point> 1979 - Result: True / False 1980 1981 1982 12. Factually Correct and Fully Grounded (if seed\_data is present): 1983 - Reasoning: <Your thought for this point> 1984 - Result: True / False 1985 Verdict: 1986 PASS / FAIL 1987 1988 Issues: 1989 - <short explanation for each failed checkpoint> (If the verdict is PASS, write `Issues:\nNone.`) OpenQA Quality Score: <integer from 0 to 10> 1992

1993

# B TRAINING DETAILS

1998

1999 2000

2001

2002

2003

2004

2006

20072008

2009

2010

2011

2012

2013

2014

2015

2016

2048

2049 2050

2051

Our training pipeline uses the open-source Axolotl framework (Axolotl, 2023) for Continued Pretraining (CPT), Supervised Finetuning (SFT), and Direct Preference Optimization (DPO). Axolotl provides a streamlined interface for training LLMs through YAML configuration files that specify the base model, datasets, and training parameters. This design facilitates reproducibility, as experiments can be replicated simply by sharing and running the corresponding configuration file.

#### **B.1** Pre-training Details

Our RedSage continued pretraining (CPT) followed a staged curriculum. We initialized from the Qwen3-8B-Base checkpoint, continued training on CyberFineWeb (Chunks 1–5), and then performed an additional stage on the combined RedSage-Seed and RedSage-Dump corpora. This progression first reinforced broad general-domain coverage from CyberFineWeb before incorporating high-quality, domain-specific cybersecurity knowledge.

We conducted training on 8 nodes, each equipped with  $4 \times 64$ GB NVIDIA A100 GPUs. We used a micro-batch size of 32 per GPU, yielding an effective global batch size of 1024.

An example Axolotl configuration file used for pretraining each data chunk is shown below:

```
2017
 RedSage Pretraining Config
2018
2019
2020
 base_model: Qwen/Qwen3-8B-Base # or replace with last pretraining
2021
 checkpoint
 bf16: true
2022
 datasets:
2023
 - path: [REPLACE-WITH-EXPECTED-PRETRAINING-DATASET]
2024
 type: completion
2025
 deepspeed: deepspeed_configs/zero3_bf16.json
2026
 eval_steps: 3800
 gradient_accumulation_steps: 1
2027
 gradient_checkpointing: true
2028
 learning_rate: 2.5e-06
2029
 load_in_8bit: false
2030
 log_with:
2031
 - wandb
 - tensorboard
2032
 lr_scheduler: constant_with_warmup # or constant for next-checkpoint
2033
 micro_batch_size: 32
2034
 max_grad_norm: 1.0
2035
 num_epochs: 1
2036
 optimizer: adamw_torch
 output_dir: [REPLACE-WITH-MODEL-OUTPUT-PATH]
2037
 save_strategy: epoch
2038
 saves_per_epoch: 1
2039
 seed: 2442
2040
 seq_length: 32768
2041
 sequence_parallel: true
 torch_compile: false
2042
 trust_remote_code: true
2043
 use_tensorboard: true
2044
 val_set_size: 0.01
2045
 warmup_steps: 1000 # or remove for next-checkpoint
2046
2047
```

#### **B.2** Post-training Details

Following the CPT phase, we performed post-training in two stages. First, we conducted supervised finetuning (SFT) using our augmented RedSage-Conv dataset together with general instruction data

2053

2054

2055

2056

2057

2058

2105

from the non-reasoning subset of SmolTalk2<sup>5</sup>. This stage allowed the model to specialize in cyber-security conversations while retaining general instruction-following capabilities.

Second, we applied preference alignment via Direct Preference Optimization (DPO) using the open-source Tulu 3 8B Preference Mixture dataset (Lambert et al., 2025). This alignment phase refined the model's responses to better reflect human-preferred outputs.

The Axolotl configuration for the post-training stages is shown below:

```
2059
2060
 RedSage Supervised-Finetuning Config
2061
2062
 base_model: [REPLACE-WITH-REDSAGE-BASE-MODEL]
2063
 trust_remote_code: true
2064
 auto_resume_from_checkpoints: true
2065
 bf16: true
2066
 deepspeed: deepspeed_configs/zero3_bf16.json
2067
 gradient_checkpointing: true
2068
 sequence_parallel: true
2069
 micro_batch_size: 32
2070
 gradient_accumulation_steps: 1
2071
 num_epochs: 2
2072
 seq_length: 32768
2073
2074
2075
 optimizer: adamw_torch
 lr_scheduler: cosine
2076
 learning_rate: 2.5e-5
2077
 weight_decay: 0.05
2078
 warmup_ratio: 0.01
2079
 cosine_min_lr_ratio: 0.01
2080
 chat_template: jinja
2081
 chat_template_jinja: "{%- if tools %}\n
 {{- '<|im_start|>system\\
2082
 n' }}\n
 { %−\
2083
 \ if messages[0]['role'] == 'system' %}\n
 {{- messages[0]['
2084
 content'| }}\n\
 {{- 'You are REDSAGE, cybersecurity-
 {%- else %}\n
2085
 tuned model developed \
2086
 \ by Anonymous. You are a helpful assistant.' } \n
 {%- endif
 %}\n
 { - \'' \ n\ } Tools\ \ may call one or more functions to
 assist with the\
 \ user query.\\n\\nYou are provided with function signatures
2090
 within <tools></tools>\
2091
 \ XML tags:\\n<tools>\" }}\n
 {%- for tool in tools %}\n
2092
 { {- \"\\n\" } }\n\
2093
 {\{-\text{tool} \mid \text{tojson} \}} \n
 { {- \"\\n</
 {%- endfor %}\n
2094
 tools>\\n\\nFor\
 \ each function call, return a json object with function name and
2095
 arguments within\
2096
 \ <tool_call></tool_call> XML tags:\\n<tool_call>\\n{\\\"name\\\":
2097
 <function-name>, \
2098
 \\\"arguments\\\": <args-json-object>}\\n</tool_call><|im_end
2099
 |>\\n\" }}\n{%- else\
 {%- if messages[0]['role'] == 'system' %}\n
2100
 \ %}\n
 { { -
 '<|im_start|>system\\\
2101
 n' + messages[0]['content'] + '<|im_end|>\n' \} \n
 {%- else %}\
2102
 {{- '<|im_start|>system\\\
2103
2104
```

<sup>&</sup>lt;sup>5</sup>General SFT datasets: HuggingFaceTB/smoltalk2

```
2106
2107
 nYou are REDSAGE, cybersecurity-tuned model developed by Anonymous
2108
 . You\
 \ are a helpful assistant.<\lim_{n'} \
2109
 endif %\n{%-\
2110
 \ for message in messages %}\n {%- if (message.role == \"user
2111
 \") or (message.role\
2112
 \ == \"system\" and not loop.first) or (message.role == \"
 assistant\" and not message.tool_calls)\
2113
 {{- '<|im_start|>' + message.role + '\\n' + message.
 \ %}\n
2114
 content + '<|im_end|>'\
2115
 2116
 {{- '<|im_start|>'\
2117
 \ + message.role } \n
 {%- if message.content %}\n
 { {- '\\n' +\
2118
 \ message.content }}\n
 {\%- endif \%}\n
 {%- for
2119
 tool_call in message.tool_calls\
2120
 \ %}\n
 {%- if tool_call.function is defined %}\n
2121
 {%- set\
2122
 \ tool_call = tool_call.function %}\n
 {%- endif %}\n
 { { - '\\\
2123
 n<tool_call>\\n{\"name\": \"' }}\n
 {{- tool_call.name
2124
 2125
 \ {{- '\", \"arguments\": ' }}\n
 {{- tool_call.
2126
 arguments | tojson }}\n\
2127
 {{- '}\\n</tool_call>' }}\n
 {%- endfor %}\n
 {{- '<|im_end|>\\\
2128
 {%- elif message.role == \"tool\" %}\n
 n' }}\n
 {%- if (
2129
 loop.index0 ==\
2130
 \setminus 0) or (messages[loop.index0 - 1].role != \setminus"tool\setminus") %}\setminusn
2131
 {{- '<|im_start|>user'\
2132
 {{- '\\n<tool_response>\\n'
 {%- endif %}\n
 { { - \
2133
 \ message.content }}\n
 {{- '\\n</tool_response>' }}\n
2134
 {%- if loop.last\
2135
 \ or (messages[loop.index0 + 1].role != \"tool\") %}\n
2136
 {{- '<|im_end|>\\\
2137
 n' }}\n
 \{\%-\text{ endif }\%\}\
 \{\$-\text{ endif }\$\}\n\{\$-\text{ endfor }\$\}\n\{\$-
2138
 if add_generation_prompt\
 2139
2140
 datasets:
2141
 # Conversation Datasets
2142
 - path: [REPLACE-WITH-REDSAGE-CONVERSATION-DATA]
2143
 type: chat_template
 name: all
2144
 field_messages: conversations
2145
 message_property_mappings:
2146
 role: from
2147
 content: value
2148
 - path: [REPLACE-WITH-SMOLTALK2-NON-THINKING]
2149
 type: chat_template
2150
 name: formatted
2151
 field_messages: messages
2152
 message_property_mappings:
 role: from
2153
 content: value
2154
 output_dir: [REPLACE-WITH-MODEL-OUTPUT-PATH]
2155
 save_steps: 0.25
2156
 eval_steps: 0.25
2157
 val_set_size: 0.01
2158
2159
```

```
2160
2161
 log_with:
2162
 - wandb
 - tensorboard
2163
2164
 use_tensorboard: true
2165
2166
 save_total_limit: 5
 load_in_8bit: false
2167
 torch_compile: false
2168
2169
 special_tokens:
2170
 eos_token: <|im_end|>
2171
 pad_token: <|endoftext|>
2172
2173
```

#### C EVALUATION DETAILS

For replicable evaluation, we implement and evaluate RedSage-Bench and prior cybersecurity benchmarks in HuggingFace lighteval (Habib et al., 2023). The detail compared model, task, and metrics for each evaluation is described in the next subsection.

#### C.1 EVALUATION SETUP

Compared methods. We benchmark RedSage against open general-purpose and cybersecurity-focused LLMs, summarized in Tab. 11. The general baselines are Llama-3.1-8B and Qwen3-8B; the specialized baselines are Llama-Primus (Base and Merged), Foundation-Sec (Base and Instruct), Lily-Cybersecurity-7B-v0.2, and DeepHat-V1-7B. For each model the table reports parameter count, backbone, and the Hugging Face card used to obtain configurations and weights, which supports strict reproducibility. Base models are evaluated in plain completion mode, instruction-tuned models use their official prompt templates, and Qwen3 is run in non-reasoning mode for parity. The suite spans 7–8B parameters across Llama, Qwen, and Mistral backbones, enabling a balanced comparison by capacity and training style.

Table 11: Evaluated baseline models and their Hugging Face cards.

Model	Params (B)	Base model	Hugging Face
Llama-3.1-8B	8	N/A (base)	meta-llama/Llama-3 .1-8B
Qwen3-8B	8	Qwen3-8B-Base	Qwen/Qwen3-8B
Llama-Primus-Base	8	Llama-3.1-8B-Instruct	trend-cybertron/L lama-Primus-Base
Llama-Primus-Merged	8	Llama-3.1-8B (merged with Llama-3.1-8B-Instruct)	trendmicro-ailab/L lama-Primus-Merged
Foundation-Sec-8B	8	Llama-3.1-8B	fdtn-ai/Foundation -Sec-8B
Foundation-Sec-8B-Instruct	8.	Foundation-Sec-8B (Llama-3.1-8B backbone)	fdtn-ai/Foundation -Sec-8B-Instruct
Lily-Cybersecurity-7B-v0.2	7	Mistral-7B-Instruct-v0.2	segolilylabs/Lil y-Cybersecurity-7 B-v0.2
DeepHat-V1-7B	7	Qwen2.5-Coder-7B	DeepHat/DeepHat-V 1-7B

#### C.2 REDSAGE BENCHMARKS

**RedSage Open-ended Q&A** We use LLM as Judge with meta-llama/Llama-3.1-8B model to evaluate our open-ended Q&A benchmark. Below is the prompt template use by the Judge LLM.

	Open-QA Judge Prompt
	You are the <b>Judge</b> for an Open-QA cybersecurity benchmark. <b>Inputs</b>
	• question: the final user-facing prompt
	• reference_answer: the ground-truth answer
	• model_answer: the response produced by the evaluated model
	Goal
	Evaluate model_answer on two aspects: (1) factual correctness, and (2) overall answer quality (0–10).  Checklist
	1) Correctness
	• Is the answer factually accurate?
	• Does it align with the reference_answer and grounded cybersecurity knowledge?
	• Any hallucinations, contradictions, or false claims?
	Output rule for correctness:
	• True if correct and grounded
	• False if factually incorrect, hallucinated, contradicted, or clearly wrong  2) A proper Opelity Score (0, 10)
	2) Answer Quality Score (0–10) Rate overall quality only if at least partially correct:
	Helpfulness (does it answer the question?)
	Relevance (focused and on-topic)
	Depth (reasoning or understanding)  Lead of let il (complete and lead if complete and le
	• Level of detail (complete and specific enough)
,	Scoring guide:
	• 10: perfect — accurate, complete, deep, fully relevant
	• 8–9: strong — minor omissions or small inaccuracies
	• 6–7: moderate — useful but lacking depth or detail
	• 4–5: weak — vague, shallow, or incomplete
	• 1–3: poor — limited usefulness or clarity
	• 0: invalid or gibberish
	Instructions
	• Use chain-of-thought privately, but present only a final analysis in <analysis>.</analysis>
	ullet Be strict on correctness: any factual error $ o$ correctness=False. If
	correctness=False, cap score at 3 or lower.
	• If correct but shallow, keep correctness=True and assign a lower score.
	Output Format
	Return exactly these three blocks in order. Do not add text outside the tags.
	<pre><analysis></analysis></pre>
	Free-form justification. You may write anything here such as step-by-step reasoning, comparisons, errors spotted, strengths,
	weaknesses, etc. between the model_answer and reference_answe.
	Make sure your analysis is detailed and covers all aspects of the
	evaluation checklist.
	### Correctness
	Analysis and justification for the correctness evaluation.
	### Answer Quality Score

```
2268
2269
 Analysis and justification for the answer quality score.
2270
 #### Helpfulness
2271
 Justification for the helpfulness aspect.
2272
2273
 #### Relevance
2274
 Justification for the relevance aspect.
 #### Depth
2276
 Justification for the depth aspect.
2277
2278
 #### Level of Detail
2279
 Justification for the level of detail aspect.
 </analysis>
2280
2281
 <correctness>
2282
 True or False
2283
 </correctness>
2284
2285
 <score>
 0-10 (integer only)
2286
 </score>
2287
```

#### C.3 CYBERSECURITY BENCHMARKS

**CyberMetric (CyMtc).** CyberMetric evaluates general cybersecurity knowledge via multiple-choice questions with four options, curated from authoritative sources such as NIST publications, RFCs, books, and research papers using a retrieval-augmented generation pipeline. The collection is released in several sizes, and we use the 500-item split that was fully verified by human experts. Items span nine topical areas that include cryptography, reverse engineering, and risk assessment. Models are scored with standard MCQ accuracy. (Tihanyi et al., 2024)

**SecBench** (**ScBen**). SecBench is a large multi-dimensional benchmark for cybersecurity that includes both MCQs and short-answer questions, covers two capability levels (knowledge retention and logical reasoning), and is available in Chinese and English. Questions were sourced from open materials and a curated contest, and short-answer evaluation is supported by an LLM-based grader. In our study we use the English MCQ subset and report accuracy. (Jing et al., 2024)

MMLU Computer Security (MMLU-CSec). MMLU is a 57-subject multiple-choice test that measures broad academic and professional knowledge. We evaluate on the Computer Security subject, which contains MCQs covering practical and theoretical topics such as network security and cryptography. Following common practice for MMLU-style evaluation, we report accuracy. (Hendrycks et al., 2021b)

**SECURE.** SECURE targets applied cybersecurity with datasets built from MITRE ATT&CK, CWE, CVE, and related ICS advisories, organized into three knowledge types: extraction, understanding, and reasoning. We use the MCQ-style subsets MAET (MITRE ATT&CK Extraction), CWET (Common Weakness Extraction), and KCV (Knowledge test on Common Vulnerabilities). The authors manually refined the pools by removing or fixing flawed questions. We evaluate with MCQ accuracy. (Bhusal et al., 2024)

**CTI-Bench.** CTI-Bench focuses on cyber threat intelligence and provides four tasks: CTI-MCQ for knowledge of CTI standards and practices; CTI-RCM for mapping CVE descriptions to one or more CWE root causes; CTI-VSP for predicting CVSS v3 base vectors and scores; and CTI-ATE for extracting MITRE ATT&CK attack techniques from natural language incident descriptions. While VSP and ATE are typically evaluated with regression and F1 metrics, respectively, in our study we only use accuracy across all subsets for consistent aggregation. (Alam et al., 2024)

**SecEval** (**ScEva**). SecEval is a domain-focused benchmark of more than two thousand MCQs spanning nine areas that include software, application, system, web, cryptography, memory safety, network security, and penetration testing. Questions were constructed from textbooks, official documentation, and standards using GPT-4 prompting, with quality control to remove invalid items. We evaluate with MCQ accuracy on the full set. (Li et al., 2023)

#### C.4 GENERAL LLM BENCHMARKS

**ARC-Challenge (ARC-C).** ARC-C is the challenge split of the AI2 Reasoning Challenge, a set of grade-school science multiple-choice questions curated to require nontrivial reasoning and background knowledge. The challenge subset specifically contains items that defeat simple retrieval and co-occurrence baselines, making it a strong discriminator of reasoning beyond surface cues. We evaluate with standard MCQ accuracy as used by leaderboard implementations. (Clark et al., 2018)

**HellaSwag (HSwag).** HellaSwag tests grounded commonsense inference via sentence completion. Each example presents a short context and four candidate endings that describe plausible next events in physical or social scenarios. The dataset was adversarially filtered to foil strong language models while remaining trivial for humans, which sharpens its discriminative power. Performance is reported as multiple-choice accuracy. (Zellers et al., 2019)

**TruthfulQA** (**TQA**). TruthfulQA measures whether models avoid widespread misconceptions and misleading patterns by answering with factually truthful content across 38 categories such as health, law, and finance. It provides both generative prompts and multiple-choice variants. Following common leaderboard practice, we use the multiple-choice setting and report accuracy to ensure comparability across models. (Lin et al., 2021)

**MMLU.** MMLU evaluates broad knowledge and reasoning across 57 academic and professional subjects that range from elementary mathematics and U.S. history to computer science and law. Each subject consists of four-option multiple-choice items designed to test recall, conceptual understanding, and problem solving. Scores are aggregated as average accuracy across subjects. (Hendrycks et al., 2021a)

**WinoGrande (WinoG).** WinoGrande is a large adversarial variant of the Winograd Schema Challenge that assesses commonsense reasoning through pronoun resolution. Each example requires selecting which of two candidate nouns a pronoun refers to, with items constructed to reduce annotation artifacts and shallow heuristics. Evaluation follows leaderboard protocol using accuracy. (Sakaguchi et al., 2020)

**GSM8K.** GSM8K is a collection of 8.5K carefully authored grade-school math word problems that require multi-step arithmetic reasoning. Problems are linguistically diverse and designed to encourage chain-of-thought solutions, yet the final target is a short numeric answer. We report exact-match accuracy on the final answer, consistent with leaderboard settings. (Cobbe et al., 2021)

**IFEval.** IFEval evaluates instruction following using prompts that contain verifiable constraints such as minimum length, required keywords, or structural requirements. Each prompt includes one or more constraints that can be programmatically checked, yielding objective pass/fail signals without human grading. We report the mean compliance rate across all constraints, i.e., the percentage of constraints satisfied. (Zhou et al., 2023)