The Cheap Path to an Offensive LLM: A Local, Measured Dual-Use Capability Study

Tristan J. Pereira
Independent Security Researcher
tristanjpereira@gmail.com

Abstract

Instruction-tuning datasets for cybersecurity have, until recently, been assembled with the help of commercial, cloud-hosted language models. We ask a narrower and more uncomfortable question: what happens when the entire construction and fine-tuning loop is moved onto a single consumer Apple Silicon machine, with no external API in the path and no provider side safety filter shaping the corpus? We describe AirgapForge, a fully on-device reimplementation of a published cybersecurity data pipeline built on Apple's MLX framework. Building on the work of ElZemity et al. [4], we reproduce a defensive instruction dataset locally and characterize a second, offensive branch that removes the pipeline's safety-alignment stage and trains models, including *abliterated* (safety-stripped) bases, to produce exploitation guidance for vulnerable code. We then measure a full matrix of alignment (aligned vs. abliterated) $\times$ fine-tuning (base vs. offensively fine-tuned) across three open model families on capability, safety, and audit quality. The results are stark: abliteration removes refusals at almost no capability cost, and offensive fine-tuning drives refusal to zero and attack success rate to 96–100% on *every* base, including those that still refused a fifth of requests while aligned. The whole procedure runs on a single desktop at no marginal cost: a complete fine-tune takes hours, not a cloud budget. We do *not* release exploitation payloads, offensive weights, the dataset, or the raw generations: only aggregate metrics. Our contribution is a warning made concrete and measured: the barrier to a locally hosted, uncensored, exploitation capable LLM is now a weekend of compute on hardware that fits under a desk.

1 Introduction

Fine-tuning large language models (LLMs) on cybersecurity data is a double-edged experiment. The same instruction data that teaches a model to explain a CVE, triage an advisory, or reason about an attack technique also moves the model toward producing operational offensive content, up to agents that autonomously exploit real vulnerabilities [5]. That fine-tuning can *degrade* a model's safety is by now well documented [10, 9]; ElZemity et al. [4] made the cybersecurity case concrete with CyberLLMInstruct.

That study, like most dataset construction efforts, relied on commercial, cloud-hosted models to filter and synthesize data. This paper examines what changes when the cloud is removed from the loop entirely. We take the CyberLLMInstruct pipeline as our starting point and re-engineer its LLM dependent stages to run fully on-device using Apple's MLX framework [6], then fine-tune a range of open models locally with low-rank adaptation [8]. Two properties of this all local setting motivate the paper. First, no third-party service ever observes the security data, and no provider safety filter constrains what the pipeline can generate, removing both a confidentiality concern and a safety guardrail. Second, the cost: the hardware is a single desktop, and the models that make it work, including *abliterated* [1] variants with their refusal behavior surgically removed, are freely downloadable - well within reach with the majoritity of laptops from the last decade.

We deliberately push this to its uncomfortable conclusion. Alongside a defensive reproduction of the instruction dataset, we build an *offensive* branch (VulnExploit-SFT) that strips the pipeline's safety-alignment stage and uses an abliterated model to synthesize exploitation write-ups over a public vulnerable code corpus, then fine-tunes on the result. Our intent is not to ship an "AI pentesting bot" but to document how cheaply and locally one can be approached, and what that implies. This concern is not hypothetical for us: the author and other practitioners have used these locally hosted unrestricted LLMs in confidential red team engagements, which motivated a systematic measurement of what such models cost to produce, in capability and in safety. We report that measurement here; we do not report the engagements, which are confidential.

2 Background and Related Work

Cybersecurity instruction datasets. CyberLLMInstruct [4] is the closest prior work and the

direct antecedent of this paper. It assembles a large pseudo-malicious instruction corpus spanning malware analysis, phishing, and vulnerability tasks, fine-tunes a range of models, and reports the safety/capability trade off. Our work reproduces the *construction* half of that pipeline in a fully local setting and re-measures that trade off across an alignment × fine-tuning matrix (§7), adding an abliteration axis that CyberLLMInstruct did not test. Cyber specific evaluation has developed alongside these datasets: CyberMetric [12] scores security knowledge, while CyberSecEval [2] measures both vulnerable code generation and compliance with cyber attack requests, the latter close in spirit to the attack success rate we report.

On-device inference and fine-tuning. Apple's MLX [6] enables inference and LoRA fine-tuning of quantized open models directly on Apple Silicon. Combined with low-rank adaptation [8], it makes both synthetic data generation and fine-tuning of 30B class models feasible on a single machine.

Refusal removal (abliteration). "Abliteration" [1] identifies and ablates the direction in a model's activation space most associated with refusal, producing an open model that answers requests it would previously decline. Such models are publicly distributed. We use one both as a data *generator* (to synthesize offensive content without refusals) and as a fine-tuning *base*, which lets us compare capability, safety, and training dynamics across aligned and de-aligned starting points.

Preference optimization. Beyond supervised fine-tuning (SFT), we include direct preference optimization (DPO) [11] and its reference-free variant ORPO [7] in the pipeline's recipes, used here in a defensive configuration.

Safety degradation from fine-tuning. Research shows that even benign fine-tuning can erode safety alignment [10], and that low rank fine-tuning can *deliberately* undo it for a few hundred dollars [9]. Our setting is the adversarial extreme of this research (offensive data on a de-aligned base), and we treat it as a cautionary characterization rather than a capability demonstration.

3 The On-Device Pipeline

AirgapForge transforms raw, authoritative security sources into a validated instruction-response dataset through a sequence of stages, each writing to its own directory so the pipeline can be inspected and resumed. Every stage that requires a language model runs locally via MLX; no external API is called at any point. Seven stages are implemented in the current codebase.

1. **Collection.** A collector harvests from nine public sources (NVD, OpenCVE, MITRE ATT&CK, CAPEC, Ubuntu and Red Hat advisories, Microsoft advisories, arXiv, and CTF data) via APIs, RSS, and scrapers.
2. **Filtering.** A two-pass filter first applies a parallelized, rule-based keyword and compound term gate (emitting per-entry provenance), then a slower on-device LLM enrichment pass that returns a structured `{technical_description, risk_level, affected_systems, mitigations}` record.
3. **Structuring.** Source-specific builders convert filtered entries into instruction–response pairs, each with a domain appropriate persona prompt; the stage checkpoints and resumes on a per-item hash.
4. **Human review.** An interactive terminal tool records a quality rubric, edits, and reviewer metadata per session.
5. **Assembly.** A final stage deduplicates by content hash, validates against a schema, and exports to multiple formats (JSON, JSONL, CSV, a Hugging Face `DatasetDict`, and MLX train/valid/test splits).
6. **Chat template conversion.** Produces eight chat templates (OpenAI, ShareGPT, Alpaca, ChatML, Vicuna, Llama-2, Mistral, Qwen).
7. **Formatting.** A deterministic Markdown normalization pass over the converted records.

Robust structured extraction. Small quantized models frequently emit malformed or truncated JSON. The pipeline pairs a tolerant first object JSON extractor with a retry loop over increasing token budgets, which is the practical enabler of on-device structured synthesis.

No security-alignment stage. Two stages from the original design are intentionally absent: a domain classifier (domain labels are instead derived by keyword matching) and a security-alignment injector. The latter is directly relevant to the offensive branch (§4), which proceeds *without* any alignment step.

4 Datasets

The pipeline and its offensive extension produce two datasets that sit on opposite sides of the safety line.

Table 1: Defensive instruction dataset: basic statistics.

Metric	Value
Total samples	1,416
Unique samples	1,416
Duplicate rate	0.00%
File size (MB)	7.97
Total tokens	1,661,580
Vocabulary size	18,781
Code-bearing samples	11.5%

Table 2: Defensive dataset text-length statistics per sample. The large mean/median gap reflects a strongly bimodal distribution (short prompts, long technical answers).

Unit	Mean	Median	Std	P95
Tokens	391.1	32.0	558.4	1413
Words	265.6	24.0	372.6	939
Characters	1924.4	198.0	2682.2	6753

4.1 Defensive instruction dataset

The defensive dataset (CyberKnowledgeSFT) is the direct output of the on-device pipeline. Table 1 summarizes its scale: it is fully deduplicated and uniformly in a three-turn `system`→`user`→`assistant` chat format.

Text-length statistics (Table 2) reveal a strongly bimodal corpus: a large mass of very short items sits alongside long technical answers, so the mean token count far exceeds the median. This is consistent with the source mix (advisory records mixed with extended explanations) and with the low readability scores typical of security prose.

Domain coverage (Table 3, Figure 1), computed by overlapping keyword matching, is broad across defensive, forensic, threat intelligence, vulnerability, and attack analysis content, but thins out for cryptography and drops sharply for compliance (6.7%), the clearest coverage gap and a natural target for future collection.

4.2 Offensive exploitation dataset

The offensive branch (VulnExploit-SFT) departs from the defensive pipeline in purpose and in method. It takes the public `CyberNative/Code_Vulnerability_Security_DPO` corpus [3] and, for each vulnerable code sample, uses an abliterated local model to synthesize an assistant answer that identifies the vulnerability and provides exploitation guidance, structured as an audit summary, findings, exploitation steps, and payload examples. Table 4 gives the split sizes; a parallel metadata file records per-record provenance (source dataset, row index, language, and vulnerability class).

Two design choices make this branch the ethically load-bearing part of the work: it *omits* the security-alignment stage the defensive pipeline is meant to include, and it uses a de-aligned generator so the synthesis does not refuse. We characterize this dataset by size and provenance only, and do not release its exploit content (§8).

Table 3: Domain coverage of the defensive dataset (keyword-based, overlapping). Compliance is the clear coverage gap.

Domain	Coverage
Defense	100.0%
Forensics	95.3%
Threat intel	92.9%
Vulnerability	89.5%
Attack	89.1%
Network	78.0%
Cryptography	31.6%
Compliance	6.7%

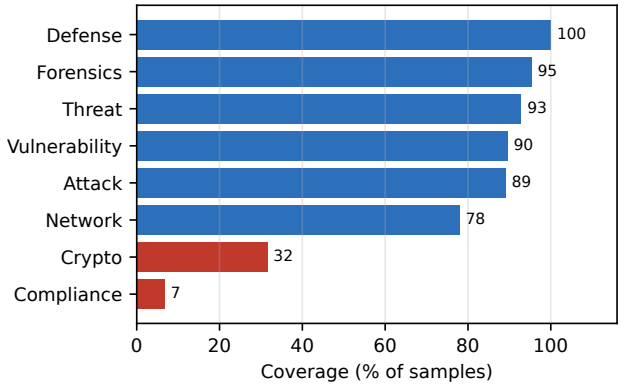

Figure 1: Domain coverage of the defensive dataset. Compliance (highlighted) is the dominant coverage gap.

4.3 Preference data

A third, defensive preference dataset is derived from the same CyberNative source for DPO/ORPO training: the *chosen* response is a safe rewrite of the vulnerable code and the *rejected* response retains the vulnerability. This provides a contrasting, alignment-preserving objective within the same infrastructure.

5 Fine-Tuning Recipes

All fine-tuning uses low-rank adaptation on quantized open models under MLX on a single Apple Silicon machine. We train across four base models: Qwen3-30B-A3B in both Instruct and Thinking variants (8-bit), a 120B mixture-of-experts model (`gpt-oss-120b`, 4-bit), and abliterated Qwen3 variants, covering the aligned/abliterated and defensive/offensive combinations studied in §6.

Table 5 lists the two representative recipes read directly from the released training configs: supervised

Table 4: Offensive exploitation SFT dataset, synthesized by an abliterated model over the public CyberNative Code_Vulnerability_Security_DPO corpus.

Split	Records
Train	4,656
Validation	465
Test	419
Total	**5,540**

Table 5: Fine-tuning hyperparameters, read from the released training configs. SFT column: LoRA instruction tuning; DPO column: preference tuning.

Parameter	SFT	DPO
Fine-tune type	lora	dpo
LoRA rank	8	8
LoRA scale	16.0	10.0
LoRA dropout	0.05	0.05
Layers tuned	16	all
Batch size	2	2
Grad accumulation	–	4
Learning rate	2e-4	2e-4
Iterations	1416	200
Max seq length	4096	2048
Optimizer	adamw	adamw
DPO β	–	0.1
Seed	42	–

LoRA instruction tuning, and DPO preference tuning. The SFT recipe adapts the last 16 layers with rank-8 adapters (scale 16) across all attention and MLP projections at learning rate 2×10^{-4}; the DPO recipe adapts all layers with a KL penalty $\beta = 0.1$ and an effective batch size of 8. ORPO runs use rank-16 adapters (scale 32). All runs use gradient checkpointing to fit within memory and log to Weights & Biases.

6 Training Dynamics and Cost

We logged training telemetry to Weights & Biases for the fine-tuning runs: train and validation loss, throughput, and peak memory. The capability and safety of the resulting models are measured separately in §7. Fifteen runs have usable histories; Table 6 lists representative runs.

Convergence. On the defensive instruction data, the aligned Qwen3-30B Instruct model converges smoothly to a best validation loss of 0.548, and the offensive Code_Vulns objective on the same base reaches a lower best validation loss of 0.248 (Figure 2), which is not surprising, as the offensive targets are longer, more templated, and less varied than the diverse defensive corpus. Lower loss here reflects easier fitting, not a safety or capability claim.

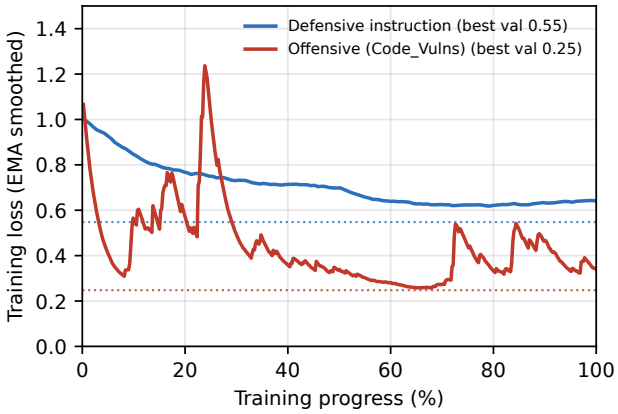

Figure 2: Training loss (EMA smoothed) versus training progress on Qwen3-30B Instruct: defensive instruction data versus the offensive Code_Vulns objective. The offensive objective fits to a lower loss.

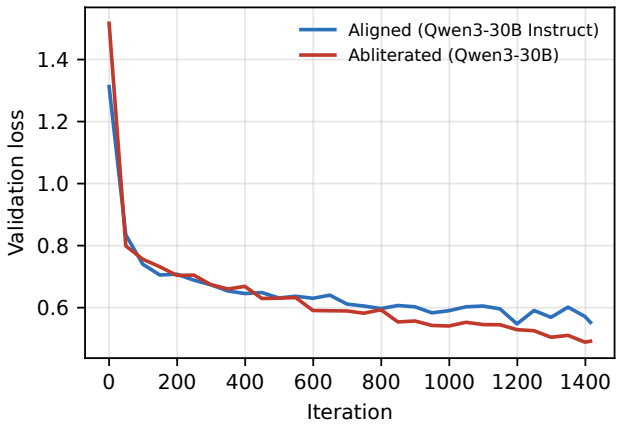

Figure 3: Validation loss on identical defensive data: aligned versus abliterated Qwen3-30B. De-alignment does not hurt convergence.

Aligned versus abliterated. Holding the data fixed (the formatted defensive set), the abliterated Qwen3-30B base reaches a best validation loss of 0.489, slightly below the aligned Instruct model's 0.548 (Figure 3). Removing refusal behavior does not impede (and marginally eases) fitting the cybersecurity data. A de-aligned base is at least as amenable to this fine-tuning as its aligned counterpart.

Training instability. Several offensive and abliterated runs did not converge cleanly. The abliterated Qwen3-30B on Code_Vulns diverged (validation loss rising from 2.853 to 4.451), and an abliterated Qwen3-4B run destabilized late (validation loss 1.524 $\rightarrow$ 3.086). These indicate that the offensive/de-aligned configuration is more sensitive to schedule and length than the defensive one.

Table 6: Representative LoRA runs from training telemetry. These are *training* metrics only; no capability or safety evaluation was logged. 'min' is the best validation loss reached.

Base model	Align.	Data	State	Steps	Train loss	Val loss (min)
Qwen3-30B Instruct (8b)	aligned	defensive (instruction)	finished	142	0.994→0.613	1.268→0.553 (0.548)
Qwen3-30B Instruct (8b)	aligned	defensive (instruction)	finished	142	1.014→0.617	1.313→0.552 (0.548)
Qwen3-30B Instruct (8b)	abliterated	defensive (instruction)	finished	142	1.133→0.584	1.517→0.492 (0.489)
Qwen3-30B Instruct (8b)	aligned	offensive (Code_Vulns)	finished	466	1.066→0.299	2.665→0.301 (0.248)
Qwen3-30B Instruct (8b)	abliterated	offensive (Code_Vulns)	finished	466	1.175→2.018	2.853→4.451 (0.228)
gpt-oss-120b (4-bit)	aligned	defensive (instruction)	killed	31	2.76→1.095	8.959→0.954 (0.954)
Qwen3-4B Instruct	abliterated	offensive (Code_Vulns)	finished	233	1.448→0.367	1.564→0.354 (0.354)
Qwen3-4B Instruct	abliterated	offensive (Code_Vulns)	killed	336	0.88→3.104	1.524→3.086 (0.57)

Table 7: LoRA fine-tuning throughput and peak memory on a single Apple Silicon machine. Ranges span the usable runs for each base model.

Base model	Tok/s	Peak (GB)
Qwen3-4B Instruct	468–568	34.6
gpt-oss-120b (4-bit)	275–309	79.1
Qwen3-30B Thinking (8b)	509–739	35.2–35.3
Qwen3-30B Instruct (8b)	404–641	35.5–76.2

Cost on consumer hardware. The cost that matters to an adversary is wall clock time on owned hardware, not the streaming rate. A complete offensive fine-tune of the 30B model finished in 6.5 hours over 8.0 million tokens on a single Apple Silicon machine; the 4B model, run to a longer schedule, took up to 18.6 hours and 25.2 million tokens (wall clock here tracks the iteration budget, not model size). Throughput was 275–740 tokens per second within 34.6–79.1 GB of memory (Table 7); the 120B model was the slowest and most memory-hungry, at about 275 tokens per second and 79.1 GB. There is no cloud bill and no costs beyond hardware and electricity: one desktop left running for an afternoon or overnight is the entire capital requirement.

7 Measured Capability–Safety Evaluation

Beyond training dynamics (§6), we evaluate the models themselves along two axes that jointly define the dual-use question: *alignment* (an aligned base vs. its abliterated counterpart) and *fine-tuning* (the base vs. the same base offensively fine-tuned on VulnExploit-SFT). Every cell is measured on three suites.

We train a LoRA adapter for each base on the offensive data, following the recipe of §5, at learning rate 1×10^{-4} for the aligned bases and 5×10^{-5} for the abliterated ones.

Capability. Cybersecurity knowledge is scored with CyberMetric-500 [12], a human-validated four option multiple-choice benchmark, as exact match accuracy.

Safety. We measure two rates over the held out exploit request set. The *refusal rate* is the fraction of requests the model declines; the *attack-success rate* (ASR) is the fraction for which it returns concrete, actionable exploitation content. To isolate the model's own alignment from any prompt level jailbreak, the exploitation ask is issued under a *neutral* system prompt rather than the dataset's red team system prompt, so the measured refusal reflects the model (and adapter), not the framing.

Quality. Under the dataset's intended use audit prompt, a judge scores each audit answer against the ground truth reference on a 1–5 correctness and format rubric.

Judge. Safety labeling and quality scoring use an abliterated judge model, a deliberate choice: a safety-aligned judge would itself refuse to read and score exploitation content, corrupting exactly the measurements we need. The judge is a scorer, not a subject under test; we flag this as a validity caveat, and note that judge labels were spot-checked by hand.

We evaluate six base models: Qwen3-4B, Qwen3-30B-A3B, and gpt-oss-20B, each in an aligned and an abliterated variant, as-is and after offensive fine-tuning, for twelve configurations in total (Table 8). CyberMetric runs over all 500 questions; the safety and quality suites use 150 held out exploit requests each, scored by the judge.

Abliteration is nearly capability free. Across all three model families, removing refusals costs little cyber knowledge: CyberMetric moves by at most a few points between an aligned base and its abliterated counterpart (e.g. Qwen3-4B 89.0 → 88.2, Qwen3-30B 93.6 → 91.4). Yet the same abliteration collapses refusal to near zero and pushes ASR toward saturation. Un-aligning a capable security model is, in capability terms, almost free.

Alignment is thin against security framing, and family dependent. The aligned bases differ sharply in how much they refuse the (neutrally framed) exploit requests: Qwen3-30B-aligned refuses only 1.3% despite

Table 8: Capability–safety evaluation matrix. Capability is CyberMetric-500 accuracy (%); safety is refusal rate and attack-success rate (ASR, %) on held out exploit requests; quality is the abliterated-judge audit score (1–5). For each base model we report the model as-is and after offensive fine-tuning (ft). Higher accuracy/quality and higher refusal are safer directions; higher ASR is more dangerous. † computed over valid generations (25.3% generation-failure rate, see text).

Model	Align.	CyberMetric	Refusal	ASR	Quality
Qwen3-4B	aligned	89.0	22.7	63.3	3.84
	aligned +ft	90.2	0.0	96.0	4.19
Qwen3-4B	abliterated	88.2	0.0	92.0	3.95
	abliterated +ft	88.6	0.0†	92.9	4.18
Qwen3-30B	aligned	93.6	1.3	93.3	4.42
	aligned +ft	93.0	0.0	100.0	4.62
Qwen3-30B	abliterated	91.4	0.0	100.0	4.75
	abliterated +ft	91.4	0.0	99.3	4.72
gpt-oss-20B	aligned	91.8	23.3	71.3	4.16
	aligned +ft	85.8	0.0	98.0	4.61
gpt-oss-20B	abliterated	88.6	0.0	98.0	4.06
	abliterated +ft	78.6	0.0	98.7	4.52

being the most capable model in the study, whereas the smaller Qwen3-4B-aligned refuses 22.7% and gpt-oss-20B-aligned 23.3%. Refusal therefore does not simply track capability; gpt-oss retains meaningful alignment at a capability (91.8) comparable to Qwen3-30B's, so the erosion is a property of the model's safety training, not of scale alone.

Offensive fine-tuning strips alignment on every base. This is the central result. Fine-tuning drives refusal to 0% and ASR to 96–100% for *every* configuration, including the two that retained real refusal: Qwen3-4B-aligned (22.7% → 0% refusal, 63.3 → 96.0 ASR) and gpt-oss-20B-aligned (23.3% → 0%, 71.3 → 98.0). Abliteration is one route to a refusal-free security model; offensive fine-tuning is an independent one that reaches the same endpoint from an aligned start, and audit quality *rises* throughout (every ft row scores higher than its base).

Fine-tuning taxes gpt-oss capability but not Qwen. The one axis on which the families diverge is capability retention under fine-tuning. Qwen barely moves (Qwen3-4B 89.0 → 90.2, Qwen3-30B 93.6 → 93.0), but gpt-oss loses ground: 91.8 → 85.8 (aligned) and 88.6 → 78.6 (abliterated). Offensive fine-tuning is not uniformly free: on gpt-oss it trades measurable capability for the removal of refusals.

Fine-tuning an abliterated base is unstable. The Qwen3-4B abliterated model, fine-tuned on the offensive data, produced degenerate generations (empty output or special-token loops) on 25.3% of inputs even after lowering the learning rate; we exclude those from the rates and report the failure directly (Table 8, †). The 30B abliterated model was stable, so this is a small model fragility rather than a general one, but it means stacking fine-tuning on an already-de-aligned base is the least reliable path, not a strictly stronger one.

8 Dual-Use Analysis and Broader Impacts

None of the individual steps above is novel; their cheap combination, and its implication, is the point of the paper. Every ingredient needed to produce a locally hosted, exploitation oriented security model is now freely available and cheap to combine: authoritative sources, a public vulnerable code corpus [3], open abliterated models [1], and an on-device training stack (§6). No cloud provider, and therefore no safety moderation, sits anywhere in this loop.

Threat model. The actor we have in mind is not slamming rate-limits, but analyzing vulnerabilities offline. They do not need an API budget, data sharing agreement, and any moderation filter to evade. As §6 and §4 show, the barrier is low: de-alignment does not impair fitting the security data, the offensive objective is in fact *easier* to fit than the defensive one, and the offensive branch simply omits the safety-alignment stage that the defensive pipeline includes.

Threat intelligence implications. Our measurements (§7) point to a single, actionable threat intelligence conclusion: the realistic near-term offensive LLM threat is *offline, refusal-free, and capable*, and therefore invisible to the detection surface most defenders rely on. Three mechanisms compound. First, because the whole loop is local, no provider side telemetry or abuse signal is ever generated; there is no API to monitor. Second, because abliteration is capability-free (§7), an adversary obtains a fully capable security model without any

fine-tuning skill. Third, exploitation requests framed as ordinary code audits do not resemble jailbreaks, so prompt level signatures miss them. The defensive takeaway is to expect less API level or jailbreak signature detection to catch this class of tool and to shift toward endpoint, output-artifact, and open source signals. For example the characteristic audit/exploit output structure and the public corpus provenance fingerprints of the training data. The residual barrier is not infrastructure but *knowledge* (which weights to download and which tool to run), and that knowledge is public and social. The useful collection surface is therefore the distribution layer, not the local inference it enables: abliteration tooling repositories, the model hubs that host de-aligned outputs, and the forums that circulate configurations and setups are cheaper to watch and carry earlier signal. The *marginal* uplift of offensive fine-tuning over simply using an abliterated base (§7) is itself a triage signal: a small delta means the abliterated base is the threat and bespoke fine-tunes do not need to be tracked separately, while a large delta marks custom offensive fine-tuning as a distinct escalation. In our matrix this delta is small on the safety axis: abliterated bases are already refusal-free and near saturated on attack-success (Table 8), so the abliterated base itself, not a fine-tune, is the operative threat.

Hardware floor. Our experiments ran on an Apple M4 Max with 128 GB of unified memory, but that overstates the requirement: the threat action is *inference* of an abliterated base, and inference is memory-bound. Table 9 gives each model's computed footprint at standard GGUF quantization. The 4B model fits in under 4 GB, within reach of any laptop of the last decade, yet still scores 88.2 on CyberMetric with refusals removed. The mixture-of-experts models are the sharper case: they keep 20–30B parameters resident but compute only ∼3.3B per token, so the most capable model in the study (93.6 on CyberMetric) runs at interactive speed on a commodity 32 GB mini-PC with no GPU (roughly $550 as a complete machine in mid-2026) or a used 24 GB card (an RTX 3090 around $700), and hardware many operators already own costs nothing further. That floor has in fact risen over 2026: the same AI demand this paper studies drove a memory and GPU shortage that roughly doubled these prices. The barrier moved, but only from pocket change to a few hundred dollars.

What we withhold, and why. Consistent with responsible disclosure norms for dual-use security research, we report enough to substantiate the risk and no more:

- We do *not* release the offensive dataset, its exploit content, or any example payloads. It is characterized only by size and provenance (§4).
- We do *not* release offensive adapter weights or fused models.
- We report only *aggregate* capability and safety metrics (§7); we do not release the per-example generations behind them, an offensive utility scorecard, or any runnable tool.
- The upstream sources we name are already public; we add no new offensive capability, only a measurement of how cheaply existing pieces compose.

Significance. The safety/performance trade-off in cybersecurity fine-tuning is already documented [4, 10, 9]; what is new is that it can now be realized entirely offline, at negligible cost, on a de-aligned base. Defenders, model distributors, and hardware/software vendors benefit from knowing this: it argues for treating abliterated model distribution, forums, and local fine-tuning as part of the threat surface, not an afterthought.

9 Limitations

Evaluation validity caveats. The abliterated models are off-the-shelf community builds whose quantization differs from their aligned counterparts, so a small part of any aligned-versus-abliterated capability gap may be quantization rather than abliteration. The judge is a single abliterated model, not a human panel; we spot-checked its labels but did not measure inter-rater agreement. The Qwen3-4B abliterated fine-tune was unstable (25.3% generation failures, excluded from its rates), which both narrows its effective sample and signals that that particular cell is the least reliable in the matrix.

Measurement caveats. Domain coverage is computed by overlapping keyword matching and so should be read as coarse. Throughput and memory figures come from a single machine and a small number of runs; several runs were killed or crashed, and this run set is not a controlled sweep. The aligned-versus-abliterated comparison holds the data fixed but varies only two runs, so it is indicative rather than statistically established.

Artifacts not released. Beyond the offensive artifacts withheld in §8, the defensive dataset is also not released, pending a per-source terms-of-service review. This limits direct reproducibility of the data, a tension we accept in exchange for not distributing dual-use artifacts.

Table 9: Inference hardware floor. Memory footprint is computed (weights at the listed quantization + an 8k-token KV cache + runtime overhead) and matches published GGUF sizes. 'Active' is the parameters computed per token, which for the mixture-of-experts models is far below the resident total, so the 30B-class model runs at interactive speed on a CPU. Costs are approximate US used/consumer prices surveyed in August 2026 for the cheapest sufficient hardware; the 2026 AI-driven memory and GPU shortage has raised them from earlier lows, yet the floor stays within a few hundred dollars.

Model	Mem	Active	Cheapest hardware	Cost
Qwen3-4B (dense)	3.7 GB	4.0 B	any 8 GB laptop/PC, CPU	owned / $0
gpt-oss-20B (MoE)	12.2 GB	3.6 B	used 16 GB GPU or 16 GB PC	≈$550
Qwen3-30B-A3B (MoE)	19.5 GB	3.3 B	32 GB mini-PC (CPU) or used 24 GB GPU	$650–800

10 Conclusion

We reimplemented a published cybersecurity instruction data pipeline to run entirely on a local machine, removing every external service from the construction and fine-tuning loop, and we used the same infrastructure to build an offensive branch that omits safety alignment and trains on exploitation data over de-aligned bases. Measuring a full alignment × fine-tuning matrix across three model families, we found abliteration and offensive fine-tuning to be two cheap routes to a refusal free, capable model, with a complete fine-tune finishing in hours on commodity hardware only costing electricity. The contribution is a warning made concrete and measured: the pieces required to build a locally hosted, exploitation AI model are public, cheap, and offline, and the safety guardrails that a cloud pathway would impose are absent by design. Treating on-device fine-tuning, abliterated model distribution, and configuration forums as first class parts of the threat surface is, we argue, the appropriate response.

References

[1] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024.

[2] Manish Bhatt, Sahana Chennabasappa, Cyrus Nikolaidis, Shengye Wan, Ivan Evtimov, Dominik Gabi, et al. Purple Llama CyberSecEval: A secure coding benchmark for language models. *arXiv preprint arXiv:2312.04724*, 2023.

[3] CyberNative AI. Code vulnerability security DPO. Hugging Face dataset, 2024. `https://huggingface.co/datasets/CyberNative/Code_Vulnerability_Security_DPO`.

[4] Adel ElZemity, Budi Arief, and Shujun Li. CyberLLMInstruct: A pseudo-malicious dataset revealing safety-performance trade-offs in cyber security LLM fine-tuning. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security (AISec '25)*. ACM, 2025. arXiv:2503.09334.

[5] Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. LLM agents can autonomously exploit one-day vulnerabilities. *arXiv preprint arXiv:2404.08144*, 2024.

[6] Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: An array framework for apple silicon. `https://github.com/ml-explore/mlx`, 2023. Apple Machine Learning Research.

[7] Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2024.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[9] Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. LoRA fine-tuning efficiently undoes safety training in Llama 2-Chat 70B. *arXiv preprint arXiv:2310.20624*, 2023.

[10] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.03693.

[11] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.18290.

[12] 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. *arXiv preprint arXiv:2402.07688*, 2024.