Evaluating Locally Hosted LLMs for CPE Identification in IoT Reconnaissance

Christopher M. Davisson and Sanmeet Kaur

***Abstract*—Vulnerability identification during penetration testing relies on rigid string matching to map network scan data to Common Platform Enumeration (CPE) identifiers and downstream CVE records. The approach fails routinely on physical IoT devices, whose truncated and non-standard service banners resist deterministic parsing. Large language models can reason through these fuzzy associations, but cloud-hosted models raise cost, latency, and operational-security concerns when the prompt contains reconnaissance data from a live network. We ask whether locally hosted open-weight LLMs can perform this task well enough to be useful, and how performance varies with model scale, family, and prompt design. We evaluate 36 LLMs across three deployment tiers (frontier hosted APIs, remotely hosted open-weight models, and local vLLM serving), plus Nmap's native CPE extraction as a no-LLM baseline, on a curated corpus of Nmap scans covering 16 IoT device entries, scored against hand-labeled ground truth with a tiered rubric that distinguishes exact matches, operationally useful partial matches, and well-formed but factually wrong identifiers. Three findings shape the result. Model family is the dominant source of variance, with a spread (0.28) wider than prompt design (0.10) or scan-suite choice (0.23) produces. The best truly-local model, an 8B security-domain fine-tune, reaches 0.332 mean match score on a single 24 GB consumer GPU, approaching the weakest frontier model (0.309). The most dangerous failure mode is not malformed output but a confident, well-formed CPE that retrieves the wrong vulnerability list: 21.9% of predictions overall, and nearly twice as frequent in the local tier (26.1%) as the frontier tier (14.5%).**

***Index Terms*—Penetration testing, reconnaissance, large language models, CPE, IoT fingerprinting, local inference, hallucination.**

I. Introduction

PENETRATION testing proceeds through planning, discovery, attack, and reporting phases [1]. Discovery is where the operator enumerates hosts, services, and software versions to establish what the network contains and what attack surface it exposes. The tooling around this phase — Nmap, Masscan, Shodan — matured under the assumption of a human operator interpreting the output. That assumption is eroding. LLM-driven frameworks such as PentestGPT [2] and AutoPT [3] now automate progressively larger portions of the workflow, and most of them route reconnaissance data — network topology, service banners, software versions, inferred vulnerabilities — through cloud-hosted frontier models operated by third parties. Using those models requires trusting that the provider does not retain the data, does not record it in a discoverable form, and does not leak it [4].

An organization whose security posture depends on the confidentiality of its internal infrastructure takes on real risk when it transmits a complete reconnaissance dataset to third-party servers [4], [5], [6]. The alternative is local inference, which eliminates external data exposure at the cost of access to proprietary frontier models. Local hosting faces a hard ceiling set by available VRAM: a workstation with one Nvidia RTX 4090 (24 GB) can serve models in the 7B–34B parameter range at moderate quantization, and pushing below 4-bit precision degrades reasoning measurably [7], [8]. The operational question is not whether a local model is preferable in principle but whether a model that fits these constraints can handle a real reconnaissance task reliably.

This paper isolates one such task: generating CPE 2.3 identifiers [12] for physical IoT devices from imperfect, incomplete Nmap scan output, suitable for downstream CVE association through NVD-style lookup [13], [14]. IoT devices are the regime where deterministic fingerprinting has the least to work with — minimal exposed services, truncated banners, embedded TCP/IP stacks that defeat OS detection [11] — and therefore the regime where a model that can reason over weak, scattered evidence has the most marginal value to offer.

The work began with a misidentification on a home network. A Toshiba Fire TV was reported by Nmap as a phone running Android 5.1. The OS family was approximately right (Fire OS is Android-derived) but the device class was wrong, the MAC OUI pointed at a contract manufacturer rather than a phone vendor, and the open ports — an AirPlay-style RTSP service on 7000, a Cast-receiver endpoint on 8009, a Netflix NRDP server header on 9080 — individually meant little but jointly identified a smart TV. A binary scoring rubric marks that answer simply wrong. A tiered rubric distinguishes a partially correct OS family from an invented vendor, and that distinction is exactly what determines whether the output is operationally usable. Designing the evaluation around it is the central methodological commitment of this study.

We make three contributions. First, we isolate CPE generation from Nmap output as a measurable reconnaissance subtask, rather than evaluating it as one step inside a larger penetration-testing agent. Second, we compare frontier hosted models, remotely hosted open-weight models, locally served open-weight models, and Nmap-native CPE extraction on the same physical and derived IoT scan corpus. Third, we score output with a tiered rubric that separates exact matches,

C. Davisson and S. Kaur are with the Department of Computer Science, Eastern Washington University, Cheney, WA. E-mail: [redacted for review].

operationally useful partial matches, malformed output, and well-formed but factually incorrect identifiers — the security-relevant form of hallucination.

Three research questions structure the evaluation:

1) **Accuracy and its drivers.** How does CPE-identification accuracy vary across deployment tier, model family, model scale, prompt design, and scan-suite choice, and which factor produces the widest spread?
2) **Local versus hosted.** How much accuracy is lost moving from hosted frontier models to local open-weight models, and do the strongest local models clear the Nmap-native baseline and approach the weakest frontier model?
3) **Failure modes.** How often do models produce well-formed but factually incorrect CPEs, and how does that risk vary with family, size, and tier?

II. Related Work

LLM-based penetration testing. PentestGPT, AutoPT, PentestAgent, and xOffense run LLMs inside multi-step reasoning loops that plan an action, read tool output, and write the next step [2], [3], [15], [16]; AutoPenBench formalizes evaluation across structured tasks [17]. These systems establish that LLMs can assist with security work, but they evaluate end-to-end agents, which obscures *where* a system fails: wrong tool, misparsed output, hallucinated vulnerability, or incorrect product identifier all collapse into one task-level failure. They also assume access to strong hosted models — an assumption that is fine for benchmarks and uncomfortable for engagements whose prompts describe live hosts. We isolate one subtask and treat local inference as part of the research problem rather than a deployment detail.

Automated CPE identification. CPE-Identifier [14] and Wåreus and Hell [18] frame CPE labeling as machine learning over CVE *descriptions* — human-authored text about known vulnerabilities. Our input is raw reconnaissance output: machine-generated, incomplete, and ambiguous. A banner can expose a library version without naming the product that embeds it; an OUI often points at the contract manufacturer rather than the brand on the case. The task is inference from weak evidence, not extraction from clean text. The output space also differs: prior work treats the task as classification over a known vocabulary, while we evaluate open-vocabulary generation of full CPE 2.3 strings, which exposes the failure class where a syntactically valid CPE names the wrong product and silently retrieves a plausible vulnerability list for it.

IoT fingerprinting. IoT Sentinel and related device-classification work show that network behavior alone often suffices to infer device type, vendor, or class [9], [10], and the same literature explains the difficulty: reused embedded stacks, minimal service surfaces, generic banners. Most of this work classifies from passive traffic features and stops short of CPE strings. A label like "camera" helps an analyst understand the network; it does not support CVE lookup. A CPE does.

Structured output and abstention. Grammar- or regex-constrained decoding can prevent malformed strings by construction, at the documented cost of occasionally forcing low-probability continuations that distort reasoning [21]. The distinction between syntactic validity and factual correctness is central here: malformed output fails loudly and cheaply, while a well-formed wrong CPE fails silently. Abstention matters for the same reason — a wildcard or partial identifier can be more useful than a specific but unsupported guess — and our rubric and prompt inventory both encode it. The doubled-prompt condition (Section IV-E) operationalizes the "lost in the middle" finding that models do not weight all regions of a long context equally [19].

III. Setting and Threat Model

Operational scenario. The intended user is a defender or authorized penetration tester running active reconnaissance against a network they are responsible for, inside a structured engagement. The operator has Nmap output for a device; that output is incomplete or ambiguous; the operator wants a structured identification suitable for CVE lookup.

Trust boundary. A reconnaissance dataset is a map of the target environment. Sending it to a hosted API extends the engagement's trust boundary to the provider, every subprocessor, every employee with log access, and every adversary capable of compromising any of them. For operators under compliance regimes — HIPAA, FedRAMP, CMMC, the financial-sector rules governing broker-dealers and registered investment advisers — the requirements are concrete: data sovereignty, audit logging, subprocessor restrictions, limits on transmitting operationally sensitive data outside the controlled environment. Enterprise API tiers address some of these; the consumer and developer APIs that LLM-augmented pentest tools actually integrate generally do not. The design constraint that follows is local inference: execution on operator-controlled hardware with no network egress at inference time. That constraint caps model size at what the hardware can host and what has been released open-weight, and that trade-off is the tension this paper measures.

CPE and partial correctness. A CPE 2.3 string carries eleven colon-separated fields after the `cpe:2.3:` prefix — part, vendor, product, version, and seven qualifiers — forming a left-to-right hierarchy in which each field narrows the set of matching products [20]. Consider three failure modes against the same device. (1) *Correct vendor and product, wildcard version:* the NVD query returns every CVE ever filed against the product. The list is wider than necessary but scoped to the right device; analyst time spent on it does real work. (2) *Correct vendor, wrong product:* the returned list is scoped to a different device, every entry is irrelevant, and nothing signals that. Analysis time here is worse than wasted — it produces confident, wrong conclusions. (3) *Structurally malformed:* the lookup never executes, the analyst sees the error immediately, no false confidence is generated. A binary rubric collapses all three into one failure bucket and erases the only distinction that matters operationally. Our tiered rubric (Section IV-F) is built around it.

IV. Experiment Design

A. Pipeline

The pipeline is six steps: scan, ingest into MongoDB alongside per-device ground truth, run every scan through every model under every prompt in normal and doubled forms, record the response, score against ground truth, aggregate. Each step is a separate script that reads database state left by the prior step; there is no orchestrator and no shared memory, so any step re-runs independently — which matters when inference takes days and re-scoring against a revised rubric takes seconds. The invariant making the heterogeneous middle tractable is that every inference backend writes the same document shape (input scan, prompt, doubled flag, trial number, model identifier and sampling parameters, raw response, parsed CPE list, scores), so scoring is implemented once and applied uniformly. An earlier design included a stage that queried the NVD with each predicted CPE; we dropped it because it measured database behavior rather than model capability — once the CPE is correct enough, the lookup is mechanical [12]. All reported analysis operates on a frozen export of the database that pins device, scan, prompt, model, and score counts at a fixed moment, so later re-runs cannot silently shift the figures.

B. Dataset

The physical testbed comprises 14 commercial IoT devices scanned on a controlled network in stock factory configuration: smart-home hubs, smart bulbs and plugs, networked cameras, multifunction printers, smart TVs and streaming devices, a cable gateway, retail WiFi routers, and a Hak5 Tetra pentest appliance as a deliberately atypical case. Three entries are the same WiFi repeater operating in AP, router, and repeater modes — intentional, since the prediction depends on observable network behavior rather than hardware identity, and the modes expose different services. Four further entries were derived from the Shodan IoT scan corpus and re-ingested through the same pipeline (a Vizio smart TV, a Nintendo Wii U, a first-generation Wink Hub, a SmartThings Hub V1). Two entries are excluded from quantitative results: the Wii U scan contained no responsive ports or banners and is retained only as a pipeline-behavior probe, and the repeater-mode scan was collected from a different layer-2 segment, so the MAC-based ground-truth resolution could not bind it. That leaves 16 analyzed entries.

Ground truth per device is a hand-curated list of accepted CPE 2.3 strings, each annotated *exact*, *partial*, or *related*. The Fire TV labels show the rule: the specific `amazon:fire_os` identifier is exact; `google:android` is partial because Fire OS is Android-derived; `linux:linux_kernel` is partial because Android runs the Linux kernel. A model that calls a Fire TV's OS Android is less specific than one that names Fire OS, but it is not equivalent to naming an unrelated product. Each ground-truth document records the rubric version under which its list was authored.

TABLE I
NMAP SCAN SUITES.

Name	Flags
01-sv-osc-top1000	`-sV --version-intensity 5 -O -sC --top-ports 1000 -T4`
02-sv-intensity1	`-sV --version-intensity 1`
03-sv-intensity5	`-sV --version-intensity 5`
04-sv-intensity9	`-sV --version-intensity 9`
05-dns-sd-udp5353	`--script dns-service-discovery -sU -p 5353`
06-upnp-udp1900	`--script upnp-info -sU -p 1900`

C. Scan Suites

Six Nmap suites per device span a deliberate range of information density (Table I), testing whether identification accuracy is more sensitive to evidence sparsity or to noise. Suite 01 is the most aggressive (service-version detection, OS detection, default scripts, top 1000 TCP ports); suites 02–04 hold port selection constant and vary version-detection intensity; suites 05–06 are protocol-specific UDP probes for the service-discovery announcements IoT devices emit in lieu of standard TCP banners.

D. Models

The lineup spans three deployment tiers plus a non-LLM baseline. *Local (vLLM):* seventeen open-weight models served on a multi-GPU workstation (4-way tensor parallel), spanning general-purpose instruction-tuned families (Mistral, Qwen, Llama, Phi, Gemma, Granite, OpenAI's open-weight release), security-domain fine-tunes (Foundation-Sec, SecurityLLM, WhiteRabbitNeo, Llama-Primus), and one community fine-tune. The multi-GPU setup extends past a single 24 GB card into the 24B–32B range; we state plainly that the local results represent a well-resourced workstation, not a single consumer GPU, except where noted. *Ollama Cloud:* ten open-weight models served through Ollama's hosted catalog. *Frontier API:* nine hosted models — three each from OpenAI (`gpt-5.5`, `gpt-5.4`, `gpt-5.4-mini`), Anthropic (`claude-opus-4-7`, `claude-sonnet-4-6`, `claude-haiku-4-5`), and Google (`gemini-3.1-pro`, `gemini-3.1-flash-lite`, `gemini-3-flash`). Frontier requests use provider-default sampling, deliberately: the goal is the response a typical API consumer receives. Local and Ollama Cloud runs fix temperature 0.0, top-p 1.0, and a constant seed; determinism remains bounded by engine-level batching and floating-point nondeterminism [8]. *Baseline:* Nmap's own CPE guesses, extracted from the scan XML and scored by the same rubric, ground every accuracy claim against what the deterministic tool already emits.

One coverage gap affects the analysis: per-token cost prevented running the doubled-prompt variant on Anthropic and OpenAI models, so doubled-versus-normal comparisons are restricted to the local, Ollama Cloud, and Gemini families.

E. Prompts

Seven prompts form a 2×2 factorial — persona (cybersecurity-analyst framing present or absent) $\times$

structure (CPE format rules, schema, and an example present or absent) — plus three probes: `evidence_first` (three-step chain-of-thought with an explicit abstention clause), `few_shot_grounded` (three labeled examples covering strong, weak, and mixed evidence), and `conversational-kind` (casual register). Prompts are immutable once referenced by a run, keyed by name and version.

Each (scan, prompt, model) cell runs twice: a normal system/user split, and a *doubled* construction that drops the system role and builds one user message as [prompt + scan + delimiter + prompt + scan]. The construction targets the lost-in-the-middle effect [19]: aggressive scan suites produce payloads of thousands of tokens, and an instruction prefixed to that payload risks under-attention, so sandwiching places a copy in the high-recency region regardless of how the model's attention prior is distributed.

F. Scoring

Inference runners extract JSON from the raw response and keep CPE-like strings that pass a structural filter (begins `cpe:2.3`, part in $\{h, o, a\}$, non-wildcard vendor and product). The scorer lowercases prediction and accepted entries, pads both to the 13 colon-separated CPE 2.3 components, and declares a match when every non-wildcard field of an accepted entry equals the prediction; when several accepted entries match, the highest tier wins. Tiers carry weights 1.0 (exact), 0.5 (partial), 0.25 (related, reserved and unused in the final labels), 0 (none). The reported mean match score over an aggregation group P of per-prediction analysis rows is

$$\text{MeanMatch} = \frac{1}{|P|} \sum_{p \in P} w(\text{tier}(p)), \tag{1}$$

where a run producing no parseable CPE contributes one zero row, so the design penalizes both abstention and overproduction: a response carrying one useful identifier alongside several unsupported ones earns credit for the useful one while the unsupported rows pull the average down. The scorer also records field-level flags (part, vendor, product, version correctness against the best-overlap accepted entry); these are field-agreement diagnostics, not success rates — a tier-*none* prediction can still register a correct vendor field.

We define a *hallucination* as a prediction that survives the structural filter (well-formed, non-wildcard vendor and product) but matches no accepted reference: vendor field wrong and match score zero. This is the silent failure class of Section III, scored separately from malformed output because the two have opposite detection properties.

Primary analyses use the unguided trial only. Trials passing a CPE-shaped regex to vLLM's `guided_regex` parameter were collected but did not behave as intended (Section VI) and feed only the methodological discussion.

V. Results

A. Model Family Dominates

Mean match score spreads 0.28 across families, from Claude (0.395) to Llama (0.116) — wider than the gap between the best and worst prompt (0.10) and between the best and worst scan suite (0.23), making family choice the single largest source of variance in the experiment (Table II). Scale tracks family at the bottom: the Llama family's position is driven by its smallest member, included deliberately to probe the open-weight floor, and the Mistral family's average is similarly dragged by a 3B entry. The task is not uniformly difficult across devices either: some devices score moderately across most models, indicating sufficient scan evidence, while others stay hard for nearly every family — failures driven by weak evidence rather than any single poor model. The devices that frontier models handle and local models do not mark the capability gap most relevant here.

TABLE II
MATCH SCORE AND HALLUCINATION RATE BY MODEL FAMILY.

Family	Mean	Exact	Halluc.	Vendor	n models
Claude	0.395	0.166	0.166	0.740	3
GLM	0.395	0.173	0.158	0.686	1
GPT	0.390	0.167	0.143	0.722	3
DeepSeek	0.381	0.167	0.135	0.681	2
Gemini	0.358	0.146	0.132	0.697	3
Kimi	0.358	0.137	0.172	0.729	2
Gemma	0.353	0.159	0.143	0.678	2
Phi	0.286	0.139	0.223	0.581	1
SecTuned	0.258	0.108	0.217	0.558	5
Qwen	0.249	0.104	0.224	0.532	4
GPT-OSS	0.245	0.104	0.292	0.611	1
Granite	0.215	0.092	0.318	0.546	1
Mistral	0.211	0.085	0.264	0.542	4
Other	0.203	0.102	0.287	0.407	2
Llama	0.116	0.051	0.373	0.380	2

TABLE III
MATCH SCORE AND HALLUCINATION RATE BY PROMPT.

Prompt	Mean	Exact	Halluc.	n
persona_structured	0.321	0.138	0.247	9,503
neutral_structured	0.312	0.132	0.247	9,052
few_shot_grounded	0.303	0.147	0.184	7,997
persona_minimal	0.270	0.116	0.230	9,730
neutral_minimal	0.262	0.109	0.241	9,249
conversational-kind	0.259	0.103	0.221	9,061
evidence_first	0.226	0.082	0.134	6,754

B. Prompt Effects Are Real but Second-Order

Prompt design moves mean score across a 0.10 range (Table III). The 2×2 marginals decompose it: structure — explicit CPE field rules, format guidance, an example — adds $+0.051$; the analyst persona adds $+0.009$. Format rules carry roughly six times the effect of role framing, and the two best prompts are precisely the two structured ones. The probes each tell a distinct story. `few_shot_grounded` posts the highest exact-match rate in the inventory (0.147) with the second-lowest hallucination rate (0.184): examples push models toward precision when evidence supports it and abstention when it does not. `evidence_first` achieves the lowest hallucination rate (0.134) and the lowest mean score (0.226) — its abstention clause does what it was designed to do, refusing identifications under weak evidence at the cost of also refusing some it could have made. The casual-register

TABLE IV
MATCH SCORE AND HALLUCINATION RATE BY SCAN SUITE.

Suite	Mean	Exact	Halluc.	n
01-sv-osc-top1000	0.343	0.133	0.212	14,607
02-sv-intensity1	0.341	0.149	0.184	11,324
03-sv-intensity5	0.340	0.151	0.187	11,262
04-sv-intensity9	0.255	0.152	0.287	7,324
06-upnp-udp1900	0.123	0.022	0.275	7,448
05-dns-sd-udp5353	0.112	0.024	0.229	6,946

TABLE V
DEPLOYMENT-TIER COMPARISON.

Tier	Mean	Exact	Halluc.	Vendor	n models
Frontier (API)	0.378	0.158	0.145	0.717	9
Ollama Cloud	0.306	0.126	0.193	0.633	10
Local (vLLM)	0.231	0.101	0.261	0.524	17

prompt performs like `neutral_minimal` despite the very different design philosophy.

The doubled-prompt variant is null in aggregate across the families where both modes ran (0.270 normal vs. 0.267 doubled, $\Delta = -0.003$) but asymmetric by size: small families gain (Llama $+0.026$, Mistral $+0.017$, Phi $+0.016$, Gemma $+0.016$) while large ones lose (DeepSeek -0.018, GLM -0.016, Gemini -0.013). The pattern is consistent with lost-in-the-middle [19]: smaller models attend less reliably to an instruction buried ahead of a long scan payload and benefit from reinforcement in the high-recency region; larger models attend to the original instruction anyway, so the duplicated payload only adds noise.

C. Scan Density: More Probing Is Not More Signal

Three patterns emerge from Table IV. The top three suites cluster within 0.003 of each other: adding OS detection and the default script set to a top-1000 probe yields essentially the same mean accuracy as the lightest version-detection-only scan. Raising version-detection intensity from 5 to 9 *drops* mean score by 0.085 and raises hallucination 10 percentage points — the intensity-9 probes elicit responses most services were never designed to give, and the extra banner text confuses more than it informs. The two UDP service-discovery suites sit 0.22 below the worst TCP suite at exact-match rates of 0.022–0.024: when the device does not participate in the protocol, the scan returns nothing identifying, and models that attempt an identification anyway are guessing.

D. Local Versus Frontier

Frontier models lead on every metric: $+0.147$ tier-mean score over local, with roughly half the hallucination rate (Table V). The Ollama Cloud tier mean understates its strongest entries — the tier mixes frontier-comparable models (`glm-5.1` at 0.395, `deepseek-v4-flash` at 0.383) with 3B models that drag the average. Tier means describe averages, not ceilings: the best model per tier is `claude-opus-4-7` at 0.4398 (frontier), `glm-5.1:cloud` at 0.3946 (Ollama Cloud), and `fdtn-ai/Foundation-Sec-1.1-8B` at 0.3320 (local). The best-to-best frontier-to-local gap is 0.108 — smaller than the tier-mean gap and approximately equal to the spread between the best and worst prompt. A defender restricted to local inference runs roughly one prompt-design tier behind the best hosted model.

Two local-tier findings sharpen the picture. Foundation-Sec-1.1-8B reaches 0.332 on a security-domain fine-tune over an 8B base small enough for a single 24 GB consumer card, within reach of the weakest frontier entry (`gemini-3-flash` at 0.309). And the local tier's hallucination rate (0.261) nearly doubles the frontier rate (0.145), compounding the accuracy gap: local predictions are both less accurate and more likely to be confidently wrong.

The comparison against the deterministic tool cuts against the framing the tier means invite. Computed per scan on inputs where Nmap emitted its own CPE, only a handful of hosted models clear the baseline: `gemini-3.1-pro` ($+0.072$), `gpt-5.5` ($+0.067$), `deepseek-v4-pro` ($+0.041$), Claude Opus ($+0.040$), `glm-5.1` ($+0.036$). The bottom of the frontier tier does not (`claude-haiku` -0.027, `gpt-5.4-mini` -0.020, `gemini-3-flash` -0.010), and the entire local tier is negative — Foundation-Sec at -0.068. No locally served model beat Nmap on Nmap's own output. That finding looks worse than it is, for a selection-effect reason worth stating plainly: Nmap emits a CPE only when its version database already matches the banner, so its predictions concentrate on the easy scans and it stays silent on the rest. Its mean of 0.382 across 144 predictions exceeds the frontier tier's 0.378 across more than ten thousand not because Nmap reasons better but because it never attempts the hard cases. A deterministic matcher with high precision and near-zero recall is hard to beat where it deigns to answer. The operational value of an LLM is producing a usable identifier on the scans where Nmap returns nothing — which is the regime this testbed was built to stress.

E. Hallucination

Across the 61,346 LLM predictions in the filtered set, 13,432 (21.9%) were flagged as hallucinations; the outcome distribution is 11.9% exact, 32.4% partial, 55.7% none, with the reserved *related* tier unassigned. The rate scales inversely with size and frontier status. The single worst offender is the smallest model in the lineup (`Llama-3.2-1B` at 43.1%); seven of the ten worst are under 10B parameters; the per-tier rate climbs from 0.145 (frontier) through 0.193 (Ollama Cloud) to 0.261 (local). One figure invites misreading: `ZySec-AI/SecurityLLM` posts the study's lowest hallucination rate (0.017), but only because it emits parseable CPE output on roughly 3% of its invocations (mean score 0.012). Near-total abstention is not reliability.

VI. DISCUSSION

A. Where Models Fail and Why

The failures that matter are the quiet ones. A malformed CPE fails to parse, the lookup never executes, and the operator knows immediately. An abstention produces an empty response. Both degrade aggregate accuracy; neither produces

dangerous output. The dangerous output is a plausible, well-formed CPE whose NVD lookup returns the wrong vulnerability list without flagging anything — and at 21.9% of predictions overall, it is not rare.

Two infrastructure failures from the study are worth reporting because practitioners will hit them. First, the constrained-decoding arm did not behave as specified: with a full CPE-JSON regex passed to vLLM's `guided_regex`, with-CPE rates for guided versus unguided runs stayed close across all seventeen served models, and SecurityLLM continued emitting analyst-narrative prose for 97% of invocations — impossible under an enforced full-match constraint, whose first token would have to be '{'. The behavior is consistent with the server treating the pattern as match-anywhere or silently dropping the parameter for some architectures. Constrained decoding is only a remedy for malformed output if the serving stack actually enforces it, and ours demonstrably did not; we therefore report unguided results and flag the verification gap. Second, two Ollama Cloud models (`kimi-k2.6`, `qwen3.5`) returned empty bodies for over 70% of invocations while sibling models on the same infrastructure responded normally. Diagnostic re-invocation surfaced the cause: these are reasoning models that consume their token budget on an internal deliberation trace stored in a separate response field, leaving the visible content empty. Batch pipelines targeting reasoning models need to disable or budget for internal thinking explicitly.

B. Accuracy, Cost, Privacy

The accuracy gradient favors frontier deployment when accuracy is the only axis: $+0.147$ tier-mean, half the hallucination. The other axes complicate the choice. Frontier API pricing at the time of the study placed the flagship models at \$15–75 per million input tokens; the Ollama Cloud tier charges a small fraction of that and reaches mid-frontier accuracy at its top end (`glm-5.1` at 0.395), making it the most economically lopsided comparison in the matrix; locally hosted inference has zero marginal cost once the hardware exists. Privacy is where local wins outright: no trust boundary extends at inference time. Hosted enterprise tiers now offer training-data exclusion, regional processing, and audit logging that approach the compliance requirements of Section III — mitigating the trust extension without eliminating it.

C. Limitations

The testbed was deliberately biased toward hard cases — white-label cameras and extenders, end-of-life hubs, minimal-surface bulbs and plugs — because general-purpose computers produce scan output rich enough that the deterministic database already identifies them, which would inflate aggregate accuracy and obscure differentiation. The numbers reported here are therefore a measurement of LLM reconnaissance accuracy on the devices conventional fingerprinting struggles with most, not a general estimate. Ground-truth labeling and the accepted-CPE tier assignments are the first author's judgment; expanding the accepted lists during the study improved fairness but introduced calls a second rater might dispute. The prompts are likewise the authors' constructions, informed by prior literature on persona effects but carrying our structure and voice. Conversational client tools (the chat interfaces shipped with the frontier models) were considered as an inference path and rejected — routing layers can substitute model variants, and conversational memory would contaminate predictions across devices — so the experiment does not measure the tool-assisted reasoning an operator might actually use. Finally, the 16-device corpus is small; per-device variance is visible in the results, and the family-level rankings should be read with that in mind.

VII. Conclusion

Which input variable moves CPE-identification accuracy most? Model family, by a clear margin: a 0.28 spread, wider than prompt design (0.10) or scan suite (0.23) produces, with scale tracking family at the bottom of the table. Prompt design matters less and specifically: format structure adds 0.051 where persona framing adds 0.009, and instruction-doubling is null in aggregate while helping precisely the small models the lost-in-the-middle hypothesis predicts it should.

What does keeping inference local cost? The gap is real but survivable: 0.147 on tier means, 0.108 best-to-best, with hallucination roughly doubling. An 8B security fine-tune on a single 24 GB card reaches 0.332, within reach of the weakest frontier model — but the bottom of the local tier scores below the Nmap-native baseline, so model choice *within* the local tier matters more than the local-versus-frontier decision itself. A defender who must keep reconnaissance data in-house can do useful work with a well-chosen 8B model; one who grabs whatever open-weight model is at hand can do worse than the deterministic tool already installed.

And the quietly wrong answers: 21.9% of predictions were well-formed, confident, and wrong, scaling inversely with model size and climbing from 14.5% at the frontier to 26.1% locally. This failure mode is silent by construction — it parses, it retrieves a plausible CVE list, and nothing tells the analyst the list belongs to a different product. Any deployment of LLM-assisted reconnaissance needs validation downstream of generation, and the smaller the model, the more it needs it.

Three directions follow. Quantization was excluded to control the variable count; adding 4-bit and 3-bit arms would test whether the size-accuracy gradient survives the precision reduction a true single-GPU operator faces. The measurement methodology extends naturally to other reconnaissance subtasks (passive intelligence, exploit-availability lookup, target prioritization) that share the ambiguous-input-to-structured-output shape. And the security fine-tunes' inconsistency — strong vendor recognition paired with elevated product invention — motivates a fine-tune trained directly against a CPE-identification objective; the 96,162 scored runs assembled here are suitable as training and evaluation corpus for exactly that, and we intend to release the dataset and pipeline alongside this paper.

References

[1] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "Technical guide to information security testing and assessment," NIST, Gaithersburg, MD, Tech. Rep. SP 800-115, Sep. 2008.

[2] G. Deng *et al.*, "PentestGPT: An LLM-empowered automatic penetration testing tool," 2024.
[3] B. Wu *et al.*, "AutoPT: How far are we from the end2end automated web penetration testing?" 2024.
[4] D. Goyal, S. Subramanian, A. Peela, and N. P. Shetty, "Hacking, the lazy way: LLM augmented pentesting," 2024.
[5] W. Jansen and T. Grance, "Guidelines on security and privacy in public cloud computing," NIST, Gaithersburg, MD, Tech. Rep. SP 800-144, Dec. 2011.
[6] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST, Gaithersburg, MD, Tech. Rep. SP 800-207, Aug. 2020.
[7] S. Liu, D. Cheng, T. Xu, Z. Gou *et al.*, "Quantization meets reasoning: Exploring LLM low-bit quantization degradation for mathematical reasoning," 2025.
[8] E. Kurtic, A. Marques, S. Pandit, M. Kurtz, and D. Alistarh, "Give me BF16 or give me death? Accuracy-performance trade-offs in LLM quantization," Nov. 2024.
[9] M. Miettinen *et al.*, "IoT Sentinel: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE ICDCS*, 2017, pp. 2177–2184.
[10] A. Sivanathan *et al.*, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, 2019.
[11] G. Lyon, "Nmap reference guide: Service and version detection," 2024. [Online]. Available: https://nmap.org/book/man-version-detection.html
[12] A. Sabetta and M. Bezzi, "Software vulnerability analysis using CPE and CVE," 2017.
[13] National Institute of Standards and Technology, "NVD developer documentation: Products (CPE)," 2024. [Online]. Available: https://nvd.nist.gov/developers/products
[14] A. Ramanujam, P. Soni, and B. Krishnamurthy, "CPE-Identifier: Automated CPE identification and CVE summaries annotation with deep learning and NLP," 2024.
[15] X. Shen *et al.*, "PentestAgent: Incorporating LLM agents to automated penetration testing," in *Proc. AsiaCCS*, 2025.
[16] P. D. Luong *et al.*, "xOffense: An AI-driven autonomous penetration testing framework with offensive knowledge-enhanced LLMs and multi-agent systems," 2025.
[17] L. Gioacchini *et al.*, "AutoPenBench: Benchmarking generative agents for penetration testing," 2024.
[18] E. Wåreus and M. Hell, "Automated CPE labeling of CVE summaries with machine learning," Jul. 2020.
[19] N. F. Liu *et al.*, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, 2024.
[20] MITRE Corporation, "Common platform enumeration: Naming specification version 2.3," MITRE, Tech. Rep., Nov. 2011.
[21] Z. R. Tam *et al.*, "Let me speak freely? A study on the impact of format restrictions on performance of large language models," 2024.