EASTERN WASHINGTON UNIVERSITY

Evaluating LLMs for CPE Identification in IoT Reconnaissance

Can local models map messy IoT scans to useful CPEs without sending recon data to the cloud?

Christopher Davisson
Computer Science & Cyber Defense
June 8, 2026

Roadmap

1. Motivation
2. My Work
3. Results
4. Conclusion

Part 1: Motivation

Why mapping scans to CPEs is hard, and why it matters

The Misfire

I scanned my home network with Nmap. The Fire TV on it came back classified as an Android phone, which made me wonder if someone had gotten onto my network. A neighbor, maybe. Until I looked closer.

Several ports were open:

- Port 7000: AirPlay-style RTSP
- Port 8009: Google Cast endpoint
- Port 9080: Netflix NRDP header

. . . and the MAC OUI belonged to an Amazon-contracted electronics maker.

Any one of these means little. **Together they point to a smart TV running Android.**

From Scan to Vulnerability

Reconnaissance runs on a chain of identifiers:

Nmap scan → product ID → CPE → CVE lookup → triage

- **CPE** (Common Platform Enumeration) is a structured name for a product: `cpe:2.3:o:amazon:fire_os:*`
- **CVE** (Common Vulnerabilities and Exposures) is a known flaw, looked up *by* CPE

The CPE is the bridge. Get it wrong and every step after it is wrong too.

How It's Done Today

Nmap already extracts CPEs by matching service banners against a fixed signature database.

- When a banner matches a known signature, Nmap emits a CPE. Precise and fast.
- When it doesn't match, Nmap says nothing at all.

That works on servers and mainstream software. It is **deterministic**: same input, same output, no guessing. The question is what happens when the input is messy.

Why a Wrong CPE Costs You

A misidentified device doesn't just produce a wrong label; it poisons everything downstream.

- Wrong CPE → wrong CVE list → wrong risk picture
- You scope compensating controls to a threat that isn't there…
- …and miss the one that is.

A malformed CPE fails loudly. A wrong-but-valid CPE fails silently. It hands you a believable answer for the wrong product.

Why IoT Breaks Deterministic Matching

Embedded devices give a signature database very little to grab onto:

- Minimal surface: often one HTTP endpoint, sometimes nothing
- Truncated or generic banners: `lighttpd` with no version, mbedTLS instead of OpenSSL
- Non-standard stacks: FreeRTOS, lwIP; no match in Nmap's OS database

A human reads *across* the weak signals. So can an LLM, but probabilistically. **A deterministic tool stays silent when unsure; a model guesses.** That trade is the whole study.

Prior Work: AI in Pentesting

A fast-growing body of work puts LLMs into offensive security:

- **PentestGPT**: walks a human tester through an engagement
- **Autonomous agents**: LLMs that plan and execute multi-step attacks on their own
- **Advisory NLP**: pulling product names and CPEs from written vulnerability text

Each is judged end-to-end: did the *agent* succeed at the *whole* task?

Prior Work: Fingerprinting & CPE

Two more strands sit close to this problem:

- **IoT device fingerprinting**: classifies device *type* or vendor from traffic, but stops short of a CPE 2.3 string
- **Automated CPE identification**: works from CVE / advisory *text*, not raw scan output

The pieces exist. But nobody has measured them on **raw IoT scan evidence, producing the CPE an NVD lookup actually needs**.

The Gap

Most LLM-pentest work throws a whole agent at the wall and measures whether the wall falls.

This thesis isolates one step

Given Nmap evidence for an IoT device, how well can a model produce a *useful* CPE on its own, not buried inside an agent?

Component-level, not end-to-end. That's the contribution.

Why Recon, Why IoT, Why Local

- **Why recon**: it's the first step; an error here compounds through the whole engagement
- **Why IoT**: it's where deterministic tools struggle most, so there's the most room to help
- **Why local**: a scan is a map of the target network. Sending it to a cloud model widens the trust boundary, hard to defend under HIPAA, CMMC, or financial regs

If local models can do this, a defender keeps the data and still gets the help.

Part 2: My Work

The pipeline, the models, and how everything was scored

Research Questions

RQ1: What drives accuracy?

Across tier, family, scale, prompt, and scan suite, which moves the needle most?

RQ2: What does staying local cost?

Do strong local models clear the Nmap baseline and approach the weakest frontier model?

RQ3: How dangerous are the failures?

How often do models emit plausible-but-wrong CPEs, and how does that scale with size?

Devices & Scans

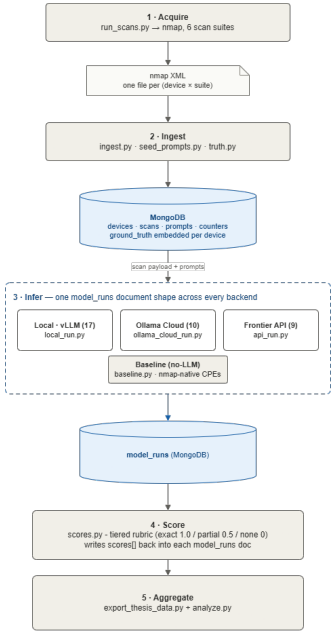
The testbed (18 entries)

- 14 physical IoT devices + 4 derived
- 16 analyzed (2 dropped: unusable scan/run data)
- TVs, streaming sticks, routers, gateways, printers, plugs, bulbs, a pentest box

Six Nmap suites

- TCP top-1000
- service-version intensity 1 / 5 / 9
- UDP discovery: DNS-SD, UPnP

Chosen to be **hard**: laptops scan cleanly and would only inflate the scores.



The Pipeline

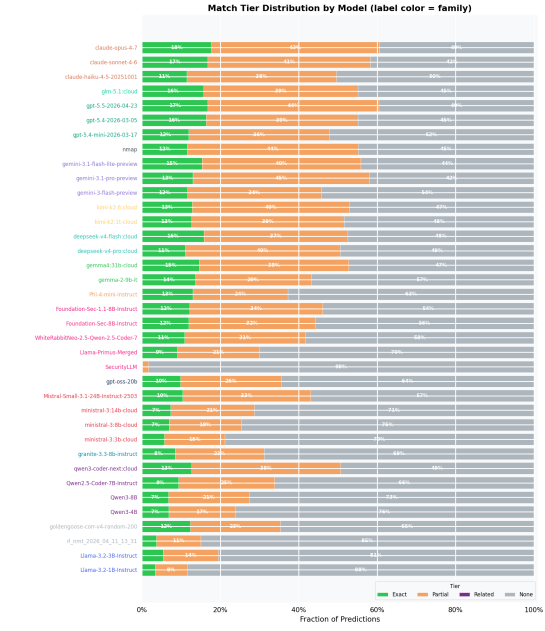
- Five stages: Acquire → Ingest → Infer → Score → Aggregate
- Every backend writes the *same* record, so scoring is uniform
- Frozen export pins every number; re-scoring is seconds, not days

Scoring Rubric

Graded credit, because a broader-but-right CPE still helps:

Tier	Weight	Operational meaning
Exact	1.00	Accepted exact identifier
Partial	0.50	Lower-specificity but still useful
None	0.00	No match, or nothing parseable

A fourth "related" tier (0.25, e.g. `linux_kernel` for the Fire TV) exists in the code but is vanishingly rare; the live scale is effectively three-tier.



Why the Tiers Matter

What a single mean score hides:

- Exact matches are *rare*
- Partial credit carries most of the useful signal
- Weak models mostly produce *no* usable CPE

Partial correctness isn't a consolation prize. A broader-but-right CPE still bounds the CVE search; a pass/fail score would erase that.

The Models

36 models across three deployment tiers, plus a no-LLM baseline:

- **Local vLLM (17)**: open weights on a 4-GPU workstation; Mistral, Qwen, Llama, Gemma, Phi, security fine-tunes
- **Ollama Cloud (10)**: hosted open models; GLM, DeepSeek, Kimi, Ministral
- **Frontier API (9)**: three each from OpenAI, Anthropic, and Google

Baseline: Nmap-native CPE extraction, the floor every model is measured against.

Why So Many Prompts?

A 2×2 over *persona* × *structure*, plus three probes (chain-of-thought, few-shot, casual).

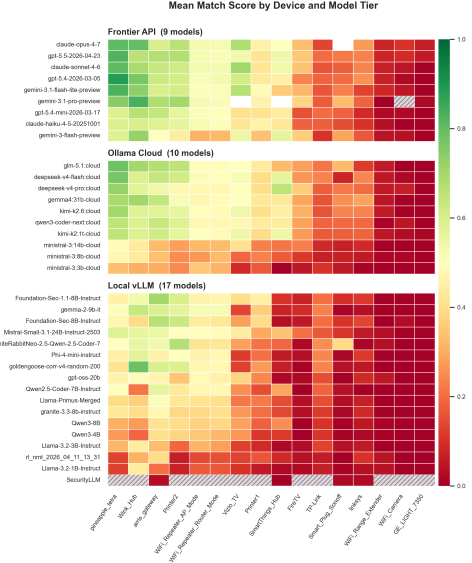
Why test instead of reason

Model output is path-dependent: the tokens before and after the scan shift what the model can recall. A clue can sit behind a wall of probability you can't cross by thinking, only by testing.

So I swept prompts empirically, and tested **placement** directly with a doubled-prompt variant (the "lost in the middle" effect).

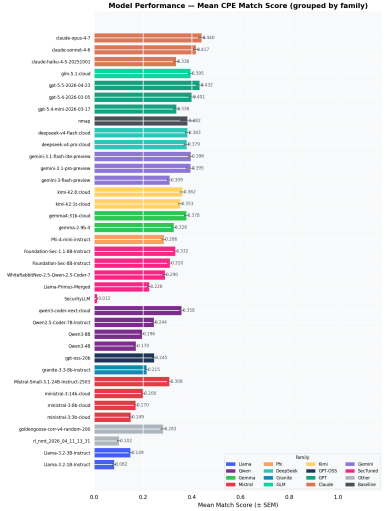
Part 3: Results

What drove accuracy, and what failed



The model matters, but the scan matters too: some devices are hard for every tier.

RQ1: Family Is the Biggest Lever



Top of the chart: Claude Opus 0.440, GPT-5.5 0.432, Claude Sonnet 0.417. Frontier families lead.

Best local lands lower: Foundation-Sec-1.1-8B 0.332, Gemma-2-9B 0.326.

The family gap dwarfs any prompt effect. Structure helps (+0.051), persona barely (+0.009), and heavier scans don't help.

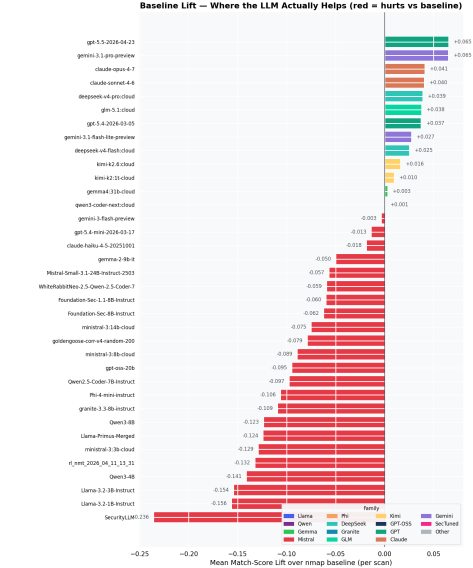
RQ2: What Staying Local Costs

Mean match falls and hallucination climbs as you move off the frontier:

Deployment tier	Mean match	Hallucination
Frontier API	0.378	0.145
Ollama Cloud	0.306	0.193
Local vLLM	0.231	0.261

Good news: the best local model (Foundation-Sec-1.1-8B, 0.332) lands near the weakest frontier model, on a single 24 GB card.

The catch: local buys data control, not frontier accuracy, and hallucinates about twice as often.



Nmap wins when it speaks; the LLM earns its place where Nmap is silent.

RQ3: The Silent Failure

- **21.9%** of predictions: clean, parseable CPEs with *no* accepted match
- Scales inverse with size: the 1B Llama hallucinates 43%; the frontier tier about 14%

The trap

SecurityLLM shows the *lowest* hallucination rate (1.7%), but it answers on only about 3% of runs. That's abstention, not reliability.

Why Some Output Looks Odd

- **Reasoning models that hurt themselves**: a few spent the whole token budget "thinking" and returned empty bodies
- **Low hallucination $\neq$ good**: the safest-looking model was just refusing to answer
- **Intensity-9 scans**: more probing added noise, not signal, and raised guessing

The rubric and the abstention tracking are what keep these from being read the wrong way.

Part 4: Conclusion

What it means, what it doesn't, and what's next

What a Defender Should Do

If the priority is...	Reach for...
Maximum accuracy	A frontier API model
Lower cost, hosted	A strong Ollama Cloud model
No data egress	A security-tuned local 8B
Safety-critical output	Any of the above + human review

Cost spans an order of magnitude per step: frontier dollars-per-million-tokens down to zero-per-call once the hardware is bought.

Limitations

- **Hard, IoT-only dataset**: worst-case numbers, not general accuracy
- **Ground truth is judgment**: the tier labels are mine
- **No CVE-lookup stage**: I scored CPE production, not the downstream NVD result
- **Constrained decoding misbehaved**: the guided-regex arm didn't bind; primary results are unguided
- **A workstation, not one card**: the single-GPU case is harder than the local numbers suggest

Conclusion

- **Family first**: it moves accuracy more than anything you tune
- **Local is survivable**: a security-tuned 8B lands near the weakest frontier model, on one card
- **Mind the silent miss**: 21.9% confident-but-wrong, worse as models shrink

The answer

Local LLMs can do useful CPE work under privacy constraints, but they are not a drop-in replacement, and the weakest ones do worse than the tool you already have.

Future Work

- **Quantization**: 4-bit and 3-bit, the case a single-GPU operator actually faces
- **A CPE-objective fine-tune**: train on the 96k-run corpus directly
- **The wider recon pipeline**: extend the tiered scoring to other subtasks
- **Release the dataset**: lower the bar for follow-up work

Thank you

Questions?

Backup

Reference slides

Backup: Full Model Roster

Local vLLM (17): Mistral-Small-3.1-24B, Qwen3-8B / 4B, Qwen2.5-Coder-7B, Llama-3.2-3B / 1B, gemma-2-9b, Phi-4-mini, granite-3.3-8b, gpt-oss-20b, Foundation-Sec-1.1-8B / 8B, SecurityLLM, WhiteRabbitNeo-Qwen-7B, Llama-Primus, goldengoose / rl_nmt

Ollama Cloud (10): glm-5.1, deepseek-v4-flash / pro, kimi-k2.6 / k2:1t, gemma4:31b, qwen3-coder-next, ministral-3:3b / 8b / 14b

Frontier API (9): gpt-5.5, gpt-5.4 / 5.4-mini, claude-opus-4-7, claude-sonnet-4-6, claude-haiku-4-5, gemini-3.1-pro, gemini-3.1-flash-lite, gemini-3-flash

Backup: Fire TV Ground Truth

The accepted CPEs for the Fire TV, by tier:

Tier	CPE
Exact	`cpe:2.3:o:amazon:fire_os:*:*:*:*:*:*:*:*`
Partial	`cpe:2.3:o:google:android:*:*:*:*:*:*:*:*`
Related	`cpe:2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:*`

Exact and partial map to operationally different lookup scopes: Fire OS versus Android isn't the same miss as a random product. Expanding the accepted set makes scoring fairer but adds researcher judgment; the labels are versioned.

Backup: Why Constrained Decoding Failed

- Trials 2–4 passed vLLM a guided regex for the CPE 2.3 structure
- With-CPE rates were nearly identical guided vs unguided, so the constraint wasn't binding
- SecurityLLM emitted prose 97% of the time even under it, likely match-anywhere or a silently dropped parameter

Backup: Why the CVE-Lookup Stage Was Cut

- Planned third stage: query the NVD with each predicted CPE
- It measured database behavior, not model capability
- Once a CPE is right enough, the lookup is mechanical; a cleaner endpoint is CPE production itself

Backup: The Ollama Empty-Body Issue

- kimi-k2.6 and qwen3.5 returned empty bodies on more than 70% of calls
- A trivial prompt answered correctly in a separate “thinking” field: reasoning models spending the budget before any visible output
- qwen3.5 was excluded from the scored matrix; documented, not silently dropped