Vulnray - X-ray for code vulnerabilities

Local AI scanning for foundational and low-level code

- Focused on C code: Bitcoin Core, hardware wallet firmware
- Runs locally with `VulnLLM-R-7B` reasoning model
- Takes under `~6 GB` VRAM/shared memory
- Allows long-running repo-scale autonomous scans
- NOT burning tons of tokens
- **NOT disclosing your findings to AI providers**

The Problem

- Security review is slow, expensive, and hard to parallelize across large repos
- Cloud LLM scanning is brittle for security work: privacy, cost, rate limits, and censorship risk
- Low-level C/C++ codebases hide subtle bugs: memory safety, integer bounds, crypto misuse

Principles: what "Good" Looks Like

- Local-first CLI (automation, scripting, background execution)
- High-recall scanning (overnight / multi-day acceptable)
- Modes: `max-recall`, `balanced`, `deterministic` (CI-friendly, temp=0)
- Model-agnostic GGUF swap, `llama.cpp` backend (Metal on Apple Silicon)
- Outputs in one run: `json`, `csv`, `md` (with executive summary + findings table)

How Vulnray Works

End-to-end pipeline

`CLI + config` -> `file discovery` -> `chunking (function/sliding)` -> `prompt profile + focus` ->
`llama.cpp inference (VulnLLM-R-7B GGUF)` -> `parse findings` -> `reports (JSON/CSV/MD)`

Per-chunk contract

- Prompts request **JSON-only** outputs (machine-parseable findings + reasoning)

Config precedence

`CLI > ENV > TOML > defaults`

Running It (Tutorial Flow)

```
python -m pip install -e .
vulnray tutorial/test_project --config tutorial/vulnray.toml
```

Produces:

- `tutorial/reports/demo_scan.json`
- `tutorial/reports/demo_scan.csv`
- `tutorial/reports/demo_scan.md`
- `tutorial/reports/demo_scan.prompt_output.md` (prompt + model output log)

What You Get (Reports)

Markdown report structure

- Executive summary (findings by severity)
- Findings table (triage-friendly)
- Per-finding detail: file, line range, function, CWE, severity, confidence, recommendation

JSON report structure (scan metadata + findings)

- `scan_metadata`: tool, model, mode, timestamp, repo_root, files scanned, chunks analyzed
- `findings[]`: `{id,file,start_line,end_line,function,vulnerability_type,severity,confidence,description,reasoning,recommendation,references}`

Demo Scan Results (tutorial/test_project)

From `tutorial/reports/demo_scan.md` (mode: `balanced`):

- **3 high-severity findings**
- `CWE-787` out-of-bounds write risk (`strcpy` into fixed-size buffer)
- `CWE-787` overflow risk in path construction (`sprintf` into fixed-size buffer)
- `CWE-190` integer overflow risk (unchecked multiplication)

Why VulnLLM-R (arXiv:2512.07533v1)

Model claim (paper)

- "First specialized reasoning LLM" for vulnerability detection
- 7B-parameter reasoning model trained via a custom recipe (data selection, teacher-generated reasoning, filtering/correction, efficiency tuning)
- Teacher models (paper): `DeepSeek-R1` and `QwQ-32B`
- Agent scaffold + context retrieval; paper reports outperforming traditional tools (e.g., CodeQL, AFL++) in real-world projects and discovering **15** zero-days

Credit (authors)

Yuzhou Nie, Hongwei Li, Chengquan Guo, Ruizhe Jiang, Zhun Wang, Bo Li, Dawn Song, Wenbo Guo.