Agents for Security Review

Agentic Security — Module 5

The Setup

Where We Are in the Arc

- Yesterday: you built ShareBox; overnight it froze
- This morning: you broke the summarizer agent (M4)
- **This lab**: audit an instructor-built ShareBox — same spec, deliberately vulnerable — with `large`
- Then **prove** each finding by exploiting the hosted instance: CTF flags, harder vulns worth more

One rule: **a finding only counts once you exploit it.** You're writing your own attack plan, not a report for a reader.

(That's this class's CTF rule — good discipline; real-world reporting rightly also publishes credible-but-unproven findings, labeled with confidence.)

Agents as Auditors

What They're Genuinely Good At

- Reading an entire codebase fast — no fatigue, no skimming
- Tracing data flow across files ("where does this filename end up?")
- Recognizing vuln **patterns**: they've read every writeup ever published
- Variant analysis: "found one path traversal — now find every similar sink"

An agent audit is a tireless junior reviewer with encyclopedic pattern knowledge. That's worth a lot — if you manage the failure modes.

The Failure Modes

- **Hallucinated vulns**: cites code and flows that aren't there
- **Severity inflation**: everything is CRITICAL; nothing is prioritized
- **Duplicates in a trench coat**: one root cause, five findings
- **Plausible-but-wrong**: names a real pattern, misses the guard clause five lines up
- **Blind spots**: business logic, authz-by-convention, anything requiring intent

The output **looks** professional either way; that tells you nothing.

Running the Audit

Two Prompting Strategies

You'll run both in Lab 5.1, in separate sessions:

Open-ended: "Find security vulnerabilities in this codebase."

- Surfaces what the model finds salient; cheap; unfocused

Checklist-driven: walk the vuln classes one at a time.

- "Check every file operation for path traversal." Then the next class.
- Focused attention per class; catches what open-ended skims past

They won't return the same list — **the model's attention goes where you point it, and unsteered attention misses things.**

Recon Before Agent

Ten minutes of your own eyes first, before any prompting:

- The stack, the auth mechanism, the routing
- Where files and sessions live; anything that smells

Why: you can't triage findings about code you've never looked at. The agent's report will be confident and voluminous — your baseline is what keeps it honest.

Triage: The Human Half

For every candidate finding, before it goes in your report:

1. Make the agent show the **exact** vulnerable line
2. Trace the path yourself: entry point → sink, is there really no guard?
3. Write the exploitation hypothesis: "POST **this** to **this** endpoint, get **this**"

If you can't complete step 3, you have a vibe, not a finding. Cut it.

Every hypothesis you write is a payload you get to try against the hosted instance — sloppy triage just wastes your own attack time.

Reviewing AI-Written Code

The target was built the way you built yours yesterday: fast, by a small model, against a happy-path functional suite.

- Expect the M3 catalog: missing authz, string-built queries, naive path handling, verbose errors
- Expect **systematic** bugs: the model that wrote one IDOR wrote the pattern everywhere — variant analysis pays off
- Don't expect subtle logic flaws; expect absent checks

Whatever model built it has habits — go after those, not just this one codebase.

The Bigger Picture

Agents in Offensive Workflows

Where this fits when you're back at your desk:

- **Recon**: unfamiliar codebase → map in minutes (Lab 2.2)
- **Audit**: this module
- **Exploitation**: payload iteration with full source context (this lab)
- **Writeups**: findings → drafts via your Lab 2.3 skill — verify before your name goes on it

The pattern everywhere: the agent drafts; **the human owns verification and judgment** — M3's accountability rule, applied to offense.

Ground Truth Is Coming

Every audit claim gets tested against the live instance before the lab ends.

- Reported and exploited → real finding, flag captured
- Reported, unexploitable → hallucination or bad triage — **yours to explain**
- Unreported, flagged by someone else → the model's blind spot, or yours

Almost no review gets this feedback; you get it today. Your reported-vs-confirmed ratio is the most honest metric of AI review you'll collect.

Wrap-Up

What You Should Take Away

- Agent auditors: tireless, pattern-rich, and confidently wrong at a steady rate
- Run open-ended **and** checklist passes; the diff is signal
- Recon first; triage everything; no hypothesis, no finding
- AI-written code fails systematically — hunt the generator's habits
- A finding is a draft until it's verified — and today, verifying means exploiting it

Lab 5.1: Audit the Vulnbox