

Answers

Reference answers for Lesson 04.

Part 1 — Lab Answers

Exercise 1 — Python Web App Audit

Expected classes of findings

- SQL injection via string concatenation into a query.
- Command injection via `os.system` / `subprocess shell=True` with user input.
- Path traversal on a file-read endpoint.
- Hard-coded secret or SECRET_KEY.
- Possibly: insecure deserialization, open redirect, or reflected XSS.

Exercise 2 — Node.js API Audit

Expected findings

- JWT verification with `algorithm: 'none'` accepted or signature skipped.
- IDOR: user-controlled ID used directly in a database lookup with no ownership check.
- NoSQL injection via object in request body passed straight to Mongo.
- Mass assignment: `Object.assign(user, req.body)` enables privilege escalation.

Exercise 3 — Legacy C Auth Audit

Expected findings

- Stack buffer overflow via `strcpy` / `gets` / `sprintf` on fixed-size buffer.
- Timing attack on `strcmp` or `memcmp` password check.
- Weak RNG (`rand()`) used for token generation.
- Integer truncation in a size calculation.

At least one finding should be exploitability-rated "likely RCE" given sufficient size controls.

Exercise 4 — Config Leak Audit

Expected findings

- Database password or API key in plaintext.
- `debug: true` or equivalent flag left on for production.
- CORS wildcard or TLS verification disabled.
- Default admin account or weak default credential.
- Internal hostname or private IP referenced directly.

Final Exercise — Audit Report

Expected cross-file chain

A secret found in `config-leak.yaml` is used as a signing key or DB password in `vulnerable-webapp.py`, which itself has an SQL injection that could exfiltrate more data. A great audit connects at least two files.

Part 2 — Homework Answers

Task 1 — False-Positive Calibration

Expected precision

For typical lab content AI precision lands around 70-85%. The most common FP pattern is flagging *any* use of a sensitive function (like `eval()`) as a bug, even when the input is a trusted constant. Misses usually involve business-logic flaws with no syntactic fingerprint.

Task 2 — Patch & Verify

Expected patch quality

Root-cause fixes replace the dangerous primitive (parameterized queries instead of string concat, `shlex.quote` instead of raw input). Symptom fixes (regex blacklist, escaping on output) are a sign the audit was not internalized.

Task 3 — Semgrep Rule Authoring

Expected rule shape

A YAML file with `pattern-either` or `pattern` clauses matching the dangerous call site, plus a `metavariable-pattern` filter to cut false positives when the argument is a string literal.