

Lab Guide

Four exercises: vuln scanner triage, SBOM review, dependency audit, and patch inventory prioritization — turn CVE noise into a weekly patch plan.

Authorized Use Only

All scans and inventories are **synthetic**. Apply patches only after staging validation.

Prerequisites

- Completed the [Theory](#) reading
- Access to an AI assistant (Claude, ChatGPT, or local LLM)
- Downloaded all four files into a folder named `vuln-lab/`
- Familiarity with: CVE / CVSS, SBOM formats (CycloneDX, SPDX), semver

Lab Scenario

AcmeCorp's scanner just dumped 2,000 findings. Leadership wants a single list of "what to patch this week" without breaking the apps. You have the raw scan, the SBOM of the flagship product, a dependency audit from CI, and a patch-inventory showing what is already deployed. Your job: produce a prioritized patch plan.

Download Lab Files

Save all 4 files into a folder named `vuln-lab/` before starting the exercises.

↓ [dependency-audit.txt](#) ~10 KB

↓ [patch-inventory.txt](#) ~2 KB

↓ [sbom-sample.txt](#) ~13 KB

↓ [vuln-scan-results.txt](#) ~16 KB

Exercise 1 — Vuln Scanner Triage

Open [vuln-scan-results.txt](#). Raw scanner output overwhelms humans — rank it.

Task 1.1 — Rank by real risk

Below is vulnerability scanner output. Rank by REAL risk, not raw CVSS:

1. Drop informational / false-positive categories.
2. For each CVE, compute a risk score combining:
 - CVSS base
 - Exploitability (public PoC? Metasploit module? CISA KEV?)
 - Asset exposure (internet-facing? internal? offline?)
 - Asset criticality (prod db vs. dev VM)
3. Group by host and by patch (one patch often fixes many CVEs).
4. Flag anything on the CISA Known Exploited Vulnerabilities list.
5. Produce the top 10 patches ranked by (reachable CVEs x criticality).

Output: a "patch this first" top-10 with rationale.

[paste vuln-scan-results.txt]

✓ Self-check: Your top item should have a specific reason ("internet facing, KEV, public PoC") not just a high CVSS.

Exercise 2 — SBOM Review

Open [sbom-sample.txt](#). A Software Bill of Materials maps what you ship.

Task 2.1 — Cross-reference with CVE

Below is a CycloneDX-style SBOM. Analyze:

1. List direct vs. transitive dependencies.
2. Identify packages with known CVEs (pin to specific versions).
3. Find abandoned / unmaintained packages (no release in 2+ years).
4. Flag license issues (GPL in a closed-source product).
5. Identify duplicate packages at multiple versions.
6. Note any package known to be typosquatted (lookalike names).
7. Suggest fixes: upgrade / replace / remove / vendor.

Output:

- CVE-impacted packages with fix versions
- "Zombie" packages to replace
- A CI gate that would have blocked new findings

[paste sbom-sample.txt]

✓ **Self-check: Every fix should include the target version, not just "upgrade".**

Exercise 3 — Dependency Audit

Open `dependency-audit.txt`. Runtime dependencies are where exploited vulns actually live.

Task 3.1 — Bucket by upgradability

Below is the output of `npm audit` / `pip-audit` / `dotnet list package --vulnerable`. For each finding:

1. Is the upgrade semver-major, minor, or patch?
2. What breaks with the upgrade (based on release notes you know about)?
3. Is there a workaround (config flag, avoid vulnerable function)?
4. Is it in the runtime path, or only in dev/test deps?
5. Rank: patch now / patch this sprint / risk-accept.

Output: a per-package decision matrix.

[paste dependency-audit.txt]

✓ Self-check: You should mark at least one finding as "risk accept" with a clear justification.

Exercise 4 — Patch Inventory

Open `patch-inventory.txt`. You cannot patch what you do not know about.

Task 4.1 — Find the coverage gap

Below is a patch-management inventory (hosts, OS, last patch date, missing KBs). Identify:

1. Hosts that have not been patched in >30 days.
2. Missing KBs that match known-exploited CVEs.
3. End-of-life OS versions (Win7, Win Server 2012, older Ubuntu).
4. Hosts reporting success but missing critical patches.
5. Clusters where patch state is consistent vs. drifted.

Output:

- A coverage heatmap (by OS, by criticality)
- A remediation sequence that minimizes downtime
- A KPI for next month (target patch compliance %)

[paste patch-inventory.txt]

✓ Self-check: You should flag every EoL host and name the replacement target.

Final Exercise — Weekly Patch Plan

Task 5.1 — The one-page plan

Combine the four analyses into a 1-page "next week's patches" plan containing:

1. Top 10 patches, in order
2. Owner per patch
3. Change window per patch
4. Rollback method per patch
5. Verification query per patch (a Splunk/OSQuery one-liner)
6. Risk-accepted items with expiry dates

Format as markdown.

✓ **Self-check: Hand this to an ops engineer — can they execute without asking you questions?**

What's Next

Lesson 08 pivots from patching to **incident response** — alerts, IOCs, affected systems, and comms.