Progress Milestone — Cybersecurity & DevOps Foundations

This document records the milestone progress and provides actionable, hands-on mini-project workflows to continue developing cybersecurity and DevOps skills. It assumes you have a Kali VM available and a working macOS development environment with Docker and Git already set up.

1) Skills Achieved
- Git version control: local commits, remote linking, branch tracking.
- Tailscale: private mesh networking and endpoint verification.
- Docker: containerized sandboxes for reproducible labs.
- Documentation: structured notes repo with lesson PDF records.

2) Why this matters
- Versioning provides an audit trail and rollback capability.
- Private networking avoids exposing services to the public internet.
- Containers isolate vulnerable services for safe experimentation.
- Documentation makes learning repeatable and sharable.

3) Starter Workflows (summaries)
A. Recon & Documentation Loop
- Launch Kali VM and target a local Docker container.
- Run nmap and capture output.
- Save findings in daily_logs/ctf_recon.md and commit.

B. Docker Sandbox CTF
- Pull and run an intentionally vulnerable web app container.
- Practice basic web vulnerabilities (SQLi, auth bypass).
- Document exploit steps in repo.

C. Mac Hardening & Audit
- Run baseline checks (firewall, SSH settings, installed packages).
- Save audit results and recommended remediations.

D. CTF Practice Loop
- Use OverTheWire, TryHackMe or VulnHub.
- Document steps, tools, and lessons learned.

4) Project Organization Suggestions
- /security
  - /labs
  - /writeups
  - /scripts
- Use commit tags such as sec-lab:, ctf:, infra:, docs:

5) Daily Routine for Labs
- Snapshot VM before testing.
- Run enumeration and tools.
- Record commands & outputs.
- Commit logs and push to GitHub.

6) Safety & Ethics
- Only test systems you own or are authorized to test.
- Use isolated networks and snapshots.
- Do not develop or deploy malware; focus on learning.

--- EXTRA LESSONS (DETAILED)
(Note: These lessons are included here for your reference. They are compact, actionable, and intended to be followed inside an isolated lab environment.)

Lesson 1: Automated Recon Script
- Purpose: build a reusable bash script that runs nmap, gobuster (optional), and saves outputs to timestamped
- Steps:
  1) Create scripts/recon.sh
  2) Make executable: chmod +x scripts/recon.sh
  3) Usage example:
     ./scripts/recon.sh 192.168.56.101
  4) Commit outputs to /security/labs/<target>/ and write a short writeup.

Lesson 2: Docker Challenge Pack
- Purpose: containerize multiple small vulnerable apps and orchestrate them with docker-compose.
- Components:

  - DVWA, Juice-Shop, and a small custom Flask app with an intentional bug.
- Steps:
  1) Create docker-compose.yml to run the stack.
  2) Expose only on host for lab testing.
  3) Practice exploitation and document fixes.

Lesson 3: Post-Exploitation Audit (Safe)
- Purpose: learn how to capture forensic evidence after a simulated compromise of a lab VM.
- Steps:
  1) Snapshot VM before test.
  2) Collect 'ps aux', 'netstat -tunapl', '/var/log/auth.log', and 'last' outputs.
  3) Store artifacts in /security/forensics/<target> and document methodology.

Lesson 4: CI for Lab Reports
- Purpose: automatically build a simple site from markdown writeups using GitHub Actions.
- Steps:
  1) Create a workflow `.github/workflows/build-site.yml` that runs on push.
  2) Convert markdown to HTML with a static site generator (e.g. mkdocs).
  3) Deploy to GitHub Pages or keep as artifacts.

Lesson 5: Defensive Monitoring Starter
- Purpose: set up a lightweight host-based IDS on macOS or a Linux VM.
- Tools:
  - osquery, Wazuh (light demo), or simple logwatch scripts.
- Steps:
  1) Install osquery and collect baseline queries.
  2) Schedule daily checks and commit summaries.

End of extra lessons.