MODULE 6

Build a Safe Hacking Lab and Experiment Workflow

Module 6

Mapping of Module

1. A short legal and ethics checklist you can copy.
2. A hardware and host decision flow so you actually pick something.
3. Detailed virtualization and network isolation steps, with snapshot workflow.
4. Logging, capture, and evidence handling templates so your findings aren't vapor.
5. Reproducible experiment templates and three slow, safe practice exercises.
6. A cleanup and archival playbook so your lab does not haunt you later.

Module 6

1. Legal and ethics checklist

Short legal rulesheet you must follow every session:

- Only experiment on systems you own, control, or have explicit written permission to test.
- Stop immediately if someone unexpectedly connects, or if real user data appears in your lab.
- Keep all networked lab gear isolated from the wider internet unless you have a controlled gateway and monitoring.
- Never reuse real credentials, personal data, or production secrets in the lab.
- Treat all findings as sensitive. Share only with authorized parties via secure channels.
- Keep a dated log of every experiment with names, scope, and outcomes.

Module 6

1. Legal and ethics checklist

Here is a template of a permission note you can ask a supervisor to sign:

- Project: Lab Testing and Research
- Scope: Tests limited to isolated lab VMs and intentionally vulnerable lab images. No testing against production or third-party networks.
- Start date: _____
- End date: _____
- Authorized by: _____ (name, title, contact)
- Signed: _____ Date: _____

Stick that in a folder. If you plan to run anything with a real network connection, get a signed authorization even for the gateway.

Module 6

2. Decide your hardware and host OS

Ask yourself, one after another:

- Am I mostly learning concepts and tools? If yes, a decent laptop with 8-16 GB RAM works.
- Will I do heavier dynamic analysis or instrumented builds? If yes, aim for 32 GB RAM and an extra SSD.
- Do I want bare-metal performance or max isolation? If you want max repeatability and snapshots, virtualization is your friend. If you want real device testing, use separate hardware.

Module 6

2. Decide your hardware and host OS

Pros & Cons for Each Host:

- Host Windows: Easy for some GUI tools and Windows-target testing. More friction setting up Linux tooling.
- Host Linux: Best for security tooling, networking control, and scripting. Stable and composes well with containers.
- Host macOS: Fine for dev work. Less community tooling for low-level network tweaks unless you install Linux VMs.

It is highly recommended for most learners that you use a Linux or Windows host & run Linux VMs inside. Keep tooling updates seperate. Use a 2nd physical machine if you're testing OS-level kernel stuff.

Module 6

Virtualization setup and network isolation, step-by-step

Goal: *Create a reproducible environment that cannot accidentally reach the web or your home network.*

Concepts you need:

- **Host:** Your daily OS.
- **Attacker VM:** Where you run tooling - it's disposable.
- **Target VM:** Intentionally vulnerable or instrumented image - it's disposable
- **Monitoring VM:** Packet capture, logging, and analysis. Persistent snapshot and storage.

Module 6

3. Virtualization setup and network isolation, step-by-step

Step 1: Choose a hypervisor

- VirtualBox: free, cross-platform, GUI friendly. Good for labs.
- VMware Workstation Player: more polished on Windows. Free for non-commercial personal use.

Hyper-V: built into Windows Pro, fine but slightly more configuration.

Pick one. Stick with it for a while.

3. Virtualization setup and network isolation, step-by-step

Step 2: Create the VM network topology

- Use an internal-only virtual network for attacker and target. This keeps traffic inside the host.
- Add a separate interface on the monitoring VM that attaches to that internal network. That VM runs your packet capture tool continuously.
- If you ever need internet for updates in the attacker VM, use a temporary NAT network that you enable consciously, then snapshot and revert afterward.

Module 6

3. Virtualization setup and network isolation, step-by-step

Step 3: Build baseline VMs

- Make 3 VMs minimum: attacker, target, monitor.
- Choose minimal OS images for each. Keep resources light so you can run multiple.
- Snapshot each immediately after a clean install and a basic hardening checklist.

3. Virtualization setup and network isolation, step-by-step

Step 4: Snapshot discipline

- Before any experiment, create a fresh snapshot labeled with date, experiment name, and short note.
- Experiment only from that snapshot. Do not modify your baseline directly.
- After the experiment, either keep the snapshot for analysis or revert and delete it. If you keep it, export the VM or save the snapshot to external storage.

3. Virtualization setup and network isolation, step-by-step

Step 5: Example VirtualBox workflow (GUI approach)

- Create VM -> attach ISO -> install OS -> install guest additions -> shutdown -> take snapshot -> label snapshot.

- Clone or copy snapshot for each experiment. Do not run multiple experiments in the same snapshot.

Module 6

3. Virtualization setup and network isolation, step-by-step

Step 6: Avoiding accidental internet leaks

- Disable host-only to NAT sharing unless you explicitly need updates.
- If you need to fetch packages, put them on a host-shared folder or a controlled update VM that has an outbound gateway you can monitor.

Ideally, this should all be doable!

Module 6

4. Monitoring and traffic capture so you can make evidence that matters

You will regret not logging. Seriously.

Why capture traffic?

- Later on, you'll want to show what happened. Packet captures are gold.
- They let you reconstruct network flows and understand tool behavior.

Module 6

4. Monitoring and traffic capture so you can make evidence that matters

Monitoring VM setup, stepwise:

- Install a stable capture tool like tcpdump or tshark. GUI alternatives exist, but command line is more reproducible.
- Mount a large virtual disk or attach host folder for long-term pcaps.
- Configure rotation: keep recent files, compress old ones, and remove junk after archiving.

Module 6

4. Monitoring and traffic capture so you can make evidence that matters

Practical capture steps:
(When you start an experiment, start capture with a command that writes to a timestamped file)

Example conceptual form:

- Start: capture interface X to file experiment-name-yyyy-mm-dd.pcap
- Stop: gracefully stop capture, verify file size > 0
- Archive: move to analysis folder and checksum it.

Module 6

4. Monitoring and traffic capture so you can make evidence that matters

Why Do Checksums Matter?

If you plan to share evidence with a team, add md5 or sha256 sums to prove files were not tampered with.

Logging the Lab Notebook:
(Every experiment requires a short header at the top of the entry!)

- Date / time range
- Snapshot name and VM IDs
- Short goal statement
- Tools started and their versions
- Files produced (pcap names, logs, screenshots)

Keep one plain text file per experiment. Version it with git or store in encrypted cloud if needed.

Module 6

5. Tool installation hygiene and package management

Small mistakes here become huge problems later.

Principles:

- Keep attacker VM for tools only. Do not mix target or monitoring tooling there.
- Use package managers and verify checksums where possible
- Prefer packaged distributions from upstream or distro repos for basics. Build from source only when you understand dependencies.

5. Tool installation hygiene and package management

Safe Tool Workflow:

- Before installing a new tool, snapshot.
- Install. Test in isolation. If something misbehaves, revert snapshot.
- Track installed packages and versions in your experiment log.

Tip: Use containerization for quick experiments!

When possible, run disposable tooling in containers. Containers roll back fast and reduce host pollution. But note: containers are not VMs. They share kernel resources. Use them carefully.

Module 6

6. Experiment Template

Use this template for every run:

- Title:
- Date:
- Snapshot: baseline-YYYYMMDD-VMname
- Goal: one sentence
- Scope: what VMs, networks, and files are in scope
- Pre-checks: list of things checked before starting (snapshots present, monitoring active)
- Steps:
- 1) Step name - short description
- 2) ...
- Expected observable results: list
- Stop conditions: when to abort
- Artifacts produced: pcap, logs, screenshots (filenames)
- Cleanup actions: revert snapshots, delete temp files
- Notes: anything weird, odd timings, follow-ups

7. Safe practice exercises

These are learning-first. Do them in the lab only.
Exercise A: Baseline network mapping
Goal: learn how services talk, make pcap, and create a network map

Steps:

1. Boot monitoring VM; start pcap on the lab network interface.
2. Boot target VM with a simple web server or intentionally vulnerable web image.
3. Boot attacker VM. From the attacker VM, enumerate open ports using non-invasive, passive techniques. Do not run intrusive scans. Note responses.
4. Stop capture, review pcap. Look for HTTP requests, TTL values, and any service banners.
5. Document findings and map out services in a diagram. Was anything unexpected observed?

Why this helps: Teaches you what normal traffic looks like. You learn to trust your baseline.

7. Safe Practice Exercises

Exercise B: Snapshot, modify, and rollback
Goal: practice snapshot discipline and reproducible rollback

Steps:

1. Create snapshot baseline-1.
2. On target VM, install a benign package or change a config. Take snapshot baseline-2.
3. Run a small experiment that modifies a file. Note changes.
4. Revert to baseline-1. Confirm the file is back to original state.
5. Reapply baseline-2 and confirm changes reappear.

Why this helps: You learn to not fear breaking things because you can revert.

7. Safe Practice Exercises

Exercise C: Controlled instrumentation
Goal: run an instrumented binary in sandbox and capture its system calls conceptually

Steps:

1. Use a benign sample known to produce observable syscalls, or write a tiny program that performs network and file operations.
2. Start the monitoring VM capture. Start a system call tracer in the target VM.
3. Run the program. Capture the pcap and syscall log.
4. Correlate timeline entries from the log and pcap. What syscall caused the network traffic? What file accesses preceded network calls?

Why this helps: Anchors dynamic observation to concrete events.

Module 6

8. Evidence Handling and Reproducibility

You want your experiments to be trusted, repeatable, and useful months from now.

Basic evidence hygiene:

1. Timestamp everything.
2. Store original pcaps and logs. Never overwrite originals. Work on copies.
3. Keep checksums and a readme for each experiment folder.

Module 6

8. Evidence Handling and Reproducibility

Archive workflow:

1. After finishing analysis, compress the experiment folder to a single archive.
2. Compute a hash and store it in a central secure location.
3. Keep a short human-readable summary file in the archive's root that explains the experiment in plain English.

Module 6

8. Evidence Handling and Reproducibility

Sharing with stakeholders:

1. Share only sanitized artifacts if they contain anything that could be sensitive.
2. Prefer anonymized timelines and summarized pcaps for initial reports. Only share raw pcaps with authorized analysts.

Module 6

9. Clean up and long-term maintenance

Daily or weekly tasks:

- Prune old snapshots you no longer need. Keep named archives for important experiments.
- Rotate and compress pcaps older than a threshold.
- Update baseline VMs on a schedule but snapshot before each major update so you can roll back if updates break things.

When you stop using the lab:

- Export final snapshots for important work.
- Wipe ephemeral drives securely.
- Destroy VMs you no longer trust.

Module 6

10. Short Example Lab Policies You Can Adapt

Minimal Lab Policy:

- No experiment leaves the virtual internal network unless a signed exception is filed.
- All research must log start time, snapshot, and lead researcher.
- No personal data in test images. Use synthetic test sets only.
- Any unexpected external contact triggers immediate halt and reporting.

Module 6

11. Common Mistakes & How To Avoid Them

Mistake: Not snapshotting before installing tools

- Fix: Snapshot first. ALWAYS snapshot first.

Mistake: Using real credentials in test configs

- Fix: Generate synthetic credentials. Use patterns like testuser+timestamp.

Mistake: Leaving monitoring off

- Fix: Make monitoring VM the first thing you boot. Make it part of your checklist.

Mistake: Mixing host network and lab network

- Fix: Use internal-only virtual networks and only open gateways consciously.