

// the attacker's mindset: continue here

The real path into offensive security.

No hype, no shortcuts. This is the order I'd learn in if I started today, with the exact free resources, tools, labs, books and people. Work it top to bottom.

Don't skip the fundamentals: that's the step that separates people who last from people who quit.

Built by Mohamed Medjahdi, Cybersecurity Engineer • medjahdi.dev

CONTENTS

- | | |
|--------------------------------|-------------------------|
| 01 Fundamentals (don't skip) | 02 The vocabulary |
| 03 Go web first | 04 The tools + commands |
| 05 Practice labs | 06 OWASP Top 10 |
| 07 The methodology | 08 Recon in depth |
| 09 AI in your workflow | 10 The report template |
| 11 Certifications | 12 Books |
| 13 People to learn from | 14 Bug bounty |
| 15 Community + staying current | 16 Reputation |
| 17 Making money | 18 Your machine |
| 19 Ethics & the law | 20 A 90-day plan |

Fundamentals: the step everyone skips

If you jump straight to "hacking" you'll spend years copy-pasting payloads you don't understand and wondering why they fail. Learn how a request actually reaches a server and the rest gets easy. Four things, roughly in this order:

Networking

TCP/IP, ports, DNS, what actually happens when you load a page.

[Professor Messer \(free\)](#) ↗

[THM: Network Fundamentals](#) ↗

Linux

The command line is your home. Learn files, permissions, processes, pipes. Best way is by doing:

[OverTheWire: Bandit](#) ↗

[Linux Journey](#) ↗

How the web works

HTTP requests & responses, methods, status codes, headers, cookies, sessions. This is the ground under every web attack.

[MDN: HTTP](#) ↗

A bit of scripting

Python first: enough to read others' tools and write small ones of your own.

[Automate the Boring Stuff \(free book\)](#) ↗

The vocabulary you'll hear everywhere

Vulnerability, a weakness that can be abused.

Payload, what the exploit delivers/runs.

Enumeration, digging deeper into what recon found.

Privilege escalation, limited access → full/admin.

CVE, a public ID for a known vuln.

SQLi, injecting into a database query.

CSRF, forcing a victim's browser to act.

RCE, running your commands on the server.

Exploit, the technique/code that abuses it.

Recon, gathering info on the target.

Foothold, your first access into a system.

Shell, command access on the target.

XSS, running your JavaScript in a victim's browser.

IDOR, changing an ID to reach others' data.

SSRF, making the server send requests for you.

LFI / RFI, including local/remote files you shouldn't.

Pick one surface: start with web

Offensive security is huge: web, networks, mobile, cloud, IoT, wireless. Don't scatter across all of them. **Start with web apps**, free to practise, browser-based, in constant demand, fastest feedback. The single best free resource on the planet for this is PortSwigger's Web Security Academy, work through it topic by topic.

[Web Security Academy \(free\)](#) ↗

[Learning paths](#) ↗

The tools: what they do, and one command

recon · port scanning

Nmap

Find open ports & services. nmap.org

web testing

Burp Suite

Intercept & edit web traffic.
portswigger.net/burp

content discovery

dirsearch / ffuf

Brute-force hidden paths. [dirsearch](https://github.com/dirsearch/dirsearch.py) · [ffuf](https://github.com/ffuf/ffuf)

sql injection

sqlmap

Automate SQLi. sqlmap.org

vuln scanning

Nuclei

Template-based checks at scale. [nuclei](https://nuclei.tools)

traffic

Wireshark

Read the packets. wireshark.org

passwords

hashcat / John

Crack hashes. [hashcat](https://hashcat.net) · [John](https://www.openwall.com/john/)

decode

CyberChef

base64, JWT, hashes, URLs. [CyberChef](https://cyberchef.org)

And the two things every one of these needs: **wordlists** (SecLists) and a **toolkit OS** (Kali Linux, in a VM). A few starter commands:

the basics

```
# find services
nmap -sV -sC target.com
# find hidden paths
dirsearch -u https://target.com -t 10
ffuf -u https://target.com/FUZZ -w ../common.txt
# test a parameter for SQLi (only on targets you own/are allowed)
sqlmap -u 'https://target.com/item?id=1' --batch
```

Practice: legal targets only

You get good by doing, thousands of reps, but only on systems built to be hacked or that you own:

- › [PortSwigger Academy](#), free, in-browser, the best web training anywhere. Start here.
- › [TryHackMe](#), guided beginner rooms. Free tier + cheap sub. Try the "Jr Penetration Tester" and "Web Fundamentals" paths.
- › [HackTheBox](#) + [HTB Academy](#), harder, realistic. Great once you have the basics.
- › [OverTheWire](#), classic Linux/terminal wargames. Free.
- › [picoCTF](#), beginner CTF challenges. Free.
- › [VulnHub](#) / [DVWA](#), vulnerable machines/apps to run yourself.
- › [OWASP Juice Shop](#), the app from the workshop. Run it locally:

```
your own practice target (needs Docker)
```

```
# then open http://localhost:3000  
docker run --rm -p 3000:3000 bkimminich/juice-shop
```

Get Docker ↗

Learn the vulnerabilities, OWASP Top 10

The reference list of the ten riskiest web flaws. Learn each, then hunt it in the labs. The workshop's XSS is **A03: Injection**. Pair this with the OWASP Testing Guide: a free, professional checklist of how to test everything.

- › **A01 Broken Access Control**, reaching data that isn't yours
- › **A02 Cryptographic Failures**, weak or missing encryption
- › **A03 Injection**, untrusted input runs as code (XSS & SQLi)
- › **A04 Insecure Design**, the flaw is in the design itself
- › **A05 Security Misconfiguration**, unsafe settings or defaults
- › **A06 Vulnerable & Outdated Components**, old libraries, known holes
- › **A07 Auth Failures**, weak logins and sessions
- › **A08 Software & Data Integrity**, trusting unverified code/updates
- › **A09 Logging & Monitoring Failures**, attacks go unseen
- › **A10 Server-Side Request Forgery**, the server tricked into requests

OWASP Top 10 ↗

OWASP Testing Guide ↗

Work a methodology, every time

Pros don't hack randomly. They run the same phases on every target:

- › **1. Recon**, learn the target (tools *and* manual poking).
- › **2. Scanning / enumeration**, map ports, services, endpoints.
- › **3. Exploitation**, turn a weakness into a foothold.
- › **4. Post-exploitation**, escalate, pivot, measure real impact.
- › **5. Reporting**, write it up clearly. This is what a client pays for.

Automated tools find hidden paths. **Manual testing**, poking every input by hand: is what finds the good bugs: broken business logic, access-control flaws, a reflected XSS a scanner walks right past. Tools narrow the search; your brain finds the bug.

Recon, in more detail

Recon is where most bugs are won. Three layers:

- › **Subdomains:** more subdomains = more attack surface. Tools: [subfinder](#), [amass](#), and the free web tool [crt.sh](#) (certificate logs).
- › **Content / endpoints:** hidden paths and files with [dirsearch](#) / [ffuf](#), plus Burp's site map as you browse.
- › **Tech fingerprinting:** what's it built with? Wappalyzer, plus [nuclei](#) for known-vuln checks.

Read the output critically. Most of a scan is noise: a wall of errors and dead ends. The skill isn't running the tool, it's spotting the two useful lines in two hundred.

Use AI the right way

In 2026 AI is part of the job. Autonomous agents now run recon and exploitation, and they drive the same classic tools you're learning, AI didn't replace Nmap and Burp, it orchestrates them. But it's a **multiplier, not a replacement**: it pays off for the person who can already tell when its output is wrong.

- › Use it to explain errors, write small scripts, and speed up the boring parts.
- › Use it to draft reports (template below).
- › **Always verify.** Never submit a finding or report you haven't confirmed yourself.

Fundamentals first. Then AI is a force multiplier in your hands, not a crutch hiding what you don't understand.

The AI vulnerability-report template

Finding a bug is half the job: the report is the other half, and it's the half that pays. Paste your finding plus this template into any AI, get a first draft, then **verify and edit it yourself**.

1, The template

vulnerability-report.md

```
# Vulnerability Report

Title: [short descriptive title]
Severity: [Critical / High / Medium / Low]
Affected asset: [URL / endpoint / feature]
Date: [date]   Tested by: [name]

## Summary
[1-2 sentences: what it is and why it matters.]

## Steps to Reproduce
1. [step]   2. [step]   3. [step]

## Proof of Concept
[the exact payload / request used]

## Impact
[what an attacker can actually do - be concrete.]

## Remediation
[how to fix it.]

## References
[OWASP link, CVE if any]
```

2, The prompt to give your AI

prompt.txt

```
You are a penetration tester writing a professional vulnerability report.
Using ONLY the finding below, fill in the template. Be precise and concise.
Do not invent details: mark anything uncertain as (to confirm).
```

```
FINDING:
```

- Vulnerability type: [e.g. DOM-based XSS]
- Location: [e.g. product search field]
- Payload used: [the script you injected]
- What happened: [script executed, session token shown]
- Target: [OWASP Juice Shop, local test instance]

TEMPLATE:

[paste the template above]

AI drafts it. **You verify it.** A report you didn't check is worse than no report.

Certifications: the honest ladder

Certs help you get past HR filters and give you structure. You don't need them to start, and skills beat paper, but here's the sensible order, cheap to expensive:

- › **CompTIA Security+**, broad entry-level, good for job filters. [comptia.org](https://www.comptia.org)
- › **eJPT (INE)**, beginner, hands-on, affordable. Great first pentest cert. [eJPT](https://www.ejpt.com)
- › **PNPT (TCM Security)**, practical, real report required, good value. [PNPT](https://www.pnpt.com)
- › **HTB CPTS**, thorough, modern, hands-on. [HTB Academy](https://www.htb.com)
- › **OSCP (OffSec)**, the well-known hard one. Aim here later, not first. [OSCP / PEN-200](https://www.offsec.com)

Don't collect certs for the wall. Pick one, use it to force yourself to study, and put the projects you build along the way on GitHub.

Books worth your time

- › **The Web Application Hacker's Handbook**, the web-hacking bible (older but foundational).
- › **Penetration Testing**, Georgia Weidman (No Starch). Best beginner all-rounder.
- › **The Hacker Playbook 3**, Peter Kim. Practical, scenario-based.
- › **Bug Bounty Bootcamp**, Vickie Li. If bounties are your goal.
- › **Real-World Bug Hunting**, Peter Yaworski. Learn from real disclosed bugs.

People to learn from (free, on YouTube)

IppSec

HackTheBox walkthroughs. [watch ↗](#)

John Hammond

Practical hacking & CTFs. [watch ↗](#)

The Cyber Mentor

Beginner-friendly pentest courses. [watch ↗](#)

NahamSec

Bug bounty & recon. [watch ↗](#)

LiveOverflow

Deep, concept-first hacking. [watch ↗](#)

PwnFunction

Web vulns explained with animation. [watch ↗](#)

Bug bounty: how to actually start

Bug bounties are the legal way to hack real companies: they invite you, you find bugs, they pay. It's hard at first: don't expect money fast. How to begin:

- › Finish PortSwigger Academy first. You can't hunt what you can't recognise.
 - › Read **disclosed reports** to see what real bugs look like: [HackerOne Hacktivity](#).
 - › Start on **VDPs** (programs with no payout) to build confidence without competition.
 - › Platforms: [HackerOne](#), [Bugcrowd](#), [Intigriti](#), [YesWeHack](#).
 - › **Always read the scope**. Testing outside it is illegal, bounty or not.
-

Community & staying current

- › **Communities**: the TryHackMe and HackTheBox Discords, [r/netsec](#), and infosec X/Twitter. Ask, share, lurk, learn.
- › **Stay current**: [Exploit-DB](#), [the CVE database](#), and the [tl;dr sec](#) newsletter.
- › **Play CTFs**: find events on [CTFtime](#). The fastest fun way to level up.
- › **Watch the AI space**, tooling is moving monthly; keep your workflow current.

Build a reputation in public

Your public work is what gets you hired: proof beats a CV full of claims.

- › **GitHub:** put your scripts, tools and notes there, keep it active.
- › **Write-ups:** solve a box or a lab, then write how you did it. A blog, Medium, or your own site.
- › **Keep notes:** a personal knowledge base (Obsidian, Notion), the writing habit is the reporting skill that pays.

`github.com/medjahdi` ↗

How you make a living from this

get hired

A job

SOC analyst, pentester, red teamer, AppSec engineer. Demand is high and pay climbs fast.

your own clients

Freelance / consulting

Run security audits for companies as an independent.

paid per bug

Bug bounties

Find real bugs on real programs: legally, and get paid per finding.

fuel for all three

Reputation

Write-ups, GitHub and solved labs open every one of these doors.

What machine do you need?

The one already in your hands. Mac, Windows or Linux: it doesn't matter when you start. Master your machine first.

- › Don't install Linux just because it looks "hacker." Run it in a **virtual machine** first ([VirtualBox](#), VMware, or UTM on Apple Silicon).
- › On a Mac you don't need to switch: it runs everything you need.
- › When ready, [Kali Linux](#) in a VM gives you a ready-made toolkit.

Ethics & the law

Everything above is legal **only** on systems you own or have explicit permission to test.

- › In Algeria, unauthorized access to a computer system is a crime, **Penal Code Article 394 bis** (up to a year in prison and a fine).
- › **Bug bounties** are the legal way to attack real systems: you're invited, and paid.
- › Labs (Juice Shop, PortSwigger, your own VMs) are always safe, built for it.

The same skill builds a career or a criminal record. Choose the one with a future.

A concrete 90-day starter plan

1 Weeks 1–3 · Fundamentals

Networking + Linux (OverTheWire Bandit) + how HTTP works. Daily terminal practice.

2 Weeks 4–7 · Web basics

Work PortSwigger Academy, XSS, SQLi, access control, auth. One topic at a time, do the labs.

3 Weeks 8–10 · Tools + methodology

Learn Burp properly, run dirsearch/ffuf and Nmap, practise on Juice Shop and TryHackMe.

4 Weeks 11–12 · Put it together

Do full boxes (recon → exploit → report). Write up every one on GitHub/your blog.

5 Ongoing

Read disclosed bug-bounty reports, play a CTF, pick a first cert (eJPT or PNPT) as a goal, and keep publishing.

Think like an attacker. Use it to defend.

Roadmap by Mohamed Medjahdi, Cybersecurity Engineer

[medjahdi.dev](#) [GitHub](#) [LinkedIn](#)

For educational and cybersecurity awareness purposes only.