FREE SAMPLE · THE METHOD + CHAPTERS 0–1

THE ADVERSARY'S MIND

A Hands-On Path from Novice to Operator

A Hands-On Field Manual — Network, Web, Wireless, Cloud, Identity & Red Team

BY

DHANANJAI SHARMA

> Drawn from the 100+ labs I built and ran myself — distilled into one progressive path, so you don't have to grind through scattered courses or an endless curriculum to learn to think like an operator. It teaches the thinking first, then turns you loose on the labs.

Network · Web · Wireless · Cloud · Identity · Red Team · June 2026

For authorized testing and your own lab environment only.

A HANDS-ON FIELD MANUAL

THE ADVERSARY'S MIND

A Hands-On Path from Novice to Operator

Dhananjai Sharma

First Edition · 2026

THE ADVERSARY'S MIND

A Hands-On Path from Novice to Operator

First edition, 2026.

Disclaimer. This book is provided for educational purposes only. Every technique in it is intended solely for use against systems you own or are explicitly authorized in writing to test. Unauthorized access to computers and networks is illegal in most jurisdictions and may carry criminal and civil liability. The author accepts no responsibility and disclaims all liability for any misuse of, or damage arising from, the information in this book. You alone are responsible for your actions.

Read this first — what's in this free sample

This is a free sample of *The Adversary's Mind*. It contains the book's first two chapters **in full** — every diagram, the under-the-hood theory, the Operator's Corner notes and trivia, exactly as they appear in the book — plus the complete table of contents so you can see precisely where the path goes.

The full book is **122 pages across 14 chapters**, recon through red team. The complete bundle adds a **120-page Lab Workbook** with 100+ hands-on labs. Everything here is taught for systems you own or are explicitly authorized to test.

If these two chapters earn your trust, the rest is one click away — and it's backed by a full refund if you haven't downloaded it.

The full book at a glance

Part I — The Mind & The Ground

- **Chapter 0 — How an Attacker Thinks** · *included in this sample*
- **Chapter 1 — Reconnaissance (OSINT)** · *included in this sample*
- Chapter 2 — Scanning & Enumeration

Part II — Gaining Access

- Chapter 3 — Exploitation
- Chapter 4 — Web Application Attacks
- Chapter 5 — Wireless Attacks
- Chapter 6 — Social Engineering

Part III — Escalation & Dominance

- Chapter 7 — Linux Privilege Escalation
- Chapter 8 — Windows Privilege Escalation
- Chapter 9 — Active Directory Attacks

Part IV — The Wider Battlefield

- Chapter 10 — Cloud Security
- Chapter 11 — Mobile Application Security

Part V — The Full Circle

- Chapter 12 — The Defender's Mind — forensics, detection & incident response
- Chapter 13 — Red Team Operations — the capstone

Appendices — A: The Wireless Audit Framework (full annotated source) · B: Universal Command Cheat Sheet · C: Certifications & Practice Platforms · D: Reporting Templates · E: Tools & Commands Reference.

The companion Lab Workbook *(included in the bundle)* — 120 pages, 100+ gated hands-on labs across 12 domains, plus a 15-lab wireless deep track.

How to read every chapter

Every chapter runs the same seven-step **Loop**, so the reasoning becomes automatic:

Concept → Hacker's Mindset → The Attack → Level Up → Cat & Mouse → Defensive Playbook → Run the Labs.

On top of the Loop, each chapter opens with a **concept diagram** and is threaded with high-signal notes — you'll see all of these in the two chapters that follow:

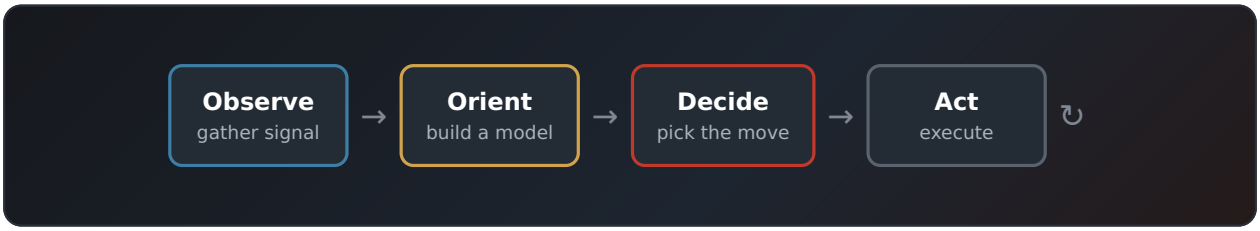
- **Theory · Under the Hood** — the real mechanism beneath the command.
- **Command Deep-Dive** — what a flag or option is actually doing.
- **Copy-Paster → Operator** — the one habit that separates running a command from understanding it.
- **Field Note & Trivia** — the history and war stories, so the technique sticks.
- **Try This & Prove It** — a safe experiment for your own machine, plus a free or paid range to test the skill for real.

Attacks carry their **MITRE ATT&CK** technique IDs, and every attack is paired with how you'd be detected and how you'd defend — because half of becoming good at offense is knowing exactly how you get caught.

CHAPTER 0 — HOW AN ATTACKER THINKS

Read this first. Re-read it after Chapter 13. It is the only chapter that matters in all the others.

Tools are cheap. Anyone can run `nmap`. What separates a script-kiddie from an operator is not the commands they know — it's the **model of the target in their head** and the **loop they run against it**. This chapter installs that model.



The OODA loop. Whoever cycles it faster — attacker or defender — controls the engagement.

The four truths every attacker internalizes

1. **You attack assumptions, not systems.** Every system is built on assumptions: "only employees can reach this," "this input is always a number," "the client validated that already." A vulnerability is just an assumption that turned out to be false. Your job is to find the assumption and break it.
2. **The defender has to be right everywhere; you have to be right once.** This asymmetry is your entire advantage. You don't need the whole wall to fall — you need one loose brick.
3. **Access is a ladder, not a door.** You rarely go from "outside" to "domain admin" in one jump. You go: no access → low-priv user → local admin → another host → domain admin. Each rung is a separate, smaller problem.
4. **Everything you do is loud unless you choose otherwise.** Every packet, every login, every file you drop is a potential alert. The amateur asks "did it work?" The operator asks "did it work, and who saw?"

THEORY · UNDER THE HOOD — ATTACK SURFACE & TRUST BOUNDARIES

Every system is a set of **trust boundaries** — lines where data or control crosses from a less-trusted zone into a more-trusted one (user to app, app to database, internet to DMZ to internal). A *vulnerability* is almost always a trust boundary that isn't actually enforced: input assumed safe, identity assumed verified, a network assumed internal. The complete set of those crossings is the **attack surface**. Defenders shrink it — least privilege, segmentation, validation; attackers enumerate it and hunt for the one crossing where the assumed check is missing. Every later chapter is just this single idea applied to a specific boundary.

The mental loop: OODA at the keyboard

Fighter pilots use OODA — Observe, Orient, Decide, Act. Operators use the same loop on every target:

```
OBSERVE  →  what's here? (recon, scan, enumerate)
ORIENT   →  what does it mean? (which assumption can I break?)
DECIDE   →  what's the cheapest path to the next rung?
ACT      →  exploit — then immediately OBSERVE again from the new position
```

The beginner's mistake is to fall in love with **Act** — to fire exploits before they've truly Observed and Oriented. The single highest-leverage habit in this entire book: **enumerate until it hurts, then enumerate more.** 80% of "I'm stuck" is actually "I didn't enumerate enough." You will see this sentence again in every chapter, because it is true in every chapter.

The kill chain — the map of an engagement

The whole book is one kill chain. Memorize the stages; every chapter is one of them:

```
RECON ──► WEAPONIZE ──► DELIVER ──► EXPLOIT ──► INSTALL ──► C2 ──► ACT ON OBJECTIVES
(Ch 1-2)   (Ch 3,6)      (Ch 6)      (Ch 3-5)   (Ch 7-9)   (Ch13)   (Ch 9-11)
```

And the parallel ladder once you're inside a host or network:

```
 FOOTHOLD ──► LOCAL ENUM ──► PRIVESC ──► CREDENTIAL ACCESS ──► LATERAL MOVEMENT ──►
DOMAIN/CLOUD DOMINANCE ──► PERSISTENCE ──► EXFIL
```

MITRE ATT&CK — the language professionals speak

Every technique in this book maps to a `T-number` in the MITRE ATT&CK framework (attack.mitre.org). This isn't bureaucracy — it's the shared vocabulary that lets a red-teamer, a SOC analyst, and a CISO talk about the same event. When you Kerberoast, that's `T1558.003`. When you deauth a client, that's part of `T1557`. **Start tagging your own actions with ATT&CK IDs now.** By Chapter 13 you'll do it reflexively, and your reports will read like a senior's.

A handful you'll meet constantly:

ID	Technique	Chapter
T1595 / T1592	Active scanning / gather host info	1, 2
T1190	Exploit public-facing application	3, 4
T1110	Brute force (incl. password spraying)	2, 5, 9
T1557	Adversary-in-the-middle	5, 9
T1068	Exploitation for privilege escalation	7, 8
T1558	Steal/forge Kerberos tickets	9
T1078	Valid accounts (the quietest attack of all)	9, 10
T1566	Phishing	6

The Cat & Mouse loop — why this book is structured the way it is

Security is not a checklist; it's an **arms race**. Every defense breeds a new attack; every new attack breeds a better defense. If you learn an attack as a static trick, you become obsolete the moment a patch ships. If you learn the *loop*, you stay dangerous forever.

Here is the loop in its purest form, using one example you'll meet in Chapter 5:

```
ATTACK         Deauth a Wi-Fi client to force a handshake, then crack it offline.
   |
DEFENSE        Enable 802.11w (Protected Management Frames) — deauth frames are now
signed; the flood is ignored.
   |
ATTACKER MOD   Skip the client entirely. Capture the PMKID straight from the AP — no
deauth, no client, no PMF to bypass.
   |
HARD DEFENSE   Patch/replace routers that leak PMKID, AND move to WPA3-SAE so the captured
material is uncrackable offline anyway.
   |
ATTACKER MOD   Can't beat SAE? Don't. Stand up an evil twin in WPA2 transition mode and
downgrade the client.
   |
HARD DEFENSE   WPA3-only network, no transition mode. Client cert validation. Now the
downgrade has nowhere to land.
```

Notice the shape: **the attacker doesn't beat the defense head-on — they move sideways to where the defense isn't.** That sideways move is the whole game. Every `Cat & Mouse` box in this book trains that instinct.

The operator's checklist for every single target

Tape this to your wall. It's the distilled mindset:

- [ ] **What do I already know?** (recon before you touch anything)

- [ ] **What is this thing's *job*?** (its purpose reveals its trusted inputs)
- [ ] **What does it trust?** (users, networks, other services, file contents, headers)
- [ ] **What's the cheapest assumption to break?** (default creds beat 0-days every time)
- [ ] **If I get in, where am I, and what can I reach from here?** (always think one rung ahead)
- [ ] **Who would see this, and do I care right now?** (OPSEC is a choice, make it consciously)
- [ ] **Can I write down exactly what I did, well enough to do it again and to fix it?** (if you can't reproduce it, you didn't really do it)

Now run the labs

There are no commands in this chapter — the lab is mental. Before Chapter 1:

- Read the MITRE ATT&CK Enterprise matrix once, top to bottom. Don't memorize; just see the shape.
- Pick a machine you'll be allowed to attack later (a TryHackMe box) and, *without touching it*, write down what you'd want to Observe first and why.

Operator's Corner

COPY-PASTER → OPERATOR — THE ONE HABIT THAT SEPARATES THE TWO

A copy-paster runs a command and waits to see what happens. An operator states — out loud or in a note — what they expect to happen **before** they hit enter, then learns from the gap between prediction and result. That gap is where real skill compounds. If you can't predict the output, you don't understand the command yet; run it in a lab until you can.

FIELD NOTE — THE WORM THAT CREATED AN INDUSTRY

In November 1988 a Cornell graduate student, Robert Tappan Morris, released a small program to "measure the size of the internet." A bug made it re-infect machines relentlessly, crippling an estimated 10% of the roughly 60,000 hosts then online. It produced the first felony conviction under the U.S. Computer Fraud and Abuse Act — and led directly to the first CERT. The lesson operators still repeat: small mistakes scale catastrophically, so you test in a lab first.

TRY THIS — MAP AN ATTACK SURFACE WITH NO TOOLS AT ALL

Pick a service you use every day — your bank, your email, your gym's app. On paper, list every way *in*: login pages, password reset, the support phone line, the people who work there, the third parties they trust. You just did threat modeling. The mindset — "where is the trust, and how is it verified?" — is the entire job. The tools only automate this question.

PROVE IT · PRACTICE ONLINE — BUILD THE MUSCLE THE REST OF THE BOOK ASSUMES

Before Chapter 1, make the Linux terminal second nature. **OverTheWire — Bandit** (free, overthewire.org) is the canonical wargame that teaches the command line by making you hack your way to each next level over SSH. Finish Bandit comfortably and you're ready for everything ahead.

TRIVIA · HACKER LORE

The first computer "bug" was a literal one. In 1947, operators of Harvard's Mark II found a moth jammed in a relay, taped it into the logbook, and noted "first actual case of bug being found." The page (moth included) lives in the Smithsonian. Your segfault has a noble lineage.

CHAPTER 0 GATE

- [ ] You can explain the OODA loop and why beginners over-invest in "Act."
- [ ] You can recite the kill chain and the privilege ladder from memory.
- [ ] You can take any attack you already know and extend it one full Cat & Mouse cycle (attack → defense → modification → hardened defense).
- [ ] You believe, in your bones, the sentence: "*80% of stuck is under-enumeration.*"

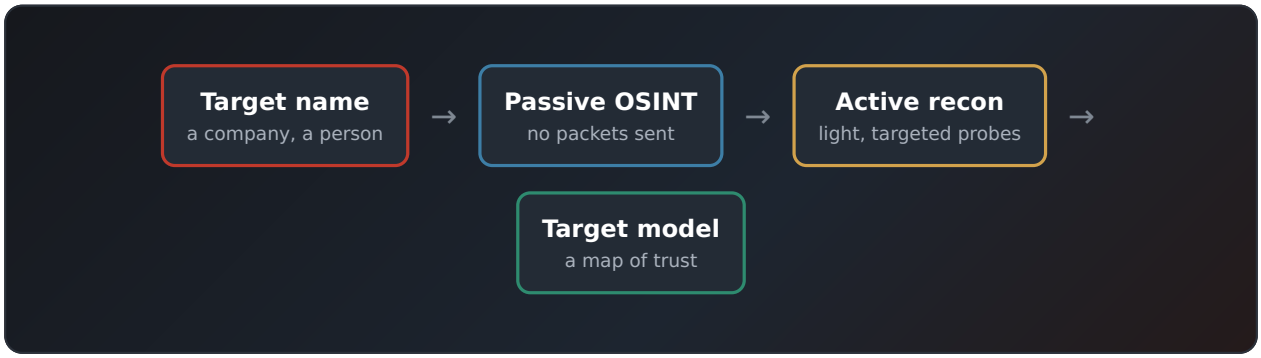
PART TWO

Finding the Way In

Reconnaissance, scanning, exploitation, and the web. The craft of turning a name into a foothold.

CHAPTER 1 — RECONNAISSANCE (OSINT)

You attack what you can find. So find more than they meant to show you.



From a name to a foothold — passive first, active only for the gaps that remain.

Concept

Reconnaissance is building a map of the target *before you send a single packet to it*. Passive OSINT (Open-Source INTelligence) uses only public data — search engines, DNS, certificate logs, breach dumps, social media — so the target has no way to know you're looking. Active recon (Chapter 2) touches the target directly and is therefore noisy. **Recon is 90% of a real engagement** and the part beginners rush. Don't.

The reason recon matters so much ties straight back to Chapter 0: you attack assumptions, and an organization's biggest assumption is "*nobody has assembled all our scattered public information into one picture.*" That assembly is your job.

THEORY · UNDER THE HOOD — HOW DNS TURNS A NAME INTO A MAP

Recon leans on DNS because the **Domain Name System** is a public, distributed database the target *must* publish to function. `A` / `AAAA` records map names to IPs; `MX` reveals mail servers; `NS` the authoritative servers; `TXT` often leaks SPF policy, verification tokens, and tooling hints; `CNAME` aliases expose third-party services (and *dangling* ones enable subdomain takeover). A misconfigured server may even permit a **zone transfer** (`AXFR`) that dumps every record at once. None of this touches the target's own hosts — you're querying the public naming layer — which is exactly why it's the quietest recon there is.

Hacker's Mindset

> *"Every company leaks. The question isn't whether — it's where, and whether anyone bothered to look. I'm the person who bothers."*

At this stage the attacker is hunting for: - **Attack surface** — every domain, subdomain, IP, and internet-facing service. Each one is a door. - **People** — names,

emails, roles, naming conventions. People are the softest target (Ch 6). - **Mistakes frozen in public** — a password in a GitHub commit, an admin panel indexed by Google, an S3 bucket left open, a forgotten subdomain pointing at a deleted cloud service. - **The shape of the org** — what tech they run, what they're hiring for (job posts leak the tech stack), who their vendors are.

The mindset shift: you're not looking for "the vulnerability." You're building a **dossier**, and the vulnerability falls out of it.

The Attack — tried and tested

> **Tools in play. whois / dig / host** — query domain registration and DNS records. **amass / subfinder** — discover subdomains from many public sources at once. **theHarvester** — scrape emails, names and hosts from search engines. **Shodan / Censys** — search engines for internet-exposed devices (they already scanned the target for you). **sherlock** — find a username across hundreds of sites. **crt.sh** — public log of every TLS certificate ever issued (leaks subdomains). **dnsx / httpx** — bulk-resolve subdomains and probe which are live web apps. **trufflehog** — scan code repos for leaked secrets. **subjack / nuclei** — automated subdomain-takeover checks.

Pick an authorized target: a bug-bounty program with `*.target.com` in scope gives you legal cover via its terms.

Google dorking (run in a browser) — `T1593`

```
site:target.com filetype:pdf
site:target.com filetype:xlsx
site:target.com inurl:admin
site:target.com intitle:"index of"
site:target.com ext:sql | ext:db | ext:log
site:target.com intext:"password" | intext:"api_key"
"target.com" site:pastebin.com
"target.com" site:github.com
```

DNS & WHOIS — `T1590`

```
whois target.com
dig target.com any
dig target.com mx
dig target.com txt
host -t ns target.com
```

Passive subdomain enumeration — `T1590.005`

```
amass enum -passive -d target.com -o subs.txt
subfinder -d target.com -o subs-finder.txt
cat subs.txt subs-finder.txt | sort -u > all-subs.txt
wc -l all-subs.txt
```

People & email OSINT — `T1589`

```
theHarvester -d target.com -b google,bing,linkedin,dnsdump -l 500 -f harvest.html
sherlock targetCEOname --output ceo-profiles.txt
# hunter.io (browser) reveals the email format, e.g. first.last@target.com
```

Infrastructure without sending a packet — Shodan/Censys — `T1596`

```
pip install shodan --break-system-packages
shodan init <API_KEY>
shodan domain target.com
shodan search "ssl.cert.subject.cn:target.com"
shodan search "org:Target Corporation"
```

You know it worked when you have a file `all-subs.txt` with subdomains the company never advertised, an email naming convention, and at least one "huh, that shouldn't be public" finding.

Level Up — novice to advanced

- **Novice:** runs `subfinder` once, gets 20 subdomains, moves on.
- **Intermediate:** chains passive sources, then resolves and probes them: `bash cat all-subs.txt | dnsx -silent -a -resp | tee resolved.txt # which actually resolve cat resolved.txt | httpx -silent -title -status-code -tech-detect # which are live web apps + their stack`
- **Advanced:** mines **Certificate Transparency logs** (every TLS cert ever issued is public) and historical DNS: `bash curl -s "https://crt.sh/?q=%25.target.com&output=json" | jq -r '.[].name_value' | sort -u` Then hunts **secrets in code** at scale: `bash trufflehog github --org=targetorg # live secret scanning across all repos`
- **Operator:** automates the whole funnel (subs → resolve → probe → screenshot → secret-scan → diff against last week) and *monitors continuously*, because new subdomains and leaked secrets appear daily. Recon is not a phase; it's a subscription.

Cat & Mouse — Subdomain Takeover

This is the cleanest recon-stage arms race to learn the loop on.

```
ATTACK          Find a dangling CNAME — a subdomain pointing at a deprovisioned cloud
                service (S3, Heroku, Azure, GitHub Pages). Claim the resource; now you
                serve content on *their* domain (phishing, cookie theft, SEO poisoning).
                Tooling: subjack -w all-subs.txt -ssl ; or  nuclei -l all-subs.txt -t
takeovers/
   |
DEFENSE         Audit DNS; delete CNAME records when you decommission a service. Monitor
                for "claimable" responses.
   |
ATTACKER MOD    Defenders only audit the *active* zone. Hunt historical/forgotten zones:
                old acquisition domains, dev/staging subs, country TLDs nobody owns
                anymore. Use crt.sh + wayback (gau, waybackurls) to find subs that DNS
                doesn't even list anymore but apps still link to.
```

```
HARD DEFENSE   Treat DNS as code: every record in version control, CI checks for dangling
               targets on every change, automated daily takeover-scanning of your OWN
               estate (run subjack/nuclei against yourself). Decommission = delete the
               record in the same change that kills the service. No orphans, ever.
```

The lesson, again from Chapter 0: the defender guards the front (active DNS); the attacker walks to the side (forgotten DNS).

Defensive Playbook

- **Minimize the surface:** every public asset is a liability. Take down what you don't need.
- **Secret scanning in CI/CD** (GitHub Advanced Security, trufflehog, gitleaks) — catch the credential *before* it's pushed, because once it's in git history it's compromised forever even if you delete it.
- **Monitor your own footprint:** run the attacker's recon against yourself on a schedule — Shodan/Censys alerts for your IP ranges, crt.sh monitoring for new certs, HIBP/dehashed for breached employee creds.
- **DNS hygiene:** no dangling CNAMEs, ever (see Cat & Mouse above).
- **Train people:** the email naming convention and org chart you just found feeds Chapter 6. Limit what public-facing roles expose.

Now run the labs

From *The Complete Ethical Hacking Lab Workbook*, Category 1:

- **Lab 1.1** — Passive OSINT: Google dorking & public data (Beginner, 2h)
- **Lab 1.2** — Email & people OSINT (Beginner, 2h)
- **Lab 1.3** — Infrastructure mapping with Shodan & Censys (Intermediate, 3h)
- **Lab 1.4** — Subdomain takeover discovery (Intermediate, 3h) — *run the Cat & Mouse loop above live*
- **Lab 1.5** — Full OSINT engagement report (Advanced, 4h) — *this is where you learn to write*
- **Lab 1.6** — Dark web & breach intelligence (Advanced, 3h)

Operator's Corner

COMMAND DEEP-DIVE — PASSIVE VS ACTIVE RECON — AND WHY THE DIFFERENCE IS EVERYTHING

"Passive" means you never send a single packet to the target; you query third parties who already collected the data. `crt.sh` reads public certificate-transparency logs, `amass enum -passive -d target.com` pulls from dozens of OSINT sources, and Shodan shows you ports *it* scanned. "Active" recon — DNS brute-forcing, `amass enum -active`, port scans — touches the target and can be logged. The operator does **all** passive recon first, builds a complete picture, and only goes active for the few gaps that remain.

COPY-PASTER → OPERATOR — A LIST OF SUBDOMAINS IS NOT RECON

The copy-paster pastes a 300-line subdomain dump into a report and calls it done. The operator *correlates*: which host is `dev.` or `staging.` (weaker, often forgotten), which runs an old framework, which employee owns which public repo, what the email format is. Recon output is raw material; the deliverable is a **target model** — a map of trust you can attack.

FIELD NOTE — THE SEARCH ENGINE BUILT FOR HACKERS

In 2009 John Matherly launched **Shodan**, which continuously scans the entire IPv4 internet and indexes the banners — so you can search for "every device of type X with default creds." It reframed recon permanently: for huge swaths of the internet, the scanning has already been done *for* you. Defenders now use it too, to find their own forgotten assets before attackers do.

TRY THIS — RECON YOURSELF

Run `whois yourdomain.com`, then `dig yourdomain.com ANY` and `dig +short txt yourdomain.com`, then open `crt.sh/?q=yourdomain.com` in a browser. Read what the public internet already knows about a domain you own — registrant data, mail servers, SPF records, every subdomain that ever got a certificate. It's sobering, and it's exactly the first thing an attacker sees.

PROVE IT · PRACTICE ONLINE — SHARPEN YOUR OSINT

TryHackMe has free guided rooms (*OhSINT*, *Google Dorking*, *Sakura Room*). For the real thing, **Trace Labs** runs free OSINT CTFs where teams use open-source intelligence to help find missing persons — practice that does genuine good. Keep the **OSINT Framework** (osintframework.com) bookmarked as a tool directory.

TRIVIA · HACKER LORE

In 2012, fugitive software mogul John McAfee was pinned to a location in Guatemala because a magazine published a photo of him with the GPS coordinates still embedded in the image's EXIF data. The first rule of hiding: strip your metadata.

CHAPTER 1 GATE

- [ ] You can build a complete target profile **without sending a single packet** to the target.
- [ ] You can find subdomains, emails, cloud assets, and leaked credentials using only passive techniques.
- [ ] You can run a subdomain-takeover Cat & Mouse cycle and explain the impact (cookie scope, phishing, SEO).
- [ ] You can write a structured OSINT report good enough to hand a client — every finding backed by evidence.

That's 2 of 14 chapters

You've just read the method and the first two chapters in full. The remaining twelve take the same Loop through exploitation, web, wireless, social engineering, Linux and Windows privilege escalation, Active Directory, cloud, mobile, defense, and a full red-team capstone — each with its own diagram, theory, Operator's Corner, and labs to match.

Get the full book

- **The Book — $59.** The complete 122-page field manual, all 14 chapters and appendices A–E.
- **The Complete Bundle — $79.** The book plus the 120-page Lab Workbook: 100+ gated, hands-on labs across 12 domains and a 15-lab wireless deep track.

Read it on any device. Instant download. Free updates to this edition. And if you haven't downloaded it yet, a full refund within 7 days — no questions.

Get it at → theadversarysmind.org

Everything in this book is for systems you own or are explicitly authorized in writing to test. Used that way, it's how professionals are trained. Used anywhere else, the same skills close doors instead of opening them.