

HANDS-ON ETHICAL HACKING

The Complete Practical Course



Abhay Jadhav

ICYWALL · icywall.tech

Hands-On Ethical Hacking

The Complete Practical Course

48 modules — zero to job-ready

Lab · Linux · Networking · Recon · Scanning
Metasploit · Manual Exploitation · Web · Passwords
Shells · Privesc · Active Directory · WiFi
MITM · Pivoting · Exploit Dev · Evasion
Social Engineering · Cloud · Bug Bounty · Career
Crypto · Mobile · Containers · Hardware
Forensics · Capstone Walkthroughs · Interview Prep

Real attacks *and* real defenses — every command typed,
every technique lab-tested, every module ending in an exercise

ICYWALL

icywall.tech

Copyright & License

© 2025–2026 ICYWALL (icywall.tech). All rights reserved.

Licensed to the individual purchaser for personal, non-transferable use only. Each downloaded copy is watermarked with the buyer’s email and payment ID. You may NOT share, resell, upload, or redistribute this book, remove its watermark, or share your download link. Unauthorized distribution is traceable and may result in license termination and legal action.

Legal disclaimer: This book teaches offensive techniques that are illegal when used without authorization. Every technique here is for lab environments, CTF platforms, bug bounty scopes, and systems you own or have explicit written permission to test. Unauthorized access is a crime under the IT Act 2000 (India, §43/66), the CFAA (US), the Computer Misuse Act (UK), and equivalents worldwide. The authors and ICYWALL accept no liability for misuse. You are responsible for your own actions.

The one rule that keeps you legal: if you didn’t deploy it, buy it, or get written permission — don’t touch it.

How defenses appear: every module includes a “Defender’s corner” — how a blue teamer detects or blocks the attack you just learned. Knowing both sides is what separates a pentester from a script kiddie.

Contents

Module 0 — Read This First	22
0.1 What “ethical hacking” actually is	22
0.2 The 12 concepts you MUST know first	22
0.3 How the commands work — the pattern	22
0.4 The attack loop — what you’ll be doing	22
0.5 What the outputs look like — read the terminal	22
0.6 Your first commands — do these right now	23
0.7 The mindset — how you actually learn	23
0.8 What “being a hacker” actually means	23
1 Build Your Hacking Lab	24
1.1 Why a lab (and why everyone skips this too fast)	24
1.2 Hypervisors — pick one	24
1.3 The network — isolation is everything	24
1.4 Installing Kali	24
1.5 Your targets — the vulnerable menagerie	24
1.6 Full arsenal install (do it once)	25
1.7 Snapshots — the undo button	25
1.8 Your hacking journal — start now	25
1.9 Cloud lab option (advanced)	25
1.10 Verify the lab end-to-end	25
1.11 Defender’s corner	25
1.12 Deep dive — lab architectures beyond VirtualBox	26
1.12.1 The multi-tier attack lab	26
1.12.2 Proxmox / ESXi — the homelab upgrade	26
1.12.3 Cloud lab — careful use	26
1.12.4 Portable toolkit — your attack USB	26
1.13 Deep dive — the operator’s daily discipline	26
1.14 Lab safety — don’t attack yourself	26
2 Linux & Terminal Mastery	26
2.1 Navigation — moving like a native	26
2.2 Reading files — every way	27
2.3 Finding things — find vs grep vs locate	27
2.4 Permissions — the security model in 9 bits	27
2.5 Users & identity	27

2.6	Processes & services	27
2.7	Packages	28
2.8	Redirection — control every byte	28
2.9	Text processing — the hacker’s superpower	28
2.10	Editing — vim survival	28
2.11	tmux — never lose a session	28
2.12	Bash scripting — automate the boring	28
2.13	The /proc filesystem — live system intel	29
2.14	Logs — where attackers live and die	29
2.15	Defender’s corner	29
2.16	Deep dive — power-user patterns	30
2.16.1	One-liners that save engagements	30
2.16.2	Aliases & shell customization	30
2.16.3	Process substitution & advanced redirection	30
2.16.4	Scripting for pentesters — the reusable templates	30
2.16.5	tmux — work like a pro	30
2.16.6	SSH — the admin’s weapon	31
3	Networking for Hackers	31
3.1	The OSI model — compressed for hackers	31
3.2	Anatomy of a connection	31
3.3	TCP vs UDP — and the handshake	31
3.4	Packet anatomy — read a packet like a pro	32
3.5	IP addressing & subnetting — the only math	32
3.6	NAT, routing, gateways	32
3.7	The essential ports — with the WHY	32
3.8	DNS — the protocol that leaks everything	32
3.9	HTTP — read requests fluently	33
3.10	Wireshark — the x-ray goggles	33
3.11	tcpdump — Wireshark without the GUI	33
3.12	IPv6 — the blind spot	33
3.13	Defender’s corner	34
3.14	Deep dive — reading packets like a native	35
3.14.1	TCP flags — what each scan actually does	35
3.14.2	ICMP — more than ping	35
3.14.3	IPv6 — the shadow network	35
3.14.4	The proxy/vpn layer — traffic routing	35
3.15	Deep dive — network troubleshooting for hackers	35
3.16	The TCP handshake — the picture	35
3.17	The OSI model — where every attack lives	36
4	Reconnaissance & OSINT	37
4.1	The recon mental model	37
4.2	Company intel — understand the org	37
4.3	Certificate transparency — the free subdomain goldmine	37
4.4	Subdomain enumeration — the complete playbook	37
4.5	Google dorks — weaponized search	37
4.6	Shodan — the search engine for devices	38
4.7	GitHub/GitLab dorking — committed secrets	38
4.8	Email & people recon	38
4.9	Metadata — documents that talk	38
4.10	Cloud surface — the new perimeter	38
4.11	Wayback Machine — time travel	38

4.12	Technology fingerprinting	39
4.13	Subdomain takeover — the recon-only bounty	39
4.14	Assembling the dossier	39
4.15	Defender’s corner	39
4.16	Deep dive — advanced OSINT techniques	40
4.16.1	Email intelligence	40
4.16.2	People OSINT	40
4.16.3	Document metadata — the quiet leak	40
4.16.4	Cloud surface — the modern perimeter	40
4.16.5	Dark web / breach OSINT — the legitimate use	40
4.16.6	Physical/geolocation OSINT	40
4.17	The OSINT-to-attack bridge	40
5	Scanning & Enumeration	41
5.1	nmap — the complete operator’s guide	41
5.1.1	Scan types	41
5.1.2	Scope and speed	41
5.1.3	NSE — nmap’s script engine	41
5.2	Speed at scale	41
5.3	Firewall evasion — understand the toolbox	41
5.4	Per-service enumeration playbooks	42
5.4.1	FTP (21)	42
5.4.2	SSH (22)	42
5.4.3	Telnet (23)	42
5.4.4	SMTP (25)	42
5.4.5	DNS (53)	42
5.4.6	HTTP/S (80,443,8080)	42
5.4.7	SMB (139,445) — the Windows treasure chest	42
5.4.8	SNMP (161/udp) — the forgotten leak	42
5.4.9	LDAP (389)	43
5.4.10	NFS (2049)	43
5.4.11	Databases	43
5.4.12	VNC (5900)	43
5.4.13	RDP (3389)	43
5.4.14	Special finds	43
5.5	Vulnerability scanners	43
5.6	From scan to attack plan — the worksheet	43
5.7	Defender’s corner	43
5.8	Deep dive — enumeration mastery	45
5.8.1	When nmap lies (and how to see through it)	45
5.8.2	Service-version-to-exploit mapping	45
5.8.3	The enumeration pyramid — do it in order	45
5.8.4	Niche services — the long tail	45
5.8.5	Scripting the enum loop	45
5.9	Reading the output — nmap in your terminal	46
5.10	rustscan — the fast discovery	46
5.11	What each nmap result means	46
6	Exploitation & Metasploit	46
6.1	The mental model	46
6.2	msfconsole — the operator’s surface	47
6.3	Databases — organize real engagements	47
6.4	Five walkthroughs — the same muscle, five doors	47

6.4.1	vsftpd 2.3.4 backdoor	47
6.4.2	distccd RCE	47
6.4.3	UnrealIRCd backdoor	47
6.4.4	Tomcat manager (default creds → RCE)	47
6.4.5	Samba usermap	47
6.5	Meterpreter — the super shell	48
6.6	multi/handler — catch every shell	48
6.7	msfvenom — craft every payload	48
6.8	Post-exploitation modules — loot systematically	48
6.9	Resource scripts — automate the routine	48
6.10	Defender's corner	49
6.11	Deep dive — Metasploit mastery	50
6.11.1	Module internals — what use actually loads	50
6.11.2	Resource scripts — automate everything	50
6.11.3	Meterpreter power commands — beyond the basics	50
6.11.4	Session management at scale	50
6.11.5	MSF vs manual — know when each wins	50
6.11.6	MSFvenom advanced — every format	51
6.12	Reading the output — Metasploit in your terminal	51
6.13	msfvenom — the payload factory	51
6.14	meterpreter — the post-exploitation swiss knife	51
7	Manual Exploitation	52
7.1	The manual exploitation loop	52
7.2	searchsploit — exploit-db offline	52
7.3	Reading a PoC safely	52
7.4	Fixing the five classic breakages	52
7.4.1	1. Python 2 vs 3	52
7.4.2	2. Hardcoded offsets	52
7.4.3	3. Wrong IPs/paths	52
7.4.4	4. Missing dependencies	52
7.4.5	5. Timing/encoding	53
7.5	Exploiting without any PoC	53
7.6	Protocol-level exploitation — raw by hand	53
7.7	Compile-and-fire (C exploits)	53
7.8	When it still fails — the debug checklist	53
7.9	Defender's corner	53
7.10	Deep dive — exploitation without a framework	54
7.10.1	Manual service exploitation — the walkthroughs	54
7.10.2	Exploit anatomy — reading a PoC	54
7.10.3	When the version has no CVE	54
7.10.4	Exploit chaining — one bug is never enough	54
7.10.5	The exploitation notebook — document as you go	54
8	Web Exploitation — The Mega-Chapter	55
8.1	Burp Suite — your web weapon, fully loaded	55
8.2	How web apps break — the one idea	55
8.3	SQL Injection — read & write the database	55
8.3.1	Detection	55
8.3.2	Auth bypass classics	55
8.3.3	UNION extraction — dump by hand	55
8.3.4	Blind SQLi — nothing prints, still exploitable	56
8.3.5	sqlmap — automate the entire class	56

8.4	Cross-Site Scripting — run JavaScript in the victim’s browser	56
8.4.1	The three species	56
8.4.2	Payload grammar — one for every context	56
8.4.3	Filter bypasses	56
8.4.4	Beyond alert(1) — real impact	57
8.4.5	Blind XSS & XSS hunters	57
8.5	CSRF, Clickjacking & CORS — the confused deputy trio	57
8.5.1	CSRF — forge the victim’s request	57
8.5.2	Clickjacking — iframe tricks	57
8.5.3	CORS misconfig — cross-origin data theft	57
8.6	IDOR & Broken Access Control — the #1 beginner bounty	57
8.6.1	Direct object reference	57
8.6.2	Access-control depth	57
8.7	SSRF — make the server your proxy	58
8.7.1	The attack set	58
8.7.2	Bypass the filter	58
8.8	XXE — XML parser as file reader	58
8.9	SSTI — template injection to RCE	58
8.10	File Upload — upload yourself a shell	58
8.10.1	Bypass arsenal	58
8.11	File Inclusion & Path Traversal	59
8.11.1	Path traversal — read anything	59
8.11.2	PHP wrappers — the LFI superpower	59
8.11.3	Log poisoning → RCE	59
8.12	Command Injection — user input hits the shell	59
8.13	Deserialization — object injection	59
8.14	Authentication & Session Attacks — the complete menu	60
8.14.1	JWT — the signature that isn’t	60
8.14.2	OAuth — the flow bugs	60
8.15	Advanced classes — know they exist	60
8.15.1	HTTP request smuggling	60
8.15.2	Race conditions	60
8.15.3	Web cache poisoning	60
8.15.4	Prototype pollution	60
8.15.5	Mass assignment	60
8.16	API Exploitation — the modern surface	60
8.17	The grab-bag — smaller classes that still pay	61
8.18	The web methodology — every target, every time	61
8.19	Defender’s corner — the unified fix	61
8.20	Deep dive — the web exploitation craft	62
8.20.1	Burp mastery — the pro workflow	62
8.20.2	Input vectors — where bugs hide	62
8.20.3	The one-rule methodology	62
8.20.4	WAF bypass — the philosophy	62
8.20.5	Blind techniques — when nothing comes back	62
8.20.6	Reading JavaScript like a hunter	62
8.20.7	Web service fingerprinting deeper	63
8.20.8	The API attack checklist	63
8.21	Deep dive II — the frontier classes	63
8.21.1	HTTP request smuggling — deeper	63
8.21.2	Race conditions — beyond the theory	63
8.21.3	Web cache poisoning & deception	63
8.21.4	Server-side request forgery — expanded	63

8.21.5	Deserialization — deeper	64
8.21.6	Prototype pollution — deeper	64
8.21.7	WebSocket attacks	64
8.21.8	The 60-second web check	64
8.22	The web attack map — where every vuln lives	64
8.23	HTTP request anatomy — every attack surface	64
8.24	The SQLi pipeline — a worked view	65
8.25	XSS — the three faces, visualized	65
8.26	Reading the output — web tools in your terminal	66
8.27	ffuf — finding hidden paths	66
8.28	Burp-style interception — what a request looks like	66
9	Password Attacks & Credential Access	66
9.1	The two battlefields	66
9.2	Know your hashes	66
9.3	Online attacks — the surgical art	67
9.3.1	hydra — the workhorse	67
9.3.2	Web-form precision — Burp Intruder	67
9.3.3	Password spraying — the stealth move	67
9.4	Offline cracking — the real power	67
9.4.1	john the ripper	67
9.4.2	hashcat — GPU warfare	67
9.5	Attack modes — beyond plain dictionary	68
9.5.1	Rules — mutate every word	68
9.5.2	Masks — when you know the shape	68
9.5.3	Combinator & hybrid	68
9.6	Wordlist engineering — the real craft	68
9.7	Hash capture — get the material	68
9.8	Pass-the-hash — skip cracking entirely	69
9.9	Defender's corner	69
9.10	Deep dive — credential attacks at scale	70
9.10.1	The spray-vs-stuff-vs-brute decision	70
9.10.2	Advanced hashcat — masks + rules + hybrid	70
9.10.3	The cracking methodology — order of operations	70
9.10.4	Wordlist engineering — the pro layer	70
9.10.5	john's hidden power	70
9.10.6	Kerberoasting / AS-REP / NTLMv2 — the Windows trinity	70
9.10.7	Online attack hygiene — don't get caught	71
9.11	Reading the output — password tools	71
9.12	hashcat — the crack	71
10	Shells & Payloads	71
10.1	Reverse vs bind — pick deliberately	71
10.2	Listeners — catch the call	71
10.3	Reverse shell one-liners — the arsenal	71
10.4	Web shells — when HTTP is the channel	72
10.5	Stabilize — junk shell to real shell	72
10.6	Transfer files — tools in, loot out	72
10.7	When your listener can't be reached	73
10.8	msfvenom payloads — the complete reference	73
10.9	Defender's corner	73
10.10	Deep dive — payload craftsmanship	74
10.10.1	The staged vs stageless decision	74

10.10.2	Full TTY upgrade — every method	74
10.10.3	Transfer methods — the exhaustive list	74
10.10.4	Exfiltration — get data out quietly	74
10.10.5	Bind shell usage — when reverse won't work	74
10.10.6	The payload-delivery cheat map	74
10.11	Reading the output — shells in your terminal	75
10.12	The file transfer — two terminals at once	75
11	Linux Privilege Escalation — The Encyclopedia	75
11.1	The method — enumerate before you exploit	75
11.2	The wins — every class, every recipe	76
11.2.1	SUID binaries → GTFOBins	76
11.2.2	sudo misconfigurations	76
11.2.3	Writable cron jobs	76
11.2.4	PATH hijacking (scripts calling bare commands)	76
11.2.5	Library hijacking	76
11.2.6	Writable /etc/passwd & friends	76
11.2.7	NFS no_root_squash	76
11.2.8	Capabilities — the quieter SUID	77
11.2.9	Kernel exploits — last resort, noisy	77
11.2.10	Containers & docker — escape when you see them	77
11.2.11	Cleartext credentials everywhere	77
11.3	Loot like a pro — you're root, now prove it	77
11.4	The full checklist — in order	77
11.5	Defender's corner	77
11.6	Deep dive — the edge cases	79
11.6.1	Capabilities — the modern SUID	79
11.6.2	Wildcard injection — argument abuse	79
11.6.3	PATH / library hijacking deeper	79
11.6.4	Docker / container escapes (from inside)	79
11.6.5	Sudo version exploits	79
11.6.6	Writable /etc/passwd variants	79
11.6.7	Systemd service abuse	79
11.6.8	The "unfindable" vectors	79
11.7	Reading the output — privesc in your terminal	80
11.8	linpeas — the checklist	80
12	Windows Privilege Escalation — The Encyclopedia	80
12.1	The method — same shape, different files	80
12.2	The wins — every class, every recipe	80
12.2.1	Token privileges — the potato family	80
12.2.2	Service attacks — the four classics	81
12.2.3	Registry & autoruns	81
12.2.4	Scheduled tasks	81
12.2.5	AlwaysInstallElevated	81
12.2.6	Unattended install files	81
12.2.7	Stored credentials	81
12.2.8	DLL hijacking	82
12.2.9	Kernel exploits (Windows)	82
12.3	meterpreter shortcuts	82
12.4	The full checklist — in order	82
12.5	Defender's corner	82
12.6	Deep dive — Windows privesc edge cases	83

12.6.1	Potato deep-dive — which one when	83
12.6.2	Service exploitation — the full workflow	83
12.6.3	Registry / scheduled task hunting	83
12.6.4	DLL hijacking — the quiet path	83
12.6.5	Registry credential mines	83
12.6.6	Windows credential stores	83
12.6.7	Print Nightmare / Spooler vector	83
12.6.8	The Windows loot run — you're SYSTEM	84
12.7	Reading the output — Windows privesc	84
12.8	winpeas — what to look for	84
12.9	mimikatz-lite — the credential dump	84
13	Active Directory — The Complete Kill Chain	84
13.1	The mental model — AD is a trust graph	84
13.2	Phase 1 — Foothold (get any domain cred)	85
13.2.1	Password spraying	85
13.2.2	Username enumeration	85
13.2.3	AS-REP roasting — creds without creds	85
13.2.4	LLMNR/NTLM poisoning — Responder	85
13.3	Phase 2 — Enumerate the domain (with any cred)	85
13.4	Phase 3 — The named attacks	85
13.4.1	Kerberoasting	86
13.4.2	Pass-the-hash / pass-the-ticket	86
13.4.3	Delegation abuse	86
13.4.4	ACL abuse — the quiet paths	86
13.4.5	DCSync — the crown jewels	86
13.4.6	Silver tickets	86
13.5	Phase 4 — Lateral movement	86
13.6	Phase 5 — Persistence in AD (authorized only)	87
13.7	Lab reality — how to practice	87
13.8	Defender's corner	87
13.9	Deep dive — AD tradecraft	88
13.9.1	BloodHound — read the graph	88
13.9.2	Delegation attacks — the three flavors	88
13.9.3	Kerberos deep — the attack primitives	88
13.9.4	NTLM relay — live credential forwarding	88
13.9.5	Trust abuse — cross-domain attacks	88
13.9.6	Post-DA — what do you DO as Domain Admin	88
13.9.7	AD enum without BloodHound — the manual path	89
13.10	The AD kill chain — the whole attack at once	89
13.11	Kerberos — the ticket flow	89
13.12	Reading the output — AD in your terminal	90
13.13	responder — the hash catch	90
13.14	secretsdump — the domain loot	90
14	Wireless Attacks	90
14.1	Hardware — the one thing that must be right	90
14.2	Recon the airspace	90
14.3	WPA2 handshake attack — the classic	91
14.4	PMKID — no clients required	91
14.5	WPS attacks — the legacy backdoor	91
14.6	Evil twin — the human-layer attack	91
14.7	bettercap — WiFi + LAN in one tool	91

14.8	Enterprise WiFi (WPA2-Enterprise / 802.1X)	92
14.9	Crack efficiently	92
14.10	Wireless defense checklist	92
14.11	Deep dive — wireless tradecraft	93
14.11.1	Handshake hunting — efficiency tips	93
14.11.2	WPA3 — what changes	93
14.11.3	Evil twin + captive portal — full anatomy	93
14.11.4	WPS deep — the three attack types	93
14.11.5	bettercap — the wireless Swiss knife	93
14.11.6	Hidden SSID recovery	93
14.11.7	Antenna & positioning — the physical layer	93
14.12	Reading the output — WiFi in your terminal	94
14.13	aircrack — the crack	94
15	MITM & Network Attacks	94
15.1	The interception position	94
15.2	ARP spoofing — the classic MITM	94
15.3	Sniff the payload	94
15.4	SSL/TLS interception — the honest picture	95
15.5	DHCP attacks	95
15.6	Responder — LLMNR/NBT-NS poisoning	95
15.7	SNMP & network-device attacks	95
15.8	IPv6 MITM — the forgotten vector	95
15.9	Defender's corner	95
15.10	Deep dive — interception tradecraft	96
15.10.1	ettercap — the MITM workbench	96
15.10.2	DNS spoofing — redirect the victim	96
15.10.3	Responder deep — WPAD + poisoning	96
15.10.4	Sniffing specifics — what plaintext still leaks	96
15.10.5	SSLStrip & HSTS — the modern reality	96
15.10.6	DHCP starvation & rogue DHCP	96
15.10.7	STP/CDP & switch attacks — layer 2 depth	96
15.11	Reading the output — MITM in your terminal	97
15.12	responder — more catches	97
15.13	ARP spoofing — the MITM in one picture	97
15.14	The interception points — what's readable	97
16	Pivoting & Tunneling	98
16.1	The scenario	98
16.2	SSH tunneling — the everyday tool	98
16.3	Chisel — the modern tunnel	98
16.4	sshuttle — poor man's VPN	98
16.5	ligolo-ng — the current favorite	98
16.6	socat — the swiss-army relay	98
16.7	Metasploit pivoting	99
16.8	Multi-hop — through A into B into C	99
16.9	Discovery from the inside	99
16.10	Data exfiltration channels	99
16.11	Persistence (authorized engagements only)	99
16.12	Choosing the tunnel — decision tree	99
16.13	Defender's corner	100
16.14	Deep dive — tunneling tradecraft	101
16.14.1	Every tunnel type, mapped	101

16.14.2	Chisel — the complete reference	101
16.14.3	ligolo-ng — real networking	101
16.14.4	SSH tricks — the full surface	101
16.14.5	The pivot decision tree — expanded	101
16.14.6	Exfiltration channels — the sneaky list	101
16.14.7	Multi-hop discipline	102
16.15	Pivoting — the topology that matters	102
16.16	The multi-hop chain — real engagements	102
16.17	Reading the output — pivoting in your terminal	103
16.18	proxychains — routing your tools through	103
16.19	sshuttle — the lazy VPN	103
17	Exploit Development — From Crash to Shell	103
17.1	The x86 you actually need	103
17.2	The workflow — 6 steps, always	104
17.3	Step 1 — Fuzz to crash	104
17.4	Step 2 — Control EIP exactly	104
17.5	Step 3 — Find JMP ESP	104
17.6	Step 4 — Badchars	104
17.7	Step 5+6 — Shellcode & landing	104
17.8	Protections you'll meet	105
17.9	Practice targets	105
17.10	Defender's corner	105
17.11	Deep dive — beyond the vanilla overflow	106
17.11.1	SEH overwrites — when the stack is protected	106
17.11.2	Egghunters — when the buffer is tiny	106
17.11.3	The pwntools workflow — modern exploit dev	106
17.11.4	GDB for exploit dev — the essentials	106
17.11.5	ROP — the DEP answer	106
17.11.6	Format string — the sibling vuln	106
17.11.7	Fuzzing at scale	106
17.12	The stack overflow — visualized	107
17.13	The exploit dev workflow — the loop	107
17.14	Reading the output — exploit dev in your terminal	108
17.15	pattern_offset — finding the exact spot	108
17.16	The full exploit — the winning command	108
18	Evasion Concepts — Playing Cat and Mouse	108
18.1	How you're actually detected	108
18.2	Obfuscation — beat the signature	109
18.3	Living off the land — use what's trusted	109
18.4	AMSI & script-host evasion — concepts	109
18.5	In-memory & fileless	109
18.6	Network evasion — look like normal	109
18.7	Persistence without alerts	109
18.8	OPSEC — the operator's rules	109
18.9	Defender's corner — the whole point	110
18.10	Deep dive — the cat-and-mouse detail	111
18.10.1	The detection stack — layer by layer	111
18.10.2	Payload delivery under inspection	111
18.10.3	AMSI/ETW — the concepts	111
18.10.4	Process injection — the patterns	111
18.10.5	Network-level evasion	111

18.10.6 The OPSEC discipline	111
18.10.7 Purple-teaming it	112
19 Social Engineering & Phishing	112
19.1 Why it works	112
19.2 The campaign lifecycle	112
19.3 Email phishing — the toolkit	112
19.3.1 GoPhish — the professional framework	112
19.3.2 The SET quick path	112
19.3.3 Crafting the phish	112
19.4 Malicious documents & droppers (awareness)	113
19.5 Pretexting & vishing — the human art	113
19.6 Physical SE — beyond email	113
19.7 Smishing & quishing	113
19.8 Awareness & detection — the defense	113
19.9 Red team vs awareness — two different missions	113
19.10 Deep dive — the SE operator’s detail	114
19.10.1 Domain & infra — believable phishing	114
19.10.2 The phish that works — anatomy	114
19.10.3 Landing page cloning	114
19.10.4 Pretexting — the psychology framework	114
19.10.5 Vishing — voice scripts that work	114
19.10.6 Physical — beyond the inbox	114
19.10.7 The measurement layer — what reports track	115
20 Cloud Attacks — AWS-First	115
20.1 The cloud attack surface — a new shape	115
20.2 SSRF → metadata → keys — the crown jewel chain	115
20.3 S3 buckets — the classic exposure	115
20.4 Exposed keys — where they leak	115
20.5 Compromised keys — what can you do?	116
20.6 Lateral & persistence in cloud	116
20.7 Serverless & container surfaces	116
20.8 Multi-cloud notes	116
20.9 Defender’s corner	116
20.10 Deep dive — cloud tradecraft	117
20.10.1 AWS enumeration — the full surface	117
20.10.2 IAM privesc paths — the catalog	117
20.10.3 SSRF → metadata — the full chain again	117
20.10.4 S3 attacks — the full list	117
20.10.5 Beyond AWS — Azure & GCP	117
20.10.6 Container escape → cloud (the bridge)	117
20.11 The SSRF-to-cloud chain — the attack that matters	118
20.12 The cloud attack surface — what’s exposed	118
20.13 Reading the output — cloud in your terminal	119
20.14 Using the stolen keys	119
21 Bug Bounty Methodology	119
21.1 Pick your battlefield	119
21.2 The recon pipeline at scale	119
21.3 Bug classes that pay — ranked	120
21.4 The manual-hunter edge	120
21.5 Automation & monitoring — hunt while you sleep	120
21.6 Report writing — the payout multiplier	120

21.7	Triage psychology	121
21.8	Defender's corner — running a program	121
21.9	Deep dive — the bounty craft	122
21.9.1	Program selection — the meta-game	122
21.9.2	The automation setup	122
21.9.3	Bug-class depth — what actually pays in practice	122
21.9.4	The report that pays — formula	122
21.9.5	Triage survival	122
21.9.6	The disclosure timeline	122
22	Reporting & Career Path	122
22.1	Pentest report anatomy	123
22.2	A real finding, written properly	123
22.3	Build the proof-of-work portfolio	123
22.4	Certifications — the honest map	123
22.5	The 12-month roadmap	123
22.6	The habits that compound	123
22.7	Deep dive — the career operator	125
22.7.1	Portfolio that gets interviews	125
22.7.2	The job-hunt skill	125
22.7.3	The path decision — offense, defense, or both	125
22.7.4	The seniority ladder — what changes	125
23	Cryptography for Hackers	125
23.1	Encoding vs hashing vs encryption — the eternal confusion	125
23.2	Hash attacks — beyond cracking	126
23.2.1	Length extension — forge signed data	126
23.2.2	Collision & weak hashes	126
23.2.3	Salt-free hashes	126
23.3	Encryption failure modes — the exploitable patterns	126
23.3.1	ECB mode — the penguin problem	126
23.3.2	CBC bit-flipping & padding oracle	126
23.3.3	Key reuse & weak randomness	126
23.3.4	IV/nonce reuse & hardcoded keys	126
23.4	TLS — what to check	126
23.5	JWT — crypto edition	127
23.6	Homebrew crypto — the universal rule	127
23.7	Steganography — data in files	127
23.8	Defender's corner	127
23.9	Deep dive — crypto attack patterns	128
23.9.1	The encoding-vs-security trap — real examples	128
23.9.2	ECB oracle — the exploit pattern	128
23.9.3	CBC bit-flipping — byte surgery	128
23.9.4	Padding oracle — decrypt without the key	128
23.9.5	Hash length extension — forge signatures	128
23.9.6	Randomness failures — predictable everything	128
23.9.7	JWT — the crypto deep-cut	128
23.10	ECB vs CBC — the visual difference	129
23.11	The crypto attack map — where each lives	129
24	Mobile App Hacking — Android	130
24.1	The setup	130
24.2	Decompile & static analysis	130
24.3	The traffic layer — same as web	130

24.4	Mobile-specific vuln classes	130
24.5	Frida — the runtime scalpel	131
24.6	iOS — the parallel world	131
24.7	MobSF — automated triage	131
24.8	Defender's corner	131
24.9	Deep dive — mobile tradecraft	132
24.9.1	The static analysis checklist	132
24.9.2	Dynamic analysis — frida recipes	132
24.9.3	Insecure data storage — where creds hide	132
24.9.4	WebView & IPC attacks	132
24.9.5	Common mobile findings	132
24.9.6	iOS parallels	132
25	Container & Kubernetes Hacking	133
25.1	Am I in a container?	133
25.2	Container escapes — the catalog	133
25.2.1	Docker socket mounted	133
25.2.2	Privileged container	133
25.2.3	Excessive capabilities	133
25.2.4	Writable host mounts & kernel vulns	133
25.3	Kubernetes — the orchestrator attack surface	133
25.3.1	Recon inside a pod	133
25.3.2	Kubelet & API exposure	134
25.3.3	kube-hunter / peirates	134
25.4	Registry & image attacks	134
25.5	Defender's corner	134
25.6	Deep dive — container & k8s tradecraft	135
25.6.1	Escape vectors — the complete catalog	135
25.6.2	Kubernetes attack paths	135
25.6.3	RBAC abuse — privilege escalation in k8s	135
25.6.4	Supply chain & registry attacks	135
25.6.5	The full escape chain — a realistic sequence	135
26	IoT & Hardware Hacking — Concepts	136
26.1	Why IoT falls	136
26.2	Firmware analysis — the software side	136
26.3	Physical access — UART/JTAG	136
26.4	Radio attacks — the RF layer	136
26.5	Embedded Linux — the usual suspects	136
26.6	Automotive & CAN bus — concept	136
26.7	Defender's corner	136
27	Forensics & Incident Response	138
27.1	The IR mindset — attacker's mirror	138
27.2	The evidence pyramid	138
27.3	Timeline analysis — reconstruct the attack	138
27.4	Log forensics — read the story	138
27.5	Persistence hunting — find the implant	138
27.6	Memory forensics — Volatility	139
27.7	Network forensics — the wire remembers	139
27.8	The IR playbook — respond to a breach	139
27.9	Attacker's counter-forensics — knowing your trail	139
27.10	Defender's corner	139
27.11	Deep dive — the forensic operator	140

27.11.1	Evidence acquisition — do it right	140
27.11.2	Timeline forensics — rebuild the attack	140
27.11.3	Log deep-dive — the real transcript	140
27.11.4	Persistence hunting — every hiding spot	140
27.11.5	Memory forensics — the implant hunt	140
27.11.6	Anti-forensics — what attackers do	140
28	Capstone Walkthroughs — Full Chains, End to End	141
28.1	Walkthrough 1 — Metasploitable 2 (full multi-vector)	141
28.1.1	Recon	141
28.1.2	Vector A — vsftpd backdoor	141
28.1.3	Vector B — UnrealIRCd	141
28.1.4	Vector C — Samba usermap	141
28.1.5	Post — loot	141
28.2	Walkthrough 2 — DVWA (the web ladder)	141
28.2.1	Low — command injection to shell	141
28.2.2	Medium — filtering	141
28.2.3	High — whitelist	142
28.3	Walkthrough 3 — A realistic AD foothold (concept chain)	142
28.4	Walkthrough 4 — Cloud SSRF chain	142
28.5	The pattern — every chain is the same	142
28.6	Deep dive — two more chains	143
28.6.1	Walkthrough 5 — LFI to root (the classic web-to-system)	143
28.6.2	Walkthrough 6 — File upload to shell to pivot	143
28.6.3	The decision framework — what to try next	143
28.6.4	Chain patterns — recognize them	143
28.7	Walkthrough 7 — a full box, narrated	143
28.7.1	Step 0 — the setup	143
28.7.2	Step 1 — find it, map it	144
28.7.3	Step 2 — map the web app	144
28.7.4	Step 3 — the foothold	144
28.7.5	Step 4 — user.txt	144
28.7.6	Step 5 — root.txt	144
28.7.7	The debrief — what just happened	144
28.7.8	What you should take from this	145
29	Interview Prep — The Q&A Encyclopedia	146
29.1	Rapid-fire definitions	146
29.2	Explain-the-vuln answers	146
29.2.1	SQLi	146
29.2.2	XSS	146
29.2.3	IDOR	146
29.2.4	SSRF	146
29.2.5	CSRF	146
29.2.6	XXE	146
29.2.7	Buffer overflow	146
29.3	The scenario answers — say these	146
29.3.1	“You get a shell as www-data. Next?”	146
29.3.2	“How do you start a pentest?”	146
29.3.3	“Difference between red/blue/purple team?”	147
29.3.4	“What’s the biggest risk in a typical web app?”	147
29.3.5	“How do you stay legal?”	147
29.4	The depth-questions — what separates you	147

29.5	Technical deep-dives to rehearse	147
29.6	Defender's corner	147
29.7	Deep dive — the interview encyclopedia	148
29.7.1	Explain-it-like-I'm-5 — every vuln in one line	148
29.7.2	The “walk me through it” scenarios — scripted answers	148
29.7.3	The technical deep-dives they ask seniors	148
29.7.4	The questions you ask THEM	148
30	CTF Methodology — Competitive Hacking	149
30.1	The categories — what to expect	149
30.2	The beginner ladder	149
30.3	Web CTF strategy	149
30.4	Crypto CTF strategy	149
30.5	Forensics CTF strategy	149
30.6	Pwn CTF strategy	150
30.7	Reverse CTF strategy	150
30.8	Time management — the tournament skill	150
30.9	From CTF to career	150
31	Red Team Operations — Beyond the Pentest	151
31.1	Pentest vs red team — the difference	151
31.2	The engagement lifecycle	151
31.3	OPSEC — operating unseen	151
31.4	C2 — command and control concepts	151
31.5	Initial access — the real-world vectors	151
31.6	The detection game — playing against the SOC	151
31.7	Reporting a red team — the narrative	152
31.8	Career path — pentester to red teamer	152
32	Blue Team — The Defender's Craft	153
32.1	The defender's stack	153
32.2	Detection engineering — turn attacks into alerts	153
32.3	Log analysis — the SOC workflow	153
32.4	Honeypots & deception — the active defense	153
32.5	Threat hunting — proactive detection	153
32.6	Incident response — the operational side	153
32.7	Purple teaming — both sides, one goal	154
33	PowerShell for Pentesters	155
33.1	The essentials	155
33.2	Download cradles — every variant	155
33.3	Active Directory enum — PowerView-style	155
33.4	AMSI/ETW concepts — the script sandbox	155
33.5	Reverse shells — PowerShell	155
33.6	Privilege checks — PowerShell	156
34	Reverse Engineering — Understanding Binaries	157
34.1	The two approaches	157
34.2	Static triage — the first 60 seconds	157
34.3	Ghidra — the free NSA disassembler	157
34.4	Assembly — the 10 instructions that matter	157
34.5	Dynamic analysis — watching it run	157
34.6	Anti-analysis — how malware fights back	157
34.7	The CTF shortcut — reverse challenges	158

35 Malware Analysis & Threat Understanding	159
35.1 Malware families — the taxonomy	159
35.2 The kill chain — how malware works	159
35.3 Analyzing a suspicious file — the safe workflow	159
35.4 Behavioral indicators — what to watch	159
35.5 Ransomware — the dominant threat	159
35.6 Building the analysis lab	160
36 Steganography & Data Hiding	161
36.1 The hiding places	161
36.2 Image stego — the hunt	161
36.3 The appended-data trick	161
36.4 NTFS alternate data streams	161
36.5 Covert channels — exfiltration stego	161
36.6 Polyglot & file-trickery	161
36.7 The stego CTF checklist	162
36.8 Deep dive — firmware & embedded detail	163
36.8.1 Firmware acquisition — every channel	163
36.8.2 Inside a firmware image	163
36.8.3 Emulating firmware — run it locally	163
36.8.4 UART — the physical root shell	163
36.8.5 JTAG/SWD — chip-level access	163
36.8.6 Radio attacks — the RF surface	163
36.8.7 Automotive CAN — concept	163
36.8.8 The IoT security verdict	164
37 Advanced OSINT — People & Infrastructure	164
37.1 People intelligence — the full profile	164
37.2 Breach data — the legitimate use	164
37.3 Geolocation — physical intelligence	164
37.4 Infrastructure intelligence	164
37.5 Dark-web & forum OSINT — concept level	164
37.6 OSINT frameworks — automated aggregation	165
37.7 Defender's corner — OSINT self-assessment	165
38 Windows Internals for Hackers	166
38.1 The security model — SID, token, ACL	166
38.2 Privileges — the power flags	166
38.3 Sessions & tokens — the identity layer	166
38.4 Services — the attack surface	166
38.5 The registry — configuration goldmine	166
38.6 LSASS — the credential motherlode	166
38.7 Persistence — the Windows catalog	167
38.8 LOLBas — living off the land	167
39 Linux Internals for Hackers	168
39.1 The process model	168
39.2 Sockets & network internals	168
39.3 The filesystem — permissions deep	168
39.4 Namespaces — containers & isolation	168
39.5 Capabilities — fine-grained root	168
39.6 The boot process — persistence terrain	169
39.7 The kernel — the real target	169

40 Database Attacks & Exploitation	170
40.1 The attack surfaces	170
40.2 MSSQL — the deepest target	170
40.3 MySQL — the web favorite	170
40.4 PostgreSQL	170
40.5 Oracle — the enterprise beast	170
40.6 Redis — the unauthenticated gem	171
40.7 MongoDB — NoSQL	171
40.8 The DB privesc patterns	171
41 SSO, OAuth & Modern Auth Attacks	172
41.1 OAuth 2.0 — how it works	172
41.2 The OAuth attack surface	172
41.3 SAML — enterprise SSO	172
41.4 OIDC — OAuth + identity	172
41.5 Session management — the auth layer beyond login	172
41.6 MFA — the bypasses that still work	172
41.7 Password reset — the forgotten-auth attack	173
41.8 Defender’s corner	173
41.9 OAuth flow — where each attack lands	174
42 API Security Deep Dive	175
42.1 API reconnaissance	175
42.2 The OWASP API Top 10 — attack-mapped	175
42.3 GraphQL — the modern API	175
42.4 REST patterns — the attack checklist	175
42.5 API auth — the common failures	175
42.6 Fuzzing APIs at scale	176
42.7 Defender’s corner	176
43 Phishing Infrastructure & Campaign Craft	177
43.1 Domain strategy	177
43.2 Mail infrastructure	177
43.3 Landing page engineering	177
43.4 Tracking & campaign management	177
43.5 The phish that works — psychology	177
43.6 Attachments & payloads	178
43.7 Campaign reporting — what clients need	178
44 Advanced Exploitation — Heap, Format Strings, Kernel	179
44.1 Heap overflows — the modern bug	179
44.2 Format string attacks — the forgotten class	179
44.3 Use-after-free — the dangling pointer	179
44.4 Race conditions in code — TOCTOU	179
44.5 Integer overflows — the subtle corruption	179
44.6 Kernel exploitation — the deepest layer	179
44.7 Browser exploitation — the modern frontier	180
44.8 The exploit dev roadmap	180
45 Persistence & Post-Exploitation Mastery	181
45.1 Persistence — the goal & the ethics	181
45.2 Linux persistence — every hiding spot	181
45.3 Windows persistence — every hiding spot	181
45.4 Access tokens & credential replay	181

45.5 Data collection — the objective layer	182
45.6 Domain persistence — the AD crown jewels	182
45.7 Cleanup — the professional obligation	182
46 Advanced Network Attacks	183
46.1 IPv6 attacks — the shadow protocol	183
46.2 Switch attacks — layer 2 depth	183
46.3 Routing protocol attacks — layer 3	183
46.4 DHCP attacks — control the gateway	183
46.5 ARP deep — the MITM mechanics	183
46.6 DNS attacks — the name layer	183
46.7 NetBIOS / LLMNR — the Windows poison	184
46.8 Wireless network attacks — beyond WPA	184
46.9 Network segmentation evasion — crossing boundaries	184
47 Certification & Exam Strategy	185
47.1 The cert landscape — what each proves	185
47.2 OSCP — the exam that defines careers	185
47.3 The study method — how people actually pass	185
47.4 eJPT / PNPT — the practical entry points	185
47.5 Exam-day tactics — the performance layer	185
47.6 Beyond the cert — the real skill	186
48 Four More Chains — Narrated	187
48.1 Chain 1 — Web-only box (no system shell needed)	187
48.2 Chain 2 — the AD box	187
48.3 Chain 3 — WiFi to internal	187
48.4 Chain 4 — cloud from a web bug	187
48.5 The universal pattern — one more time	188
Appendix A — The Master Commands Wall	189
Appendix B — Wordlists & seclists Map	190
Appendix C — Ports & Protocols	190
Appendix D — Lab & Practice Directory	190
Appendix E — Hashcat Modes & John Formats	191
Appendix F — Privesc One-Page Checklists	191
Appendix G — Staying Legal	191
Appendix H — Glossary	192
Appendix I — Per-Service Attack Cheat Sheets	192
Appendix J — Capstone Lab Index	194
Appendix K — The Tools Quick-Reference	195
Appendix L — The 50 Mistakes Beginners Make	197
Appendix M — Web Attack Payload Bank	198
Appendix N — Protocol & Concept Deep-Dives	200
Appendix O — Real-World Breach Case Studies	202

Appendix P — Pre-Engagement & Hardening Checklists	203
Appendix Q — The One-Liner Arsenal	204
Appendix R — Module-by-Module Lab Exercises	207
Appendix S — Report Templates	209
Appendix T — Extended Glossary	211
Appendix U — Troubleshooting	212
Appendix V — Resources	214
Appendix W — Legal & Ethical Handbook	215
Appendix X — Reading Tool Output	217
Appendix Y — Module Self-Checks	219
Appendix Z — Topic Index	221
Appendix AB — What It Actually Looks Like	222
48.6 nmap — the scan	223
48.7 Metasploit — the exploit	223
48.8 sqlmap — the injection	224
48.9 ffuf — the hidden paths	224
48.10hydra — the brute force	225
48.11hashcat — the offline crack	225
48.12responder — the hash catch	226
48.13linpeas — the privesc checklist	226
48.14airodump + aircrack — the wifi crack	227
48.15crackmapexec — the domain spray	227
48.16meterpreter — post-exploitation	228
48.17AWS — the cloud chain	228
48.18chisel — the tunnel	228
48.19john — the hash cracker	229
48.20gobuster — the dir bruteforce	229
48.21wpscan — the CMS audit	229
48.22searchsploit — the exploit database	230
48.23enum4linux — the SMB enum	230
48.24msf handler — catching the shell	230
48.25docker.sock — the container escape	231
48.26tcpdump — the plaintext catch	231
Appendix AA — Every Term in One Line	231
Appendix AC — Attack to Tools Matrix	233

How to use this book

- **Read with a terminal open.** Every command block is meant to be typed, not skimmed. Reading about nmap teaches nothing; running it does.
- **Do the “Try it” exercises.** They are the difference between finishing a book and gaining a skill.
- **Follow the order** if you’re new; modules build on each other.
- **Take notes as you hack** — commands tried, output seen, ideas for later. Every professional keeps notes.
- **When stuck: enumerate more.** The answer is almost always a port, path, or parameter you haven’t examined yet.

Choose your reading path

Total beginner	Read straight through — modules are ordered for you
CTF player	Skim 1–3, study 5–10 deeply, use appendices during boxes
Bug bounty hunter	Modules 4–5, 7–8, 14, 21 first; the rest as needed
Job / interview	Modules 3, 15, 22 + every “Defender’s corner”

Module map

1	Build Your Hacking Lab	VMs, targets, isolation, snapshots
2	Linux & Terminal Mastery	commands, permissions, pipes, scripting
3	Networking for Hackers	TCP/IP, ports, DNS, HTTP, Wireshark
4	Reconnaissance & OSINT	subdomains, dorks, metadata, cloud, people
5	Scanning & Enumeration	nmap deep-dive, every service playbook
6	Exploitation & Metasploit	modules, payloads, meterpreter, handlers
7	Manual Exploitation	searchsploit, PoC triage, porting exploits
8	Web Exploitation	20+ vuln classes with payloads and labs
9	Password Attacks	online vs offline, john, hashcat, wordlists
10	Shells & Payloads	reverse/bind, stabilize, transfer, exfil
11	Linux Privilege Escalation	the full checklist, SUID to kernel
12	Windows Privilege Escalation	services, registry, tokens, potatoes
13	Active Directory	the complete kill chain, BloodHound
14	Wireless Attacks	WPA2, PMKID, WPS, evil twin, defense
15	MITM & Network Attacks	ARP spoof, sniffing, interception
16	Pivoting & Tunneling	SSH, chisel, sshuttle, ligolo, autoroutes
17	Exploit Development	stack, fuzzing, buffer overflow end-to-end
18	Evasion Concepts	AV/EDR internals, obfuscation, LOLBins
19	Social Engineering	phishing campaigns, pretexting, awareness
20	Cloud Attacks	AWS metadata, IAM, S3, misconfigurations
21	Bug Bounty Methodology	pipeline at scale, bug classes, reports
22	Reporting & Career	finding format, portfolio, certs, roadmap
23	Cryptography for Hackers	encoding vs crypto, ECB, oracles, keys
24	Mobile App Hacking	APK decompile, pinning, frida, storage
25	Containers & Kubernetes	escapes, docker.sock, k8s attacks
26	IoT & Hardware	firmware, UART/JTAG, radio, CAN
27	Forensics & IR	timelines, logs, persistence, memory
28	Capstone Walkthroughs	4 full chains, recon to root
29	Interview Prep	rapid-fire Q&A, scenario answers
30	CTF Methodology	categories, strategy, tournaments
31	Red Team Ops	objectives, OPSEC, C2, detection game
32	Blue Team	detection, SOC, hunting, purple team
33	PowerShell	download cradles, AD enum, shells
34	Reverse Engineering	static/dynamic, ghidra, assembly
35	Malware Analysis	families, triage, behavior
36	Steganography	LSB, metadata, covert channels
37	Advanced OSINT	people, breaches, geolocation
38	Windows Internals	tokens, privileges, services

39	Linux Internals	proc, namespaces, capabilities
40	Database Attacks	MSSQL/MySQL/Redis/Mongo RCE
41	SSO & OAuth	federation attacks, MFA bypass
42	API Security	OWASP API top 10, GraphQL
43	Phishing Ops	domains, mail infra, GoPhish
44	Advanced Exploits	heap, format strings, kernel
45	Persistence	every mechanism + cleanup
46	Advanced Network	IPv6, L2, routing, VLAN
47	Cert & Exam Strategy	OSCP/eJPT/PNPT prep
48	More Walkthroughs	web/AD/wifi/cloud chains narrated

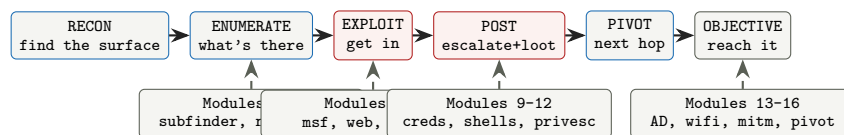
Before you start — prerequisites & difficulty

Easy	1–3, 4–5, 9–10	Foundations — anyone can start here
Medium	6–8, 11–12, 14–16, 19–21	Core pentest — needs Modules 1–5
Advanced	13, 17–18, 20, 22–30	Deep — needs Modules 1–10 solid

Modules build in order. Module 8 needs 5; 11–12 need 6, 10; 13 needs 12; 17 needs 2, 10; 20 needs 8. Stuck? Reread the foundation first.

Free lab platforms used throughout: TryHackMe, HackTheBox, picoCTF, OverTheWire, PortSwigger Web Security Academy, VulnHub, Metasploitable 2, DVWA, OWASP Juice Shop, bWAPP, GOAD.

The attack lifecycle — the whole book in one picture



Every engagement follows this loop. Each stage feeds the next — the modules are ordered to match it.

Module 0 — Read This First

In this module: The absolute-beginner primer. If you've never touched a terminal or don't know what a "shell" is, start here. Every concept you'll meet, in one line each.

Authorized use only: This is a skill that takes reps, not reading. You will not understand it by skimming — you understand it by doing the commands on a lab machine. The book is a do-along course.

0.1 What "ethical hacking" actually is

In plain words: Hackers find ways into systems. Ethical hackers do it with permission, to find weaknesses before criminals do — and write reports that fix them. That's the whole job. Every module in this book teaches one part of it.

0.2 The 12 concepts you MUST know first

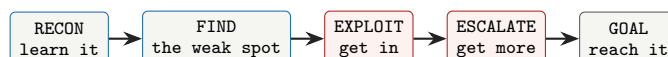
```
terminal/shell: the text interface where you type commands - your weapon
command:       an instruction you type (nmap, ls, curl)
target:        the machine you're attacking (a lab VM at first)
payload:       the code that runs on the target after you break in
exploit:       the technique that breaks in (the "how")
vulnerability: the weakness that lets you break in (the "why")
port:          a numbered door on a machine (22=ssh, 80=web, 445=smb)
service:       the program answering on a port (apache, mysql)
IP address:    a machine's address on a network (192.168.1.5)
protocol:      the language machines speak (HTTP, DNS, SMB)
hash:          a scrambled password - steal it, crack it
root/admin:    the most powerful account - the goal
```

If these 12 make sense, you can start. Everything else builds on them.

0.3 How the commands work — the pattern

```
tool -flag TARGET -option value
tool = the program (nmap, hydra, curl)
-flag = a behavior switch (-sV = version scan)
TARGET = the thing you point it at (ip, url, domain)
-option value = extra settings (-w wordlist.txt)
# read any command as: "use TOOL to do FLAG against TARGET"
# nmap -sV target = use nmap to version-scan target
```

0.4 The attack loop — what you'll be doing



Every engagement, every module, every exercise follows this loop: learn it, find the weakness, break in, get more access, reach the goal.

0.5 What the outputs look like — read the terminal

The dark boxes are what your terminal shows. The green \$ is your prompt — type after it. The gray text is what the machine answers:

```
kali@kali:~$ nmap -sV 192.168.1.10
Starting Nmap 7.94
PORT STATE SERVICE VERSION
22/tcp open  ssh OpenSSH 7.9
80/tcp open  http Apache 2.4.38
Nmap done: 1 host up
```

The command is what you type; the lines under it are what you see. Learn to read them — the output IS the answer to your question.

0.6 Your first commands — do these right now

```
# open a terminal on your Kali VM and type:
whoami      # who are you logged in as?
ip a        # what's your IP?
ls -la      # what's in the current directory?
nmap -sn 192.168.1.0/24  # what machines are on your network?
# each command teaches you something about your system
# -> typing them = you're already learning
```

0.7 The mindset — how you actually learn

- **Type every command** — reading isn't learning, doing is
- **Break things on purpose** — the lab is disposable
- **Stuck = the lesson** — that's where you learn
- **Ask “why”** — commands without understanding are useless
- **Take notes** — you'll thank yourself in 3 months
- **Google is a skill** — every pro looks things up constantly

0.8 What “being a hacker” actually means

In plain words: Not movie-hacking. It means: understanding how systems work, finding the gaps between what they SHOULD do and what they DO, and using those gaps. The skill is curiosity + method + persistence. The tools are just how you apply it.

Try it: Do the 4 commands above right now on your Kali VM. Each one takes 5 seconds and proves you can follow along — that's all you need to start Module 1.

Module 1 – Build Your Hacking Lab

In this module: Construct a private, legal playground where you attack real vulnerable systems with zero consequences. You'll learn hypervisors, network isolation, vulnerable targets, snapshot discipline, and the note-taking setup professionals use.

1.1 Why a lab (and why everyone skips this too fast)

Every skill in this book needs a target. Your lab is where failing costs nothing, where you can run exploits that would be crimes elsewhere, and where you can snapshot before every experiment. Skipping a proper lab is how beginners “hack” their own router by accident.

1.2 Hypervisors — pick one

VirtualBox	Free, simplest, runs everywhere. Best first choice.
VMware Workstation/Player	Smoother graphics, free for personal use
UTM (Apple Silicon Mac)	QEMU-based; runs Kali ARM images
Parallels (Mac)	Paid but polished ARM virtualization

ARM note: on M-series Macs use the ARM64 Kali image — some tools (hashcat GPU work, some Windows targets) need workarounds or a cloud box.

1.3 The network — isolation is everything

Three VirtualBox network modes you must understand:

NAT	VM reaches internet through host; can't see other VMs directly
Host-only	Private LAN between host and VMs; NO internet. Your attack net.
Bridged	VM sits on your real LAN — do NOT use for target VMs

The pro setup per VM:

```
# Kali: Adapter 1 = NAT (updates) Adapter 2 = Host-only (attacks)
# Targets: Adapter 1 = Host-only ONLY (they must not reach the internet)
# VirtualBox: File -> Tools -> Network Manager -> create vboxnet0
```

Result: Kali can update itself AND attack targets; targets are trapped in a sandbox that touches nothing real.

1.4 Installing Kali

```
# Use the PREBUILT VM image, not the installer ISO:
# kali.org -> Get Kali -> Virtual Machines -> VirtualBox/VMware
# Import the .ova (or unzip the .7z for VMware), boot, login kali/kali
sudo apt update && sudo apt full-upgrade -y
sudo apt install -y kali-linux-default seclists
```

Resources: 4 GB RAM minimum (8 ideal), 2+ CPU cores, 60 GB disk, video memory maxed. Enable bidirectional clipboard + drag-drop.

1.5 Your targets — the vulnerable menagerie

Metasploitable 2	Deliberately broken Linux (sourceforge) — <code>msfadmin:msfadmin</code>
DVWA	Vulnerable PHP web app — Module 8's main lab
OWASP Juice Shop	Modern Node.js vuln app — docker one-liner below
bWAPP	100+ web vulns, PHP
Windows Server eval	For AD labs — Microsoft gives 180-day trials
VulnHub VMs	Community vulnerable machines by difficulty

```
# Juice Shop via docker on Kali (cleanest):
sudo apt install -y docker.io && sudo systemctl enable --now docker
sudo docker run -d -p 3000:3000 --name juiceshop bkimminich/juice-shop
# -> http://localhost:3000 (attack a REAL modern app safely)
# DVWA on Metasploitable-style lamp stack, or THM hosts it too
```

1.6 Full arsenal install (do it once)

```
sudo apt install -y seclists gobuster feroxbuster ffuf dirb \
enum4linux-ng nbtscan nikto sqlmap john hashcat hydra \
aircrack-ng hcxtools hcxdumptool bettercap \
crackmapexec impacket-scripts netcat-traditional rlwrap \
exploitdb webshells wafw00f sslscan dnsrecon fiercer \
theharvester sherlock sublist3r whatweb \
tcpdump wireshark ncat socat proxychains4 \
steghide binwalk exiftool foremost zsteg \
bloodhound neo4j burpsuite
# wordlist check:
ls -lh /usr/share/wordlists/rockyou.txt 2>/dev/null || \
sudo gunzip /usr/share/wordlists/rockyou.txt.gz
# pip extras:
pip3 install pwntools --break-system-packages 2>/dev/null
```

1.7 Snapshots — the undo button

```
# VirtualBox: VM -> Snapshots -> Take
# naming that keeps you sane:
# "00-fresh-kali" -> after install+update
# "01-arsenal" -> after tool install
# "02-pre-box-X" -> before attacking each new VM
```

Rule: snapshot BEFORE risky experiments, not after disaster. Restoring takes 30 seconds; reinstalling takes a day.

1.8 Your hacking journal — start now

Professionals log everything. Set up before module 2:

```
# simplest: a text file per box, structured like:
# nmap.txt enum.txt exploit.txt loot.txt notes.md
mkdir -p ~/labs/msf2 && cd ~/labs/msf2
# better: Obsidian (free) or CherryTree - searchable, screenshot-friendly
# or in-terminal: 'script -a session.log' records EVERYTHING you type+see
```

Every command you run, every result, every dead end — logged. When you write reports (Module 22) or study for interviews, this journal is gold.

1.9 Cloud lab option (advanced)

For bandwidth-heavy work or ARM Mac limits: a cheap VPS running Kali/Ubuntu in the cloud. Use it **ONLY** against targets you're authorized for — a VPS is internet-facing, so sloppy work leaves the sandbox. Firewall it (**ufw allow from YOUR_IP**), disable password SSH, and shut it down when idle.

1.10 Verify the lab end-to-end

```
ip a | grep inet          # note Kali host-only IP
nmap -sn 192.168.56.0/24   # discover all lab machines
nmap -sV <metasploitable-ip> # ~20 open ports = lab alive
ping -c2 www.google.com    # NAT adapter reaches internet
```

1.11 Defender's corner

Your lab IS a lesson in defense: **network segmentation** (targets can't escape = corporate VLANs done right), **least privilege** (targets have no NAT = servers with no internet access), and **imaging/rollback** (snapshots = backups). Everything that protects your lab protects a real network — write that sentence down; it's an interview answer.

Try it: Take snapshot "lab-ready". Then deliberately break Kali (**sudo rm /etc/resolv.conf** — watch DNS die), restore, verify. Confidence in the safety net changes how aggressively you experiment.