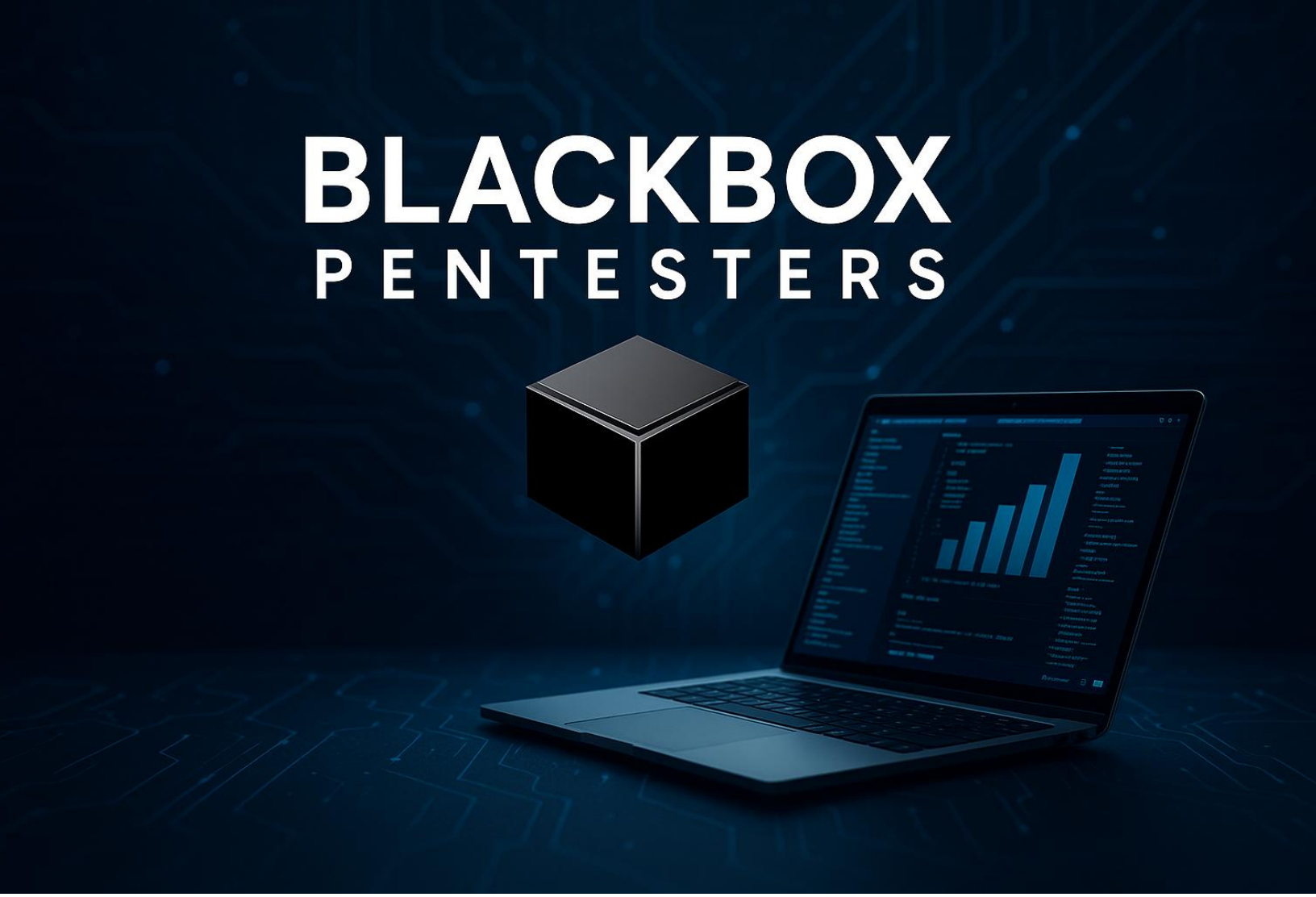

BlackBox Pentesters Training Series (Foundations)

BLACKBOX
PENTESTERS

Contents

BLACKBOX
PENTESTERS

Chapter 1: How Computers Actually Work

Before You Can Secure Systems, You Must Understand Them

Who This Chapter Is For

This chapter is for anyone entering cybersecurity who wants to:

- Stop guessing
- Stop copying commands they don't understand
- Stop being dependent on tools

If you skip this chapter, everything else you learn will be fragile.

Why This Chapter Exists

Most cybersecurity beginners jump straight to:

- Tools
- Exploits
- Certifications

And then wonder why nothing makes sense.

Here's the truth:

Cyber attacks don't exploit "vulnerabilities".
They exploit how computers work.

If you don't understand:

- How operating systems manage memory
- How processes interact
- How permissions are enforced

You are not testing security, you are pressing buttons.

What "Understanding Computers" Actually Means

You are **not** expected to be a computer scientist.

You **are** expected to understand:

1. What the operating system controls
2. What applications are allowed to do
3. Where trust boundaries exist
4. Why misconfigurations are so dangerous

This chapter focuses on **behaviour**, not theory.

BLACKBOX
PENTESTERS

1. What an Operating System Really Does

An operating system (OS) is a **control layer**.

It decides:

- What runs
- What talks to what
- What can access files, memory, and devices

Whether it's Windows, Linux, or macOS, the principles are the same.

You must understand:

- Processes & threads
- User vs kernel space
- File systems
- Permissions

Why this matters:
Privilege escalation, malware, persistence, and sandbox escapes all target OS behaviour.

Training Material

Microsoft Learn – Windows Internals & Fundamentals

Platform: Microsoft Learn
https://learn.microsoft.com

Focus on:

- Windows processes
- User accounts & permissions
- Services

Ignore product marketing. Learn how Windows *thinks*.

Linux Foundation – Introduction to Linux

Platform: Linux Foundation
https://training.linuxfoundation.org

Linux teaches clarity:

- Everything is a file
- Permissions are explicit
- Behaviour is visible

If you understand Linux, you will understand *why* many security tools exist.

2. Processes, Memory & Execution

A process is a **running program**.
That sounds simple, until it isn't.

You must understand:

- How programs are loaded into memory
- How processes interact
- Why memory corruption is catastrophic

Why attackers care:
Most serious exploits abuse how memory is managed or shared.

Beginner mistake to avoid

Thinking "memory stuff" is advanced and optional.

It isn't.

You don't need to write exploits, but you **must** understand why they work.

Training Material

Open Security Training – Intro to x86 & Memory

Platform: Open Security Training
https://opensecuritytraining.info

This teaches:

- How programs execute
- Why buffers overflow
- Why memory protections exist

Watch slowly. This knowledge compounds over years.

3. Filesystems & Permissions

Filesystems are trust models.

You must understand:

- Read / write / execute permissions
- Ownership
- Why "running as admin" is dangerous

Why this matters:
Most breaches succeed because:

- Permissions were too loose
- Secrets were stored badly
- Assumptions were made

Attackers don't break encryption if they can just read the file.

Training Material

Linux Filesystem & Permissions (Hands-on)

https://linuxjourney.com

Excellent for:

- Seeing permissions in action
- Understanding user separation
- Building intuition

If permissions confuse you, stop and fix that **now**.

4. Users, Privileges & Trust

Computers do not understand intent.
They enforce **rules**.

Understand:

- Standard users vs administrators
- Service accounts
- Why "least privilege" exists

Why attackers care:
Every privilege escalation is a failure of trust design.

Reality check

If you don't understand *why* admin access is dangerous, you should not be testing systems.

Training Material

Windows Security Fundamentals

https://learn.microsoft.com/windows/security/

Focus on:

- User rights
- Privilege boundaries
- Service behaviour

This knowledge directly feeds into Active Directory security later.

5. The Command Line (Non-Negotiable)

The command line isn't "old school".

It's:

- Faster
- Clearer
- More honest

GUIs hide behaviour.
Command lines expose it.

You must be comfortable:

- Navigating files
- Viewing processes
- Inspecting system state

Training Material

Linux Command Line Basics

https://linuxcommand.org

Learn:

- ls, ps, top, grep, chmod, chown
- What they *do*, not just how to run them

If the terminal scares you, that's exactly why you need it.

What Not to Waste Time On (Yet)

At this stage, avoid:

- Kernel exploit development
- Reverse engineering
- Low-level assembly obsession

Those come later *if needed*.

Right now, your goal is **clarity**, not depth.

How This Prepares You for Cybersecurity Roles

After this chapter, you should be able to:

- Explain what an operating system controls
- Understand why misconfigurations are dangerous
- Follow how attacks interact with systems
- Stop blindly trusting tools

This chapter underpins:

- Penetration testing
- Incident response
- SOC analysis
- Cloud security
- Endpoint security

There are no shortcuts around it.

Final Thought

Most people entering cybersecurity want to *feel* technical.

Professionals want to **understand systems**.

Tools change.
Interfaces change.
Operating system behaviour does not.

If you get this chapter right, everything else becomes easier.

BLACKBOX
PENTESTERS

Chapter 2: Networking Fundamentals

Understanding How Attacks Move

Who This Chapter Is For

This chapter is for anyone who wants to work in cybersecurity and:

- Actually understand breaches
- Stop treating networks as “background infrastructure”
- Be taken seriously in technical conversations

If networking feels overwhelming, that’s normal.
If you try to avoid it, that’s a problem.

Why Networking Is Non-Negotiable

Every cyber attack is a **networked activity**.

Credentials are transmitted over networks.
Malware communicates over networks.
Attackers move laterally over networks.
Defenders detect activity by observing networks.

If you don’t understand networking, you don’t understand attacks, full stop.

Hard truth:
Cybersecurity is not about exploits.
It’s about how systems communicate and where trust breaks down.

What “Networking Fundamentals” Really Means

You are **not** expected to be a network engineer.

You **are** expected to understand:

- How data moves from one system to another
- What “normal” traffic looks like
- Why segmentation exists
- How attackers abuse trusted paths

This chapter focuses on **behaviour**, not vendor configuration.

6. TCP vs UDP – Why Protocol Choice Matters

At a basic level:

TCP

- Reliable
- Ordered
- Session-based
- Used where accuracy matters

UDP

- Fast
- Stateless
- No delivery guarantee
- Used where speed matters

Why attackers care

- TCP provides reliability for data exfiltration
- UDP is often abused for stealth and amplification
- Detection tools behave differently for each

If you don't understand the protocol, you can't interpret the traffic.

Training Material

Cisco Networking Academy – Introduction to Networks

Platform: Cisco Networking Academy
https://www.netacad.com

Focus on:

- TCP/IP fundamentals
- Packet flow
- Basic routing concepts

Ignore certification pressure. Consume the fundamentals.

7. IP Addressing & Subnets – Trust Boundaries in Disguise

An IP address isn't just an identifier, it's a **location and trust signal**.

You must understand:

- Private vs public IP ranges

- What a subnet represents
- Why segmentation exists

Why attackers care

Flat networks turn one compromised device into a full compromise.

Most breaches aren't sophisticated, they're **poorly segmented**.

Training Material

Professor Messer – Network+ (Free, Clear)

Platform: Professor Messer
https://www.professormesser.com/network-plus/

Use this for:

- Subnetting logic
- Addressing clarity
- Protocol awareness

Watch for understanding, not exam prep.

8. Ports & Services – What Is Actually Exposed

Every open port is a decision someone made, or failed to make.

Common examples:

- 80 / 443 → Web services
- 22 → Remote administration
- 3389 → Remote desktop
- 445 → File sharing (historically abused)

The mistake beginners make

Memorising port numbers without understanding the **service behaviour**.

The professional question

"Should this service be reachable from here at all?"

Training Material

Nmap Book – Port Scanning Concepts

https://nmap.org/book/

Read for:

- How service discovery works
- Why scanning reveals design flaws
- The limits of automated discovery

This is about **understanding exposure**, not tool worship.

9. DNS – The Most Abused Protocol You'll Ever Touch

DNS is not just name resolution.
It's a **reconnaissance tool**, a **control channel**, and a **detection goldmine**.

You must understand:

- Forward vs reverse lookups
- Internal vs external DNS
- Why DNS leaks information

Why attackers care

- DNS reveals internal structure
- Malware uses DNS for command and control
- Defenders rely on DNS logs for early warning

If DNS feels invisible to you, you are missing critical signals.

Training Material

Cloudflare Learning Centre – DNS & Networking

Platform: Cloudflare
https://www.cloudflare.com/learning/

This explains:

- DNS clearly
- Protocol behaviour honestly
- Internet-scale realities

It's vendor-neutral and grounded in reality.

10. Firewalls & Network Trust Assumptions

Firewalls don't make networks secure.
They enforce **assumptions**.

Common failed assumptions:

- "Internal traffic is safe"
- "Only this system uses that rule"

- "Nobody would abuse this path"

Attackers don't break firewalls, they **walk through allowed traffic**.

Training Material

Open Security Training – Networking for Security

Platform: Open Security Training
https://opensecuritytraining.info

Excellent for:

- Understanding protocol abuse
- Seeing where assumptions fail
- Thinking like both attacker and defender

This is where networking becomes security-relevant.

11. How Attacks Actually Move Through Networks

A simplified real-world attack flow:

1. Initial access via exposed service
2. Internal discovery via network scanning
3. Credential reuse across trusted paths
4. Lateral movement using legitimate protocols
5. Data exfiltration disguised as normal traffic

This is why networking knowledge outlasts vulnerability lists like the **OWASP Top Ten**.

Framework Reference (Use for Context)

MITRE ATT&CK – Network Techniques

Framework: MITRE ATT&CK
https://attack.mitre.org

Use this to:

- Map techniques to network behaviour
- Understand attacker goals
- Connect traffic to intent

Do not memorise. Interpret.

Hands-On Reinforcement (Strongly Recommended)

TryHackMe – Pre-Security & Networking Paths

Platform: TryHackMe
https://tryhackme.com

Use labs to:

- Visualise packet flow
- Reinforce concepts safely
- Build confidence without shortcuts

Labs support understanding, they don't replace it.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Advanced routing protocols
- Vendor-specific firewall certs
- Packet crafting obsession

Those come later. Right now, clarity beats complexity.

What You Should Be Able to Do After This Chapter

You should now be able to:

- Follow how an attack moves through a network
- Explain why segmentation failures matter
- Understand why "allowed traffic" is dangerous
- Speak confidently about network-based risk

This knowledge transfers directly into:

- Penetration testing
- SOC & detection roles
- Incident response
- Cloud & identity security

Final Thought

Most people rush networking because it's uncomfortable.

That discomfort is your signal that it matters.

Tools will change.
Attack techniques will evolve.
Network behaviour will remain the foundation.

Get this right, and everything else builds faster, and stronger.

BLACKBOX
PENTESTERS

Chapter 3: How the Web Works (And Why It Fails)

Who This Chapter Is For

This chapter is for anyone who:

- Wants to understand why web vulnerabilities exist
- Is tired of memorising payloads without understanding impact
- Wants to test real applications, not just lab toys

If you work anywhere near application security and you don't understand how the web works, you are operating blind.

Why the Web Is Still the #1 Attack Surface

Most organisations run on web applications.

Customer portals.
APIs.
Admin panels.
Cloud services.

And most breaches still start here.

Why?
Because web applications are where:

- Users meet logic
- Authentication meets trust
- Speed beats security
- Business pressure overrides engineering discipline

Hard truth:
Most web vulnerabilities aren't "bugs".
They're design failures.

What "Understanding the Web" Actually Means

This chapter is **not** about:

- Memorising vulnerability lists
- Learning exploit payloads
- Becoming a developer

It **is** about understanding:

- How browsers and servers communicate

- How authentication really works
- Where trust is assumed, incorrectly
- Why logic flaws are more dangerous than technical bugs

1. HTTP: A Simple Protocol With Serious Consequence

HTTP is:

- Stateless
- Text-based
- Trust-agnostic

Every request contains:

- A method (GET, POST, PUT, DELETE)
- Headers
- Optional body data

Every response contains:

- A status code
- Headers
- Content

Why attackers care

HTTP does exactly what it's told.

If developers don't explicitly:

- Validate
- Authenticate
- Authorise

The application will happily comply.

Computers do not understand intent.
They enforce rules, or the lack of them.

Training Material

MDN Web Docs – HTTP Overview

https://developer.mozilla.org/en-US/docs/Web/HTTP

This is one of the clearest explanations of:

- Requests & responses
- Status codes
- Headers
- Caching behaviour

Ignore the front-end noise. Focus on protocol behaviour.

12. Statelessness, Sessions & the Illusion of Identity

HTTP has no memory.

So web applications invent identity using:

- Cookies
- Session IDs
- Tokens
- Headers

Why this fails

Developers often assume:

- Users won't tamper with tokens
- Sessions won't be guessed or reused
- State can't be manipulated

Attackers assume the opposite.

This is how you get:

- Session fixation
- Account takeover
- Privilege escalation

Training Material

OWASP WebGoat – Session & Auth Lessons

Project: OWASP
https://owasp.org/www-project-webgoat/

Use this to:

- Break authentication safely
- See how small mistakes cascade
- Understand why controls fail

Do not rush the labs. Observe behaviour.

13. Authentication vs Authorisation (Where Things Really Break)

This is one of the most common, and damaging, misunderstandings.

- Authentication: Who are you?

- Authorisation: What are you allowed to do?

Many applications authenticate users correctly…
…and then fail to enforce authorisation consistently.

Why attackers love this

Because logic flaws don't trigger alerts.
They don't crash systems.
They just quietly expose data.

If a user can access something "they shouldn't",
that's not a bug, that's a failure of design.

14. Client-Side Trust (A Mistake That Never Dies)

Browsers are **untrusted environments**.

Yet applications still:

- Trust hidden fields
- Trust JavaScript validation
- Trust client-side logic

Attackers can:

- Modify requests
- Replay traffic
- Bypass UI restrictions completely

If logic exists only in the browser, it does not exist.

Training Material

PortSwigger Web Security Academy

https://portswigger.net/web-security

This is one of the best free resources for:

- Understanding logic flaws
- Seeing how simple changes break apps
- Learning without cargo-cult payloads

Focus on *why the vulnerability exists*, not the exploit.

15. APIs: The Quiet Expansion of the Attack Surface

Modern applications are API-driven.

That means:

- More endpoints
- More trust boundaries
- Less visibility

Common API failures:

- Missing authentication
- Broken object-level authorisation
- Excessive data exposure

APIs fail quietly, and at scale.

Training Material

OWASP API Security Top 10

https://owasp.org/www-project-api-security/

This complements the traditional **OWASP Top Ten**, but reflects how modern applications actually work.

Use it to:

- Understand systemic API risk
- Spot architectural weaknesses
- Avoid web-only thinking

16. Why Scanners Miss the Real Risk

Automated tools are good at:

- Known patterns
- Input validation issues
- Surface-level misconfigurations

They are bad at:

- Business logic flaws
- Privilege abuse
- Multi-step attacks
- Contextual risk

This is why:

- Apps “pass” scans
- Breaches still happen
- Leadership gets confused

Tools don’t understand intent. People do.

What You Should Be Able to Do After This Chapter

You should now be able to:

- Explain how a web request is processed
- Identify where trust is assumed
- Spot authentication vs authorisation failures
- Understand why logic flaws matter more than payloads
- Read the **OWASP Top Ten** with understanding, not memorisation

What Not to Obsess Over (Yet)

At this stage, avoid:

- Advanced exploit chains
- Framework-specific deep dives
- Bypass tricks without context

Depth comes later.
Right now, you're building judgement.

Final Thought

Web security doesn't fail because developers are careless.
It fails because systems are complex, rushed, and built on assumptions.

Attackers don't need to break the web.
They just let it behave exactly as designed.

If you understand how the web works,
you'll understand why it keeps failing, and how to test it properly.

BLACKBOX
PENTESTERS

Chapter 4: Programming for Security

Only What You Need

Who This Chapter Is For

This chapter is for anyone who:

- Thinks programming is “for developers”
- Has copied scripts without fully understanding them
- Wants to stop being blocked by code they can’t read

If you avoid programming entirely, you cap your career early.

The Programming Myth (Let’s Kill It)

You do **not** need to:

- Build full applications
- Master multiple languages
- Write perfect, elegant code

You **do** need to:

- Read code confidently
- Understand logic and data flow
- Recognise dangerous patterns
- Modify scripts safely

Hard truth:
Most security work fails at the point where someone says
“I don’t really understand the code, but…”

This chapter exists to remove that sentence from your vocabulary.

What “Programming for Security” Actually Means

Security programming is about **thinking**, not syntax.

You must understand:

- How data enters a system
- How it’s processed
- Where decisions are made
- What assumptions exist

Once you grasp that, languages become interchangeable.

17. Logic, Flow & Decision-Making

Before languages, understand logic.

Every program relies on:

- Input
- Conditions (if / else)
- Loops
- Output

Why attackers care

Logic errors create:

- Authorisation bypasses
- Validation failures
- Business logic flaws

If the logic is wrong, the application behaves *perfectly*, and insecurely.

Training Material

Harvard CS50 – Computer Science Basics (Selected Sections)

Platform: Harvard University
https://cs50.harvard.edu

Focus on:

- Logic
- Conditionals
- Data handling

You don't need to finish the course. Build intuition.

18. Python: The Security Industry's Swiss Army Knife

If you learn **one** language, make it Python.

Python is used for:

- Automation
- Proof-of-concept exploits
- Parsing data
- Writing internal tools
- Modifying existing scripts

What you actually need

- Variables
- Functions
- Loops
- Dictionaries
- Basic file handling

Not frameworks. Not optimisation. Not style debates.

Training Material

Python for Everybody

https://www.py4e.com

Clear, practical, beginner-friendly.

Learn:

- Reading inputs
- Processing data
- Writing simple scripts

If you can write a script that:

- Takes input
- Processes it
- Outputs something useful

You are already ahead of many juniors.

19. Reading Code > Writing Code

Most security professionals:

- Modify existing tools
- Review application logic
- Debug scripts mid-engagement

You will read far more code than you write.

What to practise

- Follow variable names
- Trace data from input to output
- Identify where validation should exist

If you can explain what a script does in plain English, you understand it.

Training Material

GitHub – Read Real Code

https://github.com

Pick:

- Simple Python tools
- Small scripts
- Clear projects

Don't fork. Don't rush. Read slowly.

20. JavaScript: Understanding the Web's Control Layer

JavaScript controls:

- Front-end logic
- Client-side validation
- API calls
- State handling

You don't need to be a JavaScript developer.

You **do** need to:

- Read it
- Understand what it's enforcing
- Know why it can't be trusted

Training Material

MDN Web Docs – JavaScript Basics

https://developer.mozilla.org/en-US/docs/Web/JavaScript

Focus on:

- Variables
- Functions
- Requests
- Event handling

Ignore frameworks. Understand behaviour.

21. Common Dangerous Coding Patterns

These patterns appear everywhere:

- Trusting user input
- Client-side validation only
- Hard-coded secrets
- Poor error handling
- Overly complex logic

Security failures are often **obvious in code**, if you know what to look for.

Training Material

OWASP Secure Coding Practices

Project: OWASP
https://owasp.org/www-project-secure-coding-practices/

Use this as:

- A reference
- A pattern-spotting guide
- A way to explain issues clearly

Not as a checklist to memorise.

22. Automation: When Programming Becomes Leverage

A small script can:

- Save hours
- Reduce human error
- Make testing repeatable

But automation without understanding just scales mistakes.

Rule to live by

Never automate something you don't fully understand manually.

Training Material

Automate the Boring Stuff with Python

https://automatetheboringstuff.com

This teaches:

- Practical automation
- Real-world scripting
- Confidence through doing

Security benefits are a side effect, and that's a good thing.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Competitive programming
- Algorithm theory
- Language wars
- Framework deep dives

Your goal is **clarity and control**, not elegance.

What You Should Be Able to Do After This Chapter

You should now be able to:

- Read and understand simple scripts
- Trace how data moves through code
- Identify insecure logic patterns
- Modify scripts safely
- Stop being blocked by "I don't code"

That's enough to be dangerous, in the right way.

Final Thought

Programming isn't a gatekeeper skill.
It's a force multiplier.

You don't need to write beautiful code.
You need to **understand what code is doing to trust and data**.

If you get this chapter right, tools stop feeling magical, and start feeling predictable.

BLACKBOX
PENTESTERS

Chapter 5: Linux as a Daily Driver

If You Can't Use Linux, You're Renting Your Skillset

Who This Chapter Is For

This chapter is for anyone who:

- Uses Linux only "when they have to"
- Relies on copy-pasted commands
- Gets uncomfortable when the GUI disappears

If Linux feels intimidating, that's expected.
If you try to avoid it, that's a career limiter.

Why Linux Matters in Cybersecurity

Linux runs:

- The majority of servers
- Most cloud infrastructure
- Almost all security tooling
- Many security appliances
- Containers, CI/CD, and automation pipelines

If you work in cybersecurity, Linux is not a niche skill, it's the **default environment**.

Hard truth:
Many people don't "know cybersecurity".
They know how to use tools *someone else built on Linux*.

What "Linux as a Daily Driver" Actually Means

You do **not** need to:

- Be a kernel developer
- Memorise every flag
- Customise your desktop endlessly

You **do** need to:

- Navigate confidently without a GUI
- Understand permissions and ownership
- Inspect processes and services
- Read system state without panic

This chapter is about **control**, not mastery.

23. The Linux Philosophy (Why It Feels Different)

Linux is honest.

- Everything is a file
- Processes are visible
- Permissions are explicit
- Errors tell you what's wrong, if you read them

Linux doesn't protect you from yourself.
That's why it's powerful, and dangerous.

Why attackers care

Because Linux:

- Exposes behaviour clearly
- Allows chaining small actions
- Makes misconfigurations obvious (and exploitable)

Training Material

Linux Foundation – Introduction to Linux

Platform: Linux Foundation
https://training.linuxfoundation.org

Focus on:

- Filesystem layout
- Users and groups
- Command-line navigation

Ignore distro politics. Learn the fundamentals.

24. Filesystem Layout (Knowing Where You Are)

You must understand:

- /etc – configuration
- /home – user data
- /var – logs & variable data
- /bin / /usr/bin – executables
- /tmp – temporary (often abused)

Why attackers care

- Credentials live in predictable places

- Logs reveal activity
- Misplaced permissions expose secrets

If you don't know *where* to look, you won't see problems, or attacks.

Training Material

Linux Journey – Filesystem & Permissions

https://linuxjourney.com

Excellent for:

- Seeing permissions in action
- Building intuition
- Understanding ownership clearly

This is foundational knowledge. Don't rush it.

25. Permissions & Ownership (The Real Security Model)

Linux security relies heavily on:

- Read / write / execute permissions
- User ownership
- Group membership

You must understand:

- Why chmod 777 is a red flag
- What sudo actually does
- Why running as root is dangerous

Reality check

Most Linux compromises succeed because:

- Permissions were too loose
- Services ran with excessive privileges
- Admin access was misused

Attackers don't "hack" permissions, they **use them**.

26. Processes & Services (Seeing What's Really Running)

Linux makes process behaviour visible.

You must be comfortable with:

- Viewing running processes

- Understanding parent/child relationships
- Identifying suspicious behaviour
- Knowing what shouldn't be running

Why this matters

- Malware is just a process
- Persistence is just a service
- Evasion is hiding in plain sight

If you can't read process output, you can't investigate incidents properly.

Training Material

DigitalOcean – Linux Process Management

https://www.digitalocean.com/community/tutorials

Clear explanations of:

- ps, top, htop
- Signals
- Service behaviour

This knowledge transfers directly to IR and SOC work.

27. The Command Line (Where the Real Work Happens)

The terminal is:

- Faster
- Scriptable
- Transparent

You must be comfortable with:

- Navigating directories
- Searching files
- Chaining commands
- Reading output critically

This is where confidence is built.

Core Commands You Should Understand

(Not memorise, understand)

- ls, cd, pwd
- cat, less, grep

- ps, top, htop
- chmod, chown
- find, awk, sed (basic use)

Training Material

The Linux Command Line (William Shotts)

https://linuxcommand.org

This teaches:

- Why commands behave the way they do
- How to combine tools
- How to think in pipelines

This is one of the most underrated resources in the industry.

28. Linux in Real Security Work

Linux shows up everywhere:

- Kali / testing distros
- Cloud VMs
- Containers
- SOC tooling
- Security automation

If Linux feels natural, everything else feels lighter.

If Linux feels hostile, everything feels harder.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Custom kernels
- Deep system internals
- Distro hopping
- Over-engineering your setup

Your goal is **fluency**, not identity.

What You Should Be Able to Do After This Chapter

You should now be able to:

- Use Linux confidently without a GUI
- Understand permissions and ownership
- Inspect system state calmly
- Read logs and process output
- Feel comfortable living in the terminal

That's the baseline for serious security work.

Final Thought

Linux doesn't make you smarter.
It makes you **honest**.

GUIs hide behaviour.
Linux exposes it.

If you can work comfortably in Linux, you stop depending on tools, and start understanding systems.

BLACKBOX
PENTESTERS

Chapter 6: Identity, Active Directory & Enterprise Reality

Why Most Breaches Are About Who You Are, Not What You Exploit

Who This Chapter Is For

This chapter is for anyone who wants to work in:

- Penetration testing
- SOC & detection
- Incident response
- Security consulting
- Internal security teams

If you want to work with *real organisations*, you cannot avoid identity.
And in most enterprises, identity means **Active Directory**.

The Reality Most Beginners Miss

Modern breaches rarely start with:

- Kernel exploits
- Zero-days
- Sophisticated malware

They usually start with:

- A compromised account
- Excessive privileges
- Poor identity hygiene
- Blind trust in internal users

Hard truth:
Once attackers control identity, the network belongs to them.

What “Enterprise Reality” Actually Looks Like

Real organisations have:

- Legacy systems
- Technical debt
- Shared accounts
- Overworked admins
- “Temporary” permissions that never get removed

Security decisions are made under pressure, not in textbooks.

This chapter teaches you how identity **actually behaves** in the wild.

29. What Identity Really Means in Cybersecurity

Identity is not just usernames and passwords.

It includes:

- Users
- Service accounts
- Machine accounts
- Tokens
- Trust relationships

Identity answers two questions:

- **Who are you?**
- **What are you allowed to do?**

When those answers are wrong, or assumed, breaches happen.

30. What Active Directory Actually Is

Active Directory is not just “Windows login”.

It is:

- A central identity database
- An authentication authority
- An authorisation engine
- A trust model for the entire organisation

It controls:

- Access to systems
- Permissions to data
- Administrative power
- Trust between machines and users

Compromise AD, and you compromise the organisation.

Training Material

Microsoft Learn – Active Directory Fundamentals

Platform: Microsoft Learn
https://learn.microsoft.com

Focus on:

- Domains
- Users & groups
- Authentication flow
- Trust relationships

Ignore marketing. Learn how identity is enforced.

Authentication vs Authorisation (Again, But This Time It Matters)

This distinction becomes critical in enterprise environments.

- Authentication: Proving identity
- Authorisation: Granting access

Many organisations authenticate users correctly…
…and then **authorise far too much**.

Why attackers love this

Because they don't need admin exploits if:

- Users are over-privileged
- Groups are mismanaged
- Permissions are inherited blindly

Most privilege escalation is **administrative failure**, not technical genius.

31. Kerberos, NTLM & Trust (Conceptually)

You do not need to be a protocol expert.

You **do** need to understand:

- Credentials are reused
- Trust is transitive
- Authentication tokens are valuable

Why this matters

Attackers don't "crack" identity systems.
They:

- Steal credentials
- Replay trust
- Abuse delegation
- Move laterally using legitimate mechanisms

From the logs' perspective, it often looks like normal activity.

32. Service Accounts: The Quiet Disaster

Service accounts are everywhere.

They often:

- Never expire
- Have excessive permissions
- Run critical systems
- Are poorly monitored

Why attackers care

Service accounts are:

- Stable
- Powerful
- Ignored

They are one of the fastest routes to domain-wide compromise.

If a service account is compromised, attackers inherit **everything it can access**, instantly.

33. Lateral Movement: How Breaches Spread

Once identity is compromised, attackers:

- Enumerate users and groups
- Identify privileged paths
- Reuse credentials
- Access new systems
- Escalate quietly

This is not “hacking”.
This is **walking through allowed access**.

Framework Reference

MITRE ATT&CK – Credential Access & Lateral Movement

Framework: MITRE ATT&CK
https://attack.mitre.org

Use this to:

- Understand attacker goals
- Map identity abuse to real techniques
- Connect logs to intent

Interpret, don’t memorise.

34. Why Identity Failures Are Hard to Detect

Identity abuse is dangerous because:

- It uses legitimate accounts
- It blends into normal activity
- Alerts are noisy or ignored
- Context is missing

Most SOCs don't fail because they lack tools.
They fail because identity behaviour is **poorly understood**.

35. What Attackers Rarely Need to Do

In real enterprise breaches, attackers often **don't** need to:

- Exploit vulnerabilities
- Bypass authentication
- Deploy advanced malware

They just:

- Log in
- Move around
- Abuse trust
- Escalate quietly

That's why identity security is now a board-level issue.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Building your own AD lab from scratch
- Deep protocol reverse engineering
- Advanced attack tooling

Focus first on:

- Understanding identity relationships
- Recognising over-permissioning
- Seeing how trust spreads risk

What You Should Be Able to Do After This Chapter

You should now be able to:

- Explain what Active Directory really controls
- Understand why identity compromise is catastrophic
- Spot common enterprise identity weaknesses
- Follow how breaches spread laterally
- Talk confidently about identity risk to technical and non-technical audiences

This knowledge is essential for:

- Pentesting
- SOC & detection
- Incident response
- Security leadership roles

Final Thought

Most organisations don’t get breached because their passwords are weak.

They get breached because:

- Trust is excessive
- Identity is misunderstood
- Permissions accumulate silently

If you control identity, you control the environment.

Understand that, and you understand modern enterprise security.

BLACKBOX
PENTESTERS

Chapter 7: How Attacks Actually Happen

Kill Chains & Chaining Weaknesses

Who This Chapter Is For

This chapter is for anyone who:

- Understands individual security concepts but struggles to see the whole picture
- Wonders how “low-risk” issues become major breaches
- Wants to think like a real attacker, without becoming reckless

If you still think breaches happen because of *one big vulnerability*, this chapter will reset your thinking.

The Biggest Myth in Cybersecurity

The myth:

“Attackers exploit a vulnerability and you’re breached.”

The reality:

Attackers **chain small weaknesses** until the organisation collapses under its own assumptions.

Breaches are not moments.
They are **processes**.

What a “Kill Chain” Really Is

A kill chain is not a checklist.

It’s a **mental model** for how attackers:

- Progress
- Adapt
- Re-use access
- Avoid detection

Understanding kill chains allows you to:

- Anticipate next steps
- Break attacks early
- Prioritise controls intelligently

A Simple, Realistic Attack Lifecycle

Most real-world attacks follow a variation of this flow:

1. **Initial Access**
2. **Internal Discovery**
3. **Credential Access**
4. **Privilege Escalation**
5. **Lateral Movement**
6. **Persistence**
7. **Impact**

This isn't theory.
This is what happens in environments every day.

36. Initial Access: The Door Is Already Open

Attackers don't start with exploits.
They start with **opportunity**.

Common initial access paths:

- Exposed web applications
- Phishing
- Credential reuse
- Misconfigured remote access
- Third-party access

Key insight

Initial access is often:

- Low privilege
- Noisy
- Easy to detect, if you're looking

The real damage comes later.

37. Internal Discovery: Learning the Environment

Once inside, attackers slow down.

They look for:

- Network structure
- Identity relationships
- Valuable systems
- Weak trust boundaries

This stage is quiet and critical.

The better the attacker understands your environment,
the less noise they need to make.

Discovery is where many breaches could be stopped, but aren't.

38. Credential Access: Turning Access into Control

Credentials are the real prize.

Attackers want:

- Passwords
- Hashes
- Tokens
- Session material

Why?
Because credentials:

- Don't trigger exploits
- Don't crash systems
- Look legitimate in logs

Once credentials are obtained, technical controls matter far less.

39. Privilege Escalation: Expanding Trust

Privilege escalation is often misunderstood.

In real environments, it's usually:

- Over-permissioned users
- Weak group management
- Misconfigured services
- Inherited access nobody reviewed

Not clever exploitation, **administrative drift**.

Attackers don't force privilege.
They accept what's already been granted.

40. Lateral Movement: The Breach Spreads

With credentials and trust:

- Attackers move system to system
- Reuse access paths
- Blend into normal activity

From the outside, it often looks like:

- Admin activity
- Maintenance work
- Normal logins

This is where breaches become enterprise-wide incidents.

Framework Reference

MITRE ATT&CK – End-to-End Attack Techniques

Framework: MITRE ATT&CK
https://attack.mitre.org

Use this to:

- Map real behaviour to techniques
- Understand attacker intent
- Connect isolated alerts into narratives

This is how professionals think about attacks.

41. Persistence: Staying Without Being Seen

Attackers rarely rush.

They establish:

- Scheduled tasks
- Service modifications
- Backdoor accounts
- Token persistence

Persistence is about **patience**.

If attackers lose access once, they don't want to lose it again.

42. Impact: Only at the End

Impact is the final stage:

- Data theft
- Ransomware
- Espionage
- Disruption

By the time this happens, the attack is already **months old**.

The visible incident is just the last chapter.

Why Vulnerability Lists Miss This Entire Picture

Vulnerability lists, including the **OWASP Top Ten**, describe *entry points*, not attacks.

They do not show:

- How weaknesses combine
- How trust multiplies risk
- How attackers adapt

This is why organisations “fix findings” and still get breached.

How Attackers Chain Weaknesses (A Simple Example)

Individually:

- A low-impact web issue
- An over-privileged user
- Poor network segmentation

Together:

- Full domain compromise

No single failure caused the breach.
The **system did**.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Advanced exploit chains
- Zero-day hunting
- Tool collections

Focus on:

- Thinking in stages
- Understanding attacker goals
- Seeing connections between weaknesses

That mindset outlasts every tool.

What You Should Be Able to Do After This Chapter

You should now be able to:

- Describe a breach as a sequence, not an event
- Understand why “low risk” issues matter
- Explain how attackers adapt mid-engagement
- Think in terms of attack paths, not findings
- Prioritise security controls intelligently

This mindset is essential for:

- Red teams
- Blue teams
- SOC analysts
- Security leadership

Final Thought

Attackers don’t win because they’re smarter.

They win because:

- Systems are complex
- Trust accumulates
- Small decisions compound silently

Security doesn’t fail loudly.
It fails gradually, until it doesn’t.

If you understand how attacks actually happen, you stop chasing noise, and start breaking chains.

BLACKBOX
PENTESTERS

Chapter 8: Defensive Thinking (Even for Attackers)

Why the Best Attackers Understand Defence Better Than Defenders

Who This Chapter Is For

This chapter is for anyone who:

- Wants to work in offensive security without tunnel vision
- Plans to work in SOC, blue team, or incident response
- Thinks “defensive security is boring” (it isn’t, it’s harder)

If you only think like an attacker, you will always miss half the problem.

The Misunderstanding That Holds People Back

Many newcomers believe there’s a clean divide:

- Attackers break in
- Defenders stop them

Reality is messier.

Good attackers:

- Understand logging
- Anticipate detection
- Modify behaviour to blend in

Good defenders:

- Think like attackers
- Understand attack paths
- Prioritise realistically

Hard truth:
Offensive skill without defensive understanding is shallow.
Defensive skill without offensive understanding is blind.

What “Defensive Thinking” Actually Means

Defensive thinking is not about:

- Buying more tools
- Blocking everything
- Chasing alerts blindly

It *is* about:

- Understanding what matters
- Knowing what "normal" looks like
- Spotting deviation
- Making trade-offs consciously

Security is not absolute.
It's about **managing risk under constraint**.

43. Visibility Beats Perfection

You cannot defend what you cannot see.

Perfect security does not exist.
Good visibility does.

Defensive thinking starts with one question:

"If this happened, would we notice?"

Most organisations wouldn't.

Why attackers care

Attackers don't avoid controls.
They avoid **visibility**.

If something isn't logged, monitored, or reviewed, it's fair game.

44. Logging Is the Foundation (Not an Afterthought)

Logs are:

- Evidence
- Context
- Memory

Without logs:

- Alerts are meaningless
- Investigations stall
- Breaches go undetected for months

Defensive mindset shift

Instead of asking:

"Is this blocked?"

Ask:

“Is this observable?”

Training Material

Microsoft Learn – Security Logging & Monitoring

Platform: Microsoft Learn
https://learn.microsoft.com

Focus on:

- Authentication logs
- Audit trails
- Identity-related events

Ignore SIEM branding. Learn what data matters.

45. Detection Over Prevention (Most of the Time)

Prevention will fail.
Attackers only need to succeed once.

Detection allows you to:

- Respond
- Contain
- Learn
- Improve

Why this matters

Most mature security teams assume compromise.
They optimise for **early detection**, not perfection.

This is why frameworks like **MITRE ATT&CK** are detection-focused, not prevention-focused.

46. What “Suspicious” Actually Looks Like

Suspicious activity is rarely dramatic.

It looks like:

- A login at an unusual time
- A user accessing systems they don’t normally touch
- Admin actions without a clear reason
- Gradual, quiet changes

Defensive thinking is about **pattern recognition**, not signature matching.

Training Material

MITRE ATT&CK – Detection & Analytics

https://attack.mitre.org

Use this to:

- Understand what should be detectable
- Think in behaviours, not alerts
- Map attacker actions to signals

Professionals read ATT&CK *backwards*, from detection to behaviour.

47. Noise vs Signal (Why SOCs Burn Out)

Most environments generate:

- Too many alerts
- Too little context
- Too much pressure to “close tickets”

Defensive thinking asks:

“Does this alert matter *in context*?”

Attackers rely on noise.
They hide where defenders are overwhelmed.

48. Defence in Depth (As a Reality, Not a Diagram)

Defence in depth is not:

- A stack of tools
- A vendor slide

It is:

- Multiple opportunities to detect or stop an attack
- At different stages
- By different teams or controls

Even if one layer fails, the chain can still be broken.

Example

- Web app compromised
- Identity logging flags unusual access

- Endpoint monitoring spots credential reuse
- SOC intervenes

No single control saved the day.
The *system* did.

49. Why Attackers Study Blue Teams

Good attackers ask:

- What is logged?
- What is ignored?
- What alerts get actioned?
- What happens after detection?

They adapt accordingly.

If you don't understand defence, you won't understand attacker behaviour, because attackers are reacting to it.

50. Communicating Risk (The Defender's Superpower)

Defensive thinking includes communication.

Security professionals must:

- Explain risk clearly
- Prioritise realistically
- Avoid fear-driven messaging

If leadership doesn't understand:

- Why something matters
- What could realistically happen
- What decision they're making

Security will always lose.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Building custom SIEMs
- Writing complex detection rules
- Chasing "perfect" coverage

Focus instead on:

- Understanding logs
- Recognising behaviour

- Asking good questions

What You Should Be Able to Do After This Chapter

You should now be able to:

- Think in terms of detection, not just prevention
- Understand what attackers try to avoid
- Explain why visibility matters
- Recognise early indicators of compromise
- Work effectively with blue teams, even as an attacker

This mindset is critical for:

- Red teams
- SOC analysts
- Incident responders
- Security consultants
- Leadership roles

Final Thought

Security doesn't fail because attackers are invisible.

It fails because:

- Signals are ignored
- Context is missing
- Trade-offs are misunderstood

The strongest attackers understand defence.
The strongest defenders understand attack paths.

If you can think both ways, you stop playing checkers, and start playing the real game.

BLACKBOX
PENTESTERS

Chapter 9: Tools Are Last

Why Automation Isn't Security

Who This Chapter Is For

This chapter is for anyone who:

- Thinks buying tools equals improving security
- Feels productive because dashboards are busy
- Is overwhelmed by alerts, scans, and outputs
- Wants to understand why breaches still happen

If your security confidence comes from tools alone, it isn't confidence, it's dependency.

The Most Dangerous Assumption in Cybersecurity

The assumption:

"If we run enough tools, we're secure."

The reality:

Tools don't understand risk.
People do.

Automation processes data.
Security requires **judgement**.

What Tools Are Actually Good At

Tools excel at:

- Repetition
- Scale
- Speed
- Consistency

They are ideal for:

- Enumerating assets
- Identifying known patterns
- Monitoring large environments
- Reducing manual effort

Used correctly, tools are force multipliers.

Used blindly, they create **false confidence**.

What Tools Are Bad At (But Marketed as Solving)

Tools are bad at:

- Understanding business context
- Chaining weaknesses
- Interpreting intent
- Explaining impact clearly
- Prioritising realistically

No scanner understands:

- Why a vulnerability matters
- How it would actually be abused
- What risk it creates for the organisation

That judgement is human.

51. Automation Without Understanding Scales Mistakes

If you automate:

- A misunderstood process
- A misconfigured control
- A poor assumption

You don't fix the problem, you **replicate it faster**.

Never automate what you don't understand manually.

This rule applies to:

- Scanning
- Alerting
- Remediation
- Reporting

52. Scanners Find Issues, Not Risk

Automated scanners are good at:

- Known vulnerabilities
- Missing patches
- Weak configurations

They are not good at:

- Business logic flaws
- Privilege abuse

- Attack paths
- Real-world impact

This is why organisations:

- Fix “highs”
- Ignore “lows”
- Still get breached

Risk is contextual. Scanners are not.

Framework Context

Lists like the **OWASP Top Ten** describe *classes of issues*, not attack narratives.

Tools highlight entries.
Humans build stories.

53. Dashboards Create Activity, Not Security

A full dashboard feels reassuring.

It shouldn’t.

Dashboards often:

- Measure tool output
- Reward noise
- Hide blind spots
- Shift focus from outcomes

The question is not:

“How many findings do we have?”

It’s:

“Do we understand our most realistic attack paths?”

Most dashboards don’t answer that.

54. Why Manual Testing Still Matters

Manual testing:

- Identifies logic flaws
- Understands context
- Chains weaknesses
- Adapts dynamically

Automation is static.
Attackers are not.

This is why high-quality security work still relies on **human-led assessment**, with tools supporting, not replacing, judgement.

55. Tools as Enablers, Not Authorities

Professionals treat tools as:

- Assistants
- Sensors
- Accelerators

Not as:

- Decision makers
- Risk owners
- Substitutes for thinking

The moment a tool becomes the authority, understanding stops.

56. Alert Fatigue Is a Tooling Problem, Not a People Problem

Most SOC burnout is caused by:

- Too many alerts
- Poor tuning
- Lack of context
- Pressure to “close” issues

Automation created the noise.
Humans are blamed for missing the signal.

Defensive thinking means:

- Reducing noise
- Improving context
- Valuing fewer, better alerts

57. Why Attackers Love Tool-Heavy Environments

Attackers know:

- What tools are commonly deployed
- What they detect well

- What they miss

They adapt accordingly.

Environments that rely heavily on:

- Default tool configs
- Automated decisions
- Untuned alerts

Are predictable.

Predictability is the attacker's advantage.

58. The Role of Automation Done Right

When automation works, it:

- Handles scale
- Frees humans for judgement
- Provides reliable visibility
- Reduces repetitive work

Good automation supports:

- Humans making decisions
- Clear prioritisation
- Measured improvement

Bad automation hides ignorance behind activity.

What Not to Obsess Over (Yet)

At this stage, avoid:

- Tool collecting
- "Stack envy"
- Vendor promises
- Chasing coverage metrics

Focus on:

- Understanding what tools show you
- Knowing what they can't show
- Asking better questions

What You Should Be Able to Do After This Chapter

You should now be able to:

- Explain why tools don’t equal security
- Use automation without surrendering judgement
- Identify where manual analysis is required
- Push back on “tool-driven” security decisions
- Articulate risk beyond scanner output

This mindset is essential for:

- Penetration testers
- SOC analysts
- Security consultants
- Security leadership

Final Thought

Tools are necessary.
Tools are powerful.
Tools are dangerous when misunderstood.

Automation doesn’t make you secure.
Understanding does.

If you put tools last, after knowledge, context, and judgement, they become allies instead of illusions.

BLACKBOX
PENTESTERS

Chapter 10: Professionalism, Ethics & Communication

Why Trust Is Your Most Valuable Skill

Who This Chapter Is For

This chapter is for anyone who:

- Wants a long-term career in cybersecurity
- Intends to work with real organisations and real data
- Understands that technical skill alone isn't enough

If you treat cybersecurity like a game, this industry will eventually reject you.

The Part Nobody Likes to Talk About

Cybersecurity is a **trust profession**.

Clients trust you with:

- Sensitive systems
- Confidential data
- Business-critical access
- Legal and reputational risk

You are often one bad decision away from:

- Legal consequences
- Reputational damage
- Permanent career limitations

Hard truth:
You can be technically brilliant and still be unemployable.

What "Professionalism" Actually Means in Security

Professionalism is not:

- Wearing a suit
- Using corporate language
- Quoting frameworks

It *is*:

- Acting responsibly with access
- Respecting boundaries
- Communicating clearly
- Knowing when not to act

Professionalism is what allows organisations to trust you at all.

59. Ethics: The Line You Don't Cross

Ethics in cybersecurity are not abstract.

They are practical decisions made every day:

- What you test
- How far you go
- What you report
- What you keep private

Core ethical principles

- Only test what is authorised
- Do no unnecessary harm
- Minimise exposure of sensitive data
- Report honestly, even when it's uncomfortable

If something feels wrong, stop.

Framework Reference

(ISC)² Code of Ethics

Organisation: ISC2
https://www.isc2.org/ethics

You don't need the certification to understand the responsibility.

60. Authorisation Is Not a Technicality

No scope. No permission. No test.

Written authorisation exists to:

- Protect the organisation
- Protect staff
- Protect you

Testing without proper approval is not "hacking".
It's **illegal**.

"But I was helping" is not a legal defence.

61. Responsible Disclosure (Doing the Right Thing)

Finding a vulnerability is not the end of the job.

Professional behaviour means:

- Reporting clearly
- Giving organisations time to respond
- Avoiding public disclosure that causes harm

Security is about **reducing risk**, not seeking attention.

Industry Reference

Responsible Disclosure Guidelines

Organisation: OWASP
https://owasp.org

This sets the baseline for ethical behaviour across the industry.

62. Communication: Your Real Differentiator

Most security professionals:

- Understand risk
- Struggle to explain it

This is where careers stall.

Good communication means:

- Explaining impact, not exploits
- Avoiding jargon when unnecessary
- Framing findings in business terms

If leadership doesn't understand you, they won't act.

63. Writing Reports That Actually Get Read

A good security report:

- Tells a story
- Explains risk clearly
- Provides actionable guidance
- Avoids ego and theatrics

A bad report:

- Lists tools and payloads

- Drowns readers in screenshots
- Misses the “so what?”

Your report may be the **only thing** a decision-maker ever sees.

64. Saying “I Don’t Know” (And Why It’s Powerful)

Insecurity makes people bluff.

Professionals:

- Ask questions
- Admit uncertainty
- Validate assumptions

Saying “I don’t know, but I’ll find out” builds trust.

Pretending to know destroys it.

65. Managing Client Relationships & Expectations

Security work often reveals uncomfortable truths.

Professional behaviour includes:

- Staying calm
- Avoiding blame
- Being honest without being alarmist

You are not there to:

- Embarrass
- Shame
- Show off

You are there to **help improve security**.

66. Reputation Is Everything

This industry is smaller than it looks.

Your reputation is built on:

- How you handle access
- How you treat clients
- How you communicate risk
- How you behave when nobody is watching

Technical mistakes can be fixed.
Ethical failures follow you.

What Not to Obsess Over (Ever)

Avoid:

- Ego
- “Hacker” theatrics
- Social media clout chasing
- Crossing lines for attention

None of these build real careers.

What You Should Be Able to Do After This Chapter

You should now be able to:

- Understand your ethical responsibilities
- Communicate security risk clearly
- Write reports that drive action
- Set and respect boundaries
- Behave like a trusted professional

These skills apply to **every role** in cybersecurity.

Final Thought

Cybersecurity is not about proving how clever you are.

It’s about:

- Earning trust
- Protecting people and organisations
- Making responsible decisions under pressure

Your skills get you access.
Your professionalism determines what you do with it.

If you get this chapter right, you don’t just enter cybersecurity, you belong in it.