



GHOST
PROTOCOL



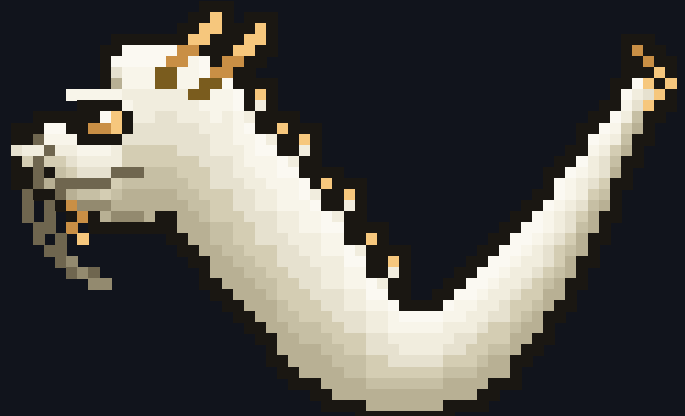
THE FIELD LIBRARY

OPERATIONS

THE WORK BEYOND THE TEST

The Consulting Craft

From tester to consultant, scoping and the paper, running the engagement, delivering findings, and the long relationship.



Ghost Protocol (Pvt) Ltd · Colombo · ghosts.lk

Security internship training · The Field Library · Edition 1 · 2026

THE GHOST PROTOCOL FIELD LIBRARY · OPERATIONS

The Consulting Craft

From tester to consultant, scoping and the paper,
running the engagement, delivering findings, and
the long relationship.

Ghost Protocol (Pvt) Ltd · Colombo

The Field Library, Edition 1 · 2026

Copyright 2026 Ghost Protocol (Private) Limited. All rights reserved.

Sold as a PDF for the buyer's own reading. Please do not share or resell the file.

The library

The Field Library is a whole shelf, not one book. The Core Path is the spine everyone reads. The rest is depth and breadth, taken on as the work calls for it. The shelf is complete now, and you are reading the one marked here.

A · THE CORE PATH

O The Oath BUILT

I Groundwork BUILT

II The Web BUILT

III The Wider Field BUILT

IV The Work BUILT

V The Craft BUILT

B · DEEP TRACKS

AI and Agents BUILT

Reading Code BUILT

Internal and Active Directory BUILT

The Web, Deeper BUILT

Cloud and Containers BUILT

Mobile and APIs BUILT

Reverse Engineering BUILT

Hardware and the Edges BUILT

C · THE OPERATIONS

Red and Purple Team BUILT

OSINT and Recon BUILT

Report Writing Mastery BUILT

The Consulting Craft YOU ARE HERE

Bug Bounty BUILT

D · THE OTHER SIDE

Detection and Response BUILT

Threat Hunting and Forensics BUILT

Hardening and Architecture BUILT

E · THE GROUND RULES

The Law, Deeper BUILT

Standards and Compliance BUILT

Ethics and Disclosure BUILT

F · COMPANIONS

The Reference Pack BUILT

The Lab Workbook BUILT

Case Files BUILT

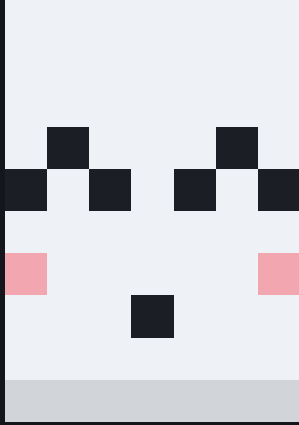
The Bestiary BUILT

The Glossary and the Ledger BUILT

Contents

- Opening 5
- 1** From Tester to Consultant 6
- 2** Scoping and the Paper 9
- 3** Running the Engagement 14
- 4** Delivering Findings 18
- 5** Trust and the Long Relationship 22
- Worked answers 27

The chair on the other side



A new client had come in, and Blink wanted to open the laptop and start testing before the call had even ended.

BLINK "They hired us to test. So we test, right now."

SOL "They hired your judgement about their risk, and the testing is only how you earn it. Before a single probe you agree what is in scope, what is off limits, and what you may do, in writing. You run it cleanly. You deliver it so they understand it and trust you with the next one. Even the free scan we lead with waits for a written yes from someone with authority over the target, never a surprise sprung to impress. The consultant owns the outcome, not just the output."

Worm's eye was on the one thing that was missing from the laptop entirely: the signed page.

WORM "the paper first."

So the book is the craft around the test: how to scope it, run it, deliver it, and become the one they call again, which in this trade is the whole game.

CHAPTER ONE

From Tester to Consultant

You already know how to find the bug. This chapter is the other half of the job: being the person a client trusts to tell them the truth about their systems, and owning whether they are actually safer after you leave.

The rest of the Library taught you to read a target, reason about it, and prove a flaw without overstating it. That is the whole of what a tester does, and it is not enough on its own to be worth hiring twice. A consultant does something a scanner cannot: sits across a table from a worried human, tells them what their systems really mean for their business, and stays accountable for the risk going down, not just for the list being long. This volume is about that second job. It assumes you can still do the first, because a consultant who cannot find the bug is only a salesperson in a better jacket.

The output is the Finding. The outcome is the Fix.

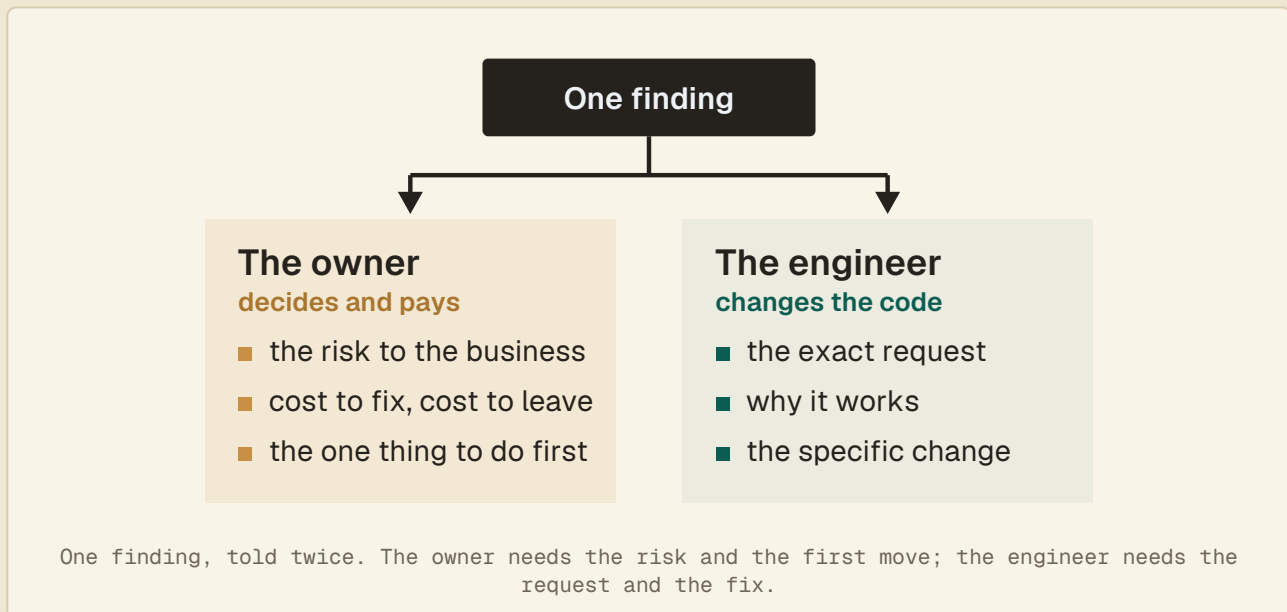
A tester is paid for output: findings, with evidence, severity, and a repro. A consultant is paid for an outcome: a client whose real risk is lower than it was. Those are not the same deliverable, and the gap between them is where most of this profession lives. You can write a flawless report of twenty findings and leave the client no safer, because the report sat unread, or the one critical that mattered was buried under nineteen notes, or nobody explained what to do first. The report is a means. The thing you are actually selling is a decision the client can make and act on.

Owning the outcome changes what you do before the report is even written. You care which finding gets fixed first, so you rank by what this client can actually reach and afford to lose, not by what was fun to exploit. You care that the fix lands, so you offer a retest and you write remediation a working engineer can follow. You care that the client understood you, so you check that they did, in the room, rather than hoping the PDF carried it. A tester's job ends at the finding. Yours ends when the risk has moved, or when you have told the client plainly that it has not and why.

You are advising a person, not a target

Volume I taught you to read a system: assume nothing, verify everything. The consultant's addition is that you now also read the person across the table. That person is a founder with a board meeting, or an engineer who built the thing you are about to find holes in, or an IT manager who asked for budget and got half. They have a deadline, a fear, and a story they tell themselves about how secure they are. None of that changes your findings. All of it changes how you deliver them, what you lead with, and whether they act.

Reading the person is not manipulation and it is not flattery. It is working out what this specific client needs to hear to make a good decision, and saying it in their language. The founder needs risk and the cost of fixing it against the cost of not. The engineer needs the exact request and the exact fix. The same finding, told two ways, because the two readers will do two different things with it. Get this wrong and your best technical work dies on contact with a tired human who did not follow a word of it.



The three questions a consultant carries

Before an engagement, and again while you work, hold three questions that a pure tester never has to ask:

- **What does this client actually need?** Not what is interesting to test, not the technique you want to practise. The smallest, most honest piece of work that answers their real question. Sometimes that is less than they asked to buy.
- **What will they be able to act on?** A finding they cannot understand, afford, or prioritise is, to them, no finding at all. Shape the work so the output is a short list of decisions, not a wall of raw results.
- **What happens after you leave?** Who fixes this, with what budget, by when, and who checks it held. If you have no answer, you have a report, not an engagement.



SOL SAYS

The day you start caring more about whether the critical got fixed than about how cleverly you found it, you have become a consultant. It is a quieter satisfaction than the catch, and it is the one clients pay for again. The finding is yours for an afternoon. The outcome is what has your name on it a year later.

What does not change

None of this lowers the technical bar, and it is worth being blunt about that, because "consulting" is where weak testers go to hide. The judgment is still judgment: a severity is argued from evidence or it is a feeling, exactly as Volume 0 and Volume II drilled it. The line is still the line: you test only what you are authorised, in writing, to test, and that is the next chapter. Clean conduct is unchanged and it is the whole of Chapter Five. The consultant's skills sit on top of the tester's; they do not replace them. A warm manner over an empty report is worse than a cold manner over a true one, because the warmth makes the client believe it.

The register: calm, not eager

There is a house voice to this, and you should learn it early because it is the opposite of what a nervous beginner does. You do not oversell. You do not perform certainty you have not earned. You do not fear-sell, meaning you never reach for the client's fear to close a deal, because manufacturing fear is both a dirty move and, in the end, a cheap one that smart buyers see through. You are the steady person in the room. You state what you found, what it means, and what you would do, in plain words and with numbers where you have them. Steadiness reads as competence, and in your case it will be true, which is the point.

■ EXERCISE

Take a real finding you made on your own lab (a DVWA SQL injection, say, or an IDOR you built and proved). Write it up twice, for nobody but yourself. First, one paragraph for the non-technical owner of a small business: what this means for them, in money and trust, and the single thing to do first, with no jargon. Then three or four lines for the engineer who has to fix it: the exact request, why it works, and the specific change. Read them back. The owner's version is the one most testers never learn to write, and it is half of the job you are training for.

■ ON YOUR FEET

You finish an authorised web test and hand over nine findings: three you have rated critical, four high, two medium. On the readout call the owner, who runs a ten-person company, tells you flatly that they can fund fixing two issues this quarter and the rest waits. You believe that is genuinely all they have. What do you do with that constraint, and what is the one thing you make sure you do not do?



WHAT WYRM KEEPS

A tester is paid for the finding. A consultant is paid for the outcome: a client whose real risk is lower, and who understood what to do about it.

- The report is a means, not the end. Rank by what this client can reach and afford to lose, write a fix they can follow, and offer the retest that proves it landed.
- You advise a person, not a target. The same finding is told one way to the owner (risk, cost) and another to the engineer (repro, fix), because they will act on different things.
- Carry three questions: what does this client actually need, what can they act on, and what happens after you leave. No answer to the third means you have a report, not an engagement.
- The technical bar does not drop and the line does not move. Clean conduct and evidence-backed judgment are unchanged. Consulting sits on top of the tester's craft, never in place of it.

CHAPTER TWO

Scoping and the Paper

The written authorisation is the thing that exists before any probe, not a form you tidy up afterwards. This chapter is the consultant's craft of producing that paper with a client, and the discipline of touching nothing until it is signed.

Volume 0 gave you the law and the shape of the rules of engagement: authorisation is binary, in writing, and in Sri Lanka a crossed line is a criminal one. This chapter does not re-teach that; it assumes it. What it adds is the consultant's side of the same fact. You are usually producing the paper with a client who has never commissioned a test before, who does not know their own asset list, and who would happily wave you at their systems on a phone call. The craft is turning that loose permission into a scope precise enough that both of you are protected, and then not touching anything until the signature is real.



Scoping is a conversation before it is a document

A client almost never hands you a clean scope. They hand you a sentence: "test our site," "make sure we are secure," "we are raising money and the investor wants a pentest." Your first job is to turn that into an exact list of what is in play. Which hostnames, which URLs, which IP ranges, which test accounts at which privilege levels. Where it runs: their own servers, a cloud account, a platform they rent but do not own. What a successful test even looks like to them, because "secure" is not a scope and "the login and the payment flow, because that is where our risk is" is.

Part of this conversation is helping the client discover their own attack surface, because they often cannot name it. The subdomain a contractor stood up two years ago, the staging box that is quietly reachable from the internet, the third-party widget embedded in checkout. You are not testing any of it yet. You are building the list together so that the paper describes the real system and not the tidy diagram the client keeps in their head. A scope written from the diagram misses exactly the forgotten host that was going to be the finding.

Out of scope is a gift, not a limit

New consultants treat the out-of-scope list as the boring half of the paper. It is the half that protects everyone. Naming what you will not touch is how the client sleeps while you work, and it is how you stay on the right side of the one rule when you trip over something tempting. The subdomain that is not listed is not yours, however loud it is. Out of scope binds exactly as hard as in scope; Volume 0 said it and it does not soften here.

This has a consequence you will meet on nearly every job. During the work you will discover an asset nobody put in scope: a linked admin panel on a sibling domain, a second environment you did not know existed. You record it as a finding, "this exists and is reachable," and you stop. You do not follow it. The discovery is the finding; the exploration is the crime. GP's own tooling is built the same way on purpose: a scanner that discovers an out-of-scope target logs it and refuses to traverse it, because the boundary is enforced in code, not left to good intentions. Hold yourself to the standard we build into the machine.

The window, the intensity, the blast radius

A scope is not only where, it is when and how hard. The window is the client's call and it goes in the paper: some want testing only at night, some not during a sale, some not on the one day a month that matters to them. Outside the window you do nothing, not even quiet reconnaissance. The intensity is agreed, not assumed: how loud you may be, whether you throttle, what you do when a service starts returning errors (you back off, you do not push through).

And a short list of things stays forbidden unless the paper names them as allowed, with separate and explicit consent: denial of service, anything destructive, changing or deleting real data, touching real customer records at scale, social engineering of staff, brute-forcing credentials. The default for all of these is off. A client who wants them tested has to say so in writing, understanding the blast radius, because these are the actions that take a business down. When in doubt, a technique is forbidden until the paper says otherwise. Prefer a benign proof over a destructive one every time: a canary that pings a host you control, a harmless marker, a short timing delay, never the impressive demo that causes the breach to make the point.

The signature has to be one the signer can give

A verbal yes on a call is not authorisation, and an email that says "sure, go ahead" from someone who may not have the authority to say it is not much better. Two things have to be true of the paper. It is written. And the person signing actually owns, or is properly authorised to grant access to, every asset in scope. This is the per-engagement representation you capture in writing, signed, before you begin: