



GHOST
PROTOCOL

TH

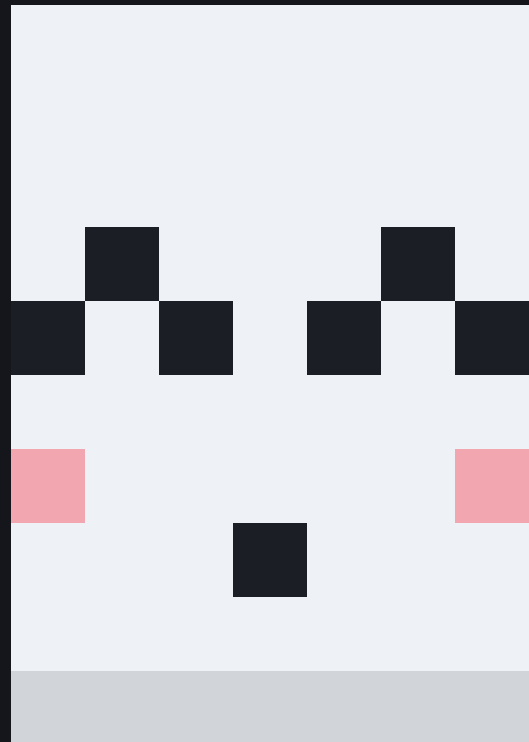
THE FIELD LIBRARY

DEFENCE

ASSUME THEY GOT IN

Threat Hunting and Forensics

Hunting versus alerting, the hunt loop, host forensics, network and log forensics, and turning a hunt into a detection.



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

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

THE GHOST PROTOCOL FIELD LIBRARY · DEFENCE

Threat Hunting and Forensics

Hunting versus alerting, the hunt loop, host forensics, network and log forensics, and turning a hunt into a detection.

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 BUILT

Bug Bounty BUILT

D · THE OTHER SIDE

Detection and Response BUILT

Threat Hunting and Forensics YOU ARE HERE

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

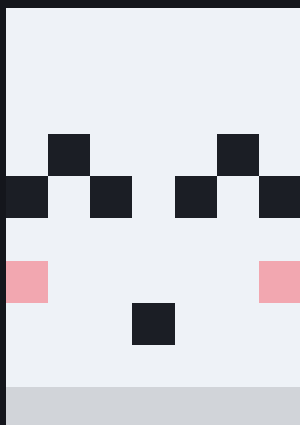
The Bestiary BUILT

The Glossary and the Ledger BUILT

Contents

- Opening 5
- 1** Hunting, and What Alerts Miss 6
- 2** The Hunt Loop 10
- 3** What the Host Remembers 15
- 4** The Wire and the Logs 21
- 5** From Hunt to Detection 25
- Worked answers 30

No alert Fired



The board was green. Every detection from the last track sat quiet, and Blink took the quiet for safety.

BLINK "Nothing fired. So nothing is wrong."

SOL "Or the one worth catching made sure nothing fired. The hunter does not wait for an alert, because the best intrusions are the ones your alerts were never written for. You start from the opposite assumption: that they are already inside, quiet, patient. Then you take a hypothesis about how, and you go looking in the data for the trace it would have left."

Worm's eye opened on the green board and did not trust it for a second.

WYRM "assume. then look."

This book is the hunt and the forensics under it: how to form a hypothesis, read the artifacts a host and a network leave behind, build the timeline, and turn what you find into a detection that will fire next time.

CHAPTER ONE

Hunting, and What Alerts Miss

An alert fires on something someone already knew to write a rule for. A hunt goes looking for the adversary no rule caught. This chapter sets the premise the whole volume stands on: assume the breach happened, and go find it.

Detection and Response sat you in the defender's chair and built the alerts. This volume starts where those alerts stop: you read what an intrusion left behind and go looking for the one your tools never flagged. Everything here is the same evidence discipline you already met in the verify-it rule, pointed at a different question: not "can I make this misbehave", but "did someone already, and would I even see it".

An alert Fires. A hunt goes looking.

An alert is a rule that someone wrote in advance. A signature in an intrusion detection system, a correlation rule in a SIEM, a behavioural detection in an endpoint detection and response (EDR) agent, a hash in an antivirus engine: each one is a sentence that says "when the data looks like this, tell me". It waits. When live data matches, it fires. That design is reactive on purpose, and it is bounded by exactly one thing, which is what the author already knew to describe. An alert can only catch what a person once thought to write down.

A hunt has no rule behind it. A person forms an idea about how an adversary might be operating, then goes into the data and looks for that activity directly, whether or not any detection exists for it. The work is proactive, and the unit of work is a hypothesis rather than a match. You are not waiting for the estate to tell you something is wrong. You are going to check a specific way it could be wrong that nothing is watching for.

The assume-breach premise

A hunt does not begin with "did an alert fire". It begins with a harder assumption: suppose a capable adversary is already inside, and suppose nothing fired. Where in the data would I see them anyway? That inversion is the discipline in one line. You stop treating a quiet console as proof of safety and start treating it as the thing to disprove.

The reason to work this way is the gap between when an intrusion starts and when anyone notices. That window, the dwell time, is often measured in weeks, sometimes much longer, and every day of it is time the adversary spends while the alerts stay green. A hunt exists to shorten that window by not waiting for a signature that may never have been written. You go looking before the tooling gets around to telling you.

Why alerting has a ceiling

Rules catch known techniques, and three kinds of activity slip under them. The first is genuinely new tradecraft that nobody has described yet, so no signature exists. The second is living off the land: the adversary uses tools already present and trusted on the host, `powershell.exe`, `certutil`, `rundll32`,

`regsvr32`, `schtasks`, or an admin tool the estate already runs such as PsExec, so nothing foreign ever lands and nothing looks out of place at a glance. The third is low and slow: activity deliberately paced and sized to stay under the thresholds the rules use. Your alert set is an accurate map of yesterday's known-bad. A hunt covers the parts of the ground the map does not draw.

What a hypothesis is

A hypothesis is a statement about adversary activity that is specific enough to be wrong. "Are we breached?" is not a hypothesis, because no query answers it and no result disproves it. Compare this: "an adversary living off the land would spawn `powershell.exe` from a Microsoft Office process, so in Sysmon Event ID 1 I should see `winword.exe` or `excel.exe` as the parent of `powershell.exe`." That names the behaviour, the data source, and the shape of the evidence, and a query over the logs either returns rows or it does not. You can act on the answer either way.

Hypotheses come from a few places. Threat intelligence hands you the techniques of a reported campaign and you ask whether you would see them here. The MITRE ATT&CK matrix (Chapter Two) hands you a technique and you ask the same. Crown-jewel analysis works backward from the data that matters and asks what an adversary would have to do to reach it. And sometimes an analyst simply notices an oddity and turns it into a question. The source matters less than the shape: if the statement cannot come back false, it is not a hypothesis, it is a mood.

A hunt is not a scanner

The pull every beginner feels, and the one Blink feels hardest, is to point a tool at the whole estate and read a verdict off the screen. A scanner has its place, but understand what it is: a set of questions somebody already encoded, which is the same ceiling as an alert. Hunting is analysis you perform over data you already hold. A tool can narrow a field of millions of events down to a handful of candidates, and that is genuine help, but the judgement about whether a candidate is an adversary or a tired administrator is yours. The tool surfaces; the analyst decides. This is the verify-it rule from the other volumes wearing a blue coat.

A hunt that finds nothing is still a result

A hunt that confirms no adversary has not failed. It produces four things worth keeping. A hypothesis disproven, written down so nobody re-runs it blind next quarter. A new detection, so the thing you just looked for by hand now fires on its own (Chapter Five is entirely this). A data gap found, which happens when you go to look and the logs were never collected or had already aged out, and that absence is itself a finding worth reporting. And documented coverage: you can now say, truthfully and with evidence, exactly what you checked and over what window.

Hold the line on that last point. Absence of evidence is a real result only when you can prove the data actually covered the question. A null result reported honestly ("we tested this hypothesis over this window with these sources and found nothing") is professional work. The same null dressed up as "the environment is clean" is not, because you checked one hypothesis over one window with whatever data you happened to have, and that is a long way from clean. The honest writeup in Chapter Five is built on exactly this distinction.

**SOL SAYS**

A hunt that finds nothing is not a failed hunt. I want you to feel that in your chest before you run your first one, because the opposite belief ruins hunters: they go looking for a trophy, and when the data does not hand them one they either wander off scope chasing a feeling, or they quietly inflate a coincidence into a finding so the day was not wasted. The failure is never the empty hunt. The failure is the hunt that claims more than it actually checked. Your credibility is the one tool in this volume you cannot reinstall.

■ EXERCISE

On your own machine, on a lab you own and nothing else: stand up a Windows virtual machine, install Sysmon with a sound logging configuration (the widely used SwiftOnSecurity sysmon-config is a reasonable starting template), and generate some ordinary activity. Then write one hypothesis in the form above, naming a behaviour, the Sysmon event that would record it, and the rows you expect, and query your own logs for it with a tool like EvtxECmd or Chainsaw. Run it once when you have done nothing suspicious, and again after you deliberately trigger the behaviour yourself. You are learning the loop on a system that is entirely yours, which is the only place to learn it.

■ VERIFY IT

Never accept a tool's "no matches" as "not present" until you have proved the query could have found it. A query that returns nothing because the data was never collected looks identical to a query that returns nothing because the adversary was not there, and the two mean opposite things. Before you record a null result, confirm you queried the right log, over a time window that actually contains the period in question, with that logging switched on for that period. A null you cannot stand behind is not evidence, it is a blank you filled in with hope.

■ THE LINE

You hunt only in an environment you are authorised, in writing, to hunt in. The law has not changed and it is not re-taught here: it is Volume O, in full. Hunt data is real data about real people, often including message content, file contents, and credentials, so authorised scope, lawful handling, and data minimisation apply from the first query you run, not from the first thing you find. A defensive seat is not a licence to read everything in the house. Know what your authorisation covers before you open the logs.

■ ON YOUR FEET

Your manager points at a quiet EDR console and says, "nothing has fired in three weeks, so we are clean, there is no need to hunt." Based only on what this chapter established, what is the flaw in that sentence, and what single hunt would you propose running first, and why that one?



WHAT WYRM KEEPS

Alerting matches what someone already knew to write a rule for. Hunting is a person testing a hypothesis about the adversary the rules missed, starting from the assumption that the breach already happened.

- Hunting is hypothesis-driven and proactive. It begins from assume-breach, not from whether an alert fired, and it exists to shorten the dwell time the alerts let run.
- Alerting has a ceiling set by its authors: new tradecraft, living off the land, and low-and-slow activity all slip under rules written for yesterday's known-bad.
- A hypothesis is a statement specific enough to come back false. It names the behaviour, the data source, and the shape of the evidence a query would return.
- A null result is real only once you have proved the data covered the question. Report absence of evidence as exactly that, and never let it become an unearned "all clear".

CHAPTER TWO

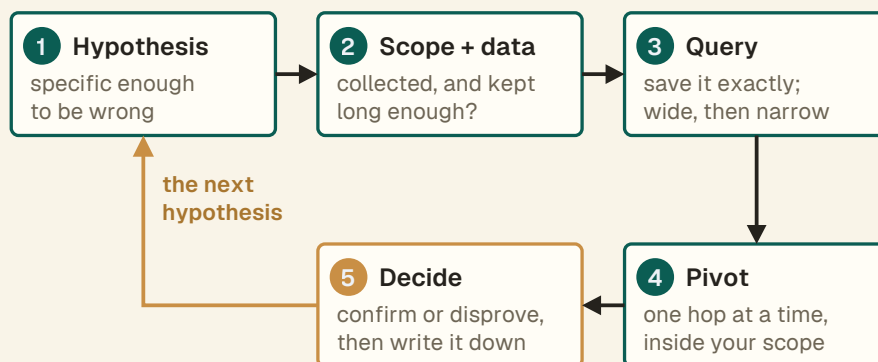
The Hunt Loop

A hunt is a loop you can run deliberately: pick a hypothesis, scope and gather the data, pivot from one indicator to the next, confirm or disprove, and document. This chapter gives you the loop and the habits that keep it honest.

Chapter One said a hunt is a hypothesis you test against data. This chapter turns that into a procedure you can run the same way every time, so the quality of a hunt comes from the method and not from whether you felt sharp that day. The loop is not a Ghost Protocol invention. It is the shape the field settled on, and two public versions of it, the Sqrrl hunting loop and the PEAK framework, are worth reading alongside this chapter.

The loop, in five moves

Strip the published models down and the same five moves are underneath all of them. One, state a hypothesis. Two, scope it and find the data that would answer it. Three, gather and query that data. Four, pivot, following each indicator to the ones connected to it. Five, decide: confirm or disprove, then document and hand the result to a durable detection. It is a loop because the fifth move feeds the first: a disproven hypothesis sharpens the next one, and a confirmed one spawns three more about how far the adversary went.



every hunt leaves one of: a new detection,
a data-gap ticket, or a written null

Five moves, run as a loop: every answer, confirmed or disproven, feeds the next hypothesis.

Two framings in the open literature map onto this cleanly. Sqrrl's threat-hunting loop runs create-hypothesis, investigate-with-tools, uncover-new-patterns, inform-and-enrich, and pairs with its Hunting Maturity Model, which grades a team from HMM0 (relying primarily on automated alerts) up to HMM4 (hunts automated into detections). Splunk's PEAK framework, from David Bianco, organises the same work as Prepare, Execute, and Act with Knowledge, and names three hunt types: hypothesis-driven, baseline (hunting for what is abnormal against a known-normal), and model-assisted. Read them. This chapter is the working version of the same idea.