# The PSIRT Field Guide

*Volume 1: Building and Running the Function*

Sylvan Press

Sylvan Press

*Field-tested, plain-English references for the people who do the work and stand behind their call.*

# Contents

**The PSIRT Field Guide**

*Volume 1: Building and Running the Function*

Published by Sylvan Press, the security imprint of Sylvan Assurance, LLC — Vermont, USA.



ISBN (paperback / hardcover / ebook): to be assigned prior to publication.

Print and ebook ISBNs are registered through Bowker before release.

First edition, 2026.

**Disclaimer.** This book provides general guidance and recommended security and compliance practices, not legal, financial, or professional advice, and it does not create any advisory or consulting relationship between the author, the publisher, and the reader. Every recommendation is optional — a widely accepted way to reduce a common risk — and the reader decides which to adopt, adapt, or decline. Following this guidance reduces exposure to common risks but does not guarantee any particular outcome and does not prevent any particular breach, incident, or loss. Where this book refers to laws, regulations, standards, or commercial products, it does so for general awareness only; confirm specific obligations with qualified counsel, and note that product references are illustrative and do not constitute endorsement.

Sylvan Press — sylvanassurance.com

# Preface

Somewhere in your inbox — or in the inbox you are about to create — is the first report. It may come from a researcher who found something real, a customer who noticed something odd, or a scanner that finally got lucky. The report is not the problem. The problem is everything the organisation does next, and whether that happens by design or by improvisation.

The practices in this book come from more than two decades of security practice inside a top-25 global company — years of it in a product security incident response function, and network security and incident response before that. A PSIRT built from the ground up — stood up alongside engineering organisations that did not yet have a model for one, then matured into a documented programme with defined roles, an intake pipeline, and measurable outcomes — leaves a residue of practices that held. This book is that residue: not everything that was tried, but the things that survived contact with an active vulnerability programme.

It builds the function from the ground up: what a PSIRT actually is and is not, the response playbook underneath most incidents, the workflow that carries a report from inbox to closure, the function built from scratch, the first seventy-two hours of a serious incident, and then the recurring crafts — researcher relationships, coordinated vulnerability disclosure, third-party and software-bill-of-materials practice, CVE and CVSS, the advisory library, crisis communications, forensic

preservation, and the cyber-insurance reality. The hiring, capability, and metrics set rides along as an appendix, because a function is eventually made of people and numbers.

Three recurring teams carry the worked examples, chosen to span the realistic range: Heron Analytics, a software company meeting its first researcher report; Mariner Networks, a network-appliance vendor with downstream customers and a component crisis waiting to happen; and the Wren Project, an open-source effort run on goodwill and three maintainers' evenings. One of them is probably close enough to your situation that you will recognise yourself.

A word on what the book promises. The practices here are framed as approaches that reduce risk and have been seen to work in real product-security operations — not as actions that guarantee any outcome, regulatory, contractual, or otherwise. Nothing here is legal advice; disclosure decisions, regulatory notifications, and contract questions belong with qualified counsel. The landscape this work sits inside keeps moving, so verify current requirements against authoritative sources before relying on specifics. What the book offers is the shape of a function that holds — and the judgement that keeps it holding when the report in the inbox is a bad one.

# How to Use This Book

### 0.0.1 The shape of this volume

**Part I — Foundations (Chapters 1 through 6).** What a PSIRT is and is not, the response playbook, workflow and escalation, the intake pipeline, building the function from scratch, and the first seventy-two hours. Read Part I in order if you are new to the work.

**Part II — The Recurring Crafts (Chapters 7 through 15).** Researcher relationships, coordinated vulnerability disclosure in depth, third-party risk and SBOM, CVE and CVSS mastery, the advisory library, communication and crisis playbook, forensics and evidence preservation, cyber insurance and liability, and the legal dimension — privilege, counsel, and the disclosure decision. Largely self-contained; read them in the order your week demands.

**Appendix A — Hiring, Capability, and Metrics.** Role definitions, the hiring kit, the vendor capability scorecard, and the metrics dashboard. Open it when the budget conversation starts, which is usually one quarter before you expected.

**The PSIRT set.** This book is complete on its own: it builds and runs the function, from the playbook and workflow through researcher relationships, coordinated disclosure, SBOM and third-party practice, CVE/CVSS, the advisory library, communications, forensics, and insurance, with the hiring and metrics set as Appendix A. Two companion books extend the practice for readers who want more. *Advanced Practice and the Hard Cases* covers bug bounty operations, AI/ML

and open-source contexts, tabletops, pre-disclosure lists, the sustained case studies, and the maturity model. *The PSIRT Operator's Workbook* holds the templates, decision trees, the multi-jurisdiction regulatory reference, the template finder with tier manifests, the maturity rubric, and the glossary. Each stands alone; none assumes you have read the others. The complete set is under $30 in ebook.

If a report is sitting in your inbox right now: Chapter 2 is the playbook, Chapter 3 is the workflow, and the *Operator's Workbook* has the triage flowchart and the response templates. Come back to the rest when the loop is closed.

# About This Book

A few conventions are worth flagging. Acronyms are spelled out at first use in each major section, not only at first use in the book — readers skim and jump, and a single first-use spell-out at page one is not enough. The Request for Comments 2119 (RFC 2119) keywords MUST, SHOULD, and MAY appear in capitalised form only where the book is quoting or modelling policy excerpts; elsewhere the book uses ordinary prose. Single quotes appear inside double quotes. UK spelling is used only for ‘organisation,’ ‘recognised,’ and ‘behaviour’; everything else follows US conventions. Where the book recommends a practice, the recommendation is framed in the defensibility language described in the Preface. The three recurring teams — Heron Analytics, Mariner Networks, and the Wren Project — and the people inside them are fictional; the patterns they illustrate are not.

Chapter 1 begins on the next page.

# 1 What PSIRT Actually Is

On a Tuesday morning in October, a senior engineer at a mid-sized SaaS vendor opened an email that began with a sentence she had seen many times before, and would see many times again: "Hi — I think I found a security issue in your product. Who do I report this to?"

She did what most senior engineers do. She forwarded it to `security@`, added a short note — "Looping in the right team" — and went back to her sprint board. She had a deploy at noon and a one-on-one at one. The email left her inbox and entered the part of the organisation where things go to be handled by somebody else.

It was not handled by somebody else. It was handled by no one for three days.

`security@` at that company was a distribution list that, at some point in the last decade, had been pointed at four people. Two had left. One was on parental leave. The fourth was a security architect who read the message, assumed it was about the corporate VPN, and replied to ask the reporter for "your employee number and the device you're using." He got back a polite, increasingly puzzled response from someone who did not work at the company and had been trying to report a privilege-escalation bug in the product's API.

The exchange went round in circles for the better part of a week. The reporter — a freelance researcher in Lisbon who had found the issue while building an integration for a paying customer — eventually escalated the only way researchers know how to escalate: a public

tweet, politely worded, with a screenshot of the unanswered emails. Within forty minutes, a person whose email signature read "Director, Product Security Incident Response Team" had replied, apologised, opened a HackerOne report on behalf of the researcher, and started a Slack channel called `#vuln-acme-api-oct`. The fix shipped eleven days later. The advisory went out three days after that. Nobody got hurt. The researcher got paid. The company added a line to its onboarding deck about not assuming `security@` knows what to do with a vulnerability report.

If you have worked in product security, you recognise this story. You may have lived it from the reporter's side, the engineer's side, or — most likely, given the book you are holding — from the director's side, watching the public tweet land and feeling the particular cold that comes with knowing the clock has already been running for three days.

This book is about the work that director does. It is about the function that catches the email when `security@` cannot, the people who decide what to do with it, and the small but consequential operational choices that determine whether the next eleven days end with a clean fix and a paid researcher — or with a regulator's letter, a customer escalation, and a paragraph in someone's threat report.

That function is PSIRT. And the first thing worth getting straight is what it actually is.

### 1.0.1 PSIRT is the team that owns vulnerabilities in the products you ship

A Product Security Incident Response Team — PSIRT — handles vulnerabilities in the products an organisation sells, licenses, or distributes to other people. That is the whole definition. Everything else is consequence.

The work has a particular shape. Reports come in from outside — from researchers, from customers, from bug bounty platforms, from CERT/CC, from a colleague who saw something on Twitter — about a flaw in the product. PSIRT triages the report, decides whether it is real, coordinates with engineering on a fix, manages the disclosure, and lives with the consequences. The lifecycle is well-trodden ground;

ISO/IEC 30111 has been describing it formally since 2013, and the FIRST PSIRT Services Framework has been refining the operational picture for nearly as long. None of this is new. What is new, in most organisations, is the volume.

PSIRT is not the only team in the building with the word "incident" in its job description, and the overlapping vocabulary is part of why the function is so often misunderstood inside its own company. So before going any further — a clarification, because the rest of the book depends on you and me drawing the lines the same way.

A Computer Security Incident Response Team — CSIRT — handles security incidents in the organisation's own corporate environment. Compromised employee laptops. Phishing campaigns against staff. Unauthorized access to the internal Jira. Ransomware on a finance share. CSIRT's customer is the organisation itself. When a report lands that says "I'm in your AWS console and I can see your production database credentials," that is CSIRT's report, not PSIRT's. The asset under attack is yours.

A Security Operations Centre — SOC — runs detection and response against the live threat environment. Watches the telemetry. Triages the alerts. Hands off confirmed incidents to CSIRT. In many organisations, the SOC and CSIRT are the same set of people wearing different hats depending on the time of day; in larger organisations they are distinct functions with distinct headcount. Either way, the SOC's domain is what is happening, right now, against your own perimeter.

A PSIRT's customer is your customer. The asset under attack — or, more often, the asset *capable* of being attacked, since most reports are not exploitation but potential — is a product the company sold to someone else. When the report lands that says "I'm logged into my own account in your product and I can read other tenants' data," that is PSIRT's report. The reporter is using the product the way the company intended; the flaw lets them do something the company did not intend. The fix has to be made by engineering, shipped to all affected customers, and ideally installed by them, before the vulnerability stops mattering. None of that is true for a CSIRT incident, where the fix is mostly an internal action and the customer never has to do anything.

I have watched senior security leaders — people who run sophisticated SOC operations and have written CSIRT runbooks I have learned from — sit through their first PSIRT incident and visibly recalibrate. The shape is different. The customer is not the attacker. The fix is not the end. The clock is not yours.

That last point is the one most worth dwelling on, because it is what makes the work distinct.

### 1.0.2 The boundary cases

The clean definition — PSIRT owns flaws in what you ship, CSIRT owns attacks on what you run — sorts most reports in seconds. The reports that consume disproportionate energy are the ones that straddle the line, and a function that has not thought about them in advance relitigates the boundary in the middle of every one. Four shapes recur.

The marketing-website report. A researcher reports cross-site scripting on `www.heron.example` — the brochure site, not the product. By the asset test this is CSIRT's: the site is corporate infrastructure, the fix is an internal action, no customer deploys anything. But the reporter came through the VDP — the published Vulnerability Disclosure Policy that invites and protects reports like theirs — expects the VDP's treatment, and does not care about the org chart. The working rule at Heron — the mid-size SaaS vendor you will meet properly a few pages on — is *route by the asset, hold the relationship where it landed*: the PSIRT keeps the researcher conversation — acknowledgement, updates, credit, bounty decision under the programme's scope terms — while the corporate security team owns the technical response. The alternative, bouncing the reporter to a different team mid-conversation, reliably reads as a brush-off, and the researcher who feels brushed off is the researcher who writes the thread.

The product-on-infrastructure report. A report says "your product is leaking other tenants' data," and validation traces the cause not to product code but to a misconfigured storage bucket in the production cloud account. The asset is corporate; the impact is customers'. This is not a routing decision at all — it is both functions' incident, run jointly: CSIRT drives containment and forensics in the environment, PSIRT drives the customer-impact assessment, the notification question, and

the external story. The failure mode is sequential handling, where the infrastructure team quietly fixes the bucket and considers the matter closed while nobody asks the PSIRT questions — who was exposed, for how long, and what do we owe them.

The published open-source report. Heron maintains two open-source client libraries on its public GitHub. A flaw in one is a product vulnerability in every sense that matters — external users, versions, advisories — even though the code is free and the "customers" never signed anything. If the organisation publishes code that others run, the PSIRT's remit covers it, scaled to its weight: a `SECURITY.md`, an advisory channel, a disclosure path. The trap is the repo nobody remembered was public, with a contact file pointing at an engineer who left; an annual inventory of published code is cheap insurance.

And the professional-services artifact. A consultant's deployment script, a sample integration in the docs, a Terraform module in a solutions-engineering repo — artifacts the company handed to customers without ever calling them products. When one carries a flaw, the instinct is to say "not a product, not ours," and the instinct is wrong in the way that matters: the customer got it from you, ran it because they trusted you, and is exposed because of you. The response runs lighter — there may be no version range, no advisory feed, sometimes just direct outreach to the customers known to have received it — but it runs.

A fifth shape arrives less often and rearranges more when it does: the acquisition. The company buys a product, and with it a codebase, a customer base, and a vulnerability-handling history that may range from a mature advisory archive to a `security@` alias pointed at a founder who has already left. From the closing date, the acquired product's flaws are the acquirer's PSIRT's flaws — researchers and customers will not wait for the integration roadmap, and the first post-acquisition report routinely arrives before the engineering teams have even met. The questions that need answers in the first week, not the first quarter: where do the acquired product's reports land now, and who watches that channel; does the acquired company's VDP still apply, and do its safe-harbor promises bind the acquirer (counsel's question, but the PSIRT has to ask it); who owns the acquired product's ex-

isting advisories and CVE records, which do not migrate themselves; and what does the acquired codebase's dependency picture look like, since the acquirer has just inherited every unpatched transitive exposure in it. Heron has been on the small end of this once — acquiring a four-person company whose entire disclosure history was one politely-handled report in a Gmail label — and the integration checklist that came out of it sits in Appendix A. The pattern to avoid is the gap: the months in which the old channel is unwatched, the new channel is unannounced, and the researcher who finds the seam concludes, correctly, that nobody is home.

None of these cases is exotic. A mid-sized vendor sees the first four within a couple of years, and the fifth whenever corporate development has a good quarter. The point of deciding the boundary policy in advance — in writing, with the CSIRT lead's agreement — is that boundary arguments conducted during an incident are conducted at the worst possible time, in front of an audience.

### 1.0.3 Why product security incidents are uniquely hard

There are at least four properties of product security incidents that, taken together, set the function apart from corporate incident response. Reasonable people order them differently; this is the order I have come to use.

The first is that **the customer is not the attacker**. In corporate incident response, when someone is doing something in your environment they should not be doing, your operating assumption is adversarial. You respond accordingly — containment, isolation, forensics, eventually attribution. In product security, the person doing the thing they should not be doing is often a paying customer using the product as designed, who has stumbled onto a flaw. Sometimes it is a researcher who has been poking deliberately. Sometimes it is another customer's red team. Almost never is it the threat actor model your CSIRT trained against. The triage move that works in CSIRT — "lock the account, preserve the evidence, escalate" — is the exact wrong move in PSIRT. Locking the researcher's account is how you get the public tweet.

The second is that **the fix is a release, not an action**. When CSIRT contains a phishing campaign, the work is done when the malicious

mail is purged and the credentials rotated. When PSIRT confirms a vulnerability, the fix exists in three stages: developed, released, and deployed. The first stage is on engineering. The second stage is on the release process. The third stage is on the customer — and in many environments, the customer cannot be compelled to deploy. Most enterprise SaaS deployments are managed; you push the update and it is live. Some are self-hosted, on-prem, air-gapped, embedded in industrial systems with maintenance windows measured in quarters. The vulnerability exists, in those environments, until the customer chooses to act on the advisory. The advisory has to be written with that in mind.

The third is that **coordinated disclosure is a multi-party negotiation**. The reporter wants credit and, increasingly, a payout. The vendor wants time to fix and a clean message. Customers want notification with enough lead time to prepare, but not so much lead time that the embargo leaks. Regulators want notification within statutory clocks that vary by jurisdiction and by category of data. Other downstream vendors who incorporate your component want pre-disclosure access. The press wants an angle. None of these parties have aligned incentives; PSIRT's job is to keep the disclosure on a timeline that holds all of them, which is mostly a matter of communicating clearly with each party while not communicating anything inconsistent across parties. It is much closer to a deal desk than to a CSIRT runbook.

The fourth is that **legal exposure shifts with disclosure timing** in ways that are not always obvious. Disclose too early, before customers can patch, and you have created a window in which exploitation is plausible and your own advisory is the road map. Disclose too late, after the researcher has lost patience, and you have undermined your own Vulnerability Disclosure Policy — which is the artifact that gives the researcher safe harbour and gives you a defensible position if it ever becomes a lawsuit. Disclose without enough internal alignment, and counsel finds out about a public advisory from the news. Disclose with too much internal alignment, and the advisory is so heavily lawyered that customers cannot tell what they are supposed to do. The job is to find the band that satisfies the disclosure obligation, gives customers a useful artifact, gives the researcher the credit they earned,

and does not create the kind of legal exposure that a competent plaintiff's attorney could later argue was foreseeable.

There are other properties — the asymmetry between the vendor's information and the customer's, the way Software Bill of Materials (SBOM) transparency is reshaping triage, the long tail of older product versions still in use — and the rest of the book picks at them in turn. But these four are the structural ones. If you find yourself explaining to a new hire why "we can't just do what CSIRT does," these are the four properties to point to.

### 1.0.4 The three teams you will meet again

The rest of this book uses three recurring case studies. They are composites — none of them is a real company — but they map closely onto organisations I have worked with or alongside, and they are sized and structured the way real PSIRTs are sized and structured. Most chapters use one of them as the worked example. A few use all three. By the time you finish the book, they will be familiar enough that I can say "Heron's triage queue on a Tuesday afternoon" and you will have a picture.

**Heron Analytics** is a mid-sized B2B SaaS vendor. About two hundred and eighty employees, fifty engineers, headquartered in Boston with a few remote-friendly hubs. They sell an analytics platform — SQL and a natural-language query layer that talks to the customer's data warehouse — to roughly eleven hundred paying organisations. Around three hundred and forty of those customers are on the higher tier that includes the GenAI feature. Heron is Series C, pre-IPO, the kind of company whose security posture is real but uneven; they have shipped good things and have things they know they need to fix.

Heron's PSIRT started two years ago as a part-time job for one engineer. It was formalised a year ago, after a near-miss incident in which a researcher's report sat unread in a shared mailbox for nine days and was eventually escalated by a customer. Priya Sundaram, the security engineering manager, runs the function. Marcus Chen is her one full-time PSIRT engineer; the rest of the security team rotates through a part-time triage shift. They publish advisories under the prefix `HSA-` — Heron Security Advisory — with a year and a sequence number. They

run a HackerOne bug bounty programme with a payout band that tops out around $20,000 for the rare critical, and they have a published Vulnerability Disclosure Policy that gives researchers safe harbour for good-faith research. They do not yet have an SBOM. They are in the messy middle of the maturity model — past ad hoc, not yet industry-leading — and most readers of this book will recognise the texture of their work. When I show you a worked example in the chapters that follow, it is usually Heron. Their work is the work most readers do.

**Mariner Networks** is a large enterprise vendor. Roughly five thousand two hundred employees globally, publicly traded, headquartered in San Jose, with development sites in Bangalore, Sofia, and Bracknell. They have been selling enterprise networking gear — switches, routers, SD-WAN platforms, security appliances — for more than two decades. Their PSIRT has existed for fifteen years; it was set up in response to a major firmware vulnerability that became a news story, and it has been continuously staffed since. Camila Reyes, the Director of Product Security, runs a team of six PSIRT engineers organised by product line, a dedicated technical writer for advisories, and a regulatory specialist who handles notifications across the US, EU, UK, Japan, India, and Brazil. They publish advisories under `MN-PSIRT-` and ship machine-readable CSAF and VEX feeds alongside the human-readable versions. They run a Bugcrowd programme that they are in the middle of migrating to HackerOne. They have a pre-disclosure list of roughly forty strategic customers under NDA, plus CERT/CC and several national CSIRTs. Mariner is what a Tier 4 PSIRT looks like in practice, and I use them when the point of a chapter is to show good — what mature processes feel like when they are working, and where the lifts are when you are trying to grow into them. They run lean for their size. Mature does not mean overstaffed.

**Wren Project** is the open-source case. It is a single Rust crate, `wren-crypto`, providing high-level cryptographic primitives — AEAD, key derivation, password hashing — for application developers who do not want to learn the underlying APIs. Six years old, started as a side project, now a transitive dependency for hundreds of crates including a few that are in production at companies whose names you would recognise. Sam Thackeray maintains it. He lives

in Manchester, has a two-year-old at home, and works full-time as a senior engineer at a UK fintech. The four trusted contributors who have merge access do not actively triage incoming reports. There is a `SECURITY.md` file in the repo pointing to a GPG-encrypted email, and there is Sam, on Tuesday evenings, after the toddler is asleep. The advisory channel is RUSTSEC. The PSIRT maturity tier is one, in the sense that there is one person. Wren is in this book because most open-source security work is closer to Sam than to Camila, and ignoring that reality would be writing about a function that does not match what the world has. When a chapter is about open-source software (OSS), or about what scaling-down looks like, it is about Wren.

These three teams will turn up in worked examples throughout the book. They have relationships with each other — Heron uses several Rust crates indirectly in its data plane, and there is a scenario later on where a vulnerability in `wren-crypto` forces Heron's PSIRT to determine whether they are transitively exposed; Mariner runs an information-sharing relationship with several mid-market vendors including Heron — but I will introduce those threads when they come up. For now, file the three teams away. They are the anchor.

### 1.0.5 Where this book fits in the standards landscape

It is worth being explicit about which standards and frameworks this book sits next to, because a few of them will come up so often that it would be inefficient to re-introduce each time.

The FIRST PSIRT Services Framework is the operational reference. If you have not read it, do; it is a freely available document that describes the services a PSIRT provides, organised by service area, with maturity indicators. The PSIRT Services Framework is to product security what NIST SP 800-61 is to incident handling — not a complete operating manual, but a map of the territory that everyone in the field shares. I will reference it by name throughout the book, and when I do, I will assume you can look up the relevant service area if you want the framework's own framing.

ISO/IEC 29147 covers the receiver-side process — how an organisation handles externally reported vulnerabilities. ISO/IEC 30111 cov-

ers the internal handling process — what happens once a report is accepted. The pair are the international standards underlying coordinated vulnerability disclosure. Some readers' organisations will be certified against them; most will not. Either way, the standards are useful as a shared vocabulary and as a defensibility reference when a customer or regulator asks how the disclosure process works.

One operational institution deserves a named place beside the standards: the CVE programme and the Common Vulnerabilities and Exposures Numbering Authority (CNA) system. A CNA is an organisation authorised to assign CVE identifiers for vulnerabilities in a defined scope — typically its own products. Mid-sized vendors increasingly become CNAs for a practical reason: the vendor that does not control its own CVE records is the vendor whose vulnerabilities get described by whoever filed first, in whatever wording and severity the filer chose, on the filer's timeline. CNA status puts the record's accuracy and timing in the PSIRT's hands — Heron became one eighteen months into its formalised existence, after exactly one experience of disputing a third-party-filed record after publication. The status is not free: it carries process obligations — assignment rules, record quality, responsiveness — and a CNA that lets its records go stale has traded one credibility problem for another. The chapter on CVE and CVSS treats the mechanics; the point here is that the CVE record is part of the disclosure surface the PSIRT owns, whether or not it has claimed the authority to write it.

There are other relevant standards — NIST SP 800-218 (SSDF) for secure development, NIST CSF for risk-management framing, ISO/IEC 27001 if your PSIRT lives inside an ISMS, the various sectoral regulations the regulatory chapter covers in depth — and the book engages them where they come up. None of them is the spine of the book; the spine is the work.

### 1.0.6 What this book is not

It is worth being explicit about that, too.

This is not a textbook. If you do not already know what a CVE is, what CVSS is meant to measure, what coordinated disclosure means and why it is structured the way it is, the book will not slow down to

explain. The glossary in the supporting *Operator's Workbook* — Volume 3 of this set, alongside the templates and runbooks — will give you the terms as you go. The book itself assumes you have done incident response before — perhaps not in the product context, but somewhere — and you are here to understand how the product context changes the work. And if that is not quite you — if you are the founder or solo operator whose first researcher email arrived this week and who could not have defined PSIRT last month — stay anyway: read this chapter and the next with the glossary at your elbow, then jump to whichever operational chapter your week demands. The pace will not slow down, but nothing here is beyond a practitioner willing to look up a term.

This is not a compliance checklist. Several chapters cover regulatory obligations, because PSIRT operates under more of them every year and the operational consequences are real. But the framing is "here is what the obligation is asking you to be able to demonstrate, and here is how that flows into how you run the team," not "here is the bullet list of requirements you can hand to your auditor." If you want the bullet list, the regulators themselves publish it, and your counsel can interpret it for your specific posture. What I can give you is the operational shape — what good looks like when the regulation lands on your desk.

This is not a beginner's guide. There are good beginner's guides; this is not one of them. The reader I have in mind is somewhere between a new PSIRT engineer wanting to understand how the function fits together and a security leader being asked to stand a PSIRT up, scale one, or repair one. If you are running an existing PSIRT and looking for the chapter that addresses the specific operational pattern that is currently keeping you up at night, the book is organised so that you can read chapters out of order and most will stand on their own. There is a recommended reading order, but it is recommended, not required.

What this book is — and what the rest of the chapters will demonstrate — is a field guide. I have spent the better part of two decades doing this work or near it. I have run the incidents, written the advisories, sat in the rooms with counsel, negotiated the disclosure timelines, written the post-incident reviews, and watched the patterns that repeat across organisations of different sizes and product types. The book is an attempt to put those patterns down in a form that another

practitioner can use. Some of the advice is opinionated. Where it is, I will say so. Where the field has reasonable disagreement — and there are several places where it does — I will note the disagreement and tell you which side I land on and why.

The voice throughout is practitioner-to-practitioner. I am writing this on the assumption that you and I would understand each other in a Slack channel. The aim is not to convince a sceptical executive that PSIRT is important; the aim is to help you do the work better than you did it yesterday.

### 1.0.7 What comes next

The arc of the book moves from foundations to the recurring crafts. Part I — the part this chapter opens — establishes the framing and the mental models, then follows the operational spine: the playbook, workflow and escalation, the intake pipeline, the build from scratch, and the first seventy-two hours of a serious incident. Part II takes up the crafts a running function returns to again and again — researchers and disclosure, the supply chain, severity and advisories, communications and forensics, insurance and the law — each treated in depth, most of them self-contained. Appendix A holds the hiring, capability, and metrics set.

The next chapter, "The Response Playbook," picks up where this one leaves off. If this chapter has been about what the function is, the next is about how the function moves — the canonical lifecycle, the phases, the artifacts produced at each step, the roles that hold the work. By the end of Chapter 2 you should have a working mental model of the operational rhythm; by the end of Part I you should have the framing to read the rest of the book in whatever order is most useful to you.

Heron's `#psirt-active` Slack channel is going to be busy in the next chapter. So is Camila's pre-disclosure list. Sam's GPG inbox is going to chime on a Tuesday evening, and he is going to weigh whether to open it before bed or in the morning, and he is going to make the choice most maintainers make.

The work is the work. The patterns are recognisable. The first thing is to get the shape of the function clear, which is what this chapter has

tried to do.

The second thing is to get the moves right.

---

*Recap: PSIRT owns vulnerabilities in the products you ship — distinct from CSIRT, which owns your corporate environment, and from the SOC, which owns live threats. The boundary cases — the marketing-site report, the product-on-infrastructure incident, the published open-source repo, the professional-services artifact — are sorted by routing on the asset while holding the researcher relationship where it landed, under a boundary policy agreed before it is needed. Product security incidents are structurally harder than corporate ones because the customer is not the attacker, the fix is a release rather than an action, disclosure is a multi-party negotiation, and legal exposure shifts with timing. The book uses three recurring case studies — Heron, Mariner, and Wren — and sits alongside the FIRST PSIRT Services Framework and ISO/IEC 29147 and 30111.*

*Next: Chapter 2 — The Response Playbook.*

*You've just read Chapter 1 of*

# The PSIRT Field Guide

The full book runs 328 pages across 15 chapters,
with a complete back-of-book index.

*Published by Sylvan Press, the book imprint of Sylvan Assurance — field-tested, plain-English references for the people who do the work.*

Scan to see the full book — and where to buy it.
**sylvanassurance.com/go/psirt**

*Not ready for the full book? See where you stand first — free, a few minutes:*

**sylvanassurance.com/go/free-psirt-response**