# Vulnerability Management

*A Practitioner's Field Guide — Volume 1: Understanding and Prioritising*

Sylvan Press

Sylvan Press

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

# Contents

**Vulnerability Management**

*A Practitioner's Field Guide — Volume 1: Understanding and Prioritising*

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

The scanner is not the programme. That sentence pays for this book, and most organisations learn it expensively — buying a tool, pointing it at the estate, and discovering that a weekly report of ten thousand findings is not a vulnerability management programme but a source of guilt.

Much of the judgement in this book was formed over more than two decades inside a top-25 global company, on the incident-response and product-security side of the work — the end of the pipe where the vulnerabilities that were not prioritised, or not patched, arrive as somebody's bad week. Vulnerability management is not, at its core, a tooling problem. It is a judgement problem: deciding which of the thousands of things a scanner can find actually matter, in what order, on whose authority, and against which clock.

This book builds that judgement layer. It covers what the discipline actually is, the asset inventory everything else stands on, the lifecycle from discovery to verification, and the maturity spectrum for locating a programme honestly. Then it works through the prioritisation stack — CVE and CVSS without the folklore, the Known Exploited Vulnerabilities catalogue and the Exploit Prediction Scoring System, business context, the severity-versus-risk distinction that trips up most programmes, and the patching ladder that turns priority into a cadence you can sustain.

Four recurring organisations carry the worked examples, chosen to

span the range of who actually runs this work: Hollybush & Cale, Saltmarsh & Linden, Tamarind Valley, and Cinderhaus. Each meets the same findings with very different resources, and what counts as a defensible decision differs accordingly. 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 operations — not as actions that guarantee any outcome. The aim is a programme you can run on an ordinary week and defend on a bad one — not a zero on the dashboard, which no real programme reaches, but a defensible order of operations that closes what matters before it is exploited.

# How to Use This Book

### 0.0.1 The shape of this volume

**Part I — Foundations (Chapters 1 through 5).** What VM is, asset inventory, the lifecycle, the maturity spectrum, and VM-PSIRT. Read in order if the discipline is new to you.

**Part II — Prioritisation (Chapters 6 through 10).** CVE/CVSS, KEV/EPSS, business context, severity versus risk, and the patching ladder.

**Part III — From Priority to Proof (Chapters 11 through 16).** Exploitability analysis, application and code-level vulnerabilities, the attacker's view, risk-scoring systems compared, threat intelligence for VM, and external findings intake.

**The Vulnerability Management set.** This book is complete on its own: it builds the judgement layer, from the inventory and the lifecycle through the full prioritisation stack to exploitability, the attacker's view, and external findings. Two companion books extend the practice for readers who want them. *Running the Programme* covers scanners, patching, SLAs, exceptions, metrics, and the specialised contexts. *The Vulnerability Management Operator's Workbook* holds the templates, with the VM Glossary and the Maturity Rubric. Each stands alone; none assumes you have read the others. The complete set is under $30 in ebook.

### 0.0.2 Reading paths

Different readers arrive with different problems, and the book supports more than one route through.

**Cover to cover** is the route for the reader building or rebuilding a programme. The chapters are sequenced so each stands on the ones before it: the inventory before the lifecycle, the lifecycle before prioritisation, the generic signals before the local judgement that overrides them. If vulnerability management is about to become part of your job, this is your path, and Part I will repay being read slowly.

**The triage desk path** serves the reader who already runs a queue and wants sharper decisions this week. Start with chapter 9 (severity is not risk), then chapters 7 and 11 (the threat signals, and exploitability analysis — the two halves of "does this finding actually matter here?"), then chapter 8 (business context). Appendix A compresses the signals into a desk card; it will make complete sense after those four chapters and partial sense before them.

**The by-tier path** suits the reader who wants to see themselves first. Chapter 4 lays out the four maturity tiers; find yours, then follow the organisation that shares it through the book — each chapter works its examples at every tier, so your tier's thread runs continuously from inventory to external findings.

**The producer's path** is for readers whose organisation ships software as well as running it: chapter 5 (VM and PSIRT) is written for you, and chapters 12 and 16 (code-level vulnerabilities, and findings that arrive from outside) complete that thread.

### 0.0.3 The four organisations as a lens

Four fictional organisations carry this book's examples, one per maturity tier, and they are worth meeting before chapter 1 uses them. **Hollybush & Cale Solicitors** is a twenty-five-person law firm with no security staff — vulnerability management at its smallest honest scale, run by an office manager in a few hours a week. **Saltmarsh & Linden Hotels** is a twelve-property hotel group with a six-person IT team and a scanner it is learning to act on — the mid-market transition from scanning to operating. **Tamarind Valley Water Authority** is a regulated utility running both corporate IT and a plant-floor OT

estate — the governed tier, where exceptions, metrics, and two different worlds of equipment must coexist. **Cinderhaus Media Group** is a thirty-thousand-person streaming platform with a dedicated platform team — the automated tier, where the judgement this book builds is encoded into pipelines. None of them is a costume for a real company; all of them are composites of patterns that recur across real programmes. They exist so that every recommendation in this book has to survive contact with four different budgets, four different estates, and four different amounts of available attention — and so that whatever your scale, one of the four is recognisably your size.

### 0.0.4 When to consult the other volumes

You will meet sentences in this book that name a practice and deliberately do not build it — scanner architecture, SLA enforcement mechanics, exception workflow design, metrics programmes, cloud and OT and supply-chain specifics. Those are *Running the Programme*'s subjects, and the pointers are signposts, not prerequisites: nothing in this volume depends on having read them. The *Operator's Workbook* earns its place the day you move from understanding a practice to installing it — its templates assume the judgement this book builds, and they go stale fastest where that judgement is missing. A reasonable division: this volume for what to fix and why, the programme volume for how to run the machine, the workbook for the paper the machine produces.

### 0.0.5 A final note

Vulnerability management has a reputation as the unglamorous corner of security, and the reputation is half right: the work is repetitive, the wins are invisible, and nobody writes films about patch windows. The other half is this: more real-world breaches begin with a known, unpatched, reachable vulnerability than with anything exotic, which means the unglamorous corner is where a large share of actual defence happens. The discipline this book teaches — knowing what you have, knowing what matters, fixing the right things first, and being able to show your reasoning — is learnable at every scale, including yours.

# 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. Where the book recommends a practice, the recommendation is framed as an approach that reduces risk — not as a guarantee of any outcome. Nothing in this book is legal advice; regulatory and contractual specifics belong with qualified counsel, and the landscape moves — verify current requirements against authoritative sources.

Chapter 1 begins on the next page.

# 1 What Vulnerability Management Actually Is

Priya Shah met vulnerability management as a question she could not answer, one Thursday in October, when Marion Cale's assistant forwarded an email with a note: "Marion says can you deal with this, she doesn't know what it means." It was from the general counsel of Hollybush & Cale's largest client — a FTSE 250 property developer — and contained a vendor security questionnaire. Question 14 read: "Does your organisation conduct regular vulnerability assessments of your IT systems, and if so, please describe your vulnerability management programme including scan frequency, remediation SLAs, and exceptions process?"

The question asked whether they had a *programme* — a word implying process, cadence, and documentation Priya was sure did not exist. She asked Eoin Brennan at TechMantis, the MSP managing the firm's IT, whether they scanned for vulnerabilities. He replied twenty minutes later: "Not in your current contract. We can do a one-off scan if you want — should probably talk about adding it to your monthly retainer actually. What are they asking for exactly?"

She forwarded it; Eoin called back next morning. "So vulnerability management is basically the process of finding the weaknesses in your systems on a regular basis and making sure someone fixes them before

they can be exploited. Think of it like a property survey — you do it regularly, you find the issues, you prioritise the ones that matter, and you fix them. The questionnaire is asking whether you have that as an ongoing thing, not just whether you've ever done it."

"Do we have that as an ongoing thing?" Priya asked.

"Not at the moment, no," Eoin said. "You've never asked for it, so we haven't included it."

TechMantis would add quarterly vulnerability scanning to the retainer, and Priya told the general counsel the firm was implementing a formal programme and could document the first scan results within sixty days. He acknowledged it and did not follow up.

The questionnaire had opened a door that did not close. That week Priya read the security sections she had always skipped: a Manchester law firm hit with ransomware that paid a six-figure sum to recover its matter management system; a Bristol professional services firm — anonymised, but recognisably in her sector — reprimanded by the ICO for a breach from a publicly known vulnerability left unpatched for eight months.

Owen Pritchard, the junior associate, noticed. "So you're basically trying to find out what's broken in the systems before someone bad finds it first?" he said. "Yes," Priya said, "that is exactly what I'm trying to do." He asked if he could help; that was the start of a partnership that carried the firm's minimum-viable programme through its first two years.

This chapter defines vulnerability management, explains what it is and what it is not, situates it within the broader security programme, and describes why organisations at every scale have a stake in it — from Hollybush & Cale's twenty-five-person law firm to Cinderhaus Media Group's thirty-thousand-person global streaming operation.

---

### 1.0.1 The working definition

Vulnerability management is the continuous discipline of identifying, assessing, prioritising, and remediating security weaknesses in an organisation's technology estate before those weaknesses can be exploited by an adversary.

That definition is concise, but each word is doing work.

**Continuous.** Vulnerability management is not a project with a completion date. It is an ongoing operational discipline that runs as long as the organisation runs technology. New vulnerabilities are disclosed every day. The US National Vulnerability Database publishes thousands of entries annually, and the rate has increased steadily over the past decade. In 2023, NIST's NVD processed over 28,000 CVE entries, and in prior years the figure was climbing year-on-year without pause. New assets are added to the estate regularly. Existing assets change: software is updated, configurations drift, new services are connected. The vulnerabilities in the estate today are not the vulnerabilities in the estate next quarter.

A programme that treats vulnerability management as a one-time exercise or an annual audit is not performing vulnerability management. It is taking a point-in-time snapshot that will be out of date before the remediation work is complete. The useful analogy is not the one-off building survey you commission when you buy a property. It is the regime of maintenance inspections that run throughout the property's life. The survey tells you the state on a single day. The maintenance regime tells you when things change, and it is the change that introduces the risk.

For Priya at Hollybush & Cale, "continuous" initially means quarterly: a scan every three months, reviewed by herself and Eoin, with the most critical findings addressed before the next scan. That cadence is proportionate to the firm's risk profile and to the time available. Continuity does not require daily scanning. It requires that the discipline repeats. For Adaeze Okonkwo's VM platform team at Cinderhaus, continuous means something different: automated scanning that runs daily against the cloud estate, with container images scanned on every build and software composition analysis on every code commit. The underlying principle — the discipline repeats — is identical. Only the operational implementation differs, calibrated to scale and risk.

**Identifying.** You cannot fix what you cannot see. The identification function — finding the vulnerabilities that exist in the estate — is the technical foundation of the programme. Identification happens through multiple mechanisms.

Authenticated vulnerability scanning runs a scanner with credentials, so it can examine the internal state of systems rather than just their external appearance. An unauthenticated scan sees what an external observer would see: open ports, service banners, the software version visible from the network. An authenticated scan logs in and inspects the installed software versions, the running processes, the applied patches, and the configuration settings. The difference in coverage is substantial. Authenticated scans find significantly more vulnerabilities, because they see the system's true state rather than its presented surface.

Agent-based scanning deploys a software agent to each system. The agent performs local assessment and reports results to a central platform. This approach is particularly valuable for assets that are frequently off-network — laptops used by remote staff, mobile devices, systems in network segments that are hard to reach from a central scanner. The agent runs locally and reports when the asset has connectivity, so it does not need a network path at scan time.

Passive traffic analysis observes network traffic to infer the software versions and protocols present, without active scanning. It is especially relevant in OT and ICS environments, where active scanning can disrupt industrial processes. Passive analysis sees what is communicating on the network without sending probes that might cause disruption.

Cloud security posture management (CSPM) assesses cloud configurations against security baselines. CSPM tools use cloud provider APIs to continuously discover and assess cloud resources — checking storage bucket permissions, reviewing IAM role assignments, identifying publicly exposed services, and detecting insecure encryption settings. CSPM is the identification mechanism for cloud misconfiguration vulnerabilities that a traditional network scanner does not reach.

Software composition analysis (SCA) inventories the third-party and open-source components in a codebase or container image and checks them against known vulnerability databases. SCA is the identification mechanism for vulnerabilities in code you did not write — the open-source libraries, frameworks, and utilities that form the foundation of virtually all modern software. At Cinderhaus, SCA is integrated into the build pipeline. Every container image build triggers an SCA

scan, and findings are reported before the image is promoted to production.

Manual assessment — penetration testing, code review, red-team exercises — finds vulnerabilities that automated tools miss: logic errors that require human reasoning, business-logic flaws, vulnerabilities in bespoke code that no CVE database indexes. Manual assessment is not a continuous mechanism. It is a periodic complement to the continuous automated programme.

**Assessing.** Not every identified vulnerability is equally dangerous. The assessment function evaluates the severity of each finding in the context of the organisation's environment.

A critical-rated vulnerability in a widely exploited software component is a different risk from a medium-rated vulnerability in an internal tool with no external exposure. The scanner produces a CVSS score, which measures the inherent severity of the vulnerability in isolation: how easily it can be exploited, what access it requires, what the impact would be. The assessment function then adjusts that base score for the organisation's specific context.

Is the affected asset internet-facing, or entirely internal? An internet-facing asset is reachable by any attacker with an internet connection, without requiring network access to the organisation's infrastructure. A CVSS 9.8 vulnerability in a database server accessible only from an internal VLAN is serious, but it requires that the attacker already has a foothold on the internal network. The same vulnerability in the internet-facing API gateway is a direct external attack surface.

What data does the asset process? An asset handling regulated personal data, payment card information, or trade-secret material carries a higher consequence in the event of compromise than an asset handling non-sensitive operational data.

Is the vulnerability already being exploited in the real world? CISA's Known Exploited Vulnerabilities catalogue lists vulnerabilities confirmed to be exploited in the wild. KEV listing is the strongest signal that a vulnerability is not merely theoretically exploitable — it is actively being used by attackers against real systems. A KEV-listed vulnerability, regardless of its CVSS score, warrants immediate attention.

Assessment is the function that turns a raw list of findings into a pri-

oritised work queue. Without it, every critical-scored finding is treated identically, and the programme misallocates its limited remediation capacity.

**Prioritising.** The output of the assessment function is a priority order for remediation. In any organisation larger than a handful of people, there are more open vulnerabilities at any point in time than there is capacity to fix them immediately. Prioritisation is the answer to "which vulnerabilities should we fix first?"

The answer must account for multiple inputs: the severity of the vulnerability, the criticality of the affected asset, the availability of a patch or other remediation, the difficulty of remediation, and the organisation's risk appetite. Poor prioritisation — treating all vulnerabilities as equal, or ordering remediations by technical severity alone without business context — results in high-effort remediation of low-risk findings while high-risk vulnerabilities remain open.

At Hollybush & Cale, prioritisation is simple and manual. Priya and Eoin review the scan results together, discuss the critical and high findings, and agree on which to address first, using a common-sense mix of severity and remediation difficulty. There is no formal scoring framework, and none is needed at this scale. At Cinderhaus, prioritisation is automated. A platform calculates a composite priority score for every finding, drawing on CVSS base score, EPSS probability, asset criticality tags, KEV status, and business context — is this an internet-facing production system or a development workstation? For most findings, the automated score sets the SLA tier and the assigned owner without analyst intervention.

Both approaches reflect the same underlying logic. The tools and the formality differ. The intent — address the most dangerous vulnerabilities in the most critical systems first — is identical.

**Remediating.** Remediation is what happens when the patch is applied, the configuration is changed, or the compensating control is put in place. This is the part of the programme that requires the broadest organisational involvement: system owners, application developers, platform engineers, database administrators, and network engineers all play a role in remediation.

The VM programme does not fix vulnerabilities itself. It provides

the signal — here is a vulnerability, here is its priority, here is the remediation guidance — and the governance — here is the SLA for this severity class, here is the exception process if remediation is not possible, here is the escalation path if the SLA is missed. The actual fix is the responsibility of the team that owns the system.

This is a critical distinction, and one that many new programme managers initially miss. The security function that operates the VM programme is not the function that applies the patches. A server team applies patches to servers. An application team updates libraries in application code. A network team updates firmware on switches and routers. The VM programme is the conductor; the system owners are the players. A programme that forgets this — that imagines the security team will handle all remediation — will fail the moment it meets a server team lead who reasonably asks why they are taking instructions from a security analyst rather than their own management chain.

Remediation governance — the SLAs, the escalation path, the exception process, the tracking — is the mechanism through which the VM programme creates accountability for remediation in teams that do not sit within the security function. That governance structure is what separates a functioning programme from a scanner.

---

### 1.0.2 What vulnerability management is not

The definition above defines vulnerability management by what it does. Equally important is understanding what it is not, because the conflations that lead to poorly designed programmes are consistent across organisations of all sizes.

**Vulnerability management is not penetration testing.** A penetration test is a structured attempt by a skilled assessor to exploit vulnerabilities in a defined scope. It simulates what an attacker would do and demonstrates the impact of a successful attack. Penetration tests produce valuable findings, particularly for complex systems where automated scanning misses logical vulnerabilities that need human reasoning to spot. A skilled tester may find a set of individually low-severity misconfigurations that chain together into administrative ac-

cess. No automated scanner would surface that, because the chain requires judgement to construct.

But penetration tests are point-in-time assessments. A test conducted in January does not tell you whether new vulnerabilities have appeared by July. The scope is fixed; the organisation's estate changes. Vulnerability management is the continuous baseline; penetration testing is a periodic deep-dive that supplements it. An organisation that funds an annual penetration test and considers its vulnerability programme satisfied has one annual snapshot and eleven months of growing risk in the gap between tests.

**Vulnerability management is not patch management.** Patch management is the process of deploying software updates — patches — to systems on a defined schedule. Vulnerability management informs patch management by identifying which patches are most urgently needed and on which systems. But patch management and vulnerability management are separate functions.

Patch management without vulnerability management produces systems that receive patches on a schedule regardless of whether those patches address actual risks. An organisation that patches all software on a thirty-day cycle — applying every available update, regardless of severity — is doing patch management. It may still have critical vulnerabilities: zero-days published in week one of the cycle, patches whose deployment failed silently, components with no vendor patch available.

Vulnerability management without patch management produces a prioritised list of vulnerabilities that nobody has the process infrastructure to remediate. The programme knows what needs fixing; it has no mechanism for getting the fixes deployed at scale. Both are necessary; neither is a substitute for the other. The relationship between them is the subject of Chapter 10.

**Vulnerability management is not compliance.** Many compliance frameworks — PCI DSS, HIPAA Security Rule, ISO 27001, DORA, Cyber Essentials, NIS2 — include requirements for vulnerability scanning, patch management, or both. Implementing vulnerability management to satisfy these requirements is a reasonable starting point, and the compliance driver is often what motivates the initial invest-

ment.

But a vulnerability management programme designed solely to produce auditor-satisfying artefacts — scan reports stored in a folder, a policy document attesting to a process — is not a programme that reduces risk. It is a programme that produces compliance evidence. Picture two file servers, each with 47 critical vulnerabilities. One has been scanned, and its scan report sits in a folder. The other has been scanned, triaged, prioritised, and remediated. The gap between them is the gap between compliance evidence and risk reduction.

Saoirse Devlin, compliance lead at Saltmarsh & Linden — the hotel group among the four organisations this book follows, all introduced properly later in this chapter — is explicit about this distinction with its PCI assessors: "We scan for PCI compliance. We also manage vulnerabilities. The two are not the same activity, though they share some inputs." PCI requires quarterly external scanning. Saltmarsh & Linden's full VM programme runs weekly internal scanning, with daily scanning against the PCI-scoped card processing systems. The PCI compliance requirement is the floor. The programme is the building above it.

**Vulnerability management is not a tool.** A vulnerability scanner — Tenable, Qualys, Rapid7 Nexpose, OpenVAS, or any of the many commercial and open-source options — is a component of the programme, not the programme itself. An organisation that has purchased and deployed a scanner has the technical capability to identify vulnerabilities. It does not yet have a vulnerability management programme.

The programme is the set of processes, policies, responsibilities, and governance structures that decide what the scanner scans, how often, who receives the results, how they are triaged, what the remediation SLAs are, how exceptions are handled, and how effectiveness is measured. The scanner is the sensor. The programme is the operational system that turns sensor output into reduced risk.

Saltmarsh & Linden, before Aarav Krishnan arrived, had a scanner. Nessus had been running for eighteen months, generating reports that were stored in a shared folder on the Edinburgh IT server and reviewed by nobody. The scan results showed critical vulnerabilities on the PMS servers across multiple properties. Nobody knew, because

nobody was reading the reports. The scanner was doing its job. The programme — the people, the process, the governance — did not exist.

**Vulnerability management is not the same as compliance scanning.** This misconception deserves its own entry because it is subtly different from the compliance conflation described above. The compliance scanning conflation is about using VM as a vehicle to satisfy auditors. This misconception is about believing that scanning once a year — because a framework requires it — constitutes an ongoing programme.

A hotel group that scans its PCI-scoped systems quarterly because the QSA requires it, and does nothing with the results between assessment cycles, is compliance scanning. The quarterly scan satisfies the auditor's checklist. It provides no continuous visibility into the organisation's vulnerability posture, no SLA-driven remediation process, and no assurance that critical vulnerabilities discovered in the scan are addressed before the next quarterly scan — let alone before an attacker discovers and exploits them first.

The distinction matters because organisations sometimes use compliance scanning as evidence that they have a VM programme when challenged by clients, insurers, or incident-response investigators. The evidence holds up against the compliance question — "did you scan?" — and collapses against the programme question — "did you remediate what you found, and how quickly?" The purpose of an ongoing programme is precisely that the answer to the second question is "yes, here is the documentation."

---

### 1.0.3 Why every organisation has a stake in it

The argument for vulnerability management sometimes gets made in terms of compliance requirements or industry standards. The more fundamental argument is about the relationship between technology and organisational risk.

Every organisation that runs technology — software, hardware, networked systems, cloud services, SaaS applications — is running code written by humans. Human-written code contains defects. Some

of those defects are security vulnerabilities: flaws that let an unauthorised party do something the system was not designed to permit. The population of known vulnerabilities grows every day as researchers, vendors, and attackers discover new defects. Attackers actively scan for and exploit known vulnerabilities. The time between disclosure and active exploitation has shortened dramatically over the past decade, and some vulnerabilities are exploited within hours of public disclosure.

The consequence of not managing vulnerabilities is not theoretical. The 2017 Equifax breach exposed the personal data of roughly 147 million people. It was caused by an Apache Struts vulnerability (CVE-2017-5638) that had been disclosed and patched two months earlier. Equifax had not patched it. The 2021 Colonial Pipeline ransomware attack shut down the largest fuel pipeline on the US East Coast. It began with a compromised VPN credential, but later analysis found the OT network carried numerous unpatched vulnerabilities that would have been exploitable had the attackers chosen that path. The 2024 Change Healthcare ransomware attack disrupted US healthcare billing for weeks. The attackers gained access through an unpatched Citrix Bleed vulnerability that had been known and patchable for months.

The 2020 SolarWinds compromise showed that the attack surface is not limited to directly operated infrastructure. The attackers inserted malicious code into software update packages distributed to thousands of customers, including US government agencies and major technology companies. Even organisations with mature VM programmes faced a question they could not answer from scan data alone: is the software we trust delivering exploitable code? The compromise extended the VM problem beyond vulnerability scanning into software supply chain integrity, a domain the companion guide's third-party chapter addresses in detail. (A word on the set, since the companion guide will keep coming up: this book is Volume 1, *Understanding and Prioritising*. Volume 2, *Running the Programme* — the companion guide — carries the operational chapters: scanners, patching, SLAs, metrics, OT, third-party. Volume 3, the *Operator's Workbook*, carries the templates and runbooks.)

For Hollybush & Cale, a twenty-five-person law firm handling confi-

dential client data, exploited vulnerabilities carry two main risks. The first is a client data breach: regulatory notification under UK GDPR, SRA disciplinary risk, professional indemnity exposure. The second is operational disruption: a ransom demand that disables the matter management system during a time-critical transaction. The firm is not a high-profile target, but most ransomware operators do not curate their victims. They scan broad IP ranges for known vulnerabilities and deliver payloads to whichever systems respond. The firm's size provides no meaningful protection. Automated exploitation does not distinguish between a FTSE 100 headquarters and a twenty-five-person law firm on the same vulnerable VPN software.

Marion Cale, when Priya eventually presented the firm's first quarterly scan results, was initially surprised by the volume of findings on systems she had assumed were "just the computers." Priya used Eoin's property survey analogy. "If you surveyed every building in Bristol simultaneously and looked for roof problems," she said, "you would find thousands. Most would be minor. A handful would be serious. The question is whether you know which buildings have the serious ones." Marion considered this for a moment. "And right now we don't know which buildings are ours," she said. "Not precisely," Priya said. "But we're learning."

For Cinderhaus Media Group, a thirty-thousand-person global streaming platform serving hundreds of millions of subscribers, the risk differs in scale but is identical in category. Known vulnerabilities in the infrastructure and application stack are attack surface. Successful exploitation of a critical vulnerability in Cinderhaus's subscriber data platform would expose personal data at a scale that triggers notification obligations under GDPR, the California Consumer Privacy Act, and a dozen other jurisdictions at once. For a streaming service that competes on trust, the reputational damage would extend well beyond the regulatory exposure. A fourteen-person VM platform team running continuous scanning, software composition analysis, and container security scanning across a multi-cloud, multi-service estate is the proportionate investment at that scale and risk profile.

The gap between Hollybush & Cale and Cinderhaus is not a gap in the category of risk. It is a gap in scale, sophistication, and the propor-

tionate investment each is expected to make. The principle is the same at both ends of the spectrum and at every point between. The organisation runs technology, the technology has vulnerabilities, some of those vulnerabilities will be exploited if left unaddressed, and the discipline of finding and fixing them before attackers do is vulnerability management.

Tamarind Valley Water Authority adds a dimension that neither the law firm nor the streaming company faces: the risk is not only to data but to physical infrastructure. The water treatment and distribution systems serving a Pacific Northwest population of roughly 400,000 people depend on SCADA and industrial control systems with their own vulnerability profiles. A successful attack on a treatment system is not a data breach. It is a public health incident. Niamh O'Donnell-Ward's framing of the programme's purpose to the Board is explicit: "We manage vulnerabilities so that someone with a browser and a grudge cannot turn off the water for half a million people." The stakes reframe the discussion entirely. For critical infrastructure, vulnerability management is not a security-team concern. It is a public safety obligation.

---

### 1.0.4 The vocabulary of vulnerability management

Vulnerability management has a specific vocabulary that practitioners use consistently and that non-practitioners often use loosely. Clarity about the terms is useful before the technical chapters begin.

A **vulnerability** is a weakness in a system that can be exploited to violate the security policy of the system. Vulnerabilities can exist in software (a buffer overflow, an injection flaw, a logic error), in configuration (a service exposed to the internet that should be internal, a credential with excessive permissions, an encryption algorithm that is no longer considered secure), in hardware (a chip-level flaw that cannot be patched by software), or in process (a deployment procedure that introduces a known vulnerability on every release).

A **CVE** (Common Vulnerabilities and Exposures) is a unique identifier assigned to a specific publicly known vulnerability. CVE identifiers are issued by CVE Numbering Authorities (CNAs), the organi-

sations authorised to assign CVE IDs, and maintained in the MITRE CVE database, which is the authoritative source. A vulnerability with a CVE ID is a known, named weakness with a published description. Many vulnerabilities have no CVE ID yet: ones discovered but not yet publicly disclosed, or ones in products whose vendors have not engaged with the CVE programme.

A **CVSS score** (Common Vulnerability Scoring System score) is a number from 0 to 10 that is the inherent severity of a vulnerability. It is built from a set of characteristics: how easy the vulnerability is to exploit, what access it requires, what the impact on confidentiality, integrity, and availability would be, and (in CVSS v3 and later) environmental and temporal modifiers. CVSS scores are useful as a consistent baseline. But they are frequently misused as a proxy for priority, without accounting for the organisation's specific context. A CVSS 9.8 vulnerability in a system with no external exposure is a different operational risk from a CVSS 9.8 vulnerability in an internet-facing system in active use.

A **patch** is a software update that fixes a vulnerability. Not every vulnerability has a patch: some require configuration changes (disabling a feature, restricting a service, changing a permission), some require architectural changes (removing a component entirely, redesigning a data flow), and some — particularly in legacy systems, medical devices, and OT environments — may have no vendor-provided remediation available.

A **remediation** is the action that addresses a vulnerability. Patching is the most common remediation; others include configuration change, compensating control (a workaround that reduces the risk without fixing the underlying vulnerability), or risk acceptance (a documented decision that the risk is accepted without remediation, typically because remediation is not possible or is disproportionately costly relative to the risk).

An **exception** is a documented decision to defer or decline remediation for a specific vulnerability on a specific system, with a named owner, a defined risk acceptance, and a review date.

An **SLA** (service-level agreement) in the VM context is a defined time limit within which a vulnerability of a given severity class must

be remediated. SLAs create organisational accountability for remediation. Without them, vulnerabilities sit in the queue indefinitely without consequence. SLAs are typically set by severity tier — critical: patch within 7 days; high: 14 days; medium: 30 days; low: 90 days or the next scheduled maintenance window. They are the governance mechanism through which the VM programme drives remediation behaviour in teams outside the security function.

**Mean Time to Remediate (MTTR)** measures the average time between a vulnerability being identified and being remediated. It is one of the primary outcome metrics for a VM programme. A declining MTTR over time indicates that the programme is improving its remediation throughput relative to the volume of findings.

**An exploit** is a piece of code, a technique, or a procedure that takes advantage of a vulnerability to produce a specific effect: executing arbitrary code, escalating privileges, bypassing authentication, exfiltrating data. A publicly available exploit raises the effective priority of a finding. The vulnerability is no longer only theoretically dangerous. The mechanism for exploiting it is in attackers' hands, which lowers the skill barrier for exploitation.

**An exploit kit** is a packaged tool, typically sold or leased in criminal markets, that automates the delivery and execution of multiple exploits. Exploit kits lower the barrier to entry for opportunistic attackers. A non-specialist can deploy one that attempts dozens of known vulnerabilities across a scanned network, without understanding the technical details of any of them. Vulnerabilities in widely distributed exploit kits should be treated as practically zero-day risks regardless of patch availability, because the exploitation infrastructure already exists.

**A zero-day** is a vulnerability that is unknown to the vendor — and therefore has no patch — at the time it is discovered or exploited. The term comes from the idea that the vendor has had zero days to respond. Zero-days are the most dangerous class, because there is no vendor-provided patch. The only defences are compensating controls (firewall rules, access restrictions, monitoring for exploitation indicators), network-level mitigations, or immediate architectural changes. When a zero-day is eventually discovered by the vendor or publicly

disclosed, it becomes a known vulnerability with a published CVE, and the patch cycle begins. The window before disclosure and patching is the most valuable period for attackers and the most precarious for defenders.

**Patch Tuesday** is Microsoft's practice, established in 2003, of releasing security patches on the second Tuesday of each month. The predictable schedule lets system administrators plan patch deployment in advance. Most major operating systems and many enterprise software vendors have adopted similar regular release schedules. Patch Tuesday creates a rhythm for VM teams: the monthly release is a known trigger for scanning, triage, and patch deployment planning. It also creates a risk window. Attackers who reverse-engineer Microsoft's patches can identify the underlying vulnerabilities and begin developing exploits immediately after release, before most organisations have finished deploying.

---

### 1.0.5 The relationship between vulnerability management and other security disciplines

Vulnerability management does not operate in isolation. It sits within a broader security programme, and its effectiveness depends on the health of the functions that surround it.

**Asset management** is the prerequisite. The VM programme cannot scan what it does not know exists. An accurate, current asset inventory is the foundation on which vulnerability scanning and triage are built. The relationship is directional: asset management feeds VM. A VM programme that identifies assets the asset management register did not know about should feed those discoveries back to the register — but the starting point is the inventory, not the scanner. Chapter 2 addresses this relationship in detail.

**Patch management** is the delivery mechanism for a significant proportion of VM remediations. The VM programme identifies which vulnerabilities need addressing and in which priority order; the patch management function deploys the patches that address them. An organisation with a functioning patch management process but no VM programme is patching without prioritisation — applying updates on

a schedule regardless of risk. An organisation with a VM programme but no patch management process is prioritising without delivery — knowing what needs fixing but lacking the infrastructure to fix it at scale. The two functions are tightly coupled. An immature patch management function is frequently the bottleneck that prevents a VM programme from achieving good MTTR metrics, regardless of the quality of the scanning and triage.

**Incident response** is the downstream safety net. When the VM programme fails — a vulnerability not detected, or detected but not remediated within the exploitation window — incident response is the next line of defence. The relationship runs both ways. IR findings should feed back into VM: if an investigation finds that exploitation was via a known CVE that should have been in scope, that is a gap to close. VM data should feed into IR: when an incident occurs, the VM team's inventory of systems and their vulnerability state is one of the first inputs the IR team needs to understand the blast radius.

Aarav Krishnan at Saltmarsh & Linden keeps a standing procedure: when a security incident is declared at any property, the first thing he produces for the incident team is the Nessus findings for the affected system — not just the current scan, but the scan history showing when the vulnerability was first detected and whether it was within or outside SLA when the incident occurred. "It tells us two things," he explains. "Whether the finding was in our known scope, and whether the programme responded to it correctly. If both answers are yes and we still had an incident, we have a detection gap. If the finding was in scope but not remediated in SLA, we have a process failure. If the finding was outside scope, we have an inventory gap." The three-way diagnostic comes directly from the relationship between VM and IR.

**Threat intelligence** sharpens prioritisation. Raw CVSS scores are static. They measure inherent severity at the time of publication and do not change as the threat landscape evolves. Threat intelligence is the dynamic input that adjusts the programme's priorities in response to real-world activity: which vulnerabilities are actively being exploited, which threat actors are targeting which sectors, and what exploitation techniques are in current use. CISA's KEV catalogue is the most widely accessible threat intelligence feed for VM purposes. It lists vulnerabil-

ities confirmed as exploited in the wild and recommends remediation timelines. EPSS (Exploit Prediction Scoring System) is a probabilistic model that estimates the likelihood a vulnerability will be exploited in the next 30 days, based on its characteristics and observed exploitation patterns. Both inputs let the programme move faster on the vulnerabilities most likely to cause harm, and deprioritise findings that represent theoretical risk without current exploitation activity.

**Compliance** sets requirements the VM programme is typically expected to satisfy. PCI DSS requires quarterly external scanning and internal scanning after significant changes. ISO 27001 requires a systematic approach to identifying and addressing technical vulnerabilities. NIS2 requires appropriate and proportionate technical measures for network and information security. DORA requires vulnerability management as part of digital operational resilience testing. These create a compliance floor: a minimum level of activity the programme must demonstrate to satisfy auditors. The programme's goal is to exceed that floor. The floor was designed as evidence that a discipline exists, not as a specification for operating it effectively. The companion guide, *Running the Programme*, addresses the compliance relationship in detail.

**Software supply chain security** is the emerging extension of VM into the dependencies the organisation does not directly control. When Log4Shell was disclosed in December 2021, the question every organisation faced was not "do we use Log4j?" — almost universally, the answer was yes, somewhere in the stack. The real question was "where do we use it, in which products, and in which versions?" Organisations with mature SCA practices and SBOMs for their software products could answer within hours. Organisations without them took days or weeks of manual triage, during which exploitation was already underway. The supply chain security function — software bills of materials, SCA integration into the build pipeline, vendor software transparency requirements — extends the VM programme's identification function into code the organisation uses but does not fully control.

### 1.0.6 Who actually does what

One ambiguity sinks more vulnerability programmes than any technical failure, and it is worth resolving in the first chapter because everything after assumes it is resolved: the vulnerability management function does not fix vulnerabilities. The fixing — the patching, the configuration change, the upgrade, the redeploy — is done by the people who own and operate the systems: the infrastructure team, the desktop team, the database administrators, the development squads, the OT engineers. The VM function finds, prioritises, routes, chases, and verifies. It owns the *process* end to end; it owns almost none of the *changes* the process exists to produce. An organisation that has not internalised this split either builds a VM team that is blamed quarterly for remediation numbers it has no hands to move, or — worse — builds one that tries to do the patching itself, at which point it can manage precisely as much estate as its own small team can touch, which is never the estate.

The split deserves to be written down per activity, because the failure lives in the seams. Detection and the scanning infrastructure: the VM function's, including the credentials and agent health that authenticated coverage depends on. Triage, prioritisation, and the exploitability judgement: the VM function's, with system owners consulted for the context no tool supplies. The remediation itself: the system owner's, full stop — with the VM function setting the clock (the SLA), supplying the evidence (why this one, why now), and providing the route (the ticket in the owner's own queue, not a spreadsheet in the VM team's). Verification and closure: the VM function's again, because the owner who fixed it should not also be the one who marks it fixed. Exception and risk-acceptance decisions: the system owner's business leadership, documented through the VM function's process — the function administers the register; it does not absorb other people's risk. Reporting: the VM function's, and the numbers reported are deliberately *about the owners' performance as much as the programme's*, which is the design choice that makes the split function: a remediation SLA report that names owning teams turns the monthly review into the owners' meeting rather than the security team's apology.

Stated this way, the split also explains what the VM function fun-

damentally is: a small team whose product is *decisions and pressure*, exercised through other teams' hands. That has two practical consequences the rest of the volume keeps meeting. First, the function's authority has to come from somewhere, because it carries responsibility for outcomes it cannot directly produce — which is why the SLA framework, the escalation path, and the executive sponsorship the later chapters describe are not bureaucratic decoration but the load-bearing substitutes for the direct control the function does not have. Second, the function's relationships are its infrastructure — as the four organisations introduced in the next section illustrate. Priya Shah at Hollybush can lean across the office; Aarav Krishnan's programme at Saltmarsh works because Diego Almeida's infrastructure team treats his tickets as real work; and at Tamarind Valley the entire OT half of the programme runs on the standing negotiated trust between Yuki Hartmann's corporate function and Marcus Whitefeather's plant engineers. Where the volume says "the programme decided" or "the organisation remediated", this split is what the words are hiding, and a reader building a programme should hear every such sentence as shorthand for the same machinery: security deciding and chasing, owners changing things, and a written understanding of which is which.

---

### 1.0.7 The four organisations and where they start

The four organisations that anchor this book each encounter vulnerability management from a different starting point. Understanding where they start is the context for understanding the arc each follows.

**Hollybush & Cale Solicitors** starts from zero. There is no scanner, no scan history, no SLA framework, no exception process, and no concept of what a vulnerability register is. The firm's arc is the minimum-viable programme: from nothing to a sustainable, lightweight programme that a non-technical office manager can operate in two hours per week and that satisfies the basic expectations of enterprise clients.

**Saltmarsh & Linden Hotels** starts with a scanner and no programme. The Nessus instance is configured and running. The scan reports exist. Nothing is done with them. The firm's arc is the transformation from scanning to operating: establishing the triage process,

the remediation ownership, the SLAs, and the metrics that turn scan results into closed vulnerabilities.

**Tamarind Valley Water Authority** starts with a mature IT VM programme and a separate, less mature OT VM programme. The firm's arc is convergence: developing a shared vocabulary and a unified governance model across the two programmes without forcing the OT programme into IT-shaped processes that do not work in an operational technology context.

**Cinderhaus Media Group** starts with a sophisticated programme and a metrics problem. The fourteen-person team is capable, the tools are thorough, and the programme produces results. The arc is evolution: redesigning the measurement framework to demonstrate risk reduction rather than activity, and building a genuine open-source dependency programme to address the transitive vulnerability risk that the existing scanning toolset does not fully capture.

The four arcs run in parallel through this book. Each chapter uses whichever organisations fit the topic. Chapter 2 (asset inventory) draws heavily on Hollybush & Cale and Saltmarsh & Linden, where the inventory problem is foundational. The companion guide's OT/ICS chapter draws on Tamarind Valley Water Authority, whose OT-IT convergence challenge is the working example for the whole chapter. Its metrics chapter draws on Cinderhaus, whose metrics redesign is the central narrative.

---

*Recap: Vulnerability management is the continuous discipline of identifying, assessing, prioritising, and remediating security weaknesses before they can be exploited. It is not a project, not a tool, not a penetration test, not patch management, not compliance evidence, and not annual compliance scanning — though it overlaps with and informs all of these. Every organisation that runs technology has a stake in it, proportionate to its scale and risk. The vocabulary — CVE, CVSS, patch, remediation, exception, SLA, MTTR, exploit, exploit kit, zero-day, Patch Tuesday — is the shared language practitioners use across organisations of all sizes. And VM connects to and depends on asset management, patch management, incident response, threat intelligence, compliance, and supply chain security. Underneath it all sits the who-does-*

*what split, written down per activity: detection, triage, prioritisation, and verification belong to the VM function, the fixing to the system owners — the function's product is decisions and pressure, exercised through other teams' hands, and the failures live in the seams.*

*Next: Chapter 2 — Asset Inventory: The Bedrock You Cannot Skip. Vulnerability management depends on knowing what you have. Chapter 2 argues that most organisations significantly undercount their technology estate, describes the four categories of asset that are most consistently missed, and presents the minimum-viable inventory practice that every tier of organisation can implement and maintain.*

*You've just read Chapter 1 of*

# Vulnerability Management

The full book runs 328 pages across 16 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/vulnerability-management**

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

**sylvanassurance.com/go/free-vulnerability-management**