ITSecuConsult

IT SECURITY CONSULTING

GETTING A GRIP ON

GRC

A Practical Guide to Risk and Compliance for Smaller Organizations

Erik Nieuwenhuis

ITSecuConsult · Amsterdam

June 2026

About this guide

This guide brings together a five-part article series originally published on LinkedIn between May and June 2026. It is written for owners, IT managers, security officers and management at small and medium-sized organizations who are responsible for risk and compliance and for the independent consultants and contractors who help them.

The aim is straightforward: to show what a serious GRC program looks like for a small organization, in the order that actually works, without the weight of enterprise platforms or the abstraction of standards documents.

Each chapter walks through one part of the method. The final chapter introduces ReguLight, a macOS application built to put the method into practice.

Author

Erik Nieuwenhuis (CISM) is the founder of ITSecuConsult, an Amsterdam-based IT security, risk and compliance consultancy, and the developer of ReguLight. He has spent more than two decades inside IT operations, cybersecurity, IT risk and compliance teams in financial services, medical and retail sectors.

Contact

ITSecuConsult

Website: www.itsecuconsult.com

Email: info@itsecuconsult.com

LinkedIn: linkedin.com/company/itsecuconsult

Copyright

Table of contents

Introduction

Most smaller organizations that decide to take governance, risk and compliance seriously do so for a familiar reason. A customer asks for a security questionnaire. A regulator publishes a new directive. An incident at a peer makes the board uneasy. A potential investor wants to see a risk register. The exact trigger varies; the response is always the same - a sudden need to look organized about something that, until that moment, had not been organized at all.

What happens next depends almost entirely on the order in which the work is approached. Done well, GRC for a small organization is a few weeks of focused effort that produces a living, computable, reportable picture of the company's exposures. Done poorly, it becomes a multi-month project that produces a stack of policies nobody reads and a risk register nobody trusts.

This guide is the order that works. Twenty-one steps grouped into five chapters, with a final chapter on the tool that puts the method into practice. The tone is pragmatic; the assumption is that the reader has limited time, limited budget and a healthy skepticism for GRC theater. None of the chapters are long. All of them are practical.

How to read this guide

Read the chapters in order on the first pass. They build on each other - the inventory in Chapter 1 is what makes risks in Chapter 2 meaningful; the gaps in Chapter 3 are what get registered in Chapter 4; the registered picture in Chapter 4 is what gets reviewed in Chapter 5. After the first pass, treat each chapter as a reference for that stage of the work.

If you want to skip ahead to ReguLight, go straight to Chapter 6. The application is built around the method described in chapters 1 through 5; if you understand the method, the application explains itself.

Chapter 1 — What are we actually trying to protect?

The challenge: starting in the wrong place

Most smaller companies that begin a serious risk and compliance effort start in the wrong place. They buy a tool. Or they download an ISO 27001 control list. Or they write a policy. Or they hire someone to “do the GRC project.”

None of those things are wrong, but they share a problem: they all assume you already know what you are trying to protect. In my experience, you usually don't. Not in the structured way GRC needs.

I have seen smart, well-resourced companies discover, three months into their compliance program, that nobody could produce a clean list of their critical applications. I have seen consultants tasked with a risk assessment without first being told which assets, data and processes were actually in scope. I have seen auditors find gaps that turned out to be nothing more than “we forgot this system existed.”

Step 1 — Creating the inventory

Risk and compliance management without a clear inventory is guesswork dressed up in a spreadsheet. Before you talk about risks, threats, controls or frameworks, you have to answer one disarmingly simple question: *what are we trying to protect?*

This is step one. And for smaller organizations especially, it is the most important step of the entire program.

Why the inventory matters more than people think

Every risk you identify later is a risk to *something*. Every control you put in place protects *something*. Every audit finding ties back to *something* you should have known about. That “something” is the inventory.

Without a serious inventory you will:

- miss critical assets entirely (the classic forgotten file server, the CRM owned by sales, the third-party tool nobody put on the list)
- over-protect trivial things and under-protect the crown jewels
- waste budget on controls that don't map to any asset
- fail every external audit on the first question

Conversely, a decent inventory unlocks everything that comes after. Risks suddenly have an object. Controls have a target. Compliance scope becomes negotiable based on facts. Reports stop being abstract.

This is true at large enterprises too, but they have armies of CMDB engineers, asset managers and process owners. Smaller organizations don't. Which is exactly why a pragmatic, fit-for-purpose inventory is non-negotiable for them.

How to do it without it becoming a project from hell

Here is the practical part. Drawing on what actually works for SMBs and individual risk officers, this is the order I recommend.

1. Pick the right level of detail. You are not building a CMDB. You are building a working inventory. Aim for 80% coverage in a week, not 100% in a year. Perfection is the enemy here.

2. Cover all the categories, not just IT. A good baseline:

- *People:* employees, contractors, key roles
- *Data:* customer data, financial data, intellectual property, personal data under GDPR scope
- *Applications:* critical business apps, supporting tools, SaaS subscriptions
- *Infrastructure:* networks, servers (cloud and on-prem), endpoints
- *Premises:* offices, server rooms, locks and physical access
- *Processes:* core business processes, support processes, financial processes
- *Third parties:* suppliers, service providers, outsourcing, integrations

Most “IT inventories” stop at applications and infrastructure. That is a mistake. Data, processes and third parties are where the real risk and the real compliance scope tend to live.

3. Talk to the people who run the place. IT operations, finance, HR, sales operations. They know what is in use, what is dependent on what, and what would actually break the company if it stopped working. The inventory you build at your desk from documentation will always be wrong. The one you build by walking around — physically or in Teams — will be much closer to reality.

4. Mark what is critical. Not every asset is equal. A lightweight categorization — for example *critical* / *important* / *supporting* — is enough at this stage. You will use this later when you assess impact.

5. Capture ownership. For every asset of any importance, name a human being. “IT” is not an owner. “The CFO” is.

6. Stop, even when it feels incomplete. It will feel incomplete. That is fine. Inventories are living things. The point of step one is to get *something* on paper that you can build on. You will refine it continuously as you move into the next steps.

Chapter 2 — What could go wrong?

The challenge: jumping straight to controls

In Chapter 1 we made the case that a GRC program starts with knowing what you are protecting. Once that inventory is on paper, there is a strong temptation to jump straight to controls. “We have customer data, so we need encryption. We have employees, so we need an awareness training. We have laptops, so we need MDM.”

That kind of reasoning is not wrong, but it is shallow. It produces a list of controls without a clear understanding of *why* those controls exist or *what would happen if they failed*. When the auditor asks “why this control and not that one,” you have no answer beyond “it seemed sensible.” When management asks “are we exposed?” you can only point at a list, not at a reasoned picture of the business.

The missing layer is risk. And risk doesn't appear out of nowhere — you build up to it in three deliberate moves: events, scenarios, threats. Then, and only then, you get to risks with likelihood and impact.

For smaller organizations this can feel academic. It isn't. Done in a lightweight way, this layer takes a couple of focused sessions and gives the entire rest of your GRC program its backbone.

Step 2 — What could possibly happen?

Start broad. Sit down with one or two people who know the business - someone from operations, someone from IT, ideally someone from finance - and brainstorm everything that could conceivably harm the company. Don't filter yet. Don't rate. Don't argue likelihood. Just collect.

A useful prompt list:

- A critical system goes down for a day, a week, a month
- Customer data is leaked
- Money is misdirected through a fraudulent invoice
- A key employee leaves with no handover
- A supplier is breached and our data is in their backup
- A laptop is lost in a taxi
- Ransomware encrypts the file server
- A regulator finds we don't meet a requirement we didn't know about
- The cloud provider has a regional outage
- A board member's email is spoofed and a payment is approved

You will end up with twenty to fifty raw events. That is the right number. Smaller is incomplete; larger is cluttered.

Step 3 — What scenarios would seriously hurt the business?

Now you cluster the raw events into *scenarios* — coherent stories that describe how harm actually plays out. A scenario is not a single event; it is a chain. "Phishing email → credentials stolen → attacker logs into VPN → ransomware deployed on file server → backups too old to recover → three days of business disrupted → customer trust damaged."

Scenarios are useful because they force you to think operationally. A single event ("ransomware") is abstract. A scenario is concrete enough that an operations manager can tell you whether it could really happen, and a CFO can tell you what it would cost.

Aim for five to ten serious scenarios. These are the ones where, if they materialized, you would seriously regret not having paid attention earlier. They will guide the rest of your work.

Step 4 — Threats: external and internal

A threat is the *source* of harm. It is what could trigger the events in your scenarios. Think of threats in two buckets:

External threats:

- Cyber criminals (financially motivated)
- State actors (less likely for SMBs, but real for some sectors)
- Suppliers and service providers being breached
- Regulatory change
- Macro events (energy, geopolitics, pandemics)
- Physical threats (fire, flood, theft)

Internal threats:

- Human error (still the largest single cause)
- Disgruntled employees
- Misconfiguration
- Inadequate change management
- Lack of capacity, knowledge or budget on the IT side
- Shadow IT and unmanaged SaaS

For smaller organizations, *human error* and *third-party breach* are usually the two threat categories that drive most realistic scenarios. Don't get distracted by APT-style threats unless your sector demands it. Be honest about what is likely.

Step 5 — Risks with likelihood and impact

Now you translate everything above into actual *risks*. A risk has a clear formulation, an owner, a likelihood and an impact. Something like:

> *"There is a risk that customer personal data is exfiltrated as a result of credential theft via phishing, leading to GDPR notification obligations, regulatory fines, and reputational damage."*

Likelihood and impact, at this stage, do not need to be precise. A simple 4×4 scale (Low / Medium / High / Very High on each axis) is more than enough. Smaller organizations get themselves in trouble trying to use 1-to-10 scales or quantitative models too early. The goal of a first risk register is to get the *relative* picture right, not to compute monetary exposures.

Aim for ten to thirty risks at this stage. Fewer and you are missing things; more and the register becomes unreadable. You can always add later.

A pragmatic tip: write the risks in a way that the board would understand. "Insufficient logging in cloud applications" is not a risk; it is a control gap. The associated risk is "Inability to detect and respond to unauthorized access to customer data." Keep that distinction. Risks are about business consequence; gaps and controls come later.

Chapter 3 — What is already in place — and how good is it really?

The challenge: the two opposite illusions

After Chapters 1 and 2 you know what you are protecting and what could go wrong. The natural next step is to look at what you are *already* doing about it. This is the part where most smaller organizations fall into one of two opposite illusions.

Illusion one: *we don't really have anything in place.* This usually shows up at companies that have never formalized their GRC. They assume that because nothing is documented, nothing exists. In reality there are usually firewalls, password policies, MFA on the main systems, backups running quietly somewhere, an antivirus package, an employee who informally watches the logs. There is more there than people think - it just hasn't been written down.

Illusion two: *we have all the basics covered.* This is the IT manager's version. Everything looks fine from the inside. Backups run. MFA is on. The vendor portal says “secure.” The truth tends to emerge under questioning: backups have never been restored, MFA exempts a handful of admin accounts, the firewall ruleset hasn't been reviewed in three years.

The honest middle ground is the only useful place. You probably have more than you think - and most of it is less effective than you assume. Step three of a serious GRC effort is to get an unflattering, accurate picture of what is in place today.

This chapter is the most uncomfortable one in the guide. It is also the one where smaller organizations get the highest immediate return on a few days of work.

Step 6 — Look honestly at the existing measures

Walk through every layer of the company and write down what is currently mitigating risk. Don't judge yet. Just inventory.

A useful structure is to walk through the categories that match well-known control frameworks: identity and access, endpoint, network, cloud, applications, data, backup and recovery, monitoring and logging, incident response, physical security, supplier management, awareness, governance. For each, ask:

- What is in place?
- Who runs it?
- When was it last reviewed?
- Is there evidence it works?

You want concrete answers like “MFA is enforced on Microsoft 365 for all users except three legacy service accounts; reviewed in February by the IT lead.” Not “we have MFA.”

A pragmatic tip: do this with the IT operational staff in the room. They know where the truth lives. They also know which measures are theoretically in place but practically rotting. If you get them to talk honestly - and you protect them from being shot for it - you will learn more in two hours than in any formal questionnaire.

Step 7 — Compare against basic hygiene

Now hold what you have against a basic hygiene baseline. There is no need to invent one. Several public frameworks (CIS Controls, NCSC Cyber Essentials, the ReguLight Security Framework I publish for free, and similar) describe what good baseline hygiene looks like for any small organization.

Things like:

- Asset inventory and ownership
- Identity and access with MFA on everything that matters
- Patch management with measurable cadence
- Endpoint protection on every device
- Email and web filtering
- Backups, *with tested restores*
- Logging and basic monitoring
- Incident response runbook (even a one-pager)
- Supplier list and basic supplier due diligence
- Employee onboarding and offboarding hygiene

Smaller organizations rarely have all of this. Larger ones rarely have all of it actually working. That's normal. The point of this comparison is not to grade yourself on a scale. It is to see, in fifteen minutes of work, where the obvious gaps are.

Step 8 — Recognize that hygiene improvements pay off immediately

This is the optimistic part of the guide. Many of the gaps you will find are *cheap to close* and *immediately reduce risk*. Closing the MFA exemption on the three legacy admin accounts is a one-day project that meaningfully drops your residual risk on credential-theft scenarios. Testing a backup restore costs you a Saturday morning and either confirms you are safe or surfaces the most important problem in your company before an attacker does.

Smaller organizations often defer these basics because they imagine GRC as a big-bang program. It isn't. The largest reductions in residual risk almost always come from the unglamorous baseline. Recognize that, and you can start showing measurable progress in weeks, not quarters.

Step 9 — List the gaps, in plain language

The output of this whole exercise is a clean list of gaps:

- *what should be in place but isn't*
- *what is in place but is ineffective in practice*
- *what is in place and effective but not documented*

Write each gap in plain language, with the IT staff who know the system. Resist the temptation to wrap them in jargon. “We have no formal process to remove leavers' access within 24 hours” is more useful than “Identity lifecycle management gaps in line with ISO A.9.2.6.”

You will use this list in Chapter 4, when you start formally registering controls and issues. For now, it is the honest mirror that the rest of your program needs.

Chapter 4 — Putting it on paper

The challenge: thinking is not the same as registering

By the end of Chapter 3 you have a remarkable amount of useful material. You know what you protect. You know what could go wrong. You know what is already in place and where the gaps are. In a smaller organization that alone puts you well ahead of where most companies sit.

And then it stalls. The whole picture lives in someone's head, in a few half-finished documents, in a meeting that everyone agreed was useful, and in the personal conviction of the security officer that “we have a handle on it now.”

That isn't GRC yet. GRC starts the moment you *register* what you know in a way that:

- you can re-read in three months and still understand
- you can show a board, an auditor or a new colleague
- you can update without rewriting from scratch
- the system can reason about (likelihood × impact, control effectiveness, residual risk)

Step four of a serious program is the move from *thinking* to *registering*. It is also, in my experience, the step where smaller organizations most often quietly give up - because spreadsheets don't do connections well, document folders don't do calculations, and enterprise GRC platforms are too heavy or too complex.

This is the chapter where order matters most. Done in the right order, registration is straightforward and motivating. Done in the wrong order, you create a mess you will have to undo.

Step 10 — Register the risks

Start with the risks you formulated in Chapter 2. Each risk gets a name, an owner, a likelihood, an impact, and a description that a non-specialist can read. Aim for ten to thirty entries. Keep the language plain and consequence-oriented. *“Inability to recover from ransomware within an acceptable timeframe due to inadequate backup testing”* is a good risk. *“Backup issues”* is not.

You are not yet doing controls. Register the risks first. The reason is psychological and practical: once controls are in the picture, people start adjusting risk descriptions to make their controls look good. Lock the risk picture before you start mitigating it on paper.

Step 11 — Register the existing measures as Internal Controls

Now go through everything you found in Chapter 3 that is *already in place* and register each as an internal control. Firewalls, MFA, backup jobs, awareness training, the leavers process - each is a control. Give each control a clear name, an owner, and a short description of what it actually does.

Smaller organizations are often surprised by how many controls they already have once they start writing them down. That is good news. It also turns the next step - rating effectiveness - into something concrete instead of abstract.

Step 12 — Rate effectiveness honestly

For each control, set its *optimal effectiveness*: how effective the control would be when it is fully and properly in place — covering everyone in scope, configured as designed, working as intended. In ReguLight this is the *Control Risk Reduction Factor* (CRRF). It is a theoretical maximum, not a description of your current implementation.

This distinction matters more than it first appears.

Take MFA. If MFA is well implemented - on every account, every relevant system, configured properly - the optimal CRRF is around 90%. It is around 90% in *your* organization too, even if three legacy admin accounts are currently exempt. The exemption does not change what the control *could* do; it changes what the control *currently* does. That is a different kind of fact, and it belongs in a different place.

The right place for it is an *Issue*. You leave the optimal CRRF at 90% and you register an Issue against the control: "*MFA enforcement excludes three legacy admin accounts.*" The issue degrades the actual effectiveness of the control in the risk engine's calculation automatically, without you ever touching the control's optimal rating. When the Issue is closed, the actual effectiveness returns to the optimal value. No re-rating required.

A few rules of thumb for setting the optimal CRRF:

- Use round numbers. 90% / 75% / 50% / 25% are honest enough. False precision is its own form of dishonesty.
- Anchor the rating in *what the control could do* if implemented properly. Not in what it currently achieves in your environment.
- Be skeptical of very high ratings. Few security controls genuinely reduce risk by more than 90% in isolation.
- Resist the urge to lower the optimal to reflect known gaps. Those gaps belong in Issues, not in the rating.

Rated this way, the optimal CRRF is a stable property of the control. It only changes if the control itself is redesigned. The day-to-day work of capturing exemptions, drift, missed

coverage and partial implementation is done by Issues, Tasks, Incidents and audit findings - exactly the inputs the engine needs to compute a living residual risk score.

Step 13 — Register improvements as Issues

For each existing control that is less effective than you would like, register the improvement as an *Issue*. Not as a vague to-do. As a tracked item with a description, severity, owner, and a link back to the control it concerns. *"Remove three legacy MFA exemptions in Microsoft 365"* is an issue. *"Improve identity management"* is not.

Issues are the connective tissue of an active GRC system. They are how the gap analysis from Chapter 3 becomes a backlog you can actually work through.

Step 14 — Register missing measures as Draft Controls

For each gap where the right answer is *we don't have this control yet*, create a new internal control with status *Draft* and effectiveness *0%*. This sounds like a paperwork exercise. It isn't. It does two things at once: it puts the missing control on the map (so the residual risk calculation knows it isn't there yet) and it gives you a place to attach the improvement work that will bring it into existence.

Step 15 — Link the Issues to the Draft Controls

Now connect the work to the target. Every issue that exists to *create* a missing control should be linked to the corresponding draft control. Every issue that exists to *improve* an existing control should be linked to that existing control.

Once these links are in place, your backlog stops being a flat list of things to do and becomes a structured improvement plan: each issue exists *for a reason*, that reason is a control, and that control exists to mitigate one or more risks.

Step 16 — Link the Controls to the Risks

The final connection. Each control - existing or draft - gets linked to the risks it mitigates. A single control will often mitigate several risks; a single risk will usually be mitigated by several controls. That web of connections is exactly what you want, because it is what makes residual risk calculable.

Once the links are in place, you can answer questions you couldn't answer before. *"What controls protect us against ransomware?"* - list. *"What risks would change if we lost this control?"* - list. *"Where does this issue actually matter?"* - line straight back to the affected risks.

Chapter 5 — Closing the loop

The challenge: GRC as a project versus GRC as a rhythm

If you have followed the guide this far, you have done something most smaller organizations never get around to. You have an inventory. You have a risk picture. You have an honest view of what is in place and what is missing. You have it all registered, connected, and computable.

It is tempting to declare victory here. Many companies do. They publish the policy, present the dashboard, get the auditor's nod, and quietly move on. Twelve months later the dashboard has not been touched, the controls have drifted, and the risk register reads like a museum exhibit.

That is GRC as a project. The whole point of a program is that it becomes GRC as a *rhythm*. A modest, recurring rhythm - review, approve, improve, audit, repeat - that takes a few hours a month and keeps the picture honest.

This last chapter is about the operational discipline that turns the work of Chapters 1 to 4 into something lasting. It is not glamorous. It is the quiet difference between a real GRC program and an expensive piece of theater.

Step 17 — Review the posture and report to management

Once your risks, controls and issues are registered and linked, the first habit to build is *reviewing the picture*. Open the risk dashboard. Look at the heatmap. Look at the controls with declining effectiveness. Look at the open issues and which risks they touch. Look at what has changed since last month.

Then translate it into something a non-specialist board member or owner can absorb in five minutes. The most common mistake here is reporting in GRC jargon. Boards do not want to hear about control effectiveness factors or framework coverage percentages. They want to hear: *what are our top three risks today, are they getting better or worse, and what do you need from us?*

A good monthly or quarterly status report has three sections: the picture (top risks, trend), the work in progress (which improvements are running, which are stalled), and the asks (decisions, budget, prioritization). Keep it to a single page. The richer detail belongs in the underlying tool, not in the report.

Step 18 — Get approval for improvements

Improvements only happen when someone says yes to them. The act of formally proposing improvement actions to management - and getting an explicit yes or no - is the single most

underused mechanism in smaller organizations. Without it, every gap quietly becomes the security officer's personal frustration, and nothing moves.

A good monthly cadence is to bring two or three concrete improvement proposals each cycle. Each proposal references the risks it reduces, the issues it closes, and the rough effort it takes. Most will be approved. Some will be deferred with a clear reason. The ones that are deferred should be visible in the risk picture, so that the deferred decision lives on the books, and not in someone's inbox.

This step is where GRC becomes genuinely useful to the business. You stop being the person who says no to things and become the person who connects security work to business decisions.

Step 19 — Improve, raise effectiveness, reduce risk

Now you actually do the work. Issues are closed. Draft controls become live controls. Effectiveness ratings go up. Residual risk scores drop. Backups get tested. MFA exemptions get removed. Suppliers get reviewed. The leavers process gets a checklist.

The thing to watch here is *visible movement*. If your tooling shows residual risk dropping as work completes, the program funds itself. People - including management - keep paying attention to things that visibly move. They stop paying attention to things that don't. Make sure the picture moves.

Step 20 — Run audits and register findings

When the basics are running, start auditing yourself. Internal audits first, before any external party. Pick a control area, design a small set of audit questions, walk through the evidence, and record what you find. Some findings will confirm that things are working. Others will surface issues you didn't know you had.

Each audit finding gets registered with a severity, an owner and a status. Where remediation is needed, escalate the finding into an issue and link it back to the affected control. The closed loop is observation → finding → escalated issue → remediation. Done well, the first internal audit is more valuable than several quarters of dashboards, because it tests the difference between what you *think* is in place and what is *actually* in place.

External audits, when they come, then become much less stressful. You are not preparing for an audit; you are sharing the picture you already maintain.

Step 21 — Plan, do, check, act

The cycle that holds everything together is the unglamorous one your quality colleagues have been talking about for decades: plan, do, check, act. In a GRC context it looks like this. *Plan:* you set your improvement priorities for the period. *Do:* you execute them, raising

effectiveness and reducing residual risk. *Check:* you review the picture, run audits, look at issues and findings. *Act:* you adjust the plan based on what you learned and start the next cycle.

For smaller organizations the cadence does not have to be heavy. A quarterly review is usually enough, supplemented by lightweight monthly check-ins. The goal is not to generate paperwork. The goal is to keep the picture honest as the business changes.

Where this guide leaves you

Five chapters, twenty-one steps, and a single underlying message: GRC for a smaller organization does not have to be heavy, and it does not have to wait. You can start tomorrow with the inventory, work through the picture in a few weeks, and have a living, calculable, reportable GRC program inside a quarter - without an enterprise platform, an army of consultants, or a budget you don't have.

The final chapter introduces the tool I built to put exactly that into practice.

Chapter 6 — How ReguLight can help you

From method to tool

The previous five chapters describe a methodology. They explain *what* to do - and just as importantly, in *what order* - to get a small organization from a standing start to a working, computable, reportable GRC program. The method is tool-agnostic: a disciplined team with a spreadsheet and a determined security officer can follow it.

But there comes a moment - usually around the end of Chapter 4 - when the method runs into the limits of spreadsheets and document folders. Risks, controls, issues and their connections start to multiply. Static documents cannot calculate residual risk. They cannot detect drift. They cannot produce a board-ready dashboard at five minutes' notice. Enterprise GRC platforms can do all of that - but they take months to implement and cost more than most small organizations can justify.

ReguLight was built specifically for that gap. This chapter explains what it does, how it maps onto the method in this guide, and how to start using it.

What ReguLight is

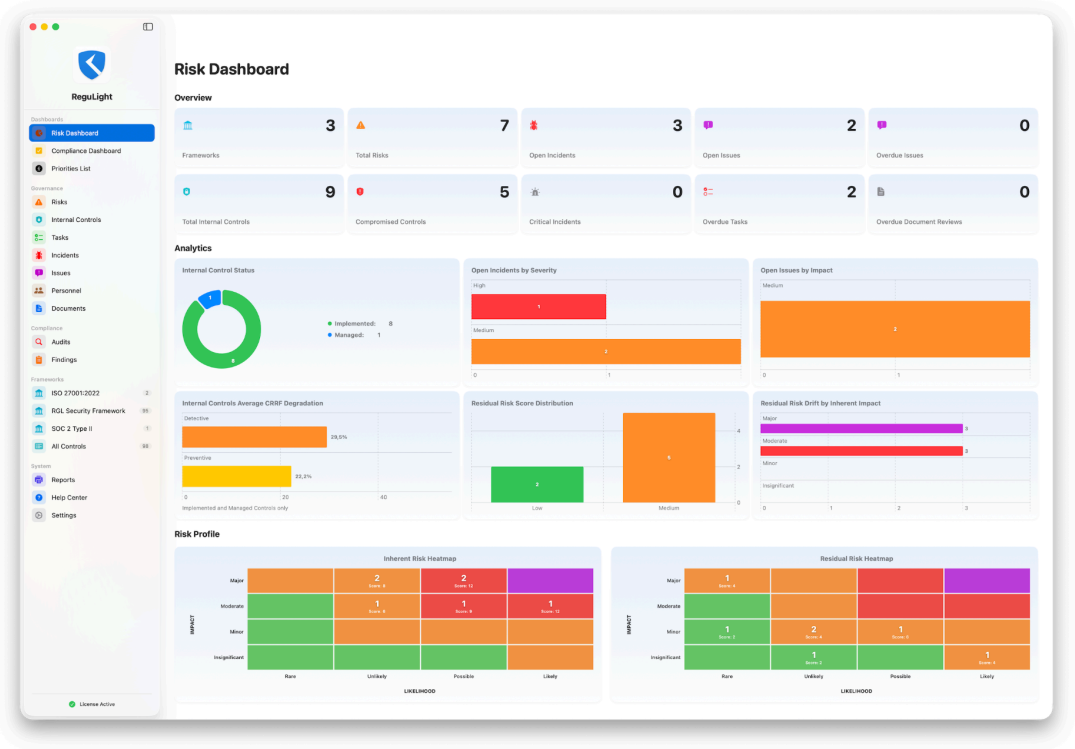
ReguLight is a lightweight Governance, Risk and Compliance application for macOS. It is published by ITSecuConsult and developed by Erik Nieuwenhuis. The full feature set runs locally on a Mac. No cloud, no shared infrastructure, no negotiation of data processing agreements. The data stays on the device, with optional iCloud sync and a one-click backup function.

The product was designed from the start for two audiences:

- Independent security, risk and compliance consultants who need a portable, professional tool to model client GRC setups quickly and consistently.
- IT professionals, security officers and management at small and medium-sized organizations who are responsible for risk and compliance but do not have a dedicated GRC team.

It is available on the Mac App Store as a free download with built-in demo data. Active use with real organizational data requires a subscription.

The tagline is plain: *Security and Compliance. Without the complexity.*



The Risk Dashboard — live inherent and residual risk, with the 4×4 heatmap.

How it maps onto this guide

The structure of ReguLight follows the same logic as the method in chapters 1 through 5. Each chapter of this guide corresponds, broadly, to a part of the application:

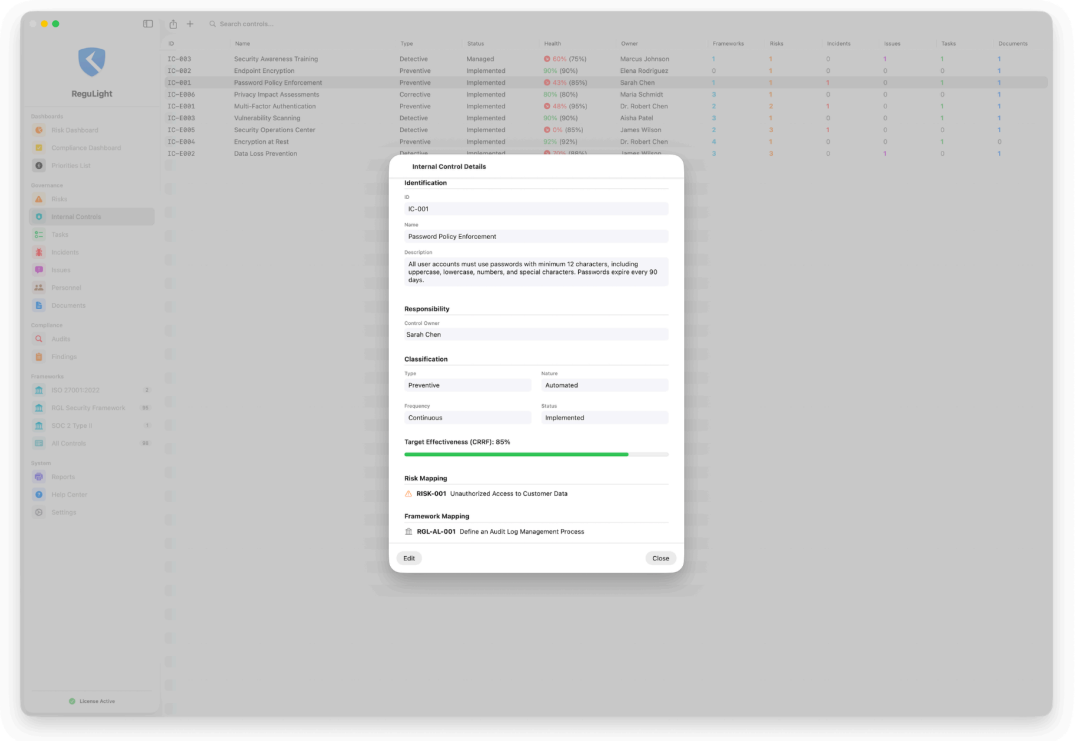
- **Chapter 1** — what to protect → *Personnel*, *Documents* and supporting reference data. Note: ReguLight is deliberately not an asset management or CMDB tool - the asset inventory lives outside the app, in whatever store of record you already use.
- **Chapter 2** — what could go wrong → the *Risks* module, with inherent risk, 4×4 likelihood/impact, and a board-ready heatmap.
- **Chapter 3** — what is already in place → the free *ReguLight Security Framework* as a hygiene baseline; the *Internal Controls* module to register what exists.
- **Chapter 4** — putting it on paper → *Risks*, *Internal Controls*, *Issues*, *Tasks*, and the typed links between them. Draft controls let you register what isn't in place yet.
- **Chapter 5** — closing the loop → *Risk Dashboard*, *Compliance Dashboard*, *Priorities List*, *Audit* module, and the *Reports* library.

The risk engine in plain words

The core of the application is its risk engine. The mechanics are deliberately straightforward.

You register each risk with an inherent likelihood and an inherent impact, on a 4×4 scale. You register the internal controls that mitigate it, each with an optimal effectiveness rating. You link the controls to the risks.

The engine then walks through the linked controls, applies their effectiveness, factors in degradation from overdue tasks, open issues and recent incidents, and produces a *live* residual risk score. As your operational reality changes - a control gets a new owner, an issue is closed, an audit finding is registered - the score updates in the dashboard automatically. The risk picture is no longer a point-in-time snapshot. It is the current state of affairs, refreshed every time you open the app.



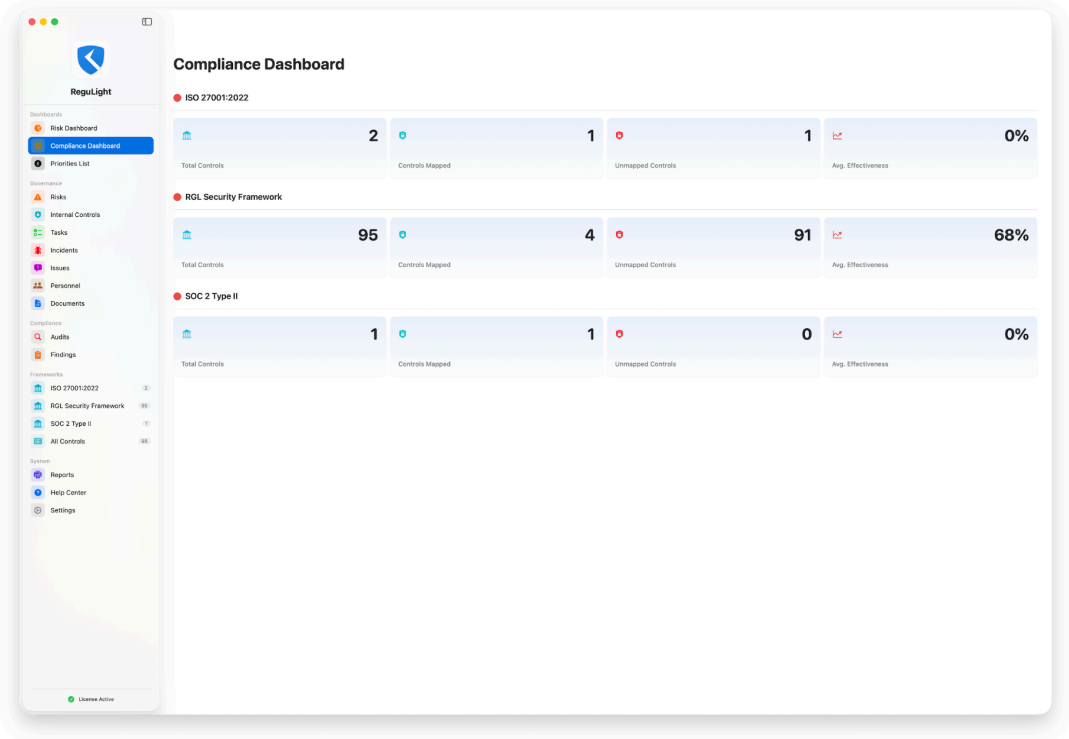
Each Internal Control carries an Optimal Effectiveness rating, an owner, and links to risks, tasks, issues and documents.

The modules at a glance

ReguLight is structured into a small number of focused modules. Connections between them are first-class - that is the point.

- **Risks** — likelihood, impact, owner, description, supporting documents, linked controls.
- **Internal Controls** — name, owner, optimal effectiveness, linked risks, linked tasks, linked issues. A *Draft* status lets you register controls you plan to put in place but haven't yet.

- **Issues** — gaps and improvements, with severity, owner, status and a link to the control or risk they concern. Audit findings can be escalated directly into issues.
- **Tasks** — assignable work items, linked to controls. Overdue tasks degrade the linked control's effectiveness.
- **Incidents** — recorded events with severity. Critical or high incidents drop linked control effectiveness to zero.
- **Personnel** — team members with roles, departments and contact details.
- **Documents** — policies and procedures, with review schedules and overdue alerts.
- **Compliance Frameworks** — imported via CSV. The free *ReguLight Security Framework* (95 controls, derived from CIS Controls v8.1) is available from regulight.eu as a ready-to-use baseline.
- **Audits & Findings** — a three-level structure (Audit → Audit Set → Findings) with severity-rated findings, escalation to issues, and a formal PDF Audit Analysis Report.
- **Dashboards** — Risk Dashboard, Compliance Dashboard, Priorities List. Live state - no spreadsheets.
- **Reports** — Risk Overview, Compliance Statement, Incident Analysis, Issue Analysis, Task Analysis, Audit Analysis - all as branded PDFs.



The Compliance Dashboard — coverage and effectiveness against the chosen framework.

The Audit module

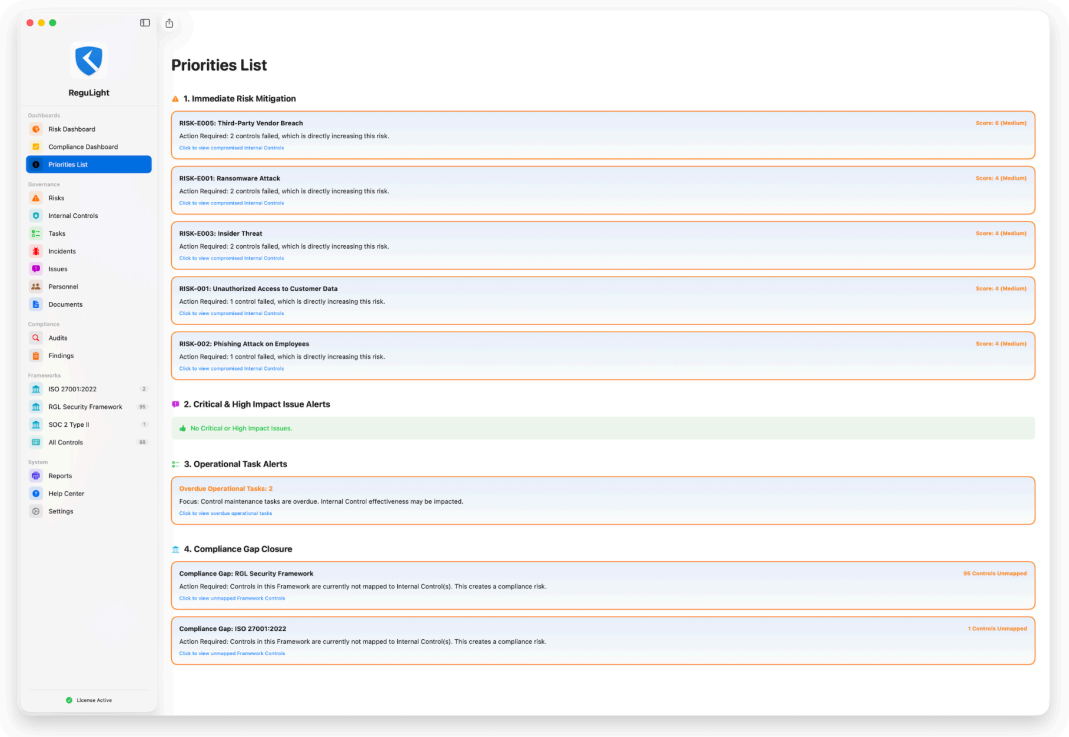
The Audit module closes the loop described in Chapter 5. You define an Audit, structure it as one or more Audit Sets, record severity-rated Findings against the controls in scope, and where remediation is required, escalate the finding directly into an Issue that links back to the affected control. The full observation-to-remediation chain stays inside one tool, on your Mac, under your control.

For internal audits, this turns what is usually a paper exercise - a list of findings in an email, a spreadsheet that nobody updates - into a structured, tracked workflow. For external audits, the *Audit Analysis Report* gives an auditor a clean PDF that shows exactly which controls were tested, what was found, and where remediation stands.

The ReguLight Security Framework

Most small organizations don't have an internal security framework, and few have the budget or knowledge to license one. ITSecuConsult publishes the *ReguLight Security Framework* - 95 controls organized by area (Identify, Protect, Detect, Respond, Recover, Govern) and domain (Asset Management, Backup & Recovery, Data Governance, Email & Web Protection, Identity & Access, Incident Response, Malware Protection, Network Security, and others).

The framework is derived from CIS Controls v8.1 with original wording, available as a free download from www.regulight.eu, and importable into ReguLight via CSV. For an organization with no formal baseline today, it is the fastest way to get one.



The Priorities List — risks, controls and issues ranked by the live engine.

A note for consultants and contractors

For independent risk and compliance consultants - one of ReguLight's two primary audiences - the application is more than a working tool. It is a delivery platform. You can model a client's GRC environment in hours rather than weeks, run formal audits, produce polished PDF deliverables (Risk Overview, Compliance Statement, Audit Analysis), and hand the client a working GRC system they can continue to maintain after the engagement.

Methodology becomes consistent across engagements. Same risk engine, same scoring, same reports - regardless of client size or sector.

Pricing and how to start

ReguLight is available from the Mac App Store. You can download the application for free and explore it with built-in demo scenarios - four populated example organizations let you see what a working GRC system looks like before you commit to anything.

Active use with your own organization's data requires a subscription:

- €49.99 per month
- €549.99 per year (a meaningful saving for steady use)

That price compares to a fraction of a single enterprise GRC seat, and includes the entire feature set described in this chapter.

More information

- **Website:** www.regulight.eu
- **Documentation:** www.regulight.eu/docs
- **Support:** support@regulight.eu
- **App Store:** search for “ReguLight” on the Mac App Store
- **About ITSecuConsult:** www.itsecuconsult.com

Twenty years inside IT operations, security, risk and compliance taught me that the value of GRC is unlocked by *structure* and *discipline*, not by complexity. ReguLight is my attempt to put that structure and discipline into a tool that any small company, individual risk officer or independent consultant can put to work the same afternoon they download it.

If you have read this far, you are exactly the kind of reader the application was built for.