ISSN: 2581-3404 (Online) IF: 5.86 (SJIF)
IJIRTM, Volume-10, Issue-6, November-2026

VIGILIS: A Lightweight Compliance Framework for Organizational Management

Rahul[1], Sakshi[2], Anju Arya[3]
Student, Department of Computer Science and Engineering (Cyber Security) [1,2]
Panipat Institute of Engineering and Technology, Samalkha, Panipat, India[1,2,3]
Affiliated to Kurukshetra University, Kurukshetra, India[1,2,3]
Assistant Professor[3]

Abstract. *In most small and mid-sized organisations, compliance work is still done in spreadsheets. Controls get updated when somebody remembers to do it, risks sit in a register with no owner attached, and vendor audit dates pass by quietly until someone notices months later. Commercial GRC platforms deal with all of this, but their pricing and the effort needed to set them up puts them out of reach for smaller teams, college groups included. This paper looks at how GRC is currently handled by different tools and then presents Vigilis, a lightweight self-hosted dashboard we are developing that sits somewhere between a spreadsheet and an enterprise suite. Vigilis tracks controls against ISO/IEC 27001 and the NIST Cybersecurity Framework, scores risks using a 5 × 5 Likelihood × Impact grid, and runs a small rule engine that points out stale controls, high risks with no owner, risks that have stayed open too long, and overdue vendor audits. Beyond that it offers a Compliance Debt Score, impact and what-if simulation, cross-framework mapping, an audit trail, and one-click PDF reports. Since Vigilis is still under development, the paper compares its design with existing approaches and does not report experimental results.*

Keywords: Governance, risk and compliance (GRC); ISO/IEC 27001; NIST CSF; risk scoring; rule engine; compliance monitoring.

Introduction

Governance, Risk, and Compliance (GRC) is really three responsibilities bundled together: deciding how security should be run inside an organisation, working out what could go wrong, and showing outside parties — clients, auditors, regulators — that the organisation is actually meeting its obligations. In the security world those obligations mostly come from known standards like ISO/IEC 27001 [1] and the NIST Cybersecurity Framework (CSF) [2], although increasingly they also come from law such as the GDPR [3], from client security questionnaires, and from cyber-insurance forms. Large companies handle this with dedicated platforms and dedicated people. Smaller ones usually have neither, so their compliance records end up in spreadsheets, shared documents, and checklists that live in somebody's head. That works fine right up until it doesn't. Nothing reminds you that a control has not been touched in four months, or that a critical risk is sitting there with no owner, or that a vendor audit was due last week. These things surface at the next review, and by then they are no longer small oversights — they are gaps. A spreadsheet also cannot answer the question every auditor eventually asks: who changed this, and when? This paper is about that gap. We start by reviewing the tools organisations use for GRC today, and then describe Vigilis, a lightweight self-hosted dashboard that gives smaller teams a portion of the

continuous monitoring commercial platforms provide, minus the subscription fee and the long configuration process. The contributions are:

- A review of commercial compliance platforms, enterprise GRC suites, spreadsheet-based practice, and academic dashboards, focusing on where each falls short for small teams.
- A description of the Vigilis architecture and the scoring, rule-based, and simulation techniques behind it.
- A qualitative comparison of Vigilis with existing options across cost, automation, simulation, customisability, and reporting.
- A clear statement of what the design covers and, just as importantly, what it does not.

Section 2 covers the background on frameworks and risk assessment. Section 3 reviews existing GRC approaches. Section 4 presents the Vigilis architecture and workflow, Section 5 explains the techniques, and Section 6 compares Vigilis with the alternatives. Section 7 discusses limitations and future work, and Section 8 concludes.

Background

A. Security frameworks and standards

ISO/IEC 27001:2022 defines the requirements for an information security management system. Its Annex A contains 93 controls organised under four themes, and an organisation pursuing certification must decide which of those controls apply to it and then show evidence that they are in place [1]. NIST CSF 2.0 looks quite different — it describes security outcomes under six functions: Govern, Identify, Protect, Detect, Respond, and Recover [2]. Although the two frameworks are structured differently, they cover much of the same ground. Both expect asset management, access control, and incident response in some form, and this overlap is what later makes cross-framework mapping possible. A few other documents tend to appear next to these two. The CIS Critical Security Controls provide a prioritised set of safeguards [4], NIST SP 800-53 is a detailed catalogue of security and privacy controls [5], and COBIT 2019 covers governance and management of enterprise IT [6]. What all of these have in common is that they describe what should be done — none of them is a tool. Recording and tracking the work still needs software, whether bought or built.

B. Risk assessment and continuous monitoring

ISO 31000 lays out general principles for managing risk in any organisation [7]. NIST SP 800-30 narrows this to information security, describing a risk assessment process around threats, vulnerabilities, likelihood, and impact [8]. OWASP's risk rating methodology works on the same logic, combining a likelihood estimate with an impact estimate to produce an overall severity [9]. Multiplying likelihood by impact on a small numeric scale is therefore a well-established way of making risks comparable, and it is the method Vigilis adopts. Continuous monitoring is the second relevant idea. NIST SP 800-137 frames it as maintaining ongoing awareness of security status, vulnerabilities, and threats rather than relying on point-in-time assessment [10], and NIST SP 800-37 places it within a broader risk management life cycle [11]. For a small team, the real question is how to approximate this without a dedicated monitoring programme. That question directly motivates the automated flagging described in Section 5.

Existing Approaches To GRC Management

A. Commercial compliance automation platforms

Vanta [12] and Drata [13] are the best-known examples here. These platforms automate compliance tasks such as collecting evidence, monitoring control status, and preparing for audits against SOC 2 [14] or

ISO/IEC 27001. They connect to an organisation's existing systems and pull evidence from them automatically, which saves a large amount of repetitive manual work. The downsides are the subscription cost, the closed-source nature of the products, and the fact that they only work with the integrations the vendor supports. For a team wanting something cheap and adaptable, these points matter.

B. Enterprise GRC suites

At the other extreme are suites like ServiceNow GRC [15], RSA Archer, and OneTrust [16]. These cover a much broader scope — enterprise risk, internal audit, policy management, regulatory requirements across departments. Large organisations with complex structures genuinely benefit from that breadth. Smaller ones mostly suffer from it: deployment typically involves heavy configuration, customised processes, administrators with specific skills, and licensing costs that put the products out of practical reach for start-ups, small firms, educational institutions, and student teams.

C. Spreadsheet-based tracking

Spreadsheets remain the most common approach in practice, and for good reason. A risk register in Excel or Google Sheets costs almost nothing, can be reshaped in minutes, and requires no training. For an organisation with few controls and light obligations, it can be perfectly adequate. The problem is that every entry depends on a human typing it in correctly and on time. A spreadsheet will not flag a stale control, score risks consistently, or record who edited a cell, so errors and omissions stay hidden until someone goes looking.

D. Academic and open-source dashboards

Academic and open-source dashboards generally exist to visualise compliance data, and they serve that purpose reasonably well, particularly in teaching. From what we have seen, though, most stop at charts and tables. Very few combine framework-based control mapping, quantitative risk scoring, and automated rule-based flagging in one tool — and combining those three is exactly what Vigilis attempts.

Table 1: Summary of existing GRC tools and approaches

Tool / approach	Type	Main features	Limitations
Vanta [12], Drata [13]	Commercial SaaS	Automated evidence collection and continuous monitoring for SOC 2 and ISO 27001	Subscription-based, closed source, tied to supported integrations
ServiceNow GRC [15]	Enterprise suite	Deep risk management, policy and audit workflows	Very high cost; heavy configuration; dedicated administrators needed
RSA Archer, OneTrust [16]	Enterprise suite	Broad GRC and privacy management	Licensing cost and complexity impractical for small teams
Excel / Google Sheets	Manual tracking	Low-cost risk registers with manual entry and basic charts	No automated flagging, scoring logic, or audit trail
Academic / open-source dashboards	Student and research projects	Visualisation of compliance data	Rarely combine framework mapping, risk scoring, and rule automation

The VIGILIS Framework

A. Design goals

Three goals guided the design of Vigilis. The first was affordability and transparency: the system should be self-hosted, built from open-source components, and carry no licensing fees. The second was a shift from periodic review towards continuous checking, so that problems appear as soon as the data reveals them instead of waiting for the next quarterly meeting. The third was readability — results should be presented in a form a manager can understand without interpreting raw control lists or technical flags.

B. System architecture

Vigilis follows a three-tier architecture,which keeps the interface, the business logic, and the storage separate.

The presentation layer is a Streamlit web interface [17] responsible for forms, buttons, charts, and dashboards — but no business logic; it only displays data and collects input. The logic layer consists of Python modules [18] that hold the scoring functions and the rule engine, and these compute compliance percentages, risk levels, and flags. The data layer is an SQLite database [19] that only the logic layer accesses; the interface never touches it directly. Keeping all data access in one place makes the system easier to reason about and to test.

C. System workflow

The workflow is straightforward. A user submits a form, the data is validated before storage, and anything incomplete or incorrect is rejected at that point and never reaches the database. One choice we made deliberately is not to compute metrics at submission time. Instead, the scoring and rule engine re-runs on every page load, reading the latest records from SQLite and recomputing the compliance percentage, risk levels, and flags from scratch. Because of this, the numbers on screen cannot drift out of date even if the data was modified through another route. Given that Vigilis targets a single organisation with modest data volumes, the cost of recalculating per load should be small. The dashboard renders these fresh results as charts and flag indicators, and a PDF snapshot of the current state can be requested at any time.

D. Functional modules

Six modules make up the system. Control Management loads the checklist for the chosen framework and lets users update each control's status, owner, and notes. The Risk Register records risks, computes their score and level automatically, and tracks their status as Open, Mitigated, Accepted, or Closed. The Vendor Audit module stores vendor audit history and shows the days remaining until the next audit is due. Reporting generates a PDF summary on demand using ReportLab [20], with charts and heatmaps drawn by Plotly [21]. The Audit Log records every create and update action with a timestamp. Finally, the Simulation and Insights module computes the Compliance Debt Score, runs the two simulators, maintains the cross-framework mapping, and produces plain-English explanations of flagged issues.

Core Techniques

A. Risk scoring

Each risk receives a likelihood rating from 1 to 5 and an impact rating from 1 to 5. The score is just the product of the two, giving a range of 1 to 25. Following the likelihood-and-impact approach found in NIST SP 800-30 [8] and the OWASP methodology [9], the score maps to one of four levels, listed in Table 2. The band boundaries are our own design choice rather than something either document prescribes, and they can be adjusted if an organisation prefers a different risk appetite.

Table 2: Risk score bands used in Vigilis

Risk level	Score range (L × I)	Typical handling
Low	1 – 6	Monitor
Medium	7 – 12	Plan treatment
High	13 – 19	Needs an owner; prioritise
Critical	20 – 25	Needs an owner; act immediately

B. Compliance percentage

Compliance for a chosen framework is the count of controls marked as Implemented divided by the number of applicable controls, multiplied by 100. Since the value is recomputed on every page load, it follows every change in control status automatically — there is no update step that could be forgotten.

C. Rule engine

The rule engine is a small set of conditional checks executed on every data refresh. Each rule targets a problem that is easy to miss in manual tracking. Table 3 lists them; the 90- and 30-day thresholds are simply the values chosen for the current design.

Table 3: Rules applied by the Vigilis rule engine

Flag	Condition	What it signals
Stale control	A control has not been reviewed for 90 days or more	Status may no longer be accurate
Ownership gap	A High or Critical risk has no assigned owner	Nobody is accountable for treatment
Ageing risk	An open High or Critical risk is older than 30 days	Serious risk is not being resolved
Overdue audit	A vendor's next audit date has already passed	Vendor assurance may have lapsed

D. Compliance Debt Score

Software developers use the term "technical debt" for the pile of shortcuts that gradually makes a system harder to maintain. We took that idea and applied it to compliance. Vigilis compresses the accumulated weight of unresolved compliance issues into a single score between 0 and 100, computed as a weighted combination of the active rule-engine flags — stale controls, ownership gaps, ageing risks, overdue audits — plus the number of open High and Critical risks. The point is a quick health check: a manager can look at one number and tell whether the overall position is improving or worsening, without reading every flag. Because the score is a heuristic, its weighting needs testing against realistic data before anyone relies on it; we return to this in Section 7.

E. Compliance Impact and What-If simulation

Two simulators let users look forward rather than only backward. The Compliance Impact Simulator takes a failed control and traces the control-to-risk and control-to-framework mappings to show which risks and frameworks would be affected. The What-If Risk Simulator goes the other way: mark a control

as implemented and preview the effect on the compliance percentage and risk levels. Both calculations run on a temporary copy of the data, so stored records are never touched. A team can therefore compare possible actions before deciding anything.

F. Cross-framework mapping

A mapping table connects equivalent ISO/IEC 27001 Annex A and NIST CSF controls. Marking one control as implemented updates its status in both frameworks simultaneously, which saves an organisation that answers to more than one framework from doing — and documenting — the same work twice.

G. Plain-English explainer

The last component is optional. It sends flags and scores to a large language model via an API and returns a short explanation of each issue's reason, severity, and duration in language a non-technical manager can follow. It requires an API key and can be left off entirely. Two cautions apply. First, an organisation should decide what information it is willing to send to an external service before enabling the feature. Second, the generated text is only an aid to readability and should not replace the underlying flags and scores, which stay authoritative.

H. Implementation stack

The system is written in Python 3.12 [18]. Streamlit [17] drives the interface, Pandas handles tabular computation, Plotly [21] draws the charts and risk heatmaps, ReportLab [20] produces the PDF reports, and SQLite [19] stores the data. Hardware requirements are modest — an Intel Core i3 or AMD Ryzen 3 with 4 GB of RAM (8 GB recommended) is enough — and the tool runs on Windows, Ubuntu Linux, or macOS.

Comparative Analysis

Table 4 compares Vigilis with the alternatives on five criteria. The entries for the other tools are qualitative judgements based on publicly available product descriptions and the review in Section 3, not results from hands-on benchmarking. Vendor features also change over time.

Table 4: Qualitative comparison of existing tools and Vigilis

Criterion	Vanta / Drata	ServiceNow GRC	Excel tracking	Vigilis
Cost	High (subscription)	Very high	Low	Low (self-hosted, no licence)
Automated risk scoring	Partial	Yes	No	Yes (L × I engine)
Compliance Debt Score	No	No	No	Yes
What-if simulation	No	No	No	Yes
Self-hosted / customisable	No	Limited	Yes	Yes
Automated PDF reporting	Yes	Yes	Manual	Yes (one click)

Commercial tools automate well, particularly around evidence collection, but cost more and adapt less readily. Excel adapts to anything but checks nothing on its own. Vigilis aims for the space between: automatic scoring and rule-based flagging at low cost, with the source code under the team's control. As far as we are aware, the Compliance Debt Score and the what-if simulator are features none of the other listed options provide. Even so, Vigilis does not beat commercial platforms everywhere — it has no automatic evidence collection and no external integrations, which remain genuine strengths of the paid products.

Scope, Limitations, and Future Work

A. Scope and limitations

The present design covers control tracking for ISO/IEC 27001 Annex A and NIST CSF, risk logging and scoring, vendor audit tracking, rule-based flagging, the simulators, an audit trail, and PDF reporting — for one organisation at a time. A few boundaries are worth stating plainly.

- The ISO/IEC 27001 and NIST CSF control lists used are simplified educational subsets, not the complete official standards.
- Only two frameworks are supported, and the system is designed for local or single-server use rather than as a multi-tenant service.
- There is no role-based access control or multi-user authentication; Vigilis is a single-workspace tool.
- Data entry is manual. The system assumes entries are accurate and timely, and it does not yet pull data from external systems such as cloud security posture tools.
- The Compliance Debt Score and rule thresholds are untested design choices that have not been validated against real organisational data.
- Vigilis is a proposed system. No measured results are reported here, and claims about its usefulness remain to be tested.

B. Future work

A number of extensions are planned. Role-based access control — Admin, Risk Owner, and Auditor or Viewer roles — would open the tool up to whole teams. Saving daily or weekly snapshots of the compliance percentage would enable trend lines. Evidence uploads, meaning screenshots and policy documents attached to individual controls, would move the tool closer to auditor expectations. Language-model summaries could be appended to the PDF report as an executive overview, and email or Slack alerts could ping owners about overdue audits and critical risks. Further control sets such as SOC 2, GDPR, and HIPAA could be added over time. Before any of these, though, comes the more important step: testing the existing design against sample data and, ideally, with real users, so that the scoring logic and debt weighting can be checked.

Conclusion

Small organisations and academic teams face an awkward choice between enterprise GRC suites they cannot afford and spreadsheets that never check anything. After reviewing the current landscape, this paper described Vigilis, a lightweight self-hosted dashboard combining framework-based control tracking, Likelihood × Impact risk scoring, and an automated rule engine, extended with a Compliance Debt Score, impact and what-if simulation, cross-framework mapping, an audit trail, and management-ready PDF reports.

The intention is not to compete head-on with commercial platforms. Vigilis simply tries to demonstrate that continuous, automated visibility of compliance posture can be achieved at low cost with open-source components. Since the system is still under construction, evaluation with sample and real data comes next, followed by the extensions in Section 7. Those results will tell us how well the design actually meets the goals described here.

References

[1] International Organization for Standardization, "ISO/IEC 27001:2022 — Information security, cybersecurity and privacy protection — Information security management systems — Requirements," ISO, Geneva, Switzerland, 2022. [Online]. Available: https://www.iso.org/standard/27001
[2] National Institute of Standards and Technology, "The NIST Cybersecurity Framework (CSF) 2.0," NIST, Gaithersburg, MD, USA, 2024. [Online]. Available: https://www.nist.gov/cyberframework
[3] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 (General Data Protection Regulation)," Official Journal of the European Union, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj
[4] Center for Internet Security, "CIS Critical Security Controls, Version 8.1," CIS, Jun. 2024. [Online]. Available: https://www.cisecurity.org/controls/v8-1
[5] R. Ross et al., "Security and Privacy Controls for Information Systems and Organizations," NIST Special Publication 800-53, Rev. 5, Sep. 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-53r5
[6] ISACA, "COBIT 2019 Framework: Governance and Management Objectives," ISACA, 2019. [Online]. Available: https://www.isaca.org/resources/cobit
[7] International Organization for Standardization, "ISO 31000:2018 — Risk management — Guidelines," ISO, Geneva, Switzerland, 2018. [Online]. Available: https://www.iso.org/standard/65694.html
[8] National Institute of Standards and Technology, "Guide for Conducting Risk Assessments," NIST Special Publication 800-30, Rev. 1, 2012. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/30/r1/final
[9] OWASP Foundation, "OWASP Risk Rating Methodology." [Online]. Available: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology
[10] K. Dempsey et al., "Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations," NIST Special Publication 800-137, Sep. 2011. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-137
[11] R. Ross et al., "Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy," NIST Special Publication 800-37, Rev. 2, Dec. 2018. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-37r2
[12] Vanta, "Security and Compliance Automation Platform." [Online]. Available: https://www.vanta.com/
[13] Drata, "Security and Compliance Automation Platform." [Online]. Available: https://drata.com/
[14] American Institute of Certified Public Accountants (AICPA), "SOC 2 Trust Services Criteria," 2022. [Online]. Available: https://www.aicpa-cima.com/
[15] ServiceNow, "Governance, Risk, and Compliance." [Online]. Available: https://www.servicenow.com/products/governance-risk-compliance.html
[16] OneTrust, "Risk and Compliance Solutions." [Online]. Available: https://www.onetrust.com/

[17] Streamlit Inc., “Streamlit Documentation.” [Online]. Available: https://docs.streamlit.io/
[18] Python Software Foundation, “Python 3 Documentation.” [Online]. Available: https://docs.python.org/3/
[19] SQLite Consortium, “SQLite Documentation.” [Online]. Available: https://www.sqlite.org/docs.html
[20] ReportLab, “ReportLab PDF Library Documentation.” [Online]. Available: https://www.reportlab.com/docs/
[21] Plotly Technologies Inc., “Plotly Python Open Source Graphing Library.” [Online]. Available: https://plotly.com/python/