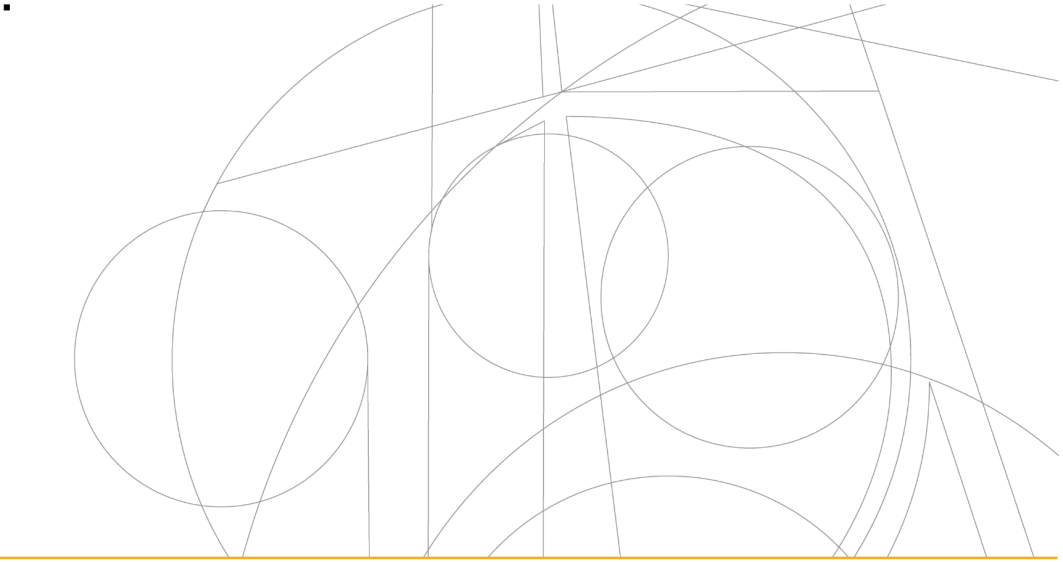

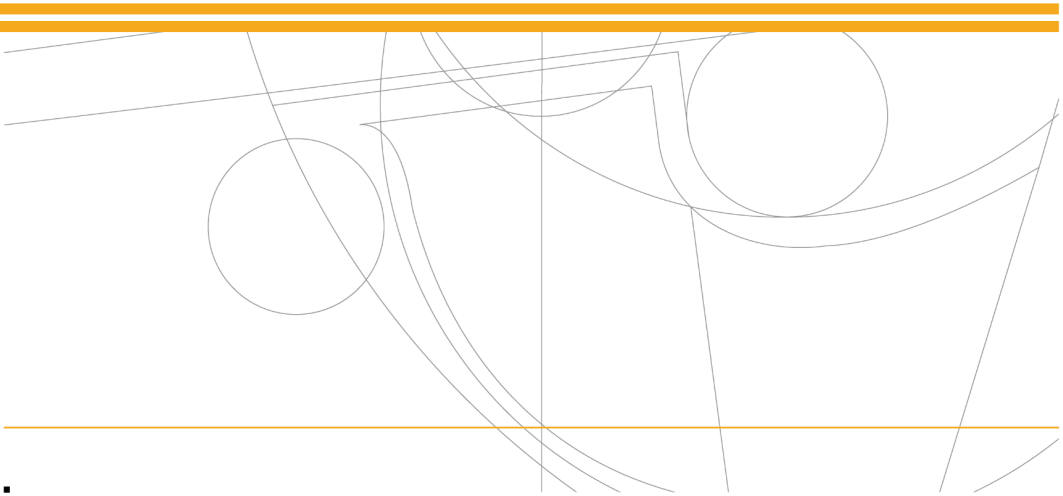

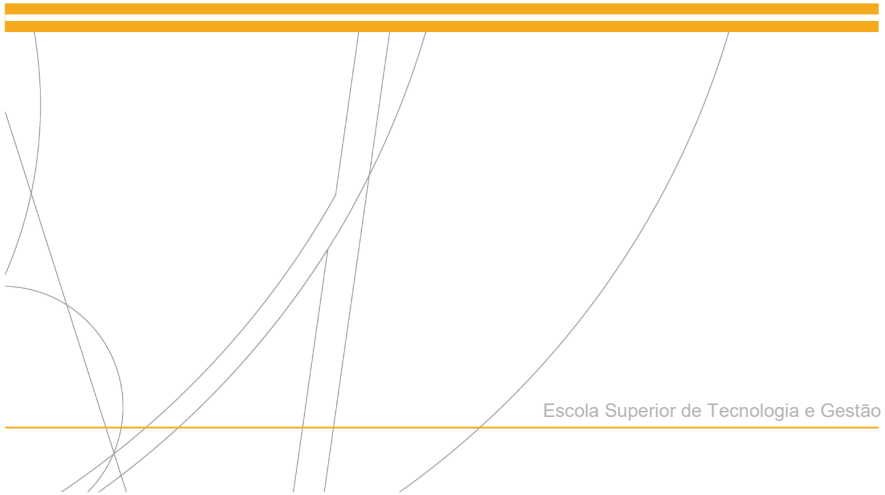

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO
ESTG

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

An Open-Source, Auditable, and Licensing-Cost-Efficient Framework for Managed Security Services: Towards Digital Sovereignty

Sérgio Tiago Gomes Serra

An Open-Source, Auditable, and Licensing-Cost-Efficient Framework for Managed Security Services: Towards Digital Sovereignty

2026

Escola Superior de Tecnologia e Gestão

Instituto Politécnico
de Viana do Castelo

An Open-Source, Auditable, and Licensing-Cost-Efficient Framework for Managed Security Services: Towards Digital Sovereignty

Autor

Sérgio Tiago Gomes Serra

Trabalho orientado por

Professor Doutor Silvestre Lomba Malta,

Professor Doutor Pedro Filipe Cruz Pinto

Mestrado em Cibersegurança

7 de setembro de 2026

Mestrado em Cibersegurança
Master in Cybersecurity

An Open-Source, Auditable, and Licensing-Cost-Efficient Framework for Managed Security Services: Towards Digital Sovereignty

a master's thesis authored by

Sérgio Tiago Gomes Serra

and supervised by

Silvestre Lomba Malta

Professor Adjunto, Instituto Politécnico de Viana do Castelo

Pedro Filipe Cruz Pinto

Professor Adjunto, Instituto Superior de Engenharia do Porto

This thesis was submitted in partial fulfillment of the requirements for the Master's degree in Cybersecurity at the Instituto Politécnico de Viana do Castelo

ipvc

September 7, 2026

Abstract

The digital transformation and the increasing sophistication of cyber threats have driven organizations to rely on Managed Security Service Providers (MSSPs) for robust cyber defense capabilities, such as Security Operations Centers (SOCs), threat intelligence, and incident response. However, the delegation of security operations often leads to vendor lock-in due to the prevalence of proprietary technologies. This dependency limits organizations' ability to independently verify security controls, ensure regulatory compliance (e.g., with Network and Information Security Directive 2 (NIS2) and General Data Protection Regulation (GDPR)), and maintain technological sovereignty over their cyber defense infrastructure. To address this gap, this dissertation proposes a comprehensive, open, and auditable MSSP framework designed to preserve technological sovereignty while delivering effective security services. Following a Design Science Research (DSR) methodology, informed by a Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)-based structured literature review, the study identifies the structural limitations of current MSSP architectures and derives verifiable design requirements. The main contribution is a five-layer reference architecture that orchestrates open-source security capabilities, ensuring multi-tenant isolation, continuous auditability, and the portability of operational artifacts. A functional prototype was developed to demonstrate the feasibility of the proposed framework, confirming that operational efficiency does not have to come at the expense of transparency. By shifting the service delivery model from opaque, proprietary ecosystems to inspectable workflows and replaceable components, this research advances the practical implementation of digital sovereignty in cybersecurity.

Keywords: Managed Security Services, Technological Sovereignty, Vendor Lock-In, Auditability, Open Security Architecture, Design Science Research.

Resumo

A transformação digital e a crescente sofisticação das ameaças cibernéticas têm levado as organizações a recorrer a Managed Security Service Providers (MSSPs) para obter capacidades robustas de ciberdefesa, tais como Centros de Operações de Segurança (SOCs), inteligência de ameaças e resposta a incidentes. No entanto, a delegação das operações de segurança resulta frequentemente em dependência de fornecedores (*vendor lock-in*) devido à prevalência de tecnologias proprietárias. Esta dependência limita a capacidade das organizações de verificarem independentemente os controlos de segurança, garantirem a conformidade regulamentar (e.g., com a Diretiva de Segurança das Redes e da Informação 2 (NIS2) e o Regulamento Geral sobre a Proteção de Dados (RGPD)) e manterem a soberania tecnológica sobre a sua infraestrutura de ciberdefesa. Para colmatar esta lacuna, esta dissertação propõe uma *framework* de MSSP abrangente, aberta e auditável, concebida para preservar a soberania tecnológica sem comprometer a eficácia dos serviços de segurança. Adotando a metodologia de Design Science Research (DSR), informada por uma revisão estruturada da literatura baseada no método Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA), o estudo identifica as limitações estruturais das atuais arquiteturas de MSSPs e deriva requisitos de *design* verificáveis. A principal contribuição é uma arquitetura de referência em cinco camadas que orquestra capacidades de segurança *open-source*, garantindo isolamento *multi-tenant*, auditabilidade contínua e a portabilidade de artefactos operacionais. Foi desenvolvido um protótipo funcional para demonstrar a viabilidade da *framework* proposta, confirmando que a eficiência operacional não tem de comprometer a transparência. Ao transitar de ecossistemas proprietários e opacos para *workflows* inspecionáveis e componentes substituíveis, esta investigação promove a implementação prática da soberania digital na cibersegurança.

Palavras-chave: Serviços Geridos de Segurança, Soberania Tecnológica, Vendor Lock-

In, Auditabilidade, Arquitetura Aberta de Segurança, Design Science Research.

Acknowledgments

This journey has been one of the most challenging and rewarding experiences of my life, and it would not have been possible without the people who stood by me along the way.

To my supervisors, Professor Silvestre Malta and Professor Pedro Pinto, I want to express my deepest gratitude for believing in this work from the very beginning and for their constant guidance. Your support was never just academic. It was thoughtful, generous, and genuinely human. You challenged me to think deeper, pushed me when I needed pushing, and offered reassurance when the path felt uncertain. I am truly grateful for everything you gave to this process.

To Engineer Pedro Pereira, my Head of the Information Systems and Administrative Modernization Division, I am thankful for his support and encouragement in my professional environment, which made it possible to pursue this work alongside my responsibilities.

To my friends and colleagues who walked alongside me through late nights, endless doubts, and small victories, you made this feel less like a solitary endeavor and more like a shared adventure. The conversations, the laughter, and the moments of mutual encouragement meant more than you know.

To my family, no words feel quite adequate. You have been my foundation throughout this entire journey. Your patience, your sacrifice, and your love carried me through the moments when I doubted myself most. This work belongs to you as much as it belongs to me.

Thank you all, from the bottom of my heart.

Contents

List of Figures

List of Tables

List of Abbreviations

ACM Association for Computing Machinery

AI Artificial Intelligence

API Application Programming Interface

ATT&CK Adversarial Tactics, Techniques, and Common Knowledge

AutoML Automated Machine Learning

CTI Cyber Threat Intelligence

DSR Design Science Research

EU European Union

GDPR General Data Protection Regulation

GLPI Gestionnaire Libre de Parc Informatique

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IaC Infrastructure as Code

ICS Industrial Control System

IEEE Institute of Electrical and Electronics Engineers

IT Information Technology

LDAP Lightweight Directory Access Protocol

LLM Large Language Model

LR Literature Review

MDR Managed Detection and Response

MISP Malware Information Sharing Platform and Threat Sharing

ML Machine Learning

MSSP Managed Security Service Provider

NFV Network Function Virtualization

NIDS Network Intrusion Detection System

NIS2 Network and Information Security Directive 2

NIST National Institute of Standards and Technology

OAuth Open Authorization

OSS Open Source Software

OT Operational Technology

PRISMA Preferred Reporting Items for Systematic Reviews and Meta-Analyses

SAML Security Assertion Markup Language

SIEM Security Information and Event Management

SLA Service-Level Agreement

SME Small and Medium-sized Enterprise

SOAR Security Orchestration, Automation and Response

SOC Security Operations Center

TLS Transport Layer Security

VPN Virtual Private Network

XDR Extended Detection and Response

Chapter 1

Introduction

This chapter introduces the research context, defining the problem space and the motivations that drive this dissertation. It outlines the research questions and objectives that guide the investigation, highlights the principal contributions of the work, and describes the methodology and structure of the remainder of the document.

1.1 Context and Motivation

The digital transformation of modern society has fundamentally altered the threat landscape faced by organizations of every size and sector. As critical processes migrate to interconnected infrastructure, the frequency, sophistication, and impact of cyberattacks have grown at a pace that consistently outstrips the defensive capacity of most institutions [11, 7]. In this environment, maintaining a dedicated and competent cybersecurity posture has become an existential imperative — yet it remains financially and operationally out of reach for the vast majority of organizations, particularly Small and Medium-sized Enterprises (SMEs) [32, 54].

This tension between the necessity and the cost of robust cyber defense has driven a well-established market shift toward the outsourcing of security functions to external specialists. Managed Security Service Providers (MSSPs) have emerged as a structural response to this challenge, delivering a broad spectrum of security services that extend beyond traditional Security Operations Centers (SOCs) operations. These include continuous monitoring, incident response, threat intelligence, vulnerability management, compli-

ance support, identity and access management, and security architecture advisory, among others. Through scalable, subscription-based models [53, 19], MSSPs enable organizations to access advanced technologies, specialized expertise, and operational maturity without incurring the full costs associated with developing and maintaining these capabilities in-house. This approach not only reduces the barrier to entry for robust cybersecurity practices but also allows organizations to focus on their core business while ensuring a continuously evolving security posture.

At the operational core of MSSP service delivery lies a constellation of interdependent technologies. Security Information and Event Management (SIEM) systems aggregate and correlate log data from heterogeneous sources, providing the analytical foundation for threat detection [45, 23]. Security Orchestration, Automation and Response (SOAR) platforms extend this capability by automating repetitive response workflows and orchestrating security toolchains, thereby reducing response latency and analyst fatigue [9, 21]. Threat hunting disciplines, increasingly augmented by Cyber Threat Intelligence (CTI) and Machine Learning (ML), allow analysts to proactively seek adversarial activity beyond the reach of automated detection rules [6, 33, 41]. Together, these components have shaped what is now understood as the modern SOC — a sociotechnical system that integrates people, processes, and technology in continuous pursuit of situational awareness and resilience [34, 7].

However, the growth of the MSSP market has introduced a set of structural tensions that go beyond operational efficiency. Organizations that delegate their security operations to external providers inevitably transfer not only responsibility but also data, visibility, and control. In contexts where sensitive operational data must traverse provider infrastructures built on proprietary software stacks, vendor dependencies accumulate silently. The inability to inspect, audit, or migrate the security mechanisms underpinning an organization's defense constitutes a form of technological subordination — one that is increasingly recognized as a strategic liability [52, 4].

These concerns are particularly acute in contexts where digital sovereignty and national security intersect. Across Europe, legislative instruments such as the Network and Information Security Directive 2 (NIS2) and the General Data Protection Regulation (GDPR) have formalized requirements for accountability, auditability, and minimum security stan-

dards [22, 23, 14]. These regulatory frameworks implicitly demand that organizations understand, document, and control the security mechanisms they rely upon — a requirement that is difficult to satisfy when those mechanisms are opaque, vendor-controlled, and contractually constrained. Consequently, a growing body of research has begun to articulate the concept of *technological sovereignty* as an organizing principle for cyber defense: the capacity of an organization, institution, or state to independently operate, audit, and adapt its security infrastructure without strategic dependence on third parties [39, 43, 22].

In this context, the concept of open, auditable managed security services acquires renewed significance. If the organizational benefits of MSSP outsourcing can be preserved while the architecture underpinning service delivery is constructed around open standards, non-proprietary formats, inspectable configurations, and auditable governance processes, then the inherent trade-off between delegation and sovereignty may be substantially resolved. Open-source components may support this goal, but the relevant architectural property is openness rather than licensing alone. This represents both a practical engineering challenge and a theoretical contribution to the emerging field of sovereignty-aware cybersecurity design.

1.2 Problem Statement

Despite the maturation of managed security services, a significant gap remains in designing MSSP frameworks that are simultaneously effective, transparent, and sovereignty-preserving.

Currently, most MSSP architectures prioritize operational throughput over architectural openness, leading to proprietary ecosystems. As a result, organizations often face *vendor lock-in* [53, 14, 5]. This dependency restricts their ability to independently verify security controls, transition to alternative providers, or efficiently demonstrate regulatory compliance [52, 5].

In the academic literature, this issue is frequently addressed in isolation. Existing research explores SOC maturity models [49, 46], log management and orchestration via SIEM and SOAR [45, 9], and the economics of security outsourcing [53, 19, 30]. However, an integrated framework that unifies open MSSP architectures, sovereignty-driven

governance, and sustainable operations is still absent.

Even when open-source components are available, orchestrating them into a cohesive, auditable, and compliant managed service presents a complex design challenge [38, 14].

This dissertation addresses this gap by proposing a comprehensive, open, and auditable MSSP framework designed to advance technological sovereignty in cyber defense.

1.3 Research Questions

This investigation is guided by the following primary research question:

> RQ: *How can an integrated, open, and auditable framework for Managed Security Services be designed to mitigate vendor lock-in and advance technological sovereignty in cyber defense?*

To operationalize this inquiry and directly address the gaps identified in the problem statement, the following subsidiary questions are proposed:

RQ1: What are the structural limitations of current MSSP architectures regarding vendor lock-in, transparency, and sovereignty, as evidenced by the academic literature?

RQ2: What specific design requirements must an MSSP framework fulfill to enable independent verification of security controls, ensure regulatory compliance, and unify open architectures with sovereignty-driven governance?

RQ3: How can disparate open-source security solutions be effectively orchestrated into a cohesive, auditable, and sustainable managed service architecture?

RQ4: To what extent does the proposed integrated framework satisfy these theoretical design requirements when evaluated against representative architectural constraints?

These questions frame the research as a dual endeavor: diagnosing the MSSP landscape to uncover architectural gaps and constructively designing a framework to resolve them.

1.4 Objectives

In alignment with the formulated research questions, this dissertation pursues the following objectives:

1. Characterize the state of the art in Managed Security Services, Security Operations Centers, and related technologies through a Literature Review (LR) based on Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) principles.

2. Identify and analyze the structural limitations of current MSSP models—particularly vendor lock-in, transparency deficiencies, and governance gaps—through a thematic synthesis.

3. Derive and formalize design requirements for an open MSSP framework that enables independent verification, ensures regulatory compliance, and supports sovereignty-driven governance.

4. Design an integrated, open, and auditable MSSP framework architecture that orchestrates security capabilities into a cohesive and deployable reference model.

5. Evaluate the proposed architecture against the derived requirements and representative architectural constraints, assessing its consistency, applicability, and potential to mitigate proprietary dependencies.

1.5 Research Methodology

This dissertation adopts two complementary methodological pillars: a LR for knowledge synthesis, and Design Science Research (DSR) for artifact design [20, 37]. The LR, detailed in Chapter 3, provides the evidential foundation upon which the proposed framework is derived and evaluated.

DSR is an established paradigm for the creation and evaluation of artifacts that address identified problems in specific domains [20, 37]. It is particularly suited to this research, as the objective extends beyond analysis to the design of a novel, open, and auditable MSSP framework.

The adoption of DSR is justified by three main factors: (i) the relevance of the identified problem, namely the lack of sovereignty-oriented MSSP frameworks; (ii) the artifact-centric nature of the research, focused on the design of an architectural framework; and

(iii) the need to derive generalizable design principles grounded in both literature and practice.

The research process follows the canonical DSR cycles:

- Relevance Cycle: Ensures alignment with real-world needs, informed by the contextual analysis, sovereignty foundations, and literature review findings. It defines the requirements that the framework must address.
- Design Cycle: Encompasses the iterative development and evaluation of the framework. This includes the derivation of design requirements, the definition of the architecture (Chapter 4), and its evaluation against those requirements and representative architectural constraints (Chapter 6).
- Rigor Cycle: Grounds the research in the existing knowledge base, drawing from the literature review, technological sovereignty principles, and established cybersecurity frameworks (e.g., National Institute of Standards and Technology (NIST), MITRE's ATT&CK).

Following the process proposed by Peffers et al. [37], this dissertation applies DSR through the following structured sequence:

1. Problem identification and motivation: Established through contextual analysis and validated by the literature review.
2. Definition of objectives: Derived from the research questions and formalized as design requirements for a sovereignty-oriented MSSP framework.
3. Design: The framework is developed through iterative refinement, integrating architectural, governance, and auditability dimensions into a cohesive reference model (Chapter 4).
4. Evaluation: The proposed architecture is assessed against the defined requirements and representative architectural constraints, analyzing its consistency, applicability, and potential to mitigate vendor lock-in (Chapter 6).
5. Communication: The research contributions are disseminated through this dissertation.

1.6 Contributions

Following this methodological process, this dissertation makes the following key contributions to the field of cybersecurity research and practice:

- A literature review on Managed Security Services, SOC architectures, and security orchestration, culminating in a thematic coverage map that identifies structural limitations, particularly vendor lock-in and transparency deficiencies.
- A set of verifiable design requirements for open MSSP frameworks, derived from the literature and aligned with regulatory frameworks such as NIS2 and GDPR, formalizing technological sovereignty as a core architectural principle.
- A novel, open and auditable MSSP framework architecture that composes security capabilities through open interfaces and non-proprietary artifacts, reducing structural dependency on proprietary vendors.
- A functional prototype that demonstrates the feasibility of the proposed architecture in a representative environment and supports its practical applicability as a reference implementation.

1.7 Dissertation Structure

The remainder of this dissertation is organized as follows:

Chapter 2 provides the background for the dissertation. It is organized around two principal sections: MSSPs and technological sovereignty, establishing the main concepts, architectural notions, governance concerns, and sovereignty-related foundations that support the remainder of the work.

Chapter 3 presents the related work and recent developments identified through the literature review. The chapter documents the PRISMA-informed review process and organizes the literature across the main dimensions of managed security services and sovereignty-aware cybersecurity, highlighting convergences, limitations, and research gaps.

Chapter 4 presents the proposed open and auditable MSSP framework. The chapter describes the derived design requirements, the architectural principles, the governance

mechanisms, and the component integration model.

Chapter 5 details the development of a functional prototype that instantiates the proposed framework, describing the technical implementation decisions, the tools adopted, and the prototype's alignment with the design requirements.

Chapter 6 discusses the findings of the research, evaluating the framework against the derived requirements, situating the contributions in the broader literature, and reflecting on limitations and future research directions.

Chapter 7 concludes the dissertation by summarizing the key contributions, the answers to the research questions, and the implications of the work for research, practice, and regulation.

Chapter 2

Background

This chapter establishes the conceptual basis of the dissertation. It first examines MSSPs as an operational and organizational model for outsourced cyber defense. It then frames technological sovereignty as the criterion that determines whether outsourcing expands defensive capability without eroding strategic control. The chapter concludes by defining the analytical lens used in Chapter 3.

2.1 Managed Security Service Providers

MSSPs have become a central mechanism through which organizations obtain continuous cyber defense without building and sustaining the full capability internally. This section characterizes the model as the operational and organizational foundation on which the remainder of the dissertation builds. It first clarifies what MSSPs are, why demand for them has grown, and how their service portfolios have evolved. It then examines their core service domains and operating model, the architectural foundations that make delivery scalable, and the organizational arrangements and governance trade-offs that accompany outsourcing. Taken together, these dimensions expose a recurring tension between the efficiency of delegated security operations and the client's ability to retain meaningful oversight, a tension that motivates the sovereignty-oriented perspective developed in the sections that follow.

2.1.1 Definition, Drivers, and Evolution

MSSPs deliver cybersecurity capabilities to client organizations through continuous service relationships rather than isolated product sales. Their expansion has been driven by the growing scale and complexity of cyber threats, the shortage of specialized personnel, and the fact that continuous monitoring remains economically difficult for many organizations to sustain internally, especially SMEs [53, 19, 32, 54]. In this sense, the MSSP market is not merely a convenience layer around existing tools; it is a response to a structural imbalance between the need for permanent cyber defense and the uneven distribution of technical, financial, and human resources across organizations [30, 52].

The MSSP model has also evolved substantially. Early offerings concentrated on perimeter management and reactive monitoring, whereas contemporary service portfolios are centered on integrated detection, response, and governance functions built around SOCs, SIEM, SOAR, threat intelligence, and incident handling workflows [45, 9, 21]. As services have expanded toward Managed Detection and Response (MDR) and Extended Detection and Response (XDR)-like operating models, the provider's technology stack has become deeper and more tightly coupled to the client's security posture, increasing both operational value and dependency on provider-controlled infrastructure [7, 53, 52].

2.1.2 Core Service Domains and Operating Model

A contemporary MSSP operates through a set of interdependent service domains rather than through a single platform. As summarized in Figure 2.1, which adapts a public MSSP service taxonomy [15], those domains typically include threat monitoring and detection, network and data protection, identity and access management, incident response and risk management, and complementary services such as malware protection, ransomware defense, and cloud security. Together, they span the full cycle from telemetry collection and correlation to containment, recovery, and reporting [12, 6, 10, 7, 28].

The operational center of this model is usually the SOC. The literature consistently describes SOC effectiveness as a socio-technical outcome produced by the interaction between people, processes, and technology, rather than as the direct consequence of tool acquisition alone [49, 12, 34]. This is particularly relevant in the MSSP context because

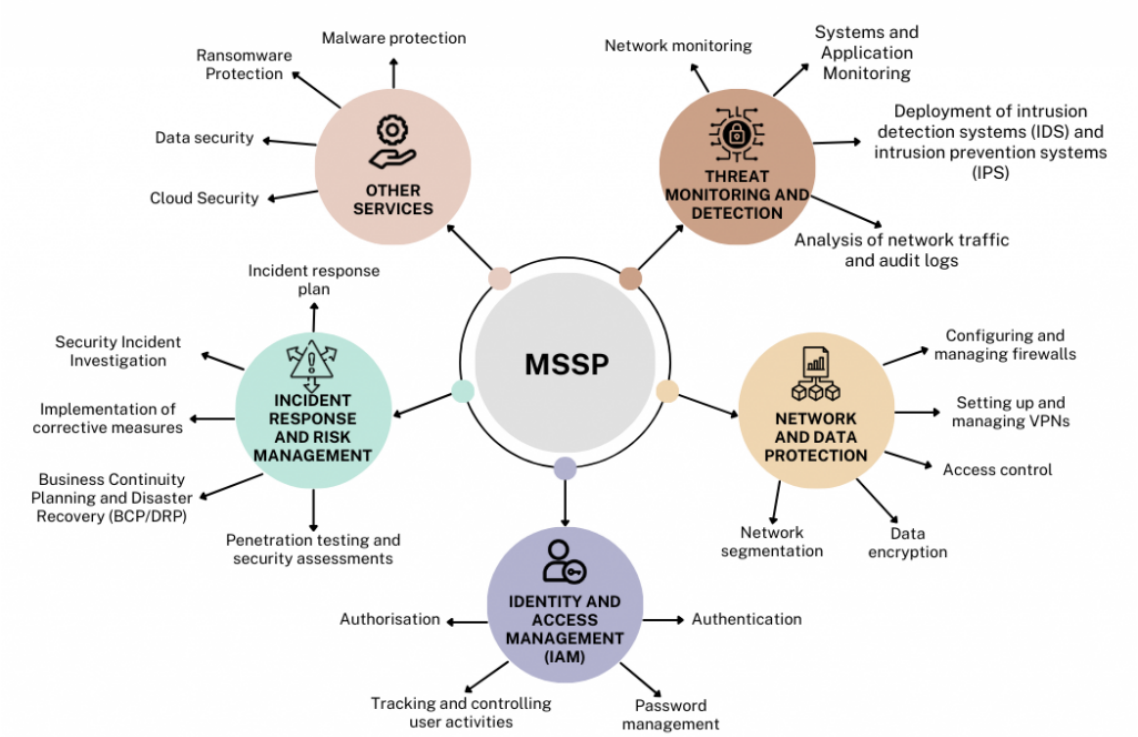

Figure 2.1: Principal MSSP service domains, grouping threat monitoring and detection, network and data protection, identity and access management, incident response and risk management, and complementary security services. Source: HTTPCS [15].

outsourced security value depends on coordination across organizational boundaries: intelligence must inform detection, detection must trigger disciplined response, and both must be documented in a way that remains understandable to the client organization [7, 4].

2.1.3 Architectural Foundations of MSSP Delivery

From a technical perspective, modern MSSPs rely on layered data and orchestration pipelines. SIEM platforms remain the analytical backbone because they collect, normalize, correlate, and retain telemetry from heterogeneous sources [45, 23, 26]. SOAR platforms extend this backbone by orchestrating tools, codifying playbooks, and automating repetitive response tasks, thereby increasing throughput and consistency when properly configured [9, 21, 48]. CTI and threat hunting capabilities complement these layers by moving the service beyond passive alert handling toward proactive detection of emerging adversarial behavior [6, 33, 24, 41].

Scalability, however, requires more than functional tooling. MSSPs must support multi-tenancy, tenant isolation, and interoperable integration across their service stack [14, 38, 54]. Multi-tenant architectures improve economic viability, but they also intensify requirements for segregation of data, reproducibility of configurations, and visibility into cross-tool workflows. In other words, the very architectural features that make MSSP delivery scalable also become the points at which governance, compliance, and sovereignty concerns materialize.

2.1.4 Organizational Arrangements and Governance Trade-Offs

At the organizational level, MSSP delivery spans a continuum ranging from full outsourcing to hybrid and shared-service arrangements. Economic and decision-oriented studies show that the attractiveness of each model depends on cost structures, attack conditions, staffing constraints, and desired degrees of retained control [53, 19, 30, 25]. Shared and community-based SOC models seek to lower entry barriers for smaller organizations by pooling infrastructure and expertise, while hybrid models attempt to preserve internal decision authority over the most critical functions [31, 54, 25]. These arrangements indicate that MSSP adoption is rarely an all-or-nothing decision; it is usually a negotiation between affordability, capability, and retained oversight.

Because MSSPs act on behalf of the client, governance is inseparable from service delivery. Service-Level Agreements (SLAs), data governance, risk allocation, audit rights, and regulatory compliance define how accountability is distributed across organizational boundaries [4, 23, 28]. The literature repeatedly points to an asymmetry of information in which the provider possesses detailed visibility into operations while the client often receives only contractual assurances and aggregated reports [4, 52]. This asymmetry is the point at which the MSSP model intersects directly with technological sovereignty: outsourced capability can improve security outcomes, but it can also reduce the client's ability to inspect, verify, and reconfigure the defensive system on which it depends.

2.2 Technological Sovereignty

Technological sovereignty is the criterion this dissertation uses to judge whether outsourced cyber defense strengthens an organization without eroding its strategic control. This section develops the concept as the analytical counterpart to the MSSP model introduced above. It first sets out the conceptual foundations, distinguishing legitimate control from mere data residency, and then draws on these foundations to define the dimensions of sovereignty that structure the analysis in the chapters that follow. The aim is to establish a precise vocabulary for reasoning about who can inspect, verify, and reconfigure a security service, and under what conditions that authority remains with the client.

2.2.1 Conceptual Foundations

Technological sovereignty extends the classical idea of sovereignty into the digital domain by focusing on meaningful control over the technological systems on which organizations and institutions depend. Floridi [17] frames this as part of the wider struggle for digital sovereignty, while Pohle and Thiel [39] show that the concept has evolved into a broad normative and policy discourse concerning data, infrastructure, platforms, and strategic autonomy. Roberts [43] sharpens the concept by distinguishing descriptive control from legitimate control, arguing that sovereignty should not be understood merely as the fact of domination over technology, but as authority that is justified, transparent, and accountable.

In cybersecurity, this distinction is critical. An organization may comply with data residency requirements while still lacking the capability to inspect detection logic, validate response workflows, or migrate between providers without significant disruption. Therefore, technological sovereignty imposes stricter requirements than data sovereignty alone [39]. The framework proposed by Kianpour et al. [22] is especially useful in this dissertation because it identifies five recurring sovereignty dimensions: governance authority, infrastructure and service control, data protection, openness balanced with autonomy, and resilience. Taken together, these dimensions shift the question from "who operates the service" to "under what conditions can the client still exercise legitimate control over that service."

Within the scope of this dissertation, technological sovereignty is therefore operationalized as the capacity of an organization to understand, operate, audit, adapt, and replace the technical and governance mechanisms through which cybersecurity services are delivered, without strategic dependence on any single external actor. This definition aligns the dissertation with a normative view of sovereignty and makes sovereignty assessable through architectural and governance properties rather than through political rhetoric alone [43, 22].

2.2.2 Sovereignty Assessment Dimensions

To move sovereignty from a normative concept to an assessable architectural property, this dissertation operationalizes technological sovereignty through four complementary assessment dimensions derived from the convergence of the Kianpour et al. framework [22] and the analytical questions established above.

Control. The degree to which the client organization governs infrastructure, detection logic, and operational decisions within the managed service. This dimension corresponds to Kianpour et al.'s governance authority and infrastructure and service control dimensions and to the analytical question of who controls the infrastructure and detection logic.

Transparency. The degree to which service operations, data flows, configurations, and decision processes are inspectable and intelligible to the client. This dimension intersects with Kianpour et al.'s data protection and governance authority dimensions, and corresponds to the analytical question of how the client can audit service execution.

Portability. The degree to which operational artifacts—detection rules, playbooks, incident records, configurations, and deployment definitions—can be exported, migrated, and reconstructed independently of the current provider. This dimension corresponds to Kianpour et al.'s openness balanced with autonomy dimension and to the analytical question of how portable data, rules, and workflows remain.

Independence. The degree to which the managed service can be operated, adapted, or

replaced without strategic dependence on any single external actor. This dimension synthesizes Kianpour et al.'s resilience dimension with the analytical questions concerning compliance operationalization across organizational boundaries and long-term sustainability.

Table 2.1 maps these four assessment dimensions to the five Kianpour et al. dimensions and to the five analytical questions, ensuring that the assessment model is traceable to its theoretical foundations.

Table 2.1: Mapping of sovereignty assessment dimensions to theoretical foundations.

Assessment dimension	Kianpour et al. dimensions	Analytical questions
Control	Governance authority; Infrastructure and service control	Who controls the infrastructure and detection logic?
Transparency	Data protection; Governance authority	How can the client audit service execution?
Portability	Openness balanced with autonomy	How portable do data, rules, and workflows remain?
Independence	Resilience; Openness balanced with autonomy	How are compliance and accountability operationalized? Is the service model sustainable?

For evaluation, each dimension is assessed on a three-level qualitative scale: *structurally enabled*, indicating that the architectural design and prototype instantiation provide direct, verifiable support for the property; *partially demonstrated*, indicating that structural support exists but full validation requires production-scale, longitudinal, or adversarial testing not performed within this dissertation; and *not addressed*, indicating that the dimension falls outside the framework's current scope. This scale is deliberately qualitative, reflecting the nature of the prototype as an architectural demonstration within a DSR process rather than as a production-grade benchmark [37].

2.2.3 Regulatory and Strategic Drivers

The relevance of technological sovereignty in cybersecurity is reinforced by regulatory and strategic pressures. NIS2 broadens organizational obligations for risk management, incident handling, and supply-chain accountability, including situations in which security functions are delivered by external providers [22, 54]. GDPR similarly transforms data handling from a matter of contractual trust into one of demonstrable compliance, which

is particularly salient when MSSPs process log data, identity records, and other security-relevant information on behalf of clients [23, 14]. The Cyber Resilience Act extends lifecycle obligations to products with digital elements and raises the compliance burden across software supply chains, including components that may form part of an MSSP platform [42].

These instruments do not abolish outsourcing, but they constrain opaque outsourcing. They imply that client organizations retain non-transferable responsibilities even when detection and response are delegated, and that providers must support verifiability rather than merely promise it [53, 52]. In this sense, regulation and sovereignty converge: both require that security controls, data flows, and governance arrangements remain intelligible enough to be audited and defended.

2.2.4 Open-Source and Interoperability as Sovereignty Enablers

If sovereignty requires inspectability and replaceability, then the openness of the technical stack becomes a structural issue. Open Source Software (OSS) can contribute to sovereignty because it improves access to source code, facilitates adaptation of detection and orchestration logic, and often encourages modular integration through open interfaces and standard formats [45, 21, 6]. From the perspective of lock-in mitigation, interoperability and portability are as important as software licensing: organizations need to preserve access to data, rules, workflows, and deployment knowledge if they are to migrate across providers or operate capabilities internally [5, 14].

However, openness is not sufficient on its own. An MSSP stack built from open components can still become opaque if integrations are ad hoc, if tenant boundaries are poorly defined, or if governance processes remain undocumented [9, 5]. The significance of open architectures therefore lies not in ideology but in the way they can support auditability, portability, and controlled customization when embedded in a disciplined service model [38, 14].

2.2.5 Auditability and Transparency

Auditability is the operational expression of sovereignty. It determines whether claims of control can be translated into evidence concerning how the service works, how inci-

dents were handled, and whether the agreed controls were actually applied [4, 52]. In the MSSP context, auditability has at least three intertwined dimensions. Technical auditability concerns access to configurations, logs, detection rules, and execution traces; process auditability concerns the traceability of incident handling and change management; governance auditability concerns reporting, oversight, and the possibility of verifying service fulfillment against contractual or regulatory obligations [49, 18, 23].

Transparency is closely related, but it is broader than logging. A transparent MSSP architecture makes data flows, component responsibilities, and automated decisions intelligible to actors outside the provider's internal operations. This requirement becomes even more important as automation and ML are introduced to reduce alert fatigue and improve decision support. Research on Automated Machine Learning (AutoML), orchestration, and automation bias shows that efficiency gains are real, but that opacity can increase if the reasoning behind prioritization and automated actions is not made inspectable to analysts and clients [40, 50, 55].

2.2.6 Resilience and Sustainability

A sovereignty-preserving MSSP must also remain viable over time. Resilience and sustainability concern the ability to maintain, evolve, and staff the service without degrading security quality or recreating dependency in a different form [22, 46]. This includes technical resilience, such as reproducible deployment and modular change; organizational resilience, such as process maturity and knowledge retention; and human sustainability, such as reducing analyst overload and supporting measurable operational competence [2, 34, 40]. It also includes the sustainability of component ecosystems themselves, since compliance and maintenance obligations may become difficult for under-resourced projects to meet over the long term [42].

2.3 Summary

This chapter has positioned MSSPs as scalable but governance-intensive instruments of cyber defense, and technological sovereignty as the condition that determines whether such scalability remains compatible with legitimate control.

Taken together, the MSSP and sovereignty literatures imply that managed security should not be evaluated only by detection efficacy or by outsourcing cost. For this dissertation, five questions are especially important: who controls the infrastructure and detection logic; how the client can audit service execution; how portable data, rules, and workflows remain; how compliance and accountability are operationalized across organizational boundaries; and whether the service model is sustainable in technical, economic, and human terms [43, 22, 4, 53, 5]. These questions have been formalized in Section 2.2.2 as four assessment dimensions—control, transparency, portability, and independence—that will guide both the derivation of design requirements and the evaluation of the proposed framework.

These criteria structure the analysis in Chapter 3 and explain why openness, auditability, and sovereignty are treated in this dissertation as architectural requirements rather than as optional add-ons. The next chapter uses this lens to compare existing work, identify its convergences and blind spots, and clarify the research gap addressed by the proposed framework.

Chapter 3

Related Work

Building on the conceptual basis established in Chapter 2, this chapter reviews the literature most directly relevant to the design of sovereign, open, and auditable managed security services. The review does not treat MSSPs only as a procurement model or as an aggregation of security tools. Instead, it examines how outsourcing arrangements [53, 19], SOC operating models and SIEM/SOAR/CTI architectures [45, 9], governance mechanisms [4], and digital sovereignty research [43] together define the design problem addressed in this dissertation.

The chapter is organized as a critical synthesis rather than as a catalog of individual publications. Its central argument is that the literature is technically and conceptually rich, but insufficiently integrated for the specific problem of designing MSSPs that preserve client control, transparency, portability, and independence. Outsourcing studies explain why organizations delegate security operations; SOC studies clarify the human and operational foundations of detection and response; technical studies show how monitoring, orchestration, and intelligence capabilities can be composed; governance studies address risk, compliance, and assurance; and sovereignty studies explain why legitimate control over digital infrastructure matters [52, 14, 22, 5]. What remains underdeveloped is a framework-level treatment of the MSSP as a managed, multi-tenant, auditable, and portable service architecture.

This gap is consequential because managed security services sit at the boundary between capability acquisition and strategic dependency. Delegation can improve cybersecurity capacity, especially for organizations that cannot sustain mature internal operations,

but it can also transfer decisive knowledge about telemetry pipelines, detection rules, response playbooks, incident records, and compliance evidence into provider-controlled environments. The purpose of this chapter is therefore to identify what existing research already contributes to the proposed framework and, more importantly, what design problems remain unresolved.

3.1 Methodology of the Structured Literature Review

To ground the chapter in a transparent and reproducible evidence base, the literature was reviewed through a structured process informed by PRISMA 2020 [36]. PRISMA was used here as a reporting and selection support framework for identification, screening, eligibility assessment, and inclusion, rather than as a claim that the chapter constitutes a formal systematic review. The aim was to assemble a corpus that was sufficiently rigorous to support thematic comparison and gap identification in a rapidly evolving research area. Specifically, the following PRISMA 2020 elements were adopted: a structured and documented search strategy, explicit eligibility criteria, a multi-stage screening and selection process, and a flow diagram reporting the disposition of records at each stage. Elements not adopted include risk of bias assessment, certainty of evidence grading, protocol preregistration, and formal meta-analysis, as these are characteristic of full systematic reviews and are not required for a structured review whose primary purpose is to support thematic synthesis and gap identification within a DSR process.

3.1.1 Search Strategy and Data Sources

The search strategy was designed to capture recent academic work at the intersection of managed security services and architectural or operational design. Three databases were selected because of their complementary coverage of cybersecurity, computer science, and engineering publications: Association for Computing Machinery (ACM) Digital Library, Institute of Electrical and Electronics Engineers (IEEE) Xplore, and Scopus. Searches were applied to title, abstract, and keyword fields, and were restricted to peer-reviewed journal articles and conference papers published between January 2023 and December 2025. The date restriction was adopted to emphasize the current generation of MSSP,

SOC, SIEM, SOAR, CTI, and sovereignty-related work, while foundational conceptual sources on digital sovereignty were added separately through manual identification where required to support the analytical framing.

The Boolean query used across the three databases was:

```
("Security Operations Center" OR "SOC as a Service" OR "SOCaaS" OR "managed
security services" OR "Security as a Service" OR "SECaaS") AND ("framework"
OR "architecture" OR "model" OR "approach" OR "design" OR
"reference architecture" OR "operating model")
```

The query was intentionally focused on SOC and MSSP architectural work. Sovereignty-specific and vendor lock-in-specific terms were not included in the primary query to avoid conflating distinct research streams; instead, these dimensions were addressed through the manually incorporated complementary sources described below, which provide the conceptual and regulatory framing for the sovereignty analysis.

Table 3.1 summarizes the number of records retrieved from each database.

Table 3.1: Records retrieved from the selected databases.

Database	Records Retrieved
ACM Digital Library	62
IEEE Xplore	87
Scopus	276
Total	425

3.1.2 Eligibility Criteria and Study Selection

Eligibility criteria were defined before screening to keep the corpus aligned with the dissertation's research problem.

- Inclusion criteria
  - Studies retrieved through the database search and published in peer-reviewed journals or conference proceedings between January 2023 and December 2025.
  - Studies proposing, evaluating, or critically discussing frameworks, architectures, operating models, or comparable design-oriented contributions related

to MSSPs, SOCs, Security-as-a-Service, or adjacent security operations capabilities.

  - Studies addressing at least one of the following domains: MSSP outsourcing, SOC design and operations, SIEM, SOAR, CTI, incident response, governance and compliance, multi-tenancy, auditability, or vendor dependency.
  - Studies available in English and accessible in full text.

- Exclusion criteria
  - Studies focused exclusively on isolated algorithms, narrowly scoped detection techniques, or single-tool optimizations without architectural or operational relevance.
  - Studies outside the MSSP and security-operations problem space.
  - Duplicate records.
  - Editorials, posters, book reviews, workshop summaries, and extended abstracts without substantive analytical content.
  - Records unavailable in full text.

The PRISMA-informed selection flow proceeded in four stages for the database-derived corpus. First, 425 records were identified through database searching. After removing 61 duplicates, 364 records remained for title and abstract screening. This stage excluded 261 studies that were outside the scope or lacked architectural relevance. The remaining 103 studies were then assessed in full text. At the eligibility stage, 58 records were excluded, primarily because they were out of scope, did not offer a meaningful framework or architectural contribution, or lacked sufficient analytical depth. This left 45 peer-reviewed studies from the database search. To support the conceptual and regulatory framing of the dissertation, 6 additional sources were manually incorporated from external sources, covering digital sovereignty, vendor lock-in, and the European regulatory context. The final qualitative synthesis therefore draws on a combined corpus of 51 sources, while preserving the distinction between the PRISMA-screened database corpus and the complementary manually added sources [17, 39, 43, 22, 42, 5].

3.1.3 Flow Diagram

The study selection process was structured around the four main phases associated with PRISMA 2020: Identification, Screening, Eligibility, and Inclusion. Figure 3.1 illustrates the flow of records through these phases as adapted for this review.

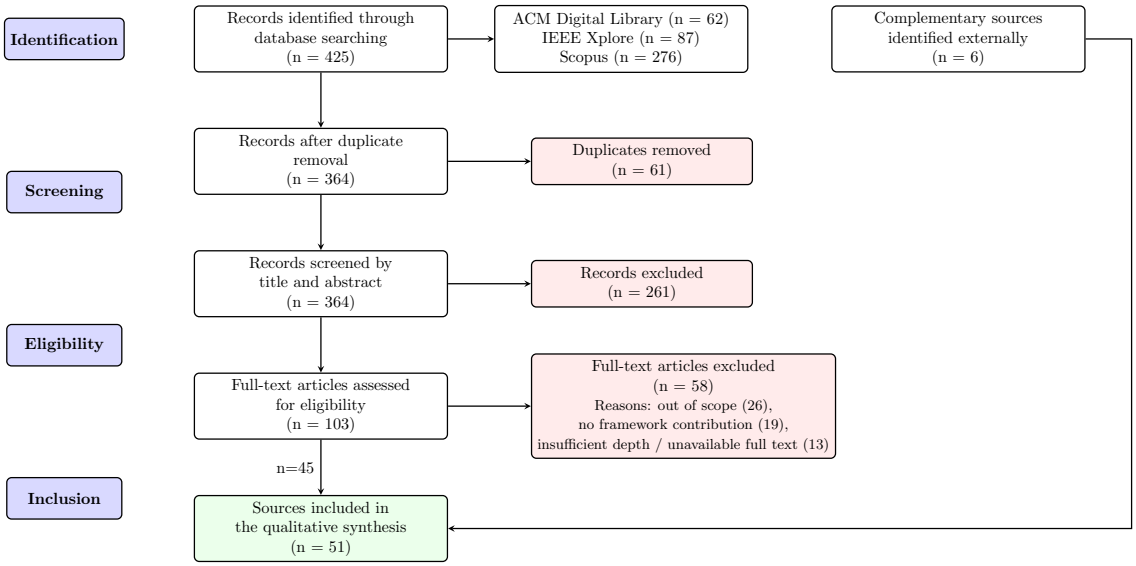

Figure 3.1: PRISMA-informed flow diagram illustrating the study selection process.

3.1.4 Data Extraction and Thematic Coverage Map

For each included study, the review extracted bibliographic metadata, research focus, methodological orientation, type of contribution, technical scope, governance or compliance dimension, and relevance to openness, auditability, portability, or sovereignty. The synthesis then proceeded through thematic coding, allowing studies to map to more than one cluster when their contributions cut across domains. The thematic coding was performed by a single researcher. While guided by explicit eligibility and thematic criteria, this approach may be subject to interpretive bias, which is acknowledged as a limitation of the review's internal validity.

Table 3.2 summarizes the thematic coverage used to structure the analysis.

This coverage map already suggests the main pattern that becomes clearer in the analysis below: the literature is rich in component-level insights but comparatively weak in integrated design thinking at the intersection of MSSP delivery, openness, auditability, and sovereignty.

Table 3.2: Thematic coverage map of the reviewed literature.

Theme	Analytical focus	Representative studies
MSSP service models and outsourcing	Cost structures, adoption incentives, shared and hybrid service models, retained control	[53, 19, 30, 52, 32, 31, 25, 54]
SOC architecture and operations	Maturity, analyst performance, technical metrics, situational awareness, deployment models	[49, 2, 1, 12, 11, 13, 18, 34, 7, 46, 29]
Open security stacks and orchestration	SIEM, SOAR, CTI, open integration, automation, federated architectures	[45, 23, 26, 27, 9, 21, 48, 6, 33, 24, 3, 41, 35, 38, 14, 16]
Governance, compliance, and risk	SLA verification, GDPR/NIS2 alignment, dynamic risk analysis, accountability mechanisms	[4, 23, 14, 28, 10, 47, 22]
Sovereignty, lock-in, and sustainability	Legitimate control, dependency, portability, openness, operational and ecosystem sustainability	[17, 39, 43, 5, 40, 51, 8, 55, 44, 50, 42]

3.2 Analysis of the Literature Review

Building on the coverage map presented above, this section develops a critical thematic reading of the selected literature rather than a purely descriptive summary. The reviewed studies are grouped into seven complementary themes: security outsourcing and retained authority; SOC capability as a socio-technical foundation; the SIEM, SOAR, and CTI operational stack; federated and service-oriented architectures; digital sovereignty, lock-in, and openness; governance, compliance, and assurance evidence; and operational sustainability and responsible automation. Each theme is examined not only for what it contributes, but also for the limitations it leaves unresolved when assessed against the dissertation's concern with open, auditable, and sovereignty-preserving MSSP delivery.

The analysis is guided by a consistent question: to what extent does each body of work enable a client to understand, verify, and retain control over a managed security service? Reading the literature through this lens exposes a recurring pattern in which technical and operational maturity is well developed within individual streams, yet the conditions under which delegation remains compatible with client sovereignty are only partially addressed. This tension motivates the comparative synthesis and gap analysis that follow.

3.2.1 Security Outsourcing and the Problem of Retained Authority

The outsourcing literature explains why organizations adopt MSSPs, but it only partially explains how outsourced security can remain governable by the client. Economic and decision-oriented studies generally model outsourcing as an allocation problem shaped by cost, risk, security externalities, and resource scarcity. Wu et al. [53] compare in-house, full outsourcing, and partial outsourcing strategies, showing that the distribution of core and non-core security functions matters under different externality conditions. Gao et al. [19] extend this view by modeling outsourcing under resource sharing and different attack modes, while Nazareth et al. [30] show that pricing structure and risk uncertainty influence the adoption of Security-as-a-Service among SMEs.

This work is valuable because it demonstrates that outsourcing is rarely a binary decision. It also supports the dissertation's focus on hybrid and modular service arrangements. Its limitation, however, is that the provider is usually represented as an economic actor, not as a technical and governance architecture whose internal opacity may itself create dependency.

The same tension appears in studies of shared or community-oriented security operations. Ntingi et al. [32, 31] argue that resource-constrained organizations often need shared security services because they lack the capital and expertise required for mature internal cyber defense. Zacharioudakis and Kakoulli [54] propose a collaborative, multi-tenant SOC model for SMEs, explicitly linking shared service delivery to regulatory needs such as NIS2. Miyashita and Contreras Pinochet [25] further show that financial SMEs may prefer hybrid SOC architectures that combine outsourced first-level response with retained internal management.

The architectural question, therefore, shifts from whether shared services are useful to how they can remain accountable. Collectively, this literature strengthens the case for managed and shared security models, but it leaves important questions unresolved: how tenants are isolated, how service actions are evidenced, how detection and response logic can be inspected, and how a client exits the arrangement without losing operational knowledge.

Empirical work on Security-as-a-Service reinforces this concern. Vedadi et al. [52] show

that post-adoption satisfaction depends on internal security resources and mature auditing practices. This finding is central to the dissertation because it contradicts the idea that outsourcing eliminates the need for client-side competence. Instead, outsourcing changes the type of competence required: clients need enough retained knowledge and evidence access to interpret provider actions, verify commitments, and preserve strategic control. The outsourcing literature therefore motivates MSSP adoption, but it does not yet define the architectural conditions under which delegation remains compatible with sovereignty.

3.2.2 SOC Capability as a Socio-Technical Foundation

The SOC literature provides the operational foundation for managed security delivery. It consistently treats security operations as a socio-technical system in which people, processes, and technologies must be coordinated to produce reliable detection, response, and situational awareness [49, 12, 34]. This is an essential correction to tool-centric views of MSSP design.

For this dissertation, the implication is direct: a managed service cannot be made sovereign merely by selecting open components. It also requires explicit processes, measurable responsibilities, and evidence-producing workflows.

Several studies attempt to make SOC capability measurable. Taqafi et al. [49] propose a maturity capability framework, Agyepong et al. [2] provide a weighted method for assessing analyst performance, and Forsberg and Frantti [18] argue that many existing metrics overemphasize operational indicators while underrepresenting technical detection performance. Chamkar et al. [12, 11, 13] examine analyst performance, SOC lifecycle practices, and MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK)-based use-case coverage.

Together, these works establish that visibility, maturity, and measurement are necessary for accountable security operations. However, they mostly frame measurement as an internal management problem. In an MSSP context, the relevant question is broader: whether those metrics, use cases, and operational records can be exposed to clients in a tenant-scoped, intelligible, and audit-ready form.

Research on situational awareness and incident response adds a second layer of insight. Ofte and Katsikas [34] show that situation awareness in SOCs is conceptually important

but unevenly operationalized, while Andreassen et al. [7] explicitly study leading MSSPs and show that incident response depends on knowledge flows across analysts, managers, and decision-makers. Agyepong and Onwubiko [1] emphasize the value of structured incident response plans for analysts with different experience levels. These studies support the need for repeatable procedures and documented escalation paths. Yet their primary concern remains operational effectiveness rather than client-verifiable governance. They clarify how SOCs work, but they do not fully define how SOC's work should be externalized as a transparent managed service.

Sector-specific work confirms that SOC architectures must be adapted to context. Murisa and Coetzee [29] derive a conceptual SOC framework for air traffic management systems, and Alaliwat et al. [3] propose an Operational Technology (OT)/Industrial Control System (ICS) threat-hunting platform aligned with MSSP services. Such studies are relevant because they show that security operations cannot be reduced to a universal tool stack. At the same time, their sectoral focus limits their contribution to the general problem of a multi-tenant, portable, and auditable MSSP architecture. The SOC literature therefore provides essential design inputs, but not a complete sovereignty-oriented managed-service model.

3.2.3 SIEM, SOAR, CTI, and the Operational Service Stack

The technical literature on SIEM, SOAR, CTI, and threat hunting is the closest to the implementation concerns of this dissertation. It converges on the idea that modern security operations require a pipeline that collects telemetry, normalizes and correlates events, enriches alerts with intelligence, orchestrates response actions, and preserves investigation records [45, 23, 26, 9, 6]. This body of work is strong at the component and workflow level, but weaker at the managed-service governance level.

SIEM research establishes the monitoring and correlation layer as the analytical backbone of security operations. Sheeraz et al. [45] propose a modular SIEM architecture that separates collection, normalization, correlation, storage, and reporting. The SIEM-SC review and proposal [23] identify a shift from basic monitoring toward compliance, service delivery, containers, and stronger audit expectations. Mohamad Hata et al. [26] emphasize log aggregation design criteria, and Mugeshwaran et al. [27] explore Artificial Intelligence

(AI)-enhanced event processing for contextual alerting and automated response.

These contributions support the dissertation's architectural decomposition of monitoring functions. Their limitation is that transparency is usually bounded by the platform: they do not specify how multiple tenants can inspect detection logic, validate normalization rules, or export correlation knowledge independently of the provider.

SOAR research addresses the next step in the service pipeline: transforming alerts into consistent operational actions. Bridges et al. [9] provide an empirical evaluation of commercial SOAR tools and show that configuration quality, efficiency gains, decision support, and ticket completeness are tightly connected. Islam et al. [21] propose a declarative Application Programming Interface (API)-driven orchestration approach, while Sworna et al. [48] propose automated API recommendation to reduce the integration burden faced by SOC teams.

These studies demonstrate that orchestration is not a secondary convenience layer; it is where operational knowledge is codified. For this dissertation, the key implication is that playbooks, connectors, response decisions, and case transitions must be treated as portability-critical artifacts. Existing SOAR research improves automation and interoperability, but it does not fully address the governance of orchestration artifacts in a provider–client relationship.

CTI and threat-hunting research reinforce the move from reactive monitoring to proactive and context-aware defense. Ammi and Jama [6] demonstrate a Malware Information Sharing Platform and Threat Sharing (MISP)-centered architecture for feeding threat intelligence into SOC tools. Nursidiq and Lim [33] propose a modified threat-hunting framework for detecting unknown threats, and Alaliwat et al. [3] align threat-hunting automation with OT/ICS and MSSP needs. More recent work explores Large Language Models (LLMs) and agentic assistance for intelligence contextualization, query generation, and analyst support [24, 41, 35]. These works indicate that open and composable intelligence-processing workflows can strengthen managed security services. Nevertheless, their evaluations focus mainly on detection, contextualization, explainability, or analyst productivity. Multi-tenancy, client-side inspectability, and migration of intelligence-processing logic remain marginal concerns.

3.2.4 Federated and Service-Oriented Security Architectures

A smaller but highly relevant body of work addresses federation, policy-driven service delivery, and distributed cybersecurity operations. PALANTIR proposes an Network Function Virtualization (NFV)-based Security-as-a-Service architecture for automated threat mitigation across cloud, edge, and on-premises environments [14]. ATHENA proposes a cross-border federated cybersecurity platform that combines threat intelligence sharing, distributed case and ticket management, synchronized playbooks, and European Union (EU) regulatory orientation [38]. Escaleira et al. [16] show how moving-target defense can be delivered as a service in cloud/edge telco environments through modular and reconfigurable protection functions.

These studies move closer to the dissertation's problem than most component-level work because they explicitly reject monolithic, single-domain security architectures. They show that distributed deployment, interoperability, policy-driven remediation, and cross-organizational coordination are technically feasible. They also align with European concerns about data governance, resilience, and strategic autonomy [14, 38, 22].

Even so, they do not fully solve the MSSP design problem. PALANTIR is strongest on automated remediation and deployment topology, but is less explicit about tenant-facing auditability and service exit. ATHENA focuses on federation among cybersecurity actors and national SOCs, whereas the MSSP context involves a structurally asymmetric provider–client relationship. Moving-target defense as a service improves technical resilience, but it does not define a full governance model for managed security delivery.

The conclusion is that federation and service orientation are necessary but insufficient. A sovereign MSSP framework requires not only distributed capabilities but also explicit identity boundaries, tenant-level evidence, portable operational artifacts, documented control flows, and governance mechanisms that make the service intelligible to the client. Existing federated architectures provide important design principles, but they stop short of a complete open and auditable MSSP operating model.

3.2.5 Digital Sovereignty, Lock-In, and Openness

Digital sovereignty research provides the normative basis for interpreting MSSP dependency as more than a procurement inconvenience. Floridi [17] frames digital sovereignty as a strategic societal issue, and Pohle and Thiel [39] show that the concept has become central but contested in digital policy discourse. Roberts [43] is particularly important for this dissertation because he distinguishes mere control from legitimate control, emphasizing transparency and accountability. Kianpour et al. [22] translate sovereignty into policy-facing dimensions such as governance authority, infrastructure and service control, data protection, openness balanced with autonomy, and resilience.

This literature changes the analytical lens of the review. It makes dependency a question of practical control, verifiability, and accountability, rather than only a question of procurement preference.

These works justify treating technological sovereignty as an architectural concern. In an MSSP relationship, the client may still own its business systems while losing practical control over the mechanisms that detect, interpret, and respond to security events. The dependency is not only infrastructural. It is also epistemic and procedural: the provider may hold the normalized telemetry model, detection assumptions, correlation rules, response playbooks, investigation history, and deployment knowledge required to reconstruct the service. A sovereignty-preserving MSSP must therefore preserve the client's capacity to understand and verify the service, not merely its contractual right to receive reports.

The vendor lock-in literature operationalizes dependency more concretely, but mostly outside the managed-security domain. The cloud vendor lock-in prediction framework proposed by Alhosban et al. [5] identifies measurable factors such as service costs, data transfer expenses, security features, compliance, scalability, and technical integration. This is useful, but cloud-generic dependency does not capture the full specificity of MSSP lock-in. In managed security, lock-in is produced by operational artifacts: telemetry schemas, parsers, detection rules, correlation logic, threat-intelligence mappings, response playbooks, case records, compliance evidence, and deployment definitions. Research on declarative orchestration and API recommendation [21, 48] points toward mitigation mechanisms, but does not yet assemble them into a full portability model.

Open-source and open-architecture approaches can reduce dependency, but the literature does not support the simplistic conclusion that open-source tools automatically produce sovereignty. Open components can still be integrated in opaque ways, governed by undocumented processes, or configured so that only the provider understands the resulting service. The relevant architectural property is therefore not source availability alone, but inspectability, standards-based interfaces, non-proprietary formats, documented control flows, and exportable operational knowledge [45, 6, 14, 38]. This distinction is central to the framework developed in Chapter 4.

3.2.6 Governance, Compliance, and Assurance Evidence

Governance and compliance literature addresses the conditions under which managed security can be trusted, verified, and aligned with external obligations. Alasmari et al. [4] are especially relevant because they examine outsourced network testing as a means of verifying whether security policies are honored, while also showing that verification itself can expose sensitive configuration information.

This captures a central dilemma of MSSP governance: clients require evidence, but evidence generation must not create new leakage or dependency risks. Vedadi et al. [52] reach a complementary conclusion from an organizational perspective by showing that mature internal auditing improves satisfaction with outsourced security services.

Compliance-oriented technical work addresses parts of this problem. The SIEM-SC proposal links SIEM evolution to auditability and GDPR-compatible security compliance [23]. PALANTIR incorporates policy-driven remediation and EU compliance considerations into its architecture [14]. Kianpour et al. [22] show that NIS2 reflects several digital sovereignty principles, particularly governance authority, while also identifying weaker treatment of the balance between openness and sovereignty. These studies establish that compliance is not external to managed security design. It affects data handling, incident documentation, control accountability, and the client's ability to demonstrate due diligence when operations are delegated.

Risk-oriented frameworks extend the discussion beyond compliance checklists. Mukhopadhyay and Jain [28] connect ransomware risk management, quantification, mitigation, resilience, and insurance. Calvo et al. [10] model risk exposure through user and entity be-

havior, while Cyber-DRIVE integrates dynamic risk analysis, situational awareness, CTI, and incident response [47]. These contributions are relevant because they show that risk evidence should be dynamic and operationally connected to detection and response workflows. The remaining limitation is that governance, compliance, and risk are frequently attached to specific tools or models rather than built into a complete service architecture. The literature does not yet provide a coherent evidence model spanning identity, telemetry, detection logic, response execution, change management, reporting, and service exit.

3.2.7 Operational Sustainability and Responsible Automation

The final relevant stream concerns the sustainability of security operations. MSSP architectures must scale not only technically, but also organizationally and cognitively. Analyst overload, alert fatigue, weak measurement, and automation bias are recurring problems in SOC environments [2, 12, 18, 40, 50]. These pressures are amplified in MSSPs, where analysts must handle multiple tenant contexts while preserving accuracy, traceability, and service-level commitments.

Automation and ML research offers partial relief. Preuveneers et al. [40] investigate AutoML for reducing false alerts, Vaarandi and Guerra-Manzanares [51] show that stream clustering and supervised learning can reduce Network Intrusion Detection System (NIDS) alert burden on commodity hardware, and Borah et al. [8] survey ML techniques for SOC optimization. Zidan et al. [55] propose evaluation parameters for automation and ML models, while Saadallah et al. [44] argue that effective SOCs require a dynamic balance between AI, automation, and human expertise. Tilbury and Flowerday [50] warn that automation can create bias if it is not task-based, process-aware, evaluated, and supported by analyst training. For the dissertation, the implication is that automation must remain auditable and governable; otherwise, it can reproduce opacity in a new form.

Sustainability also has a technical and ecosystem dimension. The Cyber Resilience Act increases expectations around maintenance, vulnerability handling, and lifecycle responsibility for products with digital elements [42]. In an open MSSP architecture, this makes component selection, update governance, documentation, reproducible deployment, and knowledge retention part of the security model itself. Long-term sovereignty cannot depend on a stack that is understandable only to its initial implementer or maintain-

able only through tacit knowledge. The literature recognizes pieces of this problem, but it rarely connects analyst workload, automation governance, software ecosystem health, compliance duties, and service reproducibility into one design perspective.

3.3 Comparative Synthesis of Limitations and Research Gaps

The thematic analysis shows that the literature is not deficient in volume or technical sophistication. Its main weakness is integration. The reviewed studies offer mature insights into outsourcing economics, SOC operation, monitoring and orchestration platforms, federation, compliance, risk, and digital sovereignty. However, these insights are usually developed in separate research streams and at different units of analysis. The MSSP design problem addressed in this dissertation requires them to be treated together, because vendor lock-in, auditability, portability, and sovereignty emerge from the interaction between service contracts, technical architectures, operational workflows, and governance evidence.

Table 3.3: Comparative synthesis of the main literature streams and unresolved limitations.

Literature stream	Established contribution	Unresolved design limitation
Outsourcing and MSSP service models	Explains adoption drivers, partial outsourcing, shared-service models, pricing sensitivity, and the need for retained client capability	Treats the provider mainly as a service actor; gives limited architectural detail on tenant evidence, audit rights, portability, and exit from the managed service
SOC capability and operations	Defines maturity, analyst performance, situational awareness, incident response, use-case coverage, and deployment practice	Remains largely organization-centric; does not fully translate internal SOC measurement into client-facing accountability within outsourced operations
SIEM, SOAR, CTI, and threat hunting	Demonstrates modular monitoring, orchestration, intelligence enrichment, and analyst-assistance workflows	Optimizes components and workflows, but rarely treats detection logic, playbooks, case records, and integrations as tenant-portable managed-service artifacts
Federated and service-oriented architectures	Shows feasibility of distributed, policy-driven, cross-organizational security operations	Provides federation and interoperability principles, but not a complete provider–client governance model for a multi-tenant, auditable MSSP
Governance, compliance, sovereignty, and sustainability	Clarifies the importance of assurance evidence, regulatory alignment, legitimate control, lock-in mitigation, and long-term maintainability	Remains fragmented across policy, cloud dependency, risk management, and human factors; does not operationalize sovereignty as an end-to-end architectural property of managed security services

From this comparative synthesis, five research gaps emerge.

1. **Absence of an integrated sovereignty-oriented MSSP framework.** Existing work contributes important partial models for outsourcing, SOC capability, SIEM/SOAR integration, federation, and compliance, but no reviewed study integrates these strands into a coherent framework explicitly designed for open, auditable, and sovereignty-preserving managed security delivery [53, 45, 14, 38, 43].

2. **Auditability is under-specified as an architectural property.** The literature addresses fragments of auditability, including SLA verification, log integrity, compliance reporting, incident documentation, and performance metrics. It does not yet define end-to-end auditability across identity events, data flows, detection logic, orchestration execution, case management, configuration changes, and client oversight within the managed service [4, 23, 52, 18].

3. **Vendor lock-in is insufficiently operationalized at the MSSP layer.** Lock-in is commonly analyzed through cloud dependency, service cost, or integration burden. In managed security, dependency also accumulates in operational artifacts such as telemetry schemas, normalization pipelines, detection rules, playbooks, investigation records, governance evidence, and deployment knowledge. These artifacts are rarely treated as explicit portability targets [5, 21, 48, 9].

4. **Open-source and federated approaches remain technically promising but organizationally incomplete.** The literature demonstrates that open components, federated platforms, and distributed security services are feasible. It does not yet provide a consolidated operating model for combining them into a sustainable, multi-tenant MSSP with clear provider–client trust boundaries, tenant-scoped evidence, verifiable control, and structured exit support [6, 38, 14, 54].

5. **Governance and sustainability remain fragmented across technical, regulatory, economic, and human dimensions.** Analyst workload, automation bias, risk management, compliance, software maintenance, and open-source ecosystem sustainability are discussed in separate bodies of work. They are rarely combined

into a single design view capable of supporting long-term, auditable MSSP operations [40, 50, 44, 28, 42].

The gap list identifies what is missing in the literature. For design purposes, however, each gap must also be translated into an architectural implication. Table 3.4 makes this bridge explicit by connecting the research gaps to the design consequences that motivate the requirements formalized in Chapter 4.

Table 3.4: Translation of research gaps into architectural implications and derived requirements.

Gap	Gap identified	Architectural implication	Derived requirement(s)
G1	No integrated sovereignty-oriented MSSP framework	The architecture must combine identity, security operations, governance, portability, and sustainability as one managed-service design, rather than as separate technical or organizational concerns	DR1–DR8
G2	Auditability is underspecified as an architectural property	Evidence generation must span identity events, telemetry flows, detection logic, orchestration execution, case management, configuration changes, and client-facing oversight	DR2, DR4, DR6
G3	Vendor lock-in is not operationalized at the managed-service layer	Operational artifacts must be treated as portability targets, and components must remain replaceable through open interfaces and non-proprietary formats	DR3, DR5, DR8
G4	Open-source and federated approaches remain organizationally incomplete	The service model must define provider–client trust boundaries, tenant-scoped visibility, federable identity control, and structured exit or migration support	DR1, DR2, DR6, DR7
G5	Governance and sustainability remain fragmented	Compliance, change management, reproducible deployment, knowledge retention, and responsible automation must be embedded as architectural functions rather than external management activities	DR4, DR6, DR8

Taken together, these implications position this dissertation at the intersection of the five gaps identified above. It treats technological sovereignty not as a rhetorical preference for local control, nor as a synonym for self-hosting, but as an architectural property of managed security services. In this view, a client organization should be able to understand, verify, govern, and, where necessary, reconstruct or replace the mechanisms through which its security is delivered [43, 22]. This requires combining research streams that are usually separated: the economic rationale for outsourcing, the socio-technical realities of SOC operation, the modular composition of SIEM, SOAR, and CTI capabilities, the governance evidence required for compliance and assurance, and the sovereignty concerns associated with dependency and lock-in [53, 49, 45, 4, 5].

The dissertation therefore does not seek to introduce another isolated SIEM, SOAR,

risk model, or outsourcing decision method. Its contribution is to translate the unresolved design limitations of the literature into an integrated reference framework for open and auditable MSSP delivery. The framework developed in Chapter 4 responds directly to the gaps identified here by deriving formal design requirements for federable identity governance, tenant visibility, modular composition, end-to-end auditability, artifact portability, compliance by design, layered security operations, and reproducible deployment.

This positioning also defines the scope of the dissertation's claims. The work does not argue that open-source components alone eliminate dependency, nor that a prototype can fully validate production-scale managed security delivery. It argues that vendor lock-in and loss of control can be structurally reduced when the MSSP service is designed around open interfaces, inspectable configurations, portable operational artifacts, auditable workflows, explicit trust boundaries, and reproducible infrastructure. The next chapter translates this argument into the proposed five-layer framework, and the subsequent prototype examines whether the framework can be instantiated with currently available technologies.

Chapter 4

Design of the Sovereign, Open, and Auditable MSSP Framework

This chapter presents the design of the proposed framework for sovereign, open, and auditable managed security services. Building upon the problem identification and knowledge synthesis established in Chapters 1–3, the framework constitutes the central artifact of the DSR methodology adopted in this dissertation [20, 37]. It translates the research gaps identified in Chapter 3 into a structured architectural response.

The chapter first defines the design requirements, then presents the architectural principles and five-layer model. It then describes each layer, the inter-layer interaction model, the trust assumptions, and the chapter-level synthesis.

It is important to note that this chapter is exclusively concerned with conceptual architecture and design. The selection of specific technologies, the implementation decisions, and the prototype instantiation are addressed separately in Chapter 5. This separation reflects the DSR distinction between the design artifact and its demonstration [37], and ensures that the architectural contributions of the framework can be evaluated independently of any particular technological realization.

4.1 Design Requirements

The design requirements are derived from three convergent sources: the research gaps identified in Chapter 3, the sovereignty dimensions articulated in Section 2.2.1,

and the analytical criteria established in Section 2.3. They translate those concerns into implementation-independent properties that can later be used as evaluation criteria in Chapter 6.

The following eight design requirements define the scope and ambition of the proposed framework:

DR1: Centralized and federable identity governance. The framework must establish a unified identity and access governance model that supports centralized authentication, role-based and attribute-based access control, privileged access management, credential lifecycle management, and federation across organizational boundaries [22, 14, 38].

DR2: Multi-tenant isolation with per-tenant visibility. The framework must support multiple client organizations with strict data and configuration segregation, while providing each tenant with independent visibility into its own security posture, detection logic, and operational artifacts [4, 52].

DR3: Modular and standards-based component architecture. All functional components must be independently replaceable and composable through open interfaces and standard data formats, preventing structural coupling to any single component or vendor [5, 21, 48].

DR4: End-to-end auditability. The framework must ensure that data flows, detection rules, orchestration actions, configuration changes, and service decisions are traceable, inspectable, and verifiable by both the provider and the client [4, 23, 52].

DR5: Portability of operational artifacts. Detection rules, correlation logic, response playbooks, incident records, configurations, and workflows must be stored in portable, non-proprietary formats that support export, migration, and independent reconstruction [5, 9].

DR6: Regulatory compliance by design. The architecture must embed compliance-relevant controls—data residency, access logging, incident documentation, and evidence generation—as native capabilities rather than as post-hoc overlays [22, 14, 42].

DR7: Layered and composable security operations. The framework must integrate telemetry collection, normalization, correlation, threat intelligence, automated response, and investigative capabilities into a coherent detection-and-response pipeline composed of independently governed modules [45, 9, 6].

DR8: Operational sustainability and reproducibility. The framework must support automated, reproducible deployment, declarative configuration management, knowledge retention, and operational continuity without dependency on tacit or non-transferable expertise [46, 2, 40, 42].

Table 4.1 maps each design requirement to the research gaps from which it is derived and to the sovereignty dimensions it addresses. This mapping ensures that the framework design is systematically grounded in the literature findings and that no identified gap is left without a corresponding architectural response. It should be noted that Gap 1—the absence of an integrated sovereignty-oriented MSSP framework—is addressed at the artifact level by the framework as a whole; each design requirement contributes to the integrated response, and the individual gap mappings in the table reflect the most direct derivation relationships.

Table 4.1: Traceability of design requirements to research gaps and sovereignty dimensions.

Req.	Description	Gaps addressed	Sovereignty dimensions
DR1	Centralized and federable identity governance	Gap 1, Gap 4	Control; Transparency
DR2	Multi-tenant isolation with per-tenant visibility	Gap 1, Gap 2, Gap 4	Control; Transparency
DR3	Modular, standards-based architecture	Gap 1, Gap 3	Portability; Independence
DR4	End-to-end auditability	Gap 1, Gap 2, Gap 5	Transparency; Control
DR5	Portability of operational artifacts	Gap 1, Gap 3	Portability; Independence
DR6	Regulatory compliance by design	Gap 1, Gap 4, Gap 5	Control; Transparency
DR7	Layered, composable security operations	Gap 1, Gap 4	Control; Independence
DR8	Operational sustainability and reproducibility	Gap 1, Gap 3, Gap 5	Independence; Portability

Table 4.2 further maps each design requirement to the four sovereignty assessment dimensions defined in Section 2.2.2, showing how the requirements collectively cover all

dimensions of the sovereignty construct.

Table 4.2: Mapping of design requirements to sovereignty assessment dimensions.

Req.	Control	Transparency	Portability	Independence
DR1	✓	✓		
DR2	✓	✓		
DR3			✓	✓
DR4	✓	✓		
DR5			✓	✓
DR6	✓	✓		
DR7	✓			✓
DR8			✓	✓

Table 4.3 defines concise evaluation anchors for each requirement, ensuring that the requirements can be assessed through concrete architectural mechanisms and evidence rather than as abstract design intentions.

Table 4.3: Evaluation anchors for the design requirements.

Req.	Architectural mechanism	Evidence expected	Evaluation method
DR1	Federated identity, access control, privileged access, credential governance	Role mappings, access logs, session records, credential lifecycle records	Architecture inspection and audit-log review
DR2	Tenant-scoped identity, network, data, and configuration boundaries	Tenant policies, segregated views, scoped artifact access	Boundary inspection and access-scope testing
DR3	Replaceable modules connected through open interfaces and formats	Interface descriptions, format specifications, component dependency map	Architecture inspection and component mapping
DR4	Traceable data flows, rules, playbooks, decisions, and changes	Audit records, rule history, workflow logs, change records	Evidence review across layers
DR5	Exportable operational artifacts and configurations	Rule exports, playbook exports, case records, configuration and deployment files	Export and reconstruction review
DR6	Compliance controls embedded in service operation	Access reports, incident records, data-handling records, compliance artifacts	Evidence mapping against regulatory obligations
DR7	Composable telemetry, detection, intelligence, response, and investigation pipeline	Telemetry flow, detection outputs, orchestration records, incident cases	End-to-end workflow inspection
DR8	Reproducible deployment, declarative configuration, documentation, migration support	Versioned deployment files, configuration history, operational documentation, migration package	Redeployment and documentation review

4.2 Guiding Architectural Principles

Before presenting the layered architecture, it is necessary to articulate the principles that govern its structural organization. These principles are not requirements in themselves; rather, they are design heuristics that constrain and orient the architectural

decisions made across all layers, ensuring internal coherence and alignment with the framework's overarching goals. Five principles guide the architecture.

Modularity and separation of concerns. The framework adopts strict modularity as a foundational principle. Each functional capability—whether identity management, telemetry collection, event correlation, or incident response—is encapsulated as a distinct module with well-defined interfaces. This reduces the blast radius of component failure, enables independent evolution, and allows modules to be replaced without disrupting the broader system [45, 21].

Defense in depth. The architecture is organized around the principle of layered defense, where no single layer is solely responsible for the security of the overall system. Each layer contributes complementary controls, and the compromise or degradation of any individual layer does not result in the complete loss of defensive capability. This principle is well established in cybersecurity engineering, but it acquires additional significance in the MSSP context because the provider must guarantee a consistent security posture across tenants with varying risk profiles and exposure surfaces [12, 46].

Sovereignty by design. Sovereignty is treated as a design constraint rather than as an emergent property or a post hoc aspiration. This means that every architectural decision—from the choice of data formats to the definition of trust boundaries—is evaluated against its impact on the client's ability to inspect, verify, govern, and, if necessary, replace or reconstruct the service components on which its cyber defense depends [43, 22]. Sovereignty by design implies that the framework does not merely permit client oversight; it structurally enables and facilitates it.

Openness as an architectural constraint. Openness is operationalized not as a preference for a particular licensing model, but as a set of technical and operational characteristics imposed as architectural constraints on interfaces, data formats, and integration patterns. All inter-component and inter-layer communication must rely on open, documented, and standards-based protocols. Detection logic, orchestration workflows, and configuration state must be expressed in non-proprietary, human-

readable formats. These characteristics—standards-based interfaces, non-proprietary data formats, documented protocols, and inspectable configurations—define openness as an architectural property. Consequently, the framework should be understood as an open-architecture framework rather than an open-source-only framework. It does not preclude the use of proprietary components, provided those components satisfy the required openness properties and do not introduce structural vendor lock-in or loss of client control. This constraint ensures that the framework remains inspectable, portable, and extensible regardless of which specific components are used to instantiate it [14, 38, 5].

Auditability as a first-class property. Auditability is elevated from a reporting function to a first-class architectural property that permeates every layer. This means that every layer must be designed to produce, preserve, and expose the evidence necessary for independent verification of its operations. Audit trails are not optional appendages; they are integral outputs of every significant system action, from identity authentication events through detection rule modifications to incident response decisions. This principle responds directly to the finding that existing MSSP architectures under-specify auditability, treating it as a feature of individual tools rather than as a systemic property of the service [4, 18, 23].

4.3 Layered Architecture Organization

The proposed framework is organized into five architectural layers, each encapsulating a coherent set of responsibilities and operating within defined trust boundaries. The layers are designed to interact through explicit, standards-based interfaces, and three cross-cutting properties—sovereignty, openness, and auditability—span the entire architecture as systemic guarantees rather than layer-specific features. Sustainability is operationalized through DR8 and Layer V, rather than as a fourth cross-cutting property. Figure 4.1 illustrates the high-level organization.

The bottom-to-top ordering reflects a trust dependency gradient: upper layers depend on the foundational trust substrate established by lower layers. Layer I establishes identity, isolation, and access governance upon which all upper layers depend. Layer II

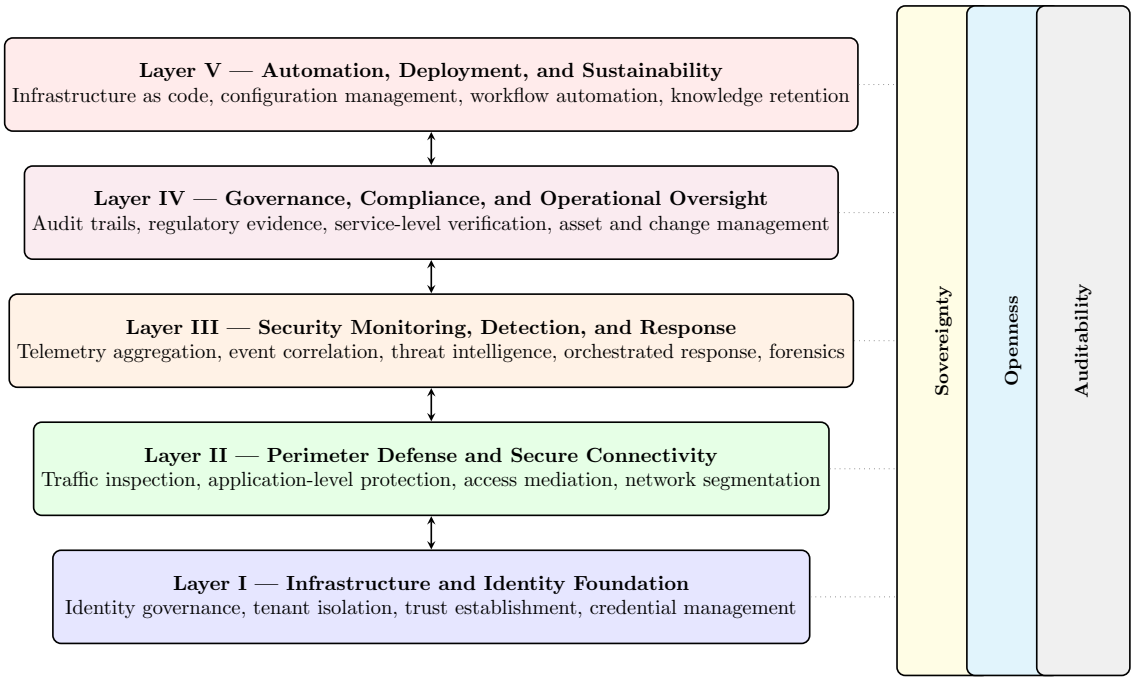

Figure 4.1: High-level layered architecture of the proposed sovereign, open, and auditable MSSP framework, illustrating the five functional layers and three cross-cutting architectural properties.

mediates the boundary between internal service infrastructure and external networks, controlling ingress and egress and enforcing application-level protection. Layer III constitutes the operational core of managed security delivery, integrating the detection, intelligence, and response capabilities that justify the MSSP relationship. Layer IV overlays governance, compliance, and operational oversight functions that ensure the service remains accountable, verifiable, and aligned with regulatory obligations. Layer V functions as a lifecycle and control-plane layer: it addresses the sustainability and reproducibility of the entire stack, providing the automation, knowledge management, and deployment discipline necessary for long-term viability. While the trust dependency gradient flows upward, operational interactions are inherently bidirectional: upper layers exercise control, policy enforcement, and orchestration over lower layers, as detailed in the inter-layer interaction model (Section 4.5).

The five-layer structure reflects three criteria: separation of concerns, alignment with MSSP operations, and support for sovereignty, openness, and auditability. This decomposition was selected after considering simpler and more granular alternatives. Simpler models would collapse identity, perimeter, detection, governance, and automation con-

cerns into broader layers, reducing the visibility needed for independent auditability and tenant-scoped control. More granular models, such as separating monitoring from response, would add complexity without materially improving the framework's ability to mitigate lock-in or preserve sovereignty. The resulting structure therefore balances auditability, replaceability, and practical implementation clarity.

The three cross-cutting properties—sovereignty, openness, and auditability—are not confined to any single layer. Instead, they impose constraints and generate requirements at every level of the architecture. Sovereignty manifests as client-inspectable configurations in Layer I, as transparent traffic policies in Layer II, as exportable detection logic in Layer III, as verifiable compliance evidence in Layer IV, and as reproducible deployment artifacts in Layer V. Openness manifests as standards-based identity protocols, open data formats, documented interfaces, and non-proprietary workflow representations throughout the stack. Auditability manifests as immutable logs, traceable configuration changes, inspectable decision chains, and tenant-accessible evidence at every layer.

The following section describes each layer in detail, specifying its definition, responsibilities, contributions to the three cross-cutting properties, and integration with adjacent layers.

4.4 Architectural Layers

Each of the five layers is characterized along four common dimensions: its definition and scope, its core responsibilities, its contributions to sovereignty, openness, and auditability, and its integration with adjacent layers. This uniform treatment allows the layers to be compared directly and supports the traceability analysis presented in Chapter 5.

Layer I — Infrastructure and Identity Foundation.

- **Definition and scope.** Layer I constitutes the foundational substrate of the framework. Its primary responsibility is to establish and enforce the trust model upon which all upper layers depend. This encompasses four interrelated concerns: centralized identity and directory services, authentication federation and single sign-on, privileged access management, and credential lifecycle gov-

ernance. Additionally, Layer I defines the tenant isolation boundaries that ensure multi-tenant operation does not compromise the confidentiality, integrity, or independent verifiability of any individual client's data and configurations.

In the context of a managed security service, identity is not merely an administrative convenience; it is the mechanism through which access decisions, audit attributions, and trust assertions are grounded [14, 38]. The framework therefore treats identity governance as an architectural foundation rather than as a supporting utility. Every action taken within the managed service—from an analyst querying a correlation engine to a playbook executing an automated containment step—must be attributable to a verified identity operating within a defined authorization scope.

- **Core responsibilities.** The first responsibility of Layer I is to maintain an authoritative directory service that serves as the single source of truth for user identities, group memberships, organizational roles, and service accounts across the framework. This directory must support hierarchical and attribute-based access structures sufficiently flexible to model the complex organizational relationships inherent in multi-tenant MSSP environments, where provider staff, client administrators, and automated processes coexist within a shared infrastructure.

  The second responsibility is authentication federation. The framework requires a centralized authentication broker that mediates access to all components through standard, open identity protocols. Single sign-on reduces credential proliferation—a significant operational and security risk in complex service environments—while federation capability allows the identity layer to integrate with client-side identity providers when hybrid governance models demand it [54, 25]. This federation capability is essential for sovereignty because it allows client organizations to retain authoritative control over their own identity assertions rather than ceding identity governance entirely to the provider.

  The third responsibility is privileged access management. Administrative access to security-critical systems—log repositories, detection engines, orchestration

platforms, and network control planes—must be mediated through a dedicated privileged access gateway that enforces session recording, just-in-time access provisioning, and granular authorization policies. This control is indispensable for auditability because it ensures that every privileged action is attributed, recorded, and independently reviewable [4, 52].

The fourth responsibility is credential lifecycle management. Passwords, secrets, cryptographic keys, and service tokens must be governed through a centralized credential management function that enforces rotation policies, prevents credential sprawl, and provides auditable records of credential issuance and usage. In multi-tenant environments, credential compromise in one domain must not propagate to other tenants, and the credential management function must therefore enforce strict tenant-scoped isolation.

- **Contributions to sovereignty, openness, and auditability.** Layer I supports sovereignty, openness, and auditability by grounding access control in open and federable identity standards rather than proprietary identity silos. This allows clients to understand and audit authentication flows, retain control over identity assertions, and decouple identities during migration. Identity-related events—including authentication outcomes, authorization decisions, privilege escalations, credential changes, and privileged session records—provide the evidential base for access-control accountability [22, 5, 23, 14].

- **Integration with adjacent layers.** Layer I provides the identity and authorization context consumed by every upper layer. Layer II relies on identity assertions from Layer I to enforce access policies at the network perimeter and to authenticate users connecting through secure remote access channels. Layer III consumes identity data for alert attribution, analyst authentication, and tenant-scoped access to detection and response interfaces. Layer IV draws upon the identity audit trails produced by Layer I as primary compliance evidence. Layer V uses service accounts and machine identities governed by Layer I to authenticate automation workflows and deployment pipelines. The trust boundary between Layer I and all upper layers is defined by the identity proto-

col: any entity that cannot present a valid, Layer I-issued credential is denied access to the managed service environment.

Layer II — Perimeter Defense and Secure Connectivity.

- **Definition and scope.** Layer II governs the boundary between the managed service infrastructure and external networks. Its scope encompasses four principal functions: reverse proxying and traffic routing for service publication, application-level inspection and web application firewall capabilities, network segmentation and micro-segmentation within the service infrastructure, and secure remote access mediation. Together, these functions define the attack surface of the framework and determine how external traffic—whether from client endpoints, monitored assets, or administrative sessions—enters and traverses the managed service environment.

  In the MSSP context, perimeter defense is not merely a matter of filtering malicious traffic. It is also a governance function, because the perimeter determines which services are exposed to which tenants, how tenant traffic is segregated, and what visibility the client retains over the access policies applied to its own data flows [14, 46].

- **Core responsibilities.** The primary responsibility of Layer II is to serve as the sole ingress and egress point for all external traffic reaching the managed service platform. A reverse proxy function terminates external connections, enforces transport-layer encryption, and routes requests to the appropriate internal service based on declarative routing policies. By centralizing traffic routing, the framework eliminates the need for individual components to manage their own external exposure, thereby reducing configuration complexity and the likelihood of misconfiguration-induced vulnerabilities.

  The second responsibility is application-level inspection. Layer II must include web application firewall capabilities that analyze incoming requests for injection attacks, cross-site scripting, protocol violations, and other application-layer threats before traffic reaches internal services. These inspection policies must be declarative, versioned, and auditable, such that changes to protection logic

are traceable and reproducible [12, 46].

The third responsibility is network segmentation. Within the service infrastructure, different functional domains—identity services, security operations, business support, privileged access management—must operate on logically separated network segments with explicitly defined inter-segment communication policies. This internal segmentation limits lateral movement in the event of compromise and enforces the principle of least connectivity, ensuring that components communicate only through sanctioned paths.

The fourth responsibility is secure remote access mediation. Administrative and operational access to the managed service infrastructure must traverse a controlled remote access gateway that enforces session authentication against Layer I, applies session policies based on role and context, and provides clientless browser-based access where possible to reduce dependency on endpoint-installed software. This function intersects directly with the privileged access management capabilities of Layer I, extending identity-based access control to the network transport layer [4].

- **Contributions to sovereignty, openness, and auditability.** Layer II supports sovereignty, openness, and auditability by expressing perimeter rules, routing configurations, inspection policies, and segmentation controls in documented, versioned formats that can be reviewed at tenant level [43, 22]. Standard network protocols and modular perimeter functions reduce coupling, while ingress and egress logs—connection metadata, inspection decisions, policy matches, and access denials—feed Layer III correlation and Layer IV compliance evidence.

- **Integration with adjacent layers.** Layer II depends downward on Layer I for identity verification. Users and services connecting through the perimeter must present credentials validated by the identity foundation before being granted access to internal resources. Upward, Layer II feeds telemetry to Layer III: connection logs, inspection findings, and access events constitute primary inputs for the security monitoring and correlation pipeline. Layer II also contributes di-

rectly to Layer IV by generating the perimeter-level audit evidence required for regulatory compliance—demonstrating, for example, that transport encryption is enforced, that application-layer protections are active, and that no unauthorized access paths exist to tenant-sensitive services.

Layer III — Security Monitoring, Detection, and Response.

- **Definition and scope.** Layer III constitutes the operational core of the managed security service. It is responsible for the continuous ingestion, normalization, correlation, and analysis of security telemetry from all monitored sources; the integration of external and internal threat intelligence; the orchestration of automated and analyst-guided response workflows; and the support of incident investigation and digital forensics. In terms of the MSSP value proposition, this is the layer that most directly transforms raw data into actionable security outcomes, and it is therefore the layer where the tension between operational efficiency and sovereignty-preserving transparency is most acute [45, 9, 6].

  The scope of Layer III encompasses the functional domains traditionally associated with SIEM, SOAR, CTI, and incident response platforms, but the framework treats these as composable functional capabilities rather than as monolithic platforms. This decomposition is essential for satisfying DR3 (modular architecture) and DR5 (portability of operational artifacts), because it prevents the tight coupling of detection, response, and intelligence functions that is a primary source of vendor lock-in in conventional MSSP architectures [21, 48, 5].

- **Core responsibilities.** The first responsibility of Layer III is telemetry acquisition and normalization. Security-relevant data—system logs, network flow records, endpoint telemetry, application events, and authentication records—must be collected from heterogeneous sources, parsed into a common schema, enriched with contextual metadata, and stored in a manner that preserves provenance and supports both real-time analysis and retrospective investigation. The normalization schema must be open and documented, ensuring that the client can interpret, export, and independently process its own telemetry without

dependency on provider-proprietary data formats [45, 26, 23].

The second responsibility is event correlation and threat detection. Normalized telemetry must be processed through correlation engines that apply rule-based, statistical, and behavioral detection methods to identify security-relevant patterns. Detection rules themselves are first-class operational artifacts: they must be versioned, peer-reviewable, expressible in standard rule languages, and exportable in portable formats. The framework explicitly rejects the model in which detection logic is an opaque, provider-controlled asset that the client cannot inspect or take with them upon service termination. This position follows from DR4 (end-to-end auditability) and DR5 (portability), and it distinguishes the proposed framework from conventional MSSP architectures where correlation logic is frequently a proprietary differentiator [4, 52].

The third responsibility is threat intelligence integration. The framework requires a dedicated function for the acquisition, normalization, enrichment, and distribution of threat intelligence, including indicators of compromise, adversary tactics and techniques, and contextual threat reporting. This function must support bidirectional intelligence sharing through open intelligence-sharing standards, enabling the MSSP to consume external intelligence feeds, contribute to community intelligence ecosystems, and provide tenant-specific intelligence products derived from the analysis of their own operational data [6, 33, 24].

The fourth responsibility is security orchestration and automated response. Response workflows—playbooks, runbooks, and automated remediation sequences—must be defined declaratively and executed through an orchestration engine that coordinates actions across multiple components. Playbooks must be expressed in open, portable formats, and their execution must produce complete audit trails documenting every decision point, action taken, and outcome observed [9, 21].

The fifth responsibility is incident investigation and digital forensics. When security events escalate to confirmed incidents, the framework must provide structured investigation environments—case management, evidence collection, timeline reconstruction, and forensic analysis—that support both the provider's re-

sponse operations and the client's need for transparent, documented, and legally defensible incident handling. Investigation records must be tenant-scoped, exportable, and retainable by the client independently of the provider [7, 34].

- **Contributions to sovereignty, openness, and auditability.** Layer III is the most sovereignty-sensitive layer because it manipulates and interprets client security data. It supports sovereignty, openness, and auditability through documented normalization schemas, portable detection rules, standard intelligence-sharing protocols, declarative playbooks, tenant-scoped investigation records, immutable ingestion logs, versioned rule repositories, orchestration execution logs, and complete case records. Together, these artifacts allow clients or successor providers to reconstruct equivalent operations and verify what was detected, how the service responded, and whether actions followed agreed policies [38, 14, 18, 4].
- **Integration with adjacent layers.** Layer III depends on Layer I for analyst and service authentication, and for tenant-scoped authorization that ensures analysts can access only the data and configurations of the tenants they are authorized to serve. Layer II provides the primary ingress telemetry—connection logs, inspection events, and access records—that feeds the correlation pipeline. Layer III produces outputs consumed by Layer IV: detection statistics, incident summaries, response metrics, and compliance-relevant evidence that support governance, reporting, and regulatory obligations. Layer III also depends on Layer V for the deployment and configuration management of its own components, ensuring that detection engines, orchestration platforms, and intelligence functions are deployed reproducibly and that their configurations are governed as code.

Layer IV — Governance, Compliance, and Operational Oversight.

- **Definition and scope.** Layer IV provides the governance and oversight functions that ensure the managed security service operates accountably, remains aligned with regulatory obligations, and produces the evidence necessary for independent verification by clients and regulators. Its scope encompasses audit

trail management, SLA monitoring and verification, regulatory compliance evidence generation, asset and configuration inventory, change management, and client-facing service reporting. While every layer contributes to auditability through its own logging and transparency mechanisms, Layer IV is the layer where these contributions are aggregated, contextualized, and presented as a coherent governance record of the managed service.

The positioning of governance as a dedicated architectural layer—rather than as a cross-cutting annotation or an afterthought—reflects the finding from the literature review that compliance and risk management are frequently layered onto existing architectures instead of shaping them from the outset [23, 22]. By elevating governance to a first-class layer, the framework ensures that auditability, compliance, and oversight are structurally supported rather than dependent on the voluntary diligence of individual component operators.

- **Core responsibilities.** The first responsibility of Layer IV is the aggregation and management of audit trails from all contributing layers. Identity events from Layer I, perimeter logs from Layer II, detection and response records from Layer III, and deployment logs from Layer V must be collected, correlated, and retained in a centralized audit repository that supports long-term retention, integrity verification, and tenant-scoped access. The audit repository must be immutable in the sense that records, once written, cannot be modified or deleted without leaving a verifiable trace, ensuring that the audit trail itself is resistant to tampering [23, 18].

  The second responsibility is SLA monitoring and service-level verification. The framework must continuously measure and report on the operational metrics specified in service-level agreements: detection latency, response time, alert acknowledgment windows, mean time to resolution, system availability, and other contractually defined indicators. These metrics must be computed from primary operational data rather than from provider-curated reports, and they must be independently accessible to the client. This independent accessibility is a direct response to the information asymmetry identified in the literature,

where clients receive only aggregated assurances that may not reflect the actual service performance [4, 52, 28].

The third responsibility is regulatory compliance evidence generation. The framework must produce structured, exportable compliance artifacts that demonstrate adherence to relevant regulatory frameworks. These artifacts include access control reports, data handling documentation, incident response records, vulnerability management summaries, and risk assessment outputs. The evidence generation function must be configurable per tenant to support the varying regulatory environments in which different client organizations operate, reflecting the multi-jurisdictional realities of MSSP delivery under GDPR, NIS2, and related instruments [22, 14, 42].

The fourth responsibility is asset and configuration inventory. Layer IV must maintain a current and auditable inventory of all hardware assets, software components, service configurations, and logical relationships within the managed service environment. This inventory supports both operational governance—enabling change tracking, impact assessment, and lifecycle management—and compliance functions that require organizations to demonstrate knowledge of and control over their information assets [49, 46].

The fifth responsibility is change management and service reporting. All changes to the managed service environment—whether to detection rules, infrastructure configurations, access policies, or automation workflows—must pass through a documented change management process that records the rationale, authorization, implementation, and verification of each change. Client-facing service reports must be generated from the aggregated governance data and presented in formats that are both technically substantive and accessible to organizational decision-makers, bridging the gap between operational detail and executive oversight [7, 34].

- **Contributions to sovereignty, openness, and auditability.** Layer IV turns sovereignty into verifiable governance evidence. By exposing audit trails, SLA metrics, compliance artifacts, asset inventories, and change records in

open, machine-readable formats, it allows clients, auditors, and successor providers to assess service quality and compliance without relying only on provider assurances [43, 52, 5]. It is therefore the consolidation point for evidence about what the service did, when it acted, under which authorization, and with what result.

- **Integration with adjacent layers.** Layer IV consumes outputs from all other contributing layers. It receives identity and access logs from Layer I, perimeter telemetry and access records from Layer II, detection statistics, incident records, and response logs from Layer III, and deployment and configuration change records from Layer V. In return, Layer IV provides governance context to the operational and lifecycle layers: compliance policies may constrain the data retention settings in Layer III, access governance requirements may inform the identity policies in Layer I, and change management controls may gate the deployment automation in Layer V. This bidirectional integration ensures that governance is not merely observational but actively shapes operational behavior across the framework.

Layer V — Automation, Deployment, and Sustainability.

- **Definition and scope.** Layer V is the framework's lifecycle and control-plane layer. It addresses the long-term operational viability of the managed security service by providing the automation, deployment, and knowledge management functions necessary for reproducibility, maintainability, and evolutionary resilience. Its scope encompasses Infrastructure as Code (IaC) practices, declarative configuration management, workflow automation for operational and administrative tasks, documentation and knowledge retention systems, and the processes that support orderly service transitions, including exit procedures and migration support. This layer is the architectural response to the sustainability gap identified in the literature, where the technical, human, and economic dimensions of service continuity are discussed in isolation rather than integrated into a coherent operational model [46, 2, 40, 42].
- **Core responsibilities.** The first responsibility of Layer V is IaC and deploy-

ment automation. The entire managed service environment—from container definitions and network configurations to application settings and inter-service integrations—must be expressed in declarative, version-controlled code. This practice ensures that the environment can be reconstructed from its definition files alone, without reliance on undocumented manual procedures or tacit knowledge held by individual operators. Reproducible deployment is fundamental to both sovereignty and sustainability: it guarantees that the service can be rebuilt, migrated, or audited from a known, inspectable baseline [46].

The second responsibility is declarative configuration management. Beyond initial deployment, the ongoing configuration state of every component must be managed through idempotent, declarative mechanisms that enforce desired-state convergence. This means that configuration drift—the gradual divergence between intended and actual system state—is continuously detected and corrected, and that every configuration change is recorded as a versioned code artifact. Configuration management at this level eliminates the class of sovereignty risks associated with opaque, ad-hoc system administration, where the actual state of the service becomes unknowable to anyone other than the operator who most recently modified it [18].

The third responsibility is workflow automation for operational tasks. Routine administrative, maintenance, and governance workflows—backup verification, certificate renewal, patch orchestration, compliance report generation, health checks—must be automated through a workflow engine that executes predefined sequences, handles exceptions, and produces execution logs for audit purposes. This automation reduces the operational burden on human analysts, addresses the alert fatigue and overload concerns identified in the literature, and ensures that critical maintenance tasks are performed consistently and traceably [40, 44, 50].

The fourth responsibility is documentation and knowledge retention. The framework requires a structured knowledge management function that captures operational procedures, architecture documentation, runbooks, configuration rationale, and lessons learned in a searchable, versioned, and accessible

repository. Knowledge retention is essential for sustainability because it decouples the service from individual expertise. In the MSSP context, where analyst turnover and team scaling are recurrent operational realities, the ability to onboard new personnel from documented knowledge rather than from oral tradition is a critical resilience factor [2, 34].

The fifth responsibility is exit and migration support. The framework must be designed from the outset to support orderly service transitions, whether from one provider to another, from outsourced to in-house operation, or from one architectural instantiation to a different one. This includes the export of all tenant data, detection rules, playbooks, incident records, configurations, and compliance evidence in portable formats, as well as the provision of sufficient documentation for a competent successor to reconstruct equivalent operations. Exit support is not an edge case; it is a sovereignty requirement, because the credibility of the client's retained control depends on the demonstrated ability to exercise that control when needed [5, 22, 43].

- **Contributions to sovereignty, openness, and auditability.** Layer V supports sovereignty, openness, and auditability through IaC, declarative configuration, open automation formats, version control, execution logs, documentation, and exit support. These mechanisms give the client—or a successor provider—a portable description of how the service is configured, deployed, maintained, and reconstructed, reducing dependency on undocumented operational knowledge [43, 22].
- **Integration with adjacent layers.** Layer V provides the deployment and configuration management substrate for the other framework layers. Layer I's identity services, Layer II's perimeter components, Layer III's detection and response platforms, and Layer IV's governance tools are all deployed, configured, and maintained through the automation functions provided by Layer V. In return, Layer V consumes governance context from Layer IV: change management policies, approval workflows, and compliance constraints shape how deployment automation operates and what controls must be satisfied before

configuration changes are applied to the production environment. Layer V also generates deployment and configuration audit records that feed into Layer IV's governance repository, closing the accountability loop between operational action and governance oversight.

4.5 Inter-Layer Interaction Model

The five layers described above do not operate in isolation; they interact through defined data flows, control flows, and trust boundaries that together constitute the inter-layer interaction model of the framework. This section describes these interactions systematically. Figure 4.2 synthesizes these relations in a single analytical view.

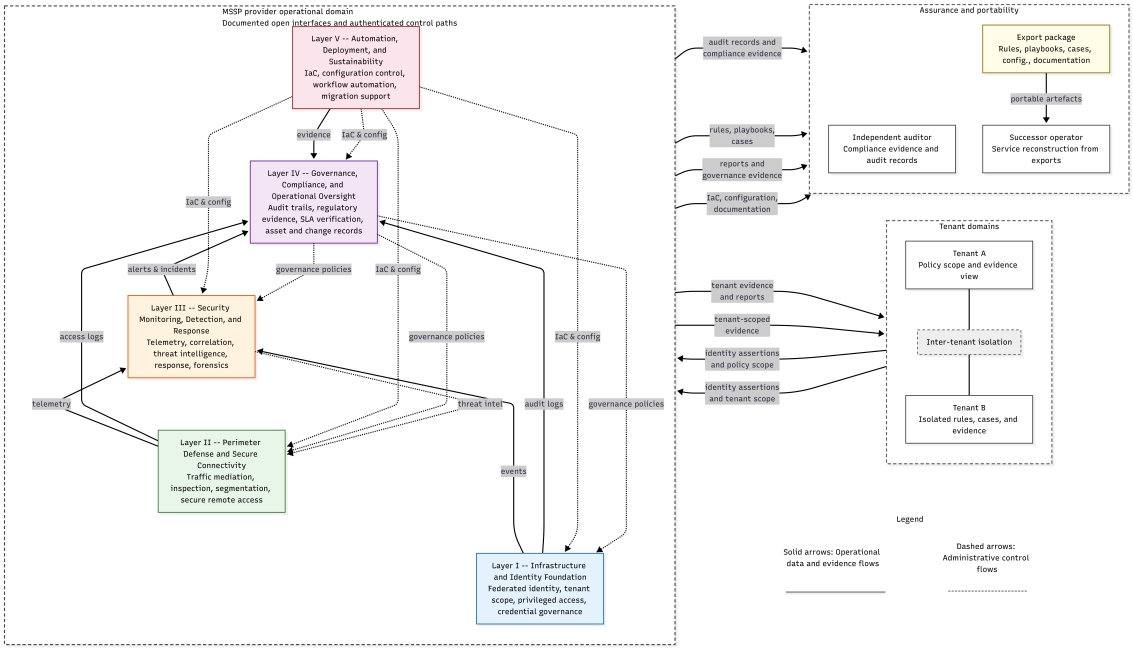

Figure 4.2: Inter-layer interaction model of the proposed MSSP framework, showing operational data and evidence flows, administrative control flows, trust boundaries, tenant isolation, audit exposure, and export paths for migration or reconstruction.

4.5.1 Data Flow

The dominant data flow direction is upward: operational telemetry and event data are generated at the lower layers and consumed, processed, and aggregated by the upper layers. Layer I produces identity and access events; Layer II produces network, perime-

ter, and inspection telemetry; Layer III processes these inputs through its correlation and detection pipeline and produces security alerts, incident records, and response logs; Layer IV aggregates the outputs of all layers into governance records, compliance evidence, and service reports; and Layer V contributes deployment and configuration change logs.

However, the data flow is not exclusively upward. Downward data flows exist where governance and intelligence outputs inform lower-layer operations. Threat intelligence processed by Layer III is disseminated downward to Layer II in the form of updated blocklists, reputation feeds, and inspection signatures. Compliance constraints defined in Layer IV flow downward to Layer III (shaping data retention policies), to Layer I (informing access governance rules), and to Layer V (constraining deployment automation). Configuration state managed by Layer V flows into every layer as the authoritative definition of how each component should be configured.

In all cases, inter-layer data exchange must use open, documented formats, and the transfer of data across layer boundaries must produce audit records that document what was transferred, by which identity, at what time, and under what authorization. This discipline ensures that data flow itself is an auditable and governable activity.

4.5.2 Control Flow

Control flow in the framework follows a dual pattern. Operational control flows upward from the infrastructure to the governance and automation layers: alerts generated by Layer III trigger escalation to Layer IV's incident management processes, and operational anomalies detected by Layer II may trigger automated responses orchestrated by Layer III. Administrative control flows downward from the automation and governance layers to the operational infrastructure: deployment commands from Layer V modify the state of components in Layers I through IV, and governance decisions from Layer IV may result in policy changes propagated to Layer I (access restrictions), Layer II (perimeter rule modifications), or Layer III (detection threshold adjustments).

The framework requires that all control flow actions be authenticated against Layer I, authorized against the role and tenant scope of the requesting entity, and logged for audit purposes. No layer may modify the state of another layer without passing through these identity and governance controls. This constraint ensures that inter-layer control flow

does not become an unmonitored channel through which unauthorized or untraceable modifications can propagate.

4.5.3 Trust Boundaries

The framework defines trust boundaries at three levels. The first is the *provider–tenant boundary*, which separates the provider's operational domain from each tenant's data and configuration scope. This boundary is enforced primarily at Layer I through identity-based access controls and at Layer II through network segmentation, and it is verified at Layer IV through governance monitoring. No provider-side operation should be able to access tenant data or modify tenant configurations without a documented and auditable authorization path.

The second is the *inter-tenant boundary*, which isolates different client organizations from one another within the shared multi-tenant environment. This boundary must be enforced at every layer: isolated directory partitions at Layer I, segregated network segments at Layer II, tenant-scoped detection pipelines and investigation environments at Layer III, tenant-specific compliance repositories at Layer IV, and tenant-parameterized configuration templates at Layer V. The inter-tenant boundary ensures that multi-tenancy—a necessary economic feature of scalable MSSP delivery—does not compromise the confidentiality, integrity, or auditability of any individual tenant's security posture [14, 54].

The third is the *inter-layer boundary*, which defines the interface contact between adjacent layers. Each layer exposes its capabilities to adjacent layers through documented, versioned interfaces and consumes the capabilities of adjacent layers through the same mechanism. This boundary prevents tight coupling between layers and ensures that the replacement of any single layer's implementation does not cascade into failures or incompatibilities in adjacent layers. The inter-layer boundary is a primary mechanism for satisfying DR3 (modularity) and for ensuring that the framework remains open and evolvable over time.

4.6 Threat Model and Trust Assumptions

This section defines the trust assumptions, threat actors, and attack surfaces that the framework is designed to address. The purpose is not to present a comprehensive penetration analysis of the prototype, but to make explicit the adversarial conditions under which the architecture's security properties are expected to hold.

4.6.1 Trust Assumptions

The framework assumes that Layer I (identity and trust infrastructure) constitutes the root of trust for the entire architecture. All authentication, authorization, and credential management decisions are anchored in this layer, and its compromise would affect the integrity of all upper layers. The host operating system and container runtime are within the trusted computing base; their hardening is a prerequisite for deployment but is not itself an architectural contribution of the framework. The MSSP provider is treated as a *partially trusted* actor: the provider is expected to operate the service in good faith, but the architecture is designed so that the client does not need to rely on provider honesty alone. Instead, auditability, inspectability, and evidence generation provide independent means of verification.

4.6.2 Threat Actors and Attack Surfaces

Three categories of threat actors are considered:

External attackers. Adversaries who target the MSSP infrastructure from outside the trust perimeter. Their primary attack surfaces are Layer II (network perimeter, exposed services) and any externally reachable management interfaces. The framework mitigates external threats through perimeter defense (Layer II), network segmentation, and identity-bound access controls (Layer I).

Compromised MSSP insiders. Provider personnel or automated processes with legitimate but abused access. Their attack surface spans all layers to which their role grants access. The framework mitigates insider threats through role-based and attribute-based access control (DR1), privileged access management with session

recording (Layer I), immutable audit trails (DR4), and the principle that all administrative actions must be authenticated, authorized, and logged.

Compromised components. Individual open-source components (e.g., Wazuh, Keycloak, Shuffle) that are exploited through software vulnerabilities. Their attack surface is the layer in which they operate and any adjacent layers reachable through their interfaces. The framework mitigates component compromise through modularity (DR3), inter-layer boundary enforcement, and the ability to replace any single component without disrupting the broader architecture.

4.6.3 Propagation Containment

The layered architecture and inter-layer trust boundaries are designed to limit the blast radius of compromise. A compromised component in Layer III (e.g., the SIEM engine) should not be able to modify identity configurations in Layer I or alter governance evidence in Layer IV, because inter-layer control flow must pass through identity and authorization checks enforced at Layer I. Similarly, a compromised deployment process in Layer V cannot bypass identity governance because the initial bootstrap of infrastructure is the only point at which Layer V operates before Layer I is established; once operational, all Layer V actions are subject to Layer I authentication.

The architectural properties most relevant to threat mitigation are: auditability (DR4), which ensures that compromise leaves detectable traces; modularity (DR3), which ensures that a compromised component can be isolated and replaced; tenant isolation (DR2), which ensures that compromise of one tenant's scope does not propagate to another; and identity governance (DR1), which ensures that all actions are attributable. These properties do not eliminate risk, but they structurally reduce the scope and detectability window of successful attacks.

4.7 Summary

This chapter has presented the design of a sovereign, open, and auditable MSSP framework organized into five architectural layers: Infrastructure and Identity Foundation, Perimeter Defense and Secure Connectivity, Security Monitoring, Detection, and

Response, Governance, Compliance, and Operational Oversight, and Automation, Deployment, and Sustainability. The framework is derived from eight requirements traceable to the research gaps identified in Chapter 3 and to the sovereignty dimensions established in Chapter 2.

Across these layers, sovereignty is realized through federable identity governance, transparent perimeter policies, portable detection and response artifacts, independently accessible compliance evidence, and reproducible infrastructure. Openness is enforced through open standards, documented interfaces, non-proprietary formats, and modular composition. Auditability is supported by treating identity actions, data flows, detection changes, response workflows, governance records, and deployment actions as inspectable evidence.

The inter-layer interaction model defines the data flows, control flows, and trust boundaries required for multi-tenant managed security delivery. The framework remains a conceptual architecture: it specifies what the managed service must provide and how its components relate, while Chapter 5 addresses the concrete technology stack and prototype.

Chapter 5

Prototype Implementation

This chapter describes the translation of the conceptual framework presented in Chapter 4 into a functional prototype. Whereas the preceding chapter defined what the managed security service must do and how its architectural layers must relate to one another, this chapter addresses how those layers are instantiated through concrete, open-source components deployed within a self-hosted infrastructure. The prototype serves as the demonstration step of the DSR methodology [37], providing architectural evidence that the proposed design can be realized with available technologies and meaningfully assessed against its design requirements.[1]

The chapter is organized as follows. Section 5.1 defines the prototype's scope and constraints, clarifying the boundary between what it implements and what it defers. Section 5.2 establishes the criteria used to select each component, grounded in principles of community maturity, openness, and enterprise adoption rather than in commercial comparison. Section 5.3 describes the instantiation of each of the five framework layers, mapping conceptual functions to specific open-source solutions and explaining the rationale for each mapping. Section 5.4 maps the prototype to the design requirements. Section 5.5 presents the deployment architecture, including the containerization strategy, network segmentation model, and deployment automation approach. Section 5.6 examines the integration logic that binds the components into a coherent platform. Section 5.7 summarizes the demonstration activities used to support the subsequent evaluation. Fi-

[1] The prototype source code and deployment automation are publicly available at `https://github.com/SergioTGSerra/mssp-prototype`.

nally, Section 5.8 summarizes the implementation decisions and their alignment with the framework's design requirements.

It is important to note that this chapter focuses on the architectural and integration dimensions of the prototype. Deep configuration details, installation procedures, and step-by-step deployment instructions are outside the scope of this dissertation; the emphasis is placed on explaining why each component was selected, how it fulfills its designated layer responsibilities, and how the components interact to produce the integrated managed security service envisioned by the framework.

5.1 Implementation Scope and Constraints

The prototype is designed as a representative instantiation of the five-layer framework, not as a production-grade deployment. Its purpose is to demonstrate that the architectural principles, design requirements, and inter-layer interactions defined in Chapter 4 can be realized through a coherent set of open-source components operating within a self-hosted environment. The prototype therefore prioritizes architectural coverage, integration coherence, and functional completeness over performance optimization, high-availability engineering, or large-scale multi-tenant operation.

Several constraints shape the implementation. First, the prototype operates within a single-site infrastructure rather than across geographically distributed data centers. While the framework's design supports distributed deployment, evaluating geographic distribution is outside the scope of this work. Second, multi-tenancy is addressed at the logical level through identity-based tenant separation rather than through physically isolated infrastructure per tenant. This reflects the operational reality of many MSSP environments, where economic viability depends on shared infrastructure with strong logical isolation [14, 54]. Third, the prototype is intentionally constrained to open-source components; no proprietary software is used in any layer. This creates a deliberate distinction: the framework defines an open architecture, while the prototype provides an open-source-only instantiation of that architecture. The open-source-only constraint is a methodological choice designed to foreground sovereignty, inspectability, and portability in the demonstration, and does not reflect a framework-level prohibition on proprietary

components. As established in Section 4.2, the framework defines openness as a set of architectural properties—standards-based interfaces, non-proprietary data formats, documented protocols, and inspectable configurations—rather than as a licensing requirement. Proprietary components that satisfy these properties would be architecturally compatible with the framework, although their use was not empirically examined in this prototype.

The prototype covers all five framework layers within the completed scope of this work. The implementation does not realize every possible operational refinement of the framework, but it instantiates the core components and integration paths necessary to assess the architecture's coherence, feasibility, and traceability to the design requirements.

5.2 Component Selection Criteria

The selection of components for the prototype follows a principled evaluation process rather than an ad hoc or opportunistic assembly of tools. Each component was assessed against a consistent set of criteria derived from the framework's design requirements and from the sovereignty, openness, and auditability principles articulated in Chapter 4. Five criteria guided all selection decisions.

Open-source availability and licensing transparency. For this prototype, every component was required to be available under an open-source license that permits inspection, modification, redistribution, and self-hosting without dependency on vendor-controlled licensing mechanisms. This is a prototype selection constraint rather than a framework-level licensing requirement. It supports DR3 (modular, standards-based architecture) and DR5 (portability of operational artifacts) by removing legal and contractual barriers to inspection, reproduction, and migration [5, 45]. The licensing model of each component was verified to ensure compatibility with the self-hosted, sovereignty-preserving deployment model used in the prototype.

Community strength and ecosystem maturity. Components were preferred when supported by active, well-governed open-source communities with a demonstrated track record of sustained development, transparent governance, and responsive issue resolution. Community strength is a proxy for long-term viability: a component with a

broad contributor base, regular release cycles, and active community forums is more likely to remain maintained, to receive timely security patches, and to evolve in response to emerging requirements than a component dependent on a single developer or a narrow contributor group [42, 46]. This criterion supports DR8 (operational sustainability and reproducibility) by reducing the risk that critical components become unmaintained.

Enterprise adoption and industry recognition. Preference was given to components with demonstrated adoption in enterprise and institutional environments, because widespread deployment serves as evidence of functional maturity, interoperability, and resilience under operational stress. Components recognized within the cybersecurity community through inclusion in reference architectures, mention in academic literature, or adoption by government and institutional users were considered more credible candidates for a sovereignty-preserving MSSP platform [6, 38, 14].

Interoperability and standards compliance. Each component was evaluated for its support of open standards and standard integration protocols. Components that expose well-documented APIs, support open data formats, and integrate through widely adopted protocols were preferred over those requiring proprietary connectors or bespoke integration logic. This criterion directly implements DR3 and supports the framework's insistence on open interfaces at every inter-layer and inter-component boundary [21, 48].

Alignment with framework layer responsibilities. Finally, each component was evaluated for its functional alignment with the specific responsibilities defined for its target layer in Chapter 4. A component was selected only if its functional scope demonstrated a credible capacity to fulfill the architectural responsibilities assigned to the function it would instantiate, as defined by the design requirements and layer specifications.

5.3 Layer-by-Layer Instantiation

This section describes how each of the five framework layers is instantiated in the prototype. For each layer, the relevant components are identified, the rationale for their selection is explained, and their functional mapping to the layer's architectural responsibilities is described. Figure 5.1 provides a consolidated view of the functional modules addressed within each layer.

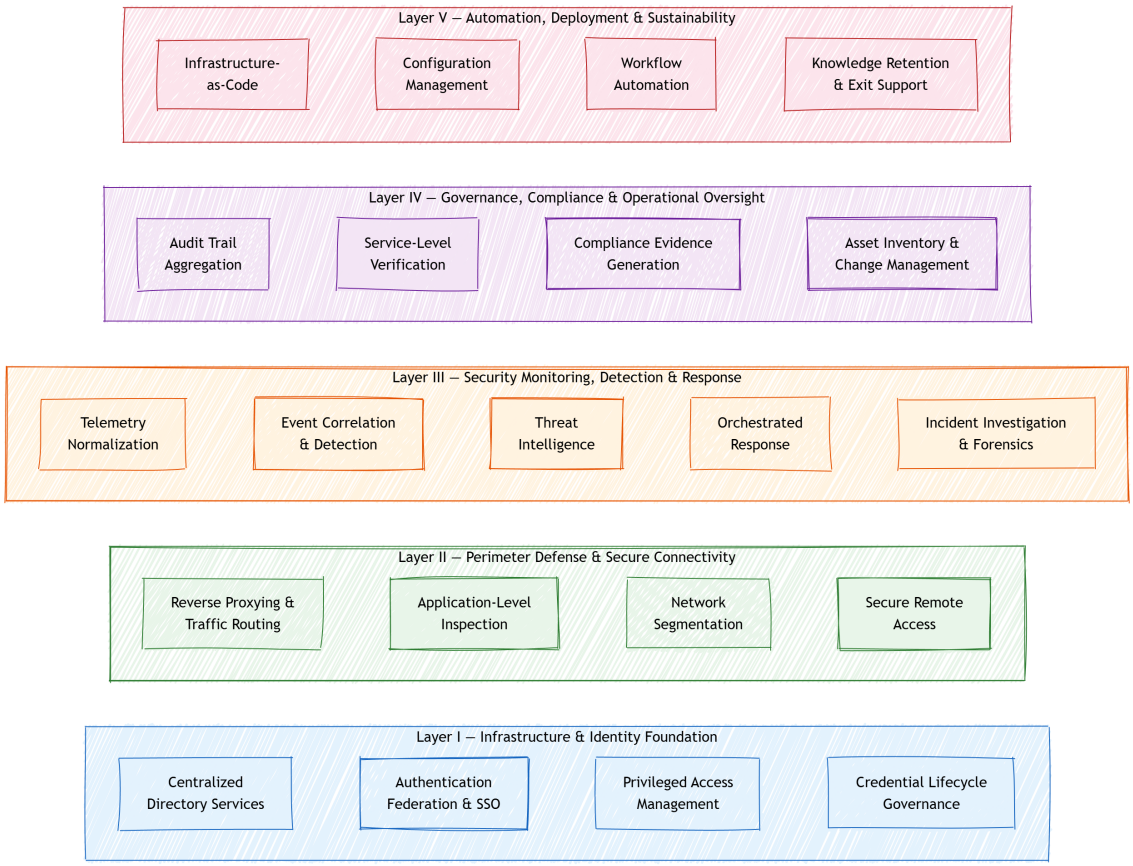

Figure 5.1: Technological architecture overview showing the functional modules within each of the five framework layers. Each layer groups the specific capabilities that are instantiated through open-source components in the prototype.

5.3.1 Layer I: Infrastructure and Identity Foundation

The Infrastructure and Identity Foundation layer, as defined in Section 4.4, requires centralized identity and directory services, authentication federation and single sign-on, privileged access management, and credential lifecycle governance. The prototype instan-

tiates these responsibilities through four complementary components.

The authoritative directory service is provided by FreeIPA, an integrated identity management system that combines directory services based on Lightweight Directory Access Protocol (LDAP), Kerberos-based authentication, certificate management, and policy enforcement within a unified platform. FreeIPA was selected because its directory and authentication capabilities match the open, inspectable identity foundation required by DR1. Its role in the prototype therefore instantiates the federated and standards-based identity substrate emphasized in service-oriented security architecture literature [14, 38]. Within the prototype, FreeIPA serves as the single source of truth for all user identities, group memberships, organizational roles, and host registrations across the platform.

Authentication federation and single sign-on are provided by Keycloak, an identity and access management platform that supports OpenID Connect, Security Assertion Markup Language (SAML) 2.0, and Open Authorization (OAuth) 2.0 protocols. Keycloak connects to FreeIPA as its user federation backend and exposes standardized authentication interfaces to all application-layer components in the stack. Its selection is justified by its protocol support and its capacity to serve as a centralized authentication broker for heterogeneous service portfolios—precisely the role required by DR1. The combination of FreeIPA and Keycloak ensures that the identity layer is both authoritative (through FreeIPA's directory) and federable (through Keycloak's protocol support), satisfying the sovereignty requirement that client organizations can integrate their own identity providers when hybrid governance models are needed [54, 38].

Privileged access management is provided by JumpServer, a platform that mediates administrative access to infrastructure and sensitive systems through session recording, granular authorization, and audit trail generation. JumpServer was selected for its open-source availability, its mature support for protocol-level session proxying, and its alignment with the framework's requirement that every privileged action be attributed, recorded, and independently reviewable (DR4). Within the prototype, JumpServer operates on a dedicated network segment and authenticates all sessions against the Keycloak identity broker, ensuring that privileged access is governed by the same identity foundation that controls all other platform access.

Credential lifecycle management is provided by Vaultwarden, a lightweight, self-hosted

implementation compatible with the Bitwarden password management ecosystem. Vaultwarden manages passwords, secrets, and shared credentials for administrators, teams, and service accounts. Its inclusion addresses the credential governance responsibilities defined for Layer I by providing centralized, auditable credential storage with support for rotation policies and team-scoped access. This self-hosted credential governance role supports the sovereignty requirement that critical control mechanisms remain inspectable and governable within the client's or provider's own operational domain [22].

Together, these four components establish the trust substrate upon which all upper layers depend. Identity assertions issued by Keycloak, grounded in FreeIPA's directory, propagate upward through the architecture, providing the authenticated context for every action taken within the managed service environment.

5.3.2 Layer II: Perimeter Defense and Secure Connectivity

The Perimeter Defense and Secure Connectivity layer, as defined in Section 4.4, requires reverse proxying and traffic routing, application-level inspection and web application firewall capabilities, network segmentation, and secure remote access mediation. The prototype instantiates these responsibilities through two core components and a container-level network segmentation strategy.

The reverse proxy and web application firewall function is provided by BunkerWeb, a security-focused reverse proxy that integrates web application firewall capabilities, automatic Transport Layer Security (TLS) certificate management, and declarative rule-based traffic inspection. BunkerWeb sits at the edge of the platform and serves as the sole ingress point for all external Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) traffic, terminating encrypted connections and routing requests to internal services based on declarative configuration. Its selection is based on its open-source availability, its integrated approach to combining reverse proxying with application-level security inspection, and its support for declarative, version-controllable configuration formats—a property that directly supports DR4 (auditability) and DR8 (reproducibility). All inspection and routing rules are expressed as configuration files that can be versioned, audited, and reproduced.

Network segmentation within the prototype is implemented through dedicated con-

tainer networks that logically separate different functional domains. The deployment architecture defines distinct network segments for public-facing access, identity services, email and communication services, privileged access management, security operations, and internal service communication. Inter-segment communication follows explicit, policy-defined paths, ensuring that the principle of least connectivity is enforced at the infrastructure level. This segmentation strategy implements the Layer II responsibility for micro-segmentation described in the framework design and supports DR2 (multi-tenant isolation) by ensuring that functional domains are separated even within a shared infrastructure [46].

Secure remote access is addressed through NetBird, which provides a zero-trust networking layer that establishes secure, peer-to-peer connectivity across distributed systems and users without relying on traditional perimeter-based Virtual Private Network (VPN) architectures. NetBird was selected for its open-source zero-trust architecture and its support for identity-driven access policies. In the prototype, remote access is governed through the Layer I identity foundation, ensuring that connectivity control remains aligned with the same trust model that regulates the broader platform.

5.3.3 Layer III: Security Monitoring, Detection, and Response

The Security Monitoring, Detection, and Response layer, as defined in Section 4.4, requires telemetry acquisition and normalization, event correlation and threat detection, threat intelligence integration, security orchestration and automated response, and incident investigation and digital forensics. The prototype instantiates these responsibilities through four specialized components.

Telemetry acquisition, normalization, event correlation, and threat detection are provided by Wazuh, a unified security monitoring platform that integrates host-based intrusion detection, log analysis, file integrity monitoring, vulnerability detection, and security event correlation within a single, open-source solution. Wazuh was selected because it represents one of the most widely adopted open-source security monitoring platforms in the community, with extensive documentation, a large and active contributor ecosystem, and demonstrated deployment across enterprise, government, and academic environments. Its functional scope aligns precisely with the framework's requirements for a composable

detection and correlation capability (DR7), while its open rule format and documented normalization schema support the portability and auditability requirements (DR4, DR5). Within the prototype, Wazuh agents are deployed across all monitored endpoints and services, collecting security-relevant telemetry that is forwarded to the central Wazuh manager for correlation and analysis. Detection rules are maintained as version-controlled, human-readable configuration files, ensuring that the detection logic is inspectable, exportable, and reproducible [45, 23].

Security orchestration and automated response are provided by Shuffle, an open-source SOAR platform designed to connect security tools, automate response workflows, and execute playbooks across heterogeneous environments. Shuffle was selected for its open-source nature, its visual workflow design approach, its extensive integration capabilities through standard APIs, and its explicit orientation toward security operations automation. Within the prototype, Shuffle connects to Wazuh for alert ingestion, to IRIS for case escalation, and to the broader platform infrastructure for executing automated containment and notification actions. Response playbooks are defined as declarative workflow definitions that can be exported, versioned, and reproduced—directly satisfying DR5 (portability of operational artifacts) as applied to orchestration logic [9, 21].

Threat intelligence integration is provided by MISP, the open-source threat intelligence platform widely recognized as a reference solution for indicator management, threat intelligence sharing, and intelligence enrichment within the cybersecurity community. MISP was selected because of its position as one of the most established and community-driven threat intelligence platforms, its support for open, standardized intelligence-sharing formats, its extensive integration ecosystem, and its demonstrated adoption across national and sectoral cybersecurity organizations [6, 33]. Within the prototype, MISP serves as the central repository for indicators of compromise, adversary tactics, and contextual threat reports. It receives intelligence from external feeds and distributes enrichment data to Wazuh for detection enhancement and to Shuffle for intelligence-informed response workflows.

Incident investigation and digital forensics are provided by IRIS, an open-source incident response platform that supports structured case management, evidence collection, timeline reconstruction, and collaborative investigation workflows. IRIS was selected for its focus on the investigative and forensic dimensions of incident response and its capacity

to serve as a tenant-scoped investigation environment—a function explicitly required by the framework design. Within the prototype, IRIS receives escalated incidents from Shuffle, maintains structured investigation records linked to the originating alerts, and provides exportable case documentation. This role reflects the literature's emphasis on documented incident-response knowledge flows and situation awareness in security operations [7, 34].

Together, these four components form the operational core of the managed security service, transforming raw telemetry into correlated alerts, enriching detection with threat intelligence, orchestrating response through automated workflows, and supporting structured investigation when incidents require deeper analysis. The decomposition into four distinct, interoperable components—rather than a single monolithic platform—is a deliberate implementation of DR3 (modularity) and a concrete demonstration that the framework's insistence on composable security operations can be realized in practice.

5.3.4 Layer IV: Governance, Compliance, and Operational Oversight

The Governance, Compliance, and Operational Oversight layer, as defined in Section 4.4, requires audit trail aggregation, SLA monitoring and verification, regulatory compliance evidence generation, asset and configuration inventory, change management, and client-facing service reporting. The prototype instantiates these responsibilities through three primary components.

Infrastructure monitoring and service-level visibility are provided by Zabbix, a mature, enterprise-grade monitoring and alerting platform with a long history of deployment across institutional and enterprise environments. Zabbix was selected for its extensive monitoring capabilities, its open-source availability, its scalable architecture, and its well-documented API support. Within the prototype, Zabbix monitors the health, performance, and availability of all platform components, generating the operational metrics necessary for SLA computation and service-level verification. Its alerting capabilities provide early warning of service degradation, while its historical data retention supports the retrospective analysis of service performance. The metrics collected by Zabbix are computed from primary operational data rather than from curated provider reports, directly addressing the information asymmetry concern identified in the literature [4, 52].

Information Technology (IT) service management and asset inventory are provided by

Gestionnaire Libre de Parc Informatique (GLPI), an open-source platform for IT service management, asset tracking, helpdesk processes, and operational governance. GLPI was selected because it offers a comprehensive approach to IT asset lifecycle management, incident ticketing, and change management within a single, open, and self-hosted solution. Within the prototype, GLPI maintains the authoritative inventory of all hardware assets, software components, service configurations, and logical relationships within the platform. It also provides the change management function required by the framework, ensuring that modifications to the managed service environment pass through a documented process that records rationale, authorization, and verification for each change. This use of GLPI operationalizes the asset, process, and change-management concerns identified in SOC maturity and infrastructure-practice literature [49, 46].

Documentation, knowledge management, and operational procedure retention are provided by BookStack, a self-hosted documentation platform that supports structured, searchable, and versioned content organized in a hierarchical book-chapter-page model. BookStack was selected for its simplicity, its self-hosted nature, and its capacity to serve as the institutional knowledge repository for operational procedures, architecture documentation, runbooks, and playbooks. While BookStack does not generate compliance evidence directly, it supports the governance layer by ensuring that the knowledge required to understand, operate, and audit the managed service is captured in a persistent, accessible, and searchable format rather than residing in the tacit expertise of individual operators [2, 34].

Compliance evidence generation in the prototype is addressed as a cross-cutting concern rather than through a single dedicated component. Audit evidence is distributed across the systems that originate it: identity events are retained by the identity and privileged-access components, perimeter and security-operation records are retained by the corresponding operational tools, Zabbix provides service availability metrics, GLPI provides asset and change management records, BookStack preserves governance documentation, and the version-controlled infrastructure definitions from Layer V provide deployment and configuration evidence. The prototype therefore implements audit aggregation as an evidence-collection pattern rather than as a single centralized audit repository. Together, these sources produce the structured, exportable compliance artifacts required

by DR6 (regulatory compliance by design) [22, 14].

5.3.5 Layer V: Automation, Deployment, and Sustainability

The Automation, Deployment, and Sustainability layer, as defined in Section 4.4, requires IaC and deployment automation, declarative configuration management, workflow automation for operational tasks, documentation and knowledge retention, and exit and migration support. The prototype instantiates these responsibilities through a combination of automation practices, deployment tooling, and documentation mechanisms.

Infrastructure-as-code and deployment automation are realized through the combination of shell-based deployment scripts and Ansible, an agentless configuration management and automation platform widely adopted in enterprise and institutional environments. The prototype's deployment is organized around a main setup script that orchestrates the sequential provisioning of all platform components, with supporting utility scripts and environment-based configuration through a centralized environment file. This modular, script-based approach ensures that the entire platform can be reconstructed from its definition files alone, without dependency on undocumented manual procedures. Ansible complements and extends this shell-based approach by introducing idempotent, declarative state management for operating-system-level provisioning, system baseline definition, and repeatable configuration tasks [46]. Ansible was selected because of its agentless architecture (which avoids introducing additional infrastructure dependencies), its dominant position in the configuration management ecosystem, its extensive module library, and its declarative playbook format that serves simultaneously as automation logic and as auditable documentation of desired system state. Together, the shell scripts and Ansible playbooks implement DR8 (operational sustainability and reproducibility) by ensuring that deployment knowledge is codified rather than tacit and that the platform can be rebuilt, audited, or migrated from a known, inspectable baseline.

The containerization substrate is provided by Podman, a daemonless container engine that provides process isolation, image management, and container lifecycle control without requiring a privileged system daemon. Podman was selected over alternative container runtimes because its daemonless, rootless architecture improves the security posture of the host system, its compatibility with standard container image formats ensures portability,

and its open-source governance model aligns with the platform's sovereignty requirements. Each service in the prototype is deployed as an isolated container, with dedicated container networks enforcing the network segmentation defined in Layer II. The use of containerization ensures that every component is deployed from a reproducible image definition, that dependencies are isolated from the host operating system, and that the deployment can be reproduced across different infrastructure environments. This supports the modular service-deployment logic emphasized in NFV-based Security-as-a-Service work [14].

Within the implemented prototype, operational and administrative automation is addressed through the deployment scripts and Ansible playbooks rather than through a separate workflow-automation platform. This keeps routine provisioning, baseline configuration, and repeatable maintenance procedures within the same auditable automation layer that defines the infrastructure state, while leaving more elaborate event-driven administrative workflows outside the prototype scope.

Documentation and knowledge retention, as noted in Section 5.3.4, are supported by BookStack at the governance layer. From the perspective of Layer V, BookStack also serves the sustainability function by providing the structured repository in which deployment procedures, architecture rationale, operational guides, and lessons learned are captured. This dual positioning reflects the cross-cutting nature of knowledge management, which serves both governance (Layer IV) and sustainability (Layer V) objectives.

Exit and migration support are addressed structurally rather than through a dedicated tool. Because every component is open-source, deployment is defined as code, configurations are declarative and version-controlled, and operational artifacts are maintained in exportable formats, the prototype provides structural support for portability. A competent successor—whether another provider or the client organization itself—can in principle reconstruct equivalent operations from the exported infrastructure definitions, detection rules, playbooks, incident records, and configuration files. This structural portability implements DR5, while a full provider-exit exercise remains outside the prototype scope [5, 43].

5.4 Traceability to Design Requirements

Table 5.1 summarizes how the prototype instantiates the eight design requirements defined in Chapter 4. The table does not replace the evaluation in Chapter 6; it identifies the concrete mechanisms and artifacts on which that evaluation is based.

Table 5.1: Traceability between design requirements and prototype mechanisms.

Req.	Prototype mechanism	Evidence or artifact produced
DR1	FreeIPA, Keycloak, JumpServer, Vaultwarden	Identity records, role mappings, authentication events, privileged session records, credential records
DR2	Tenant-scoped identity configuration, container networks, access policies	Tenant-specific access scopes, logical network boundaries, scoped operational views
DR3	Modular component stack connected through open protocols and documented interfaces	Component dependency map, protocol interfaces, replaceable service boundaries
DR4	Identity logs, privileged session records, Wazuh events, Shuffle executions, IRIS cases, change records	Traceable evidence of access, detection, response, investigation, and configuration changes
DR5	Wazuh rules, Shuffle workflows, IRIS case records, configuration files, deployment definitions	Exportable detection, orchestration, incident, configuration, and deployment artifacts
DR6	Zabbix metrics, GLPI assets and changes, retained audit evidence, BookStack documentation	Service availability records, asset inventories, change records, compliance-supporting documentation
DR7	Wazuh–Shuffle–MISP–IRIS operational pipeline	Telemetry flow, alerts, enriched intelligence, response workflows, incident cases
DR8	Podman containers, shell scripts, Ansible playbooks, version-controlled documentation	Reproducible deployment definitions, configuration history, operational knowledge base

5.5 Deployment Architecture

The deployment architecture of the prototype translates the abstract layer boundaries of the framework into concrete infrastructure decisions. This section describes the containerization strategy, network topology, and deployment sequencing that govern how the components are provisioned and organized.

5.5.1 Containerization and Isolation Strategy

All prototype components are deployed as containers managed by Podman. Each service operates within its own container, isolated from other services and from the host operating system. This approach provides several architectural benefits aligned with the framework's design requirements. Container isolation ensures that a vulnerability or misconfiguration in one component does not directly compromise adjacent components, sup-

porting the defense-in-depth principle. Container images are defined declaratively through standard image specifications, ensuring reproducibility across environments. The daemonless, rootless execution model of Podman further reduces the attack surface of the container substrate itself.

The container deployment is orchestrated through the main setup script, which provisions services in a defined sequence that respects inter-component dependencies (see Figure 5.2). Identity services (FreeIPA, Keycloak) are provisioned first because all subsequent services depend on the identity foundation for authentication. The perimeter layer (BunkerWeb) is provisioned next, establishing the traffic routing and inspection infrastructure. Security operations, governance, and automation components are then provisioned in dependency order, with each component configured to integrate with the identity and perimeter layers during its setup process.

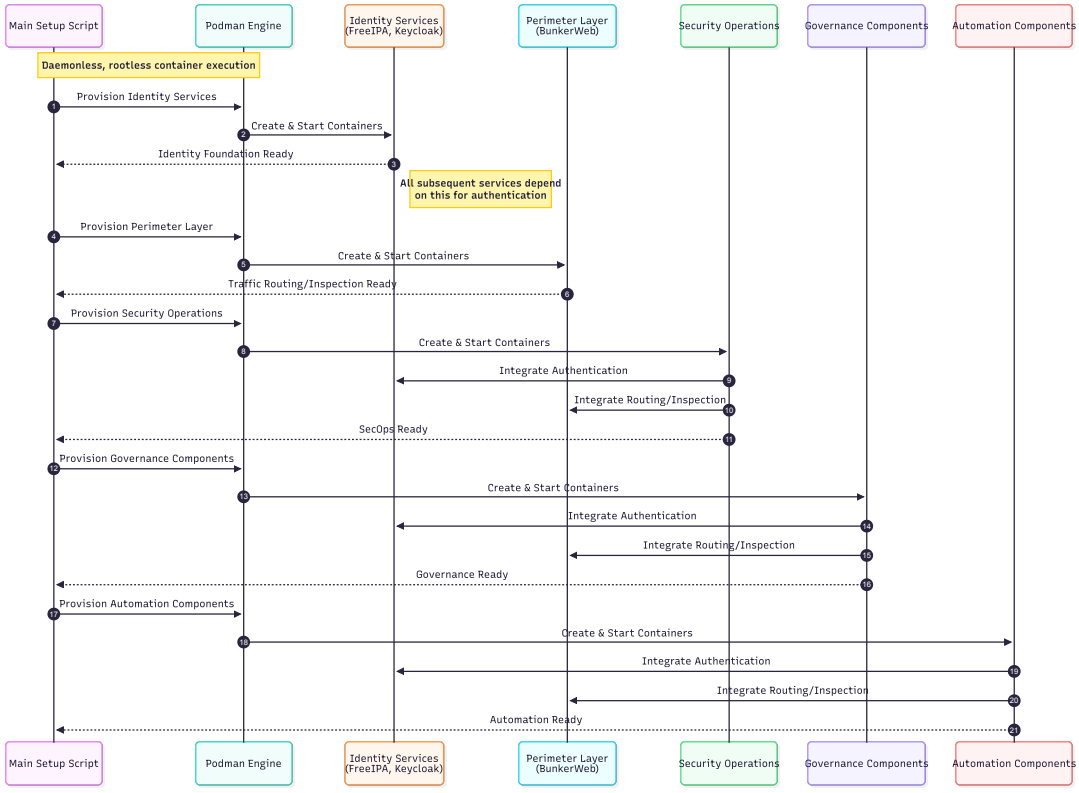

Figure 5.2: Container deployment sequence managed by the setup script, illustrating the dependency-ordered provisioning of identity, perimeter, security operations, governance, and automation components.

5.5.2 Network Segmentation

The prototype implements logical network segmentation through dedicated container networks. Each functional domain operates on a distinct network segment, and inter-segment communication is restricted to explicitly defined paths. The principal network segments include a public-facing network for ingress traffic through the reverse proxy, an identity services network connecting the directory, authentication, and credential management components, a privileged access network isolating the privileged access management gateway, a security operations network connecting the monitoring, detection, response, and intelligence components, and an internal services network for governance, collaboration, and business support functions.

This segmentation model implements the Layer II responsibility for micro-segmentation and enforces the principle of least connectivity at the infrastructure level. The network boundaries also serve as enforcement points for the inter-tenant and inter-layer trust boundaries defined in the framework's inter-layer interaction model (Section 4.5).

5.6 Integration Model

The integration of the prototype's components follows the data flow, control flow, and trust boundary patterns defined in the framework's inter-layer interaction model. Figure 5.3 presents an abstract map of the primary communication protocols and standard interfaces employed across the framework layers, ensuring an open and vendor-agnostic architecture. This section describes the principal integration pathways.

5.6.1 Identity Integration

The identity integration pattern, depicted in Figure 5.4, is the most pervasive in the prototype. FreeIPA serves as the authoritative directory, and Keycloak federates FreeIPA's identity data to all application-layer components through OpenID Connect. Services that support OpenID Connect authenticate users through Keycloak, which in turn validates credentials against FreeIPA. Services that support direct LDAP authentication bind to FreeIPA's directory. JumpServer authenticates privileged sessions through Keycloak, and Vaultwarden integrates with the identity layer for team-scoped credential access. This

Figure 5.3: Framework integration and protocols map, illustrating the open standard communication protocols (e.g., OIDC, LDAP, REST API, SSH) employed between architectural layers to prevent vendor lock-in.

consistent identity integration ensures that every action within the platform is attributable to a verified identity, satisfying DR1 and DR4 simultaneously.

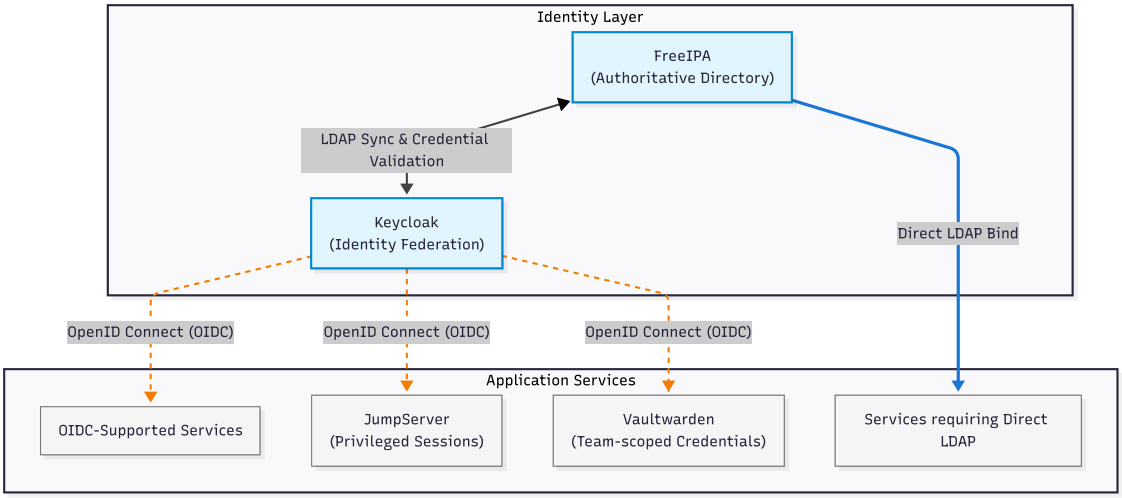

Figure 5.4: Identity integration map illustrating the authentication flows. FreeIPA acts as the authoritative directory for direct LDAP binds, while Keycloak provides OpenID Connect (OIDC) federation for compatible services, including JumpServer and Vaultwarden.

5.6.2 Security Operations Integration

The security operations integration follows a pipeline model, as depicted in Figure 5.5. Wazuh agents deployed across the infrastructure collect security telemetry and forward it to the central Wazuh manager, where correlation rules produce security alerts. Alerts meeting defined severity thresholds are forwarded to Shuffle, which executes the appropriate response playbooks. MISP provides threat intelligence enrichment to both Wazuh (for detection rule enhancement) and Shuffle (for intelligence-informed response actions). When incidents require investigation, Shuffle escalates cases to IRIS, which maintains structured investigation records linked to the originating alerts. This pipeline implements DR7 (layered, composable security operations) and ensures that each component in the security operations chain communicates through documented interfaces.

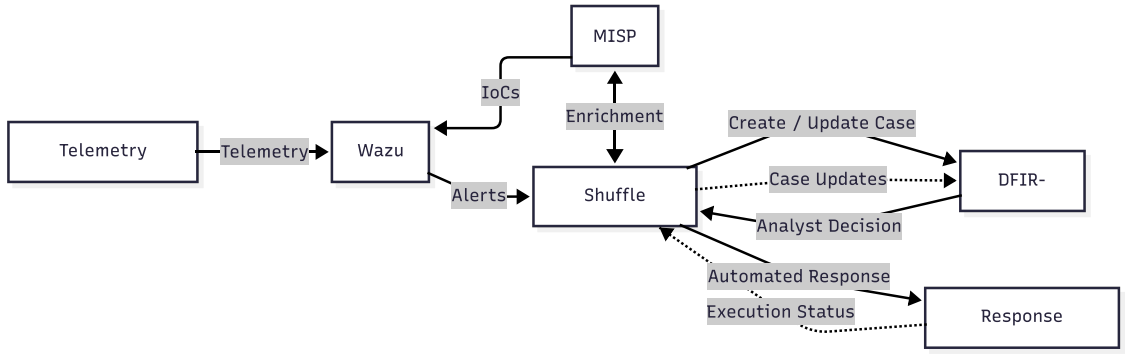

Figure 5.5: Security operations pipeline illustrating the unidirectional and bidirectional data flows across the integrated components, from raw telemetry ingestion to structured case management.

5.6.3 Governance Integration

Governance integration operates as an aggregation pattern. Zabbix collects operational metrics from all infrastructure components, providing the raw data for SLA monitoring. GLPI maintains the asset and configuration inventory, receiving updates from the deployment automation layer and from manual asset registration processes. Audit logs from identity events, perimeter telemetry, and security operations records are retained and made available for compliance reporting. BookStack provides the knowledge management substrate in which operational procedures and governance documentation are maintained.

Together, these integrations ensure that the governance layer has access to the evidence required for independent verification of the managed service's operations.

5.6.4 Automation Integration

The automation layer integrates with all other layers through the provisioning and configuration mechanisms defined in the deployment scripts and Ansible playbooks. Together, these mechanisms manage the provisioning, configuration, and repeatable maintenance of platform components across all layers, ensuring that the platform's configuration is defined as code and that changes are applied through auditable, repeatable processes. Their execution records contribute to the governance layer's audit trail.

5.7 Prototype Validation Procedures

The prototype was exercised through a set of focused demonstration activities aligned with the evaluation anchors defined in Table 4.3. These activities were intended to verify architectural coherence and traceability, not to benchmark production performance.

- Re-deployment of the platform from the version-controlled deployment scripts and Ansible definitions, validating that the environment can be reconstructed from documented artifacts.
- Authentication and access-flow verification through FreeIPA, Keycloak, JumpServer, and Vaultwarden, validating that user and privileged actions remain tied to the identity foundation.
- Security operations workflow validation through the Wazuh–Shuffle–MISP–IRIS pipeline, validating the flow from telemetry ingestion and alerting to enrichment, response, and case management.
- Governance evidence review across Zabbix metrics, GLPI asset and change records, BookStack documentation, retained audit logs, and version-controlled configuration artifacts.

- Export and reconstruction review of representative operational artifacts, including detection rules, playbook definitions, incident records, deployment files, and configuration files.

These activities support the qualitative assessment presented in Chapter 6. They demonstrate that the core architectural mechanisms are present and integrated, while leaving production-scale performance, adversarial testing, formal regulatory audit, and full provider-exit validation outside the prototype scope.

5.8 Summary

This chapter has described the translation of the proposed framework into a functional prototype built entirely from open-source components deployed within a self-hosted, containerized infrastructure. The prototype instantiates all five framework layers: the Infrastructure and Identity Foundation through FreeIPA, Keycloak, JumpServer, and Vaultwarden; the Perimeter Defense and Secure Connectivity layer through BunkerWeb, NetBird, and container network segmentation; the Security Monitoring, Detection, and Response layer through Wazuh, Shuffle, MISP, and IRIS; the Governance, Compliance, and Operational Oversight layer through Zabbix, GLPI, and BookStack; and the Automation, Deployment, and Sustainability layer through Ansible, Podman, and shell-based deployment scripts.

Every component was selected through a consistent evaluation against criteria of open-source availability, community maturity, enterprise adoption, interoperability, and functional alignment with the framework's layer responsibilities. No component depends on proprietary licensing, vendor-controlled infrastructure, or closed-source integration mechanisms. The prototype's deployment is reproducible from version-controlled definitions, and the core operational artifacts relevant to detection, response, investigation, governance, and deployment are maintained in exportable formats.

Table 5.2 summarizes the mapping between framework layers and prototype components.

The integration model demonstrates that the components can operate as a coherent platform rather than as a collection of isolated tools. Identity integration through FreeIPA

Table 5.2: Mapping of framework layers to prototype components.

Framework layer	Prototype components
Layer I: Infrastructure and Identity Foundation	FreeIPA, Keycloak, JumpServer, Vaultwarden
Layer II: Perimeter Defense and Secure Connectivity	BunkerWeb, NetBird, container network segmentation
Layer III: Security Monitoring, Detection, and Response	Wazuh, Shuffle, MISP, IRIS
Layer IV: Governance, Compliance, and Operational Oversight	Zabbix, GLPI, BookStack
Layer V: Automation, Deployment, and Sustainability	Ansible, Podman, shell scripts

and Keycloak provides the authenticated context consumed by every component. The security operations pipeline from Wazuh through Shuffle and MISP to IRIS implements the composable detection-and-response model required by the framework. Governance integration through Zabbix, GLPI, and BookStack provides the evidence base for accountability and compliance. Automation through Ansible and the deployment scripts supports reproducibility and operational sustainability.

The prototype shows that the framework can be instantiated to a substantial degree with currently available open-source technologies. It provides the practical basis for the evaluation and discussion that follow in Chapter 6, where the prototype's capacity to address the eight design requirements and the research gaps identified in the literature is assessed systematically.

Chapter 6

Validation and Discussion

This chapter interprets the results of the literature review, framework design, and prototype demonstration against the research questions. It assesses the extent to which the proposed architecture addresses the identified gaps, while keeping the claims proportionate to the prototype's demonstration scope.

6.1 Answering the Research Questions

To establish a clear framework for the ensuing analysis and to contextualize the architectural contributions of this work, the primary research questions that guided the study are restated below:

RQ1: What are the structural limitations of current MSSP architectures regarding vendor lock-in, transparency, and sovereignty, as evidenced by the academic literature?

RQ2: What specific design requirements must an MSSP framework fulfill to enable independent verification of security controls, ensure regulatory compliance, and unify open architectures with sovereignty-driven governance?

RQ3: How can disparate open-source security solutions be effectively orchestrated into a cohesive, auditable, and sustainable managed service architecture?

RQ4: To what extent does the proposed integrated framework satisfy these theoretical design requirements when evaluated against representative architectural constraints?

The primary research question asked how an integrated, open, and auditable framework for managed security services can be designed to mitigate vendor lock-in and advance technological sovereignty in cyber defense. The dissertation answers this question by showing that such a framework is feasible when sovereignty is treated as an architectural constraint rather than as a contractual aspiration. In the proposed solution, sovereignty is implemented through explicit identity governance, modular and standards-based composition, end-to-end auditability, portability of operational artifacts, compliance-oriented evidence generation, and reproducible deployment. The resulting five-layer architecture therefore addresses the problem at the level where dependence is structurally produced: not only in procurement choices, but in data flows, detection logic, orchestration workflows, governance records, and deployment knowledge.

RQ1 concerned the structural limitations of current MSSP architectures. Chapter 3 showed that the principal limitation is not a lack of technical capability, but a lack of integration across outsourcing, security operations, governance, and sovereignty perspectives. Existing approaches remain fragmented: some optimize component-level detection and response, others improve compliance or outsourcing decisions, but few address the provider–client asymmetry that makes auditability, portability, and verifiable control difficult in practice. As a result, vendor lock-in and opacity persist not merely because providers use proprietary tools, but because the managed service is rarely designed so that tenants can inspect, verify, and reconstruct its operational logic.

RQ2 asked which design requirements such a framework must satisfy. The dissertation answered this by deriving eight requirements from the literature and from the sovereignty dimensions established in Chapter 2: federable identity governance, multi-tenant isolation with per-tenant visibility, modular and standards-based composition, end-to-end auditability, portability of operational artifacts, compliance by design, layered and composable security operations, and operational sustainability with reproducibility. These requirements are theoretically significant because they translate an otherwise abstract notion of technological sovereignty into verifiable architectural properties that can guide both design and evaluation.

RQ3 asked how heterogeneous open-source security solutions can be orchestrated into a coherent MSSP architecture. The framework and prototype provide a concrete answer

at the level of architectural demonstration. The orchestration logic does not depend on a monolithic security platform; instead, it depends on explicit trust boundaries, a shared identity substrate, open interfaces, version-controlled configuration, and a layered separation of concerns. The prototype shows that identity services, perimeter controls, security monitoring, orchestration, threat intelligence, governance tooling, and automation mechanisms can be composed into a coherent service model while preserving inspectability and replaceability across layers. This does not imply that every framework instantiation must be open-source-only; the prototype uses open-source components as a controlled demonstration of the framework's open-architecture principles.

RQ4 asked to what extent the framework satisfies the derived requirements under representative architectural constraints. The answer is necessarily qualified. At the level of architectural demonstration, the prototype supports a qualitative assessment that the framework is internally coherent and technically feasible. The strongest support concerns modularity, auditability, portability, and reproducibility, because these properties are directly instantiated in the prototype architecture. The weaker areas are those that require production-scale or longitudinal validation, especially large-scale multi-tenancy, regulatory assurance across concrete jurisdictions, and long-term operational sustainability. The dissertation therefore supports a claim of feasibility and structured lock-in mitigation, not a claim that the prototype fully validates a production-ready MSSP model.

The assessment is grounded in the prototype mechanisms and demonstration activities described in Sections 5.4 and 5.7. Table 6.1 summarizes this assessment.

6.2 Addressing Sovereignty Dimensions

Beyond the evaluation of individual design requirements, the framework is assessed against the four sovereignty dimensions defined in Section 2.2.2: control, transparency, portability, and independence. Table 6.2 summarizes this assessment.

The sovereignty assessment shows that three of the four dimensions—control, transparency, and portability—are structurally enabled by the framework's design and prototype. The independence dimension is partially demonstrated: the structural foundations are in place, but full validation requires the production-scale and longitudinal work iden-

Table 6.1: Evaluation of the design requirements in the prototype.

Req.	Assessment	Basis for assessment
DR1	Architecturally instantiated	Centralized identity, federation, privileged access control, and credential governance are instantiated through FreeIPA, Keycloak, JumpServer, and Vaultwarden.
DR2	Partially instantiated	Multi-tenancy is addressed through multiple tenant configurations with identity-based access boundaries and network segmentation at the logical level. However, adversarial cross-tenant testing, production-scale tenant operations, and enforcement under heterogeneous client conditions were not performed.
DR3	Structurally supported	The prototype relies on independently replaceable open-source components connected through documented interfaces, open protocols, and non-proprietary formats.
DR4	Structurally supported	Identity events, privileged sessions, workflow execution, configuration state, incident handling, and service metrics are designed to produce inspectable evidence across layers.
DR5	Structurally supported	Detection rules, playbooks, incident records, configuration files, and deployment definitions are maintained in exportable and reproducible forms. Controlled redeployment and configuration reconstruction exercises support portability, although a full provider-exit or multi-provider migration scenario was not exhaustively tested.
DR6	Partially structurally supported	Compliance-relevant evidence is embedded in the architecture, but no external audit or formal certification exercise was performed against concrete regulatory controls.
DR7	Architecturally instantiated	The Wazuh–Shuffle–MISP–IRIS pipeline shows that layered security operations can be composed from modular components without relying on a monolithic platform.
DR8	Partially structurally supported	Reproducible deployment, documentation, and automation are implemented, but long-term maintainability, staffing sustainability, and ecosystem resilience require longitudinal observation.

tified in the future work section.

6.3 Theoretical Contributions

The dissertation makes three main theoretical contributions. First, it operationalizes technological sovereignty as a set of concrete architectural criteria for managed security services. In the literature, sovereignty is often discussed at the level of policy, strategy, or infrastructure ownership [43, 22]. Here, sovereignty is translated into inspectability, governability, portability, and reproducibility within the managed service itself. This shift makes sovereignty analytically useful for architecture design rather than merely descriptive of geopolitical preference.

Table 6.2: Assessment of the framework against sovereignty dimensions.

Dimension	Assessment	Basis
Control	Structurally enabled	The framework provides identity governance (DR1), tenant isolation (DR2), and layered composable operations (DR7) that support the client's ability to govern infrastructure and detection logic. The prototype demonstrates these capabilities through FreeIPA, Keycloak, and tenant-scoped configurations.
Transparency	Structurally enabled	End-to-end auditability (DR4) and compliance-by-design evidence (DR6) make service operations, configurations, and decisions inspectable. The prototype produces distributed traceable audit evidence across the implemented layers.
Portability	Structurally enabled	Operational artifacts are maintained in non-proprietary, exportable formats (DR5), and the architecture uses modular, standards-based composition (DR3). Re-deployment and reconstruction activities support a claim of structural portability.
Independence	Partially demonstrated	Modularity (DR3) and reproducibility (DR8) reduce strategic dependence on any single actor. However, full independence requires longitudinal validation of sustainability, component lifecycle governance, and production-scale migration exercises not performed in this work.

Second, the dissertation reframes auditability as a first-class system property rather than as a reporting afterthought. Prior work typically addresses fragments of auditability, such as log retention, SLA verification, or compliance reporting [4, 23, 52]. The proposed framework instead treats auditability as end-to-end traceability spanning identity, configuration, detection logic, orchestration, incident handling, and governance evidence. This is a stronger and more operationally demanding definition, but it is also closer to the level at which trust in managed security services must actually be justified.

Third, the dissertation extends the lock-in discussion from cloud dependence in general to MSSP-specific operational artifacts. The literature already shows that dependency is shaped by technical integration, transfer cost, and compliance constraints [5]. This work adds that, in managed security services, the decisive artifacts include normalized telemetry pipelines, detection rules, orchestration playbooks, case records, and deployment knowledge. By treating these artifacts as portability targets, the dissertation offers a more precise account of how vendor lock-in is created and how it can be reduced in practice.

Taken together, these contributions bridge research streams that are usually treated

separately. The framework links outsourcing theory, SOC architecture, governance and compliance, and sovereignty studies into a single design-oriented model. In DSR terms, the contribution is therefore not only a prototype, but a reference architecture and a set of design propositions that can structure future empirical work on sovereign and auditable managed security services.

6.4 Practical Implications for Managed Security Services

For MSSPs, the most direct implication is that openness and auditability should be designed into the operating model, not added as client-facing reporting features. A provider that wants to reduce structural dependence must expose identity boundaries, preserve exportable rules and workflows, document change history, and maintain evidence that can be independently inspected by the client. In this model, service quality derives from engineering discipline, operational competence, and governance maturity rather than from opacity or proprietary coupling.

For client organizations, the framework provides a concrete basis for procurement and oversight. Instead of evaluating providers primarily through broad SLAs or marketing claims, clients can ask whether detection logic is exportable, whether response workflows are documented, whether incident records are portable, whether privileged actions are attributable, and whether the platform can be reconstructed without provider-specific knowledge. These questions are directly aligned with the sovereignty and accountability pressures highlighted by NIS2, GDPR, and related regulatory developments [22, 14].

For regulators and auditors, the dissertation suggests that managed security assurance should focus less on whether security is outsourced and more on whether outsourced security remains governable. The key issue is not externalization per se, but whether the provider–client relationship preserves verifiable control. An architecture that generates primary evidence across all service layers is materially better aligned with this requirement than one that substitutes dashboards and summary reports for direct inspectability.

These implications also involve trade-offs. A modular open stack reduces strategic dependence, but it increases integration responsibility and requires stronger documentation, lifecycle management, and architectural discipline from the provider. The practical

value of the framework therefore lies not in promising lower complexity, but in relocating complexity from opaque vendor dependence to explicit and auditable engineering choices.

6.5 Limitations

The first limitation concerns evaluation scope. The prototype was designed to demonstrate architectural feasibility, not to serve as a production-grade benchmark. Accordingly, the dissertation does not provide quantitative evidence on detection efficacy, false-positive behavior, response latency, cost efficiency, high-availability performance, or resilience under sustained operational load. These omissions do not invalidate the architectural contribution, but they limit the strength of claims that can be made about operational superiority.

The second limitation concerns deployment realism, particularly DR2 and DR6. The prototype operates in a single-site environment and implements multi-tenancy primarily through logical separation rather than through physically isolated tenant infrastructures. This is sufficient to demonstrate the architecture's internal logic, but it does not fully test the framework under distributed, cross-jurisdictional, or highly heterogeneous client conditions. The same qualification applies to compliance: the framework embeds compliance-relevant evidence, yet no formal external audit was conducted against a concrete control catalog.

The third limitation concerns portability and sustainability claims, particularly DR5 and DR8. The dissertation shows that artifacts are stored in exportable formats and that deployment is reproducible from code-defined definitions, and portability was supported through controlled re-deployment and reconstruction activities. However, the dissertation does not include a full provider-exit exercise, a third-party takeover simulation, or a longitudinal maintenance study. As a result, the claims about lock-in mitigation and sustainability are supported by both structural design and controlled testing, but a complete end-to-end migration across providers remains empirically unvalidated.

The fourth limitation is analytical scope. The implementation is intentionally restricted to open-source components in order to foreground sovereignty, inspectability, and portability. This makes the argument clearer, but it narrows immediate generalizability

to hybrid environments in which organizations may retain proprietary tooling for specific capabilities. As established in the architectural principles (Section 4.2), the framework defines openness as a set of architectural properties rather than as a licensing requirement, and is therefore compatible with proprietary components that satisfy those properties. However, that compatibility was not empirically examined in this prototype.

The fifth limitation concerns evaluation objectivity. The framework was designed and evaluated by the same researcher, which may introduce confirmation bias and limit external validity. Involving independent practitioners, domain experts, or client organizations in the evaluation remains a direction for future work that would strengthen the assessment's credibility and generalizability.

Chapter 7

Conclusions and Future Work

This chapter concludes the dissertation. It first summarizes the main conclusions drawn from the research questions and the contributions of the proposed architecture, and then discusses the limitations of the present evaluation together with the opportunities they open for future work.

7.1 Conclusions

This dissertation addressed the problem of how managed security services can be designed so that outsourcing does not require opaque dependence on a provider's internal stack. The central conclusion is that a sovereign, open, and auditable MSSP architecture is architecturally feasible when sovereignty is operationalized as a design constraint and implemented through explicit requirements for identity governance, modularity, auditability, portability, compliance evidence, and reproducibility.

The research made this conclusion through three cumulative steps. First, the literature review showed that current research is strong in component-level insight but weak in integrated design at the intersection of managed security delivery, governance, and technological sovereignty. Second, the dissertation translated those gaps into eight traceable design requirements and a five-layer reference architecture. Third, the prototype demonstrated that the architecture can be instantiated with currently available open-source technologies in a coherent and self-hosted environment.

Through these steps, the dissertation addressed each of the formulated research ques-

tions. Addressing RQ1, the study established that the structural limitations of prevailing MSSP architectures stem primarily from opaque integration and the tight coupling of proprietary components, which hinder transparency and client-side sovereignty. In response to RQ2, the research defined a set of eight design requirements—ranging from federable identity to end-to-end auditability—that must be fulfilled to unify open architectures with sovereignty-driven governance. Regarding RQ3, the dissertation demonstrated that disparate open-source solutions can be orchestrated into a cohesive and sustainable architecture by establishing explicit trust boundaries, standardized interfaces, and a shared identity substrate. Finally, concerning RQ4, the qualitative evaluation indicated that the proposed framework satisfies the design requirements at the level of architectural demonstration: modularity, auditability, portability, and reproducibility are directly instantiated in the prototype, while multi-tenant isolation, compliance assurance, and long-term sustainability are structurally supported but still require production-scale and longitudinal validation. The evaluation therefore supports a claim of architectural feasibility and structured mitigation of vendor lock-in under representative constraints, rather than a claim of fully validated operational efficacy.

The main contribution of the dissertation is therefore not the claim that open-source tools alone solve the MSSP problem. Rather, it is the demonstration that vendor lock-in and loss of control can be structurally reduced when the service is organized around exportable operational artifacts, inspectable workflows, verifiable evidence, and replaceable components. Taken together, the dissertation's outputs include a structured synthesis of the literature, a traceable requirement set, a five-layer reference architecture, and a prototype-based demonstration. In this sense, the work advances the discussion from abstract calls for digital sovereignty to a concrete architectural model through which sovereignty can be assessed and implemented.

At the same time, the conclusions must remain proportionate to the evidence. The prototype supports the feasibility and internal coherence of the proposed architecture; it does not by itself prove production-scale performance, regulatory sufficiency in every context, or long-term operational sustainability. Even so, the dissertation establishes a defensible foundation for future empirical work and provides a rigorous reference model for organizations seeking alternatives to opaque and tightly coupled managed security service

arrangements.

7.2 Future Work

Future work could extend the dissertation in six directions:

1. Deploy and evaluate the framework in a production-like, multi-tenant environment, including quantitative assessment of detection quality, response latency, service availability, and operational overhead.

2. Conduct formal assurance studies that map the framework's evidence model to concrete regulatory and audit controls under instruments such as NIS2, GDPR, and sector-specific security obligations.

3. Execute controlled migration and provider-exit exercises to test whether operational artifacts, workflows, and governance records can in fact be transferred or reconstructed with limited loss of fidelity.

4. Perform longitudinal studies of maintainability, analyst workload, automation quality, and component ecosystem health in order to validate the framework's sustainability claims over time.

5. Conduct independent expert evaluations with MSSP practitioners, auditors, and client organizations to strengthen external validity and reduce researcher-evaluation bias.

6. Evaluate hybrid instantiations that combine open-source and proprietary components while preserving the framework's open-architecture properties.

These directions would move the framework from architectural demonstration toward externally validated and production-informed evidence.

References

[1] Enoch Agyepong and Cyril Onwubiko. "An Exemplar Incident Response Plan for Security Operations Centre Analysts". In: *Springer Proceedings in Complexity* (2025). Type: Conference paper, pp. 109–127. DOI: 10.1007/978-981-96-0401-2_7.

[2] Enoch Agyepong et al. "A systematic method for measuring the performance of a cyber security operations centre analyst". In: *Computers and Security* 124 (2023). Type: Article. DOI: 10.1016/j.cose.2022.102959.

[3] Fatimah Alaliwat et al. "OTuHunt: An Aggregated Threat Hunting & Intelligence Platform for OT/ICS Environment and MSSP Services". In: *Proceeding - 12th International Conference on Information Technology: Innovation Technologies, ICIT 2025*. Type: Conference paper. 2025, pp. 39–46. DOI: 10.1109/ICIT64950.2025.11049292.

[4] Sultan Ali Alasmari et al. "Protection of Network Security Selector Secrecy in Outsourced Network Testing". In: *Proceedings - International Conference on Computer Communications and Networks, ICCCN*. Vol. 2023-July. Type: Conference paper. 2023. DOI: 10.1109/ICCCN58024.2023.10230113.

[5] Amal Alhosban, Saichand Pesingu, and Krishnaveni Kalyanam. "CVL: A Cloud Vendor Lock-In Prediction Framework". In: *Mathematics* 12.3 (2024). ISSN: 2227-7390. DOI: 10.3390/math12030387.

[6] Meryem Ammi and Yusuf Mohamud Jama. "Cyber Threat Hunting Case Study using MISP". In: *Journal of Internet Services and Information Security* 13.2 (2023). Type: Article, pp. 1–29. DOI: 10.58346/JISIS.2023.I2.001.

[7] Jarl Andreassen et al. “InCReASE: A Dynamic Framework Towards Enhancing Situational Awareness in Cyber Incident Response”. In: *IFIP Advances in Information and Communication Technology* 672 LNBIP (2023). Type: Conference paper, pp. 230–243. DOI: 10.1007/978-3-031-34207-3_15.

[8] Samarjeet Borah et al. “Machine Learning for Security Operations Center (SOC) Optimization”. In: *Lecture Notes in Networks and Systems* 1362 LNNS (2025). Type: Conference paper, pp. 397–409. DOI: 10.1007/978-981-96-5214-3_32.

[9] Robert A. Bridges et al. “Testing SOAR tools in use”. In: *Computers and Security* 129 (2023). Type: Article. DOI: 10.1016/j.cose.2023.103201.

[10] Albert Calvo et al. “A Data-driven Approach for Risk Exposure Analysis in Enterprise Security”. In: *2023 IEEE 10th International Conference on Data Science and Advanced Analytics, DSAA 2023 - Proceedings*. Type: Conference paper. 2023. DOI: 10.1109/DSAA60987.2023.10302480.

[11] Achraf Samir Chamkar, Maleh Yassine, and Gherabi Noreddine. “Security Operations Centers: Use Case Best Practices, Coverage, and Gap Analysis Based on MITRE Adversarial Tactics, Techniques, and Common Knowledge”. In: *Journal of Cybersecurity and Privacy* 4.4 (2024). Type: Article, pp. 777–793. DOI: 10.3390/jcp4040036.

[12] Achraf Samir Chamkar, Maleh Yassine, and Gherabi Noreddine. “SOC Analyst Performance Metrics: Towards an optimal performance model”. In: *EDPACS* 68.3 (2023). Type: Article, pp. 16–29. DOI: 10.1080/07366981.2023.2259046.

[13] Achraf Samir Chamkar, Maleh Yassine, and Gherabi Noreddine. “SOC Use Cases: Deployment, Challenges and Gap Assessment Based on MITRE ATT&CK”. In: *Lecture Notes in Networks and Systems* 1312 LNNS (2025). Type: Conference paper, pp. 212–223. DOI: 10.1007/978-3-031-94620-2_19.

[14] Maxime Compastié et al. “PALANTIR: An NFV-Based Security-as-a-Service Approach for Automating Threat Mitigation”. In: *Sensors* 23.3 (2023). Type: Article. DOI: 10.3390/s23031658.

[15] Cyber Expert. *What is an MSSP – Managed Security Service Provider?* HTTPCS. May 20, 2024. URL: https://blog.httpcs.com/en/what-is-an-mssp-managed-security-service-provider/ (visited on 05/14/2026).

[16] Pedro Escaleira et al. "Moving Target Defense for the cloud/edge Telco environments". In: *Internet of Things (Netherlands)* 24 (2023). Type: Article. DOI: 10.1016/j.iot.2023.100916.

[17] Luciano Floridi. "The Fight for Digital Sovereignty: What It Is, and Why It Matters, Especially for the EU". In: *Philosophy & Technology* 33.3 (Sept. 2020), pp. 369–378. ISSN: 2210-5441. DOI: 10.1007/s13347-020-00423-6.

[18] Joonas Forsberg and Tapio K. Frantti. "Technical performance metrics of a security operations center". In: *Computers and Security* 135 (2023). Type: Article. DOI: 10.1016/j.cose.2023.103529.

[19] Xing Gao et al. "Information security outsourcing in a resource-sharing environment: The impacts of attack modes". In: *Journal of the Operational Research Society* 75.6 (2024). Type: Article, pp. 1092–1110. DOI: 10.1080/01605682.2023.2233550.

[20] Alan R. Hevner et al. "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1 (2004), pp. 75–105. DOI: 10.2307/25148625.

[21] Chadni Islam, Muhammad Ali Ali Babar, and Surya Nepal. "Design and Generation of a Set of Declarative APIs for Security Orchestration". In: *IEEE Transactions on Services Computing* 17.1 (2024). Type: Article, pp. 127–141. DOI: 10.1109/TSC.2023.3336666.

[22] Mazaher Kianpour, Peter Alexander Earls Davis, and Iwona Maria Windekilde. "Digital sovereignty in practice: analyzing the EU's NIS2 directive". In: *International Journal of Information Security* 24.4 (2025), p. 167. DOI: 10.1007/s10207-025-01090-4.

[23] Juan Miguel López Velásquez et al. "Systematic review of SIEM technology: SIEM-SC birth". In: *International Journal of Information Security* 22.3 (2023). Type: Article, pp. 691–711. DOI: 10.1007/s10207-022-00657-9.

[24] Shaswata Mitra et al. “LocalIntel: Generating Organizational Threat Intelligence from Global and Local Cyber Knowledge”. In: *Lecture Notes in Computer Science* 15533 LNCS (2025). Type: Conference paper, pp. 63–78. DOI: 10.1007/978-3-031-87496-3_5.

[25] Erik Etsushi Miyashita and Luis Hernan Contreras Pinochet. “Strategic Selection of SOC Architectures in Financial SMEs: Applying CRITIC and WASPAS to Support Cybersecurity Decisions”. In: *Procedia Computer Science*. Vol. 266. Type: Conference paper. 2025, pp. 505–513. DOI: 10.1016/j.procs.2025.08.064.

[26] Mohsen Mohamad Hata et al. “A Log Aggregation Design Criteria for Robust SIEM (Security Information and Event Management) in Enhancing Threat Detection”. In: *8th International Conference on Recent Advances and Innovations in Engineering: Empowering Computing, Analytics, and Engineering Through Digital Innovation, ICRAIE 2023*. Type: Conference paper. 2023. DOI: 10.1109/ICRAIE59459.2023.10468438.

[27] K. Mugeshwaran et al. “An Integrated Approach to AI-Enhanced Security Information and Event Management”. In: *International Conference on Computational Robotics, Testing and Engineering Evaluation, ICCRTEE 2025*. Type: Conference paper. 2025. DOI: 10.1109/ICCRTEE64519.2025.11053120.

[28] Arunabha Mukhopadhyay and Swati Jain. “A framework for cyber-risk insurance against ransomware: A mixed-method approach”. In: *International Journal of Information Management* 74 (2024). Type: Article. DOI: 10.1016/j.ijinfomgt.2023.102724.

[29] Wesley Murisa and Marijke Coetzee. “A Conceptual SOC Framework for Air Traffic Management Systems”. In: *International Conference on Information Systems Security and Privacy*. Vol. 1. Type: Conference paper. 2025, pp. 275–282. DOI: 10.5220/0013174200003899.

[30] Derek L. Nazareth, Jae J. Choi, and Thomas L. Ngo-Ye. “Investing in security-as-a-service for e-commerce infrastructure by small and medium enterprises: a Monte Carlo approach”. In: *Journal of Systems and Information Technology* 26.2 (2024). Type: Article, pp. 257–275. DOI: 10.1108/JSIT-04-2023-0071.

[31] Nombeko Ntingi, Jaco du Toit, and Basie H.(Basie) von Solms. "A Community Security Operations Centre (ComSOC) Model for SMMEs in Developing Countries". In: *Lecture Notes in Networks and Systems* 1017 LNNS (2024). Type: Conference paper, pp. 443–456. DOI: 10.1007/978-3-031-62277-9_29.

[32] Nombeko Ntingi, Jaco Du Toit, and Sebastian Von Solms. "Towards an active cyber defence framework for smmes in developing countries". In: *European Conference on Information Warfare and Security, ECCWS*. Vol. 2023-June. Type: Conference paper. 2023, pp. 341–348. DOI: 10.34190/eccws.22.1.1053.

[33] Arssy Hasyir Nursidiq and Charles Lim. "Cyber Threat Hunting to Detect Unknown Threats in the Enterprise Network". In: *Proceedings - 2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity: Cryptography and Cybersecurity: Roles, Prospects, and Challenges, ICoCICs 2023*. Type: Conference paper. 2023, pp. 303–308. DOI: 10.1109/ICoCICs58778.2023.10277438.

[34] Håvard Jakobsen Ofte and Sokratis K. Katsikas. "Understanding situation awareness in SOCs, a systematic literature review". In: *Computers and Security* 126 (2023). Type: Article. DOI: 10.1016/j.cose.2022.103069.

[35] Ciprian Ionut Păduraru, Catalina Camelia Patilea, and Alin Ştefănescu. "CyberGuardian 2: Integrating LLMs and Agentic AI Assistants for Securing Distributed Networks". In: *International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE - Proceedings*. Type: Conference paper. 2025, pp. 660–667. DOI: 10.5220/0013406000003928.

[36] Matthew J Page et al. "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews". In: *BMJ* 372 (2021). DOI: 10.1136/bmj.n71. eprint: https://www.bmj.com/content/372/bmj.n71.full.pdf. URL: https://www.bmj.com/content/372/bmj.n71.

[37] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

[38] Adamantini Peratikou et al. “ATHENA: A Federated Architecture for Cross-Border Cybersecurity Operations and Situational Awareness”. In: *Proceedings of the 2025 IEEE International Conference on Cyber Security and Resilience, CSR 2025*. Type: Conference paper. 2025, pp. 1037–1042. DOI: 10.1109/CSR64739.2025.11129994.

[39] Julia Pohle and Thorsten Thiel. “Digital sovereignty”. In: *Internet Policy Review* 9 (Dec. 2020). DOI: 10.14763/2020.4.1532.

[40] Davy Preuveneers et al. “On the Use of AutoML for Combating Alert Fatigue in Security Operations Centers”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 14399 LNCS (2024). Type: Conference paper, pp. 609–627. DOI: 10.1007/978-3-031-54129-2_36.

[41] Moumita Das Purba, Bill Chu, and Will French. “Towards Automated and Explainable Threat Hunting with Generative AI”. In: *Proceedings - 2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2025*. Type: Conference paper. 2025, pp. 664–677. DOI: 10.1109/DSN64029.2025.00067.

[42] *Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements (Cyber Resilience Act)*. http://data.europa.eu/eli/reg/2024/2847/oj. Text with EEA relevance; amending Regulations (EU) No 168/2013, (EU) No 2019/1020 and Directive (EU) 2020/1828. 2024.

[43] Huw Roberts. “Digital sovereignty and artificial intelligence: a normative approach”. In: *Ethics and Information Technology* 26.4 (2024), p. 70. DOI: 10.1007/s10676-024-09810-5.

[44] Mehdi Saadallah, Abbas Shahim, and Svetlana N. Khapova. “Harmonizing paradoxical tensions in SOCs: A Strategic model for integrating AI, automation, and human expertise in cyber defense and incident response”. In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. Type: Conference paper. 2025, pp. 6171–6180. DOI: 10.24251/hicss.2025.738.

[45] Muhammad Sheeraz et al. “Effective Security Monitoring Using Efficient SIEM Architecture”. In: *Human-centric Computing and Information Sciences* 13 (2023). Type: Article. DOI: 10.22967/HCIS.2023.13.023.

[46] Dimitri Alexandre da Silva, José Luís Costa, and João Rafael Almeida. “Infrastructural Challenges and Good Practices in a Security Operation Center”. In: *OpenAccess Series in Informatics*. Vol. 120. Type: Conference paper. 2024. DOI: 10.4230/OASIcs.SLATE.2024.13.

[47] Stefano Solari et al. “Cyber-DRIVE: Dynamic Risk Versatile Engine for Cyber Security Risk Analysis, Situational Awareness and Incident Response”. In: *CEUR Workshop Proceedings*. Vol. 3962. Type: Conference paper. 2025. URL: https://ceur-ws.org/Vol-3962/paper63.pdf.

[48] Zarrin Tasnim Sworna, Chadni Islam, and Muhammad Ali Ali Babar. “APIRO: A Framework for Automated Security Tools API Recommendation”. In: *ACM Transactions on Software Engineering and Methodology* 32.1 (2023). Type: Article. DOI: 10.1145/3512768.

[49] Issam Taqafi, Maleh Yassine, and Karim Ouazzane. “A MATURITY CAPABILITY FRAMEWORK FOR SECURITY OPERATION CENTER”. In: *EDPACS* 67.3 (2023). Type: Article, pp. 21–38. DOI: 10.1080/07366981.2023.2159047.

[50] Jack Laurie Tilbury and Stephen V. Flowerday. “The Rationality of Automation Bias in Security Operation Centers”. In: *Journal of Information Systems Security* 20.2 (2024). Type: Article, pp. 89–109. URL: https://api.semanticscholar.org/CorpusID:271163524.

[51] Risto Vaarandi and Alejandro Guerra-Manzanares. “Stream clustering guided supervised learning for classifying NIDS alerts”. In: *Future Generation Computer Systems* 155 (2024). Type: Article, pp. 231–244. DOI: 10.1016/j.future.2024.01.032.

[52] Ali Vedadi, Nita G. Brooks, and Timothy H. Greer. “Investigating the role of internal security resources in post-adoption satisfaction with the Security-as-a-Service model: an organizational mindfulness perspective”. In: *Journal of Enterprise Information Management* 36.6 (2023). Type: Article, pp. 1583–1609. DOI: 10.1108/JEIM-07-2022-0227.

[53] Yong Wu et al. “Managing partial outsourcing on information security in the presence of security externality”. In: *Expert Systems with Applications* 246 (2024). Type: Article. DOI: 10.1016/j.eswa.2023.123003.

[54] Eleftherios Zacharioudakis and Elena Kakoulli. “A Scalable Cybersecurity Model for Shared SOCs in SMEs”. In: *Lecture Notes in Networks and Systems* 1487 LNNS (2025). Type: Conference paper, pp. 257–269. DOI: 10.1007/978-3-031-95652-2_22.

[55] Kamal Zidan, Abu Alam, and Qublai K. Ali Mirza. “A Grid-Matrix Based on Industry Needs to Evaluate Automation in Security Operations Centre (SOC)”. In: *Proceedings - 2024 11th International Conference on Future Internet of Things and Cloud, FiCloud 2024*. Type: Conference paper. 2024, pp. 16–20. DOI: 10.1109/FiCloud62933.2024.00011.