



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Using AI to Continuously Monitor Cloud Workloads, Detect Threats, and Respond in Real Time

Research Supervisor: Prof. Amit Naik

Student Name: Md Ashraful Islam

Student ID: 2436063

SUBJECT: MSC IN CYBER SECURITY & DIGITAL FORENSICS

Table of Contents

1. Introduction.....	6
1.1 Context and Background.....	6
1.2 Research Problem.....	8
1.3 Research Aim and Objectives	9
Research Aim:	9
Research Objectives:	9
1.4 Significance of the Research.....	9
2. Literature Review	11
2.1 Introduction to the Literature Review	11
2.2 Thematic Overview	11
Hybrid AI and Anomaly Detection	11
Automated Threat Detection and Response	11
Cloud Integration and Scalability	12
Runtime Security and Cloud-Native Telemetry	12
2.3 Key Contributions and Gaps.....	12
Key Contributions (from literature and practice)	12
Gaps	12
2.4 Theoretical and Technical Frameworks	13
2.5 Relation to the Current Study.....	13
2.6 Literature Summary	13
2.7 Background	13
2.8 Survey Literature: From Data Mining Pipelines to Cloud-Native Defenses.....	14
2.9 Boosted Trees, Ensembles, and the Engineering of Tabular Cloud Features	15
2.10 Adversarial ML, Robustness, and the Security of the Detector Itself	15
2.11 Secure-by-Design Software and the Upstream Causes of Runtime Alerts	15
2.12 Benchmarks, Leakage, and the Epistemology of “Ground Truth” in IDS Research	15
2.13 Kubernetes, Control Planes, and the Shift of “Perimeter” to Identity	16
2.14 Risk Management Standards as a Bridge Between ML Metrics and Business Outcomes .	16
2.15 Practical ML Textbooks and Reproducible Experimental Habits	16
2.16 Federated and Privacy-Preserving Learning in Security Analytics (Emerging)	16
2.17 Time-Series and Sequence Modeling for Lateral Movement Narratives.....	17
2.18 Industry Reports and the Sociology of “AI SOC” Claims	17

2.19 OWASP Perspectives on ML in Production Applications (Adjacent Lessons)	17
2.20 Synthesis: A Layered Evidence Stack for Cloud AI Security Claims	17
2.21 Graph Learning and Relational Telemetry (Conceptual Positioning)	18
2.22 Human–Computer Interaction and Alert Design as Research Adjacent to ML	18
2.23 From SIEM to XDR, CNAPP, and Cloud Detection and Response (CDR)	18
2.24 MLOps, Model Lifecycle, and Operationalizing ML in Security Operations	19
2.25 Identity-Centric Threats, API Abuse, and Control-Plane Telemetry	19
2.26 Detection Engineering, Declarative Rules, and the Case for Hybrid Pipelines	20
2.27 Recent Contributions (2022–2025) and Positioning of the Present Study	20
3. Research Design and Methodology	21
3.1 Research Design	21
3.2 Research Methods and Materials	22
Research strategy:	23
Data sources:	23
Tools and software:	23
Hardware resources:	23
Ethical considerations:	23
3.3 Evaluation Protocol and Reporting Standards	24
3.4 Threat Model, Assumptions, and Scope Boundaries	24
3.5 Data Governance, Minimization, and Synthetic Generation	24
3.6 Risk Management Alignment (ISO 27001 / NIST CSF Narrative)	25
3.7 Validation Strategy: Triangulation Across Evidence Types	25
3.8 Toolchain Reproducibility and Environment Pinning	25
3.9 Professional Development Link: Communication, Ethics, and Teamwork	25
3.10 Methodological Limitations (Declared Upfront)	25
3.11 Experimental Factors, Blocking, and Repeated Measures	26
3.12 Baseline Hierarchy and “No Free Lunch” Controls	26
3.13 Calibration, Thresholding, and Cost-Sensitive Decision Theory	26
3.14 Feature Schema Versioning and Interface Contracts for ML	26
3.15 Load Testing, Saturation, and Latency SLO Methodology	27
3.16 Data Documentation Cards and Provenance for Security Datasets	27
3.17 Statistical Testing and Practical Significance Boundaries	27

3.18 Red-Team Orientation and Negative Controls (Methodological).....	27
3.19 Integration Testing Methodology for Hybrid Pipelines.....	28
3.20 Traceability from Requirements to Metrics (Design Science Link).....	28
3.21 Threat-Driven Test Cases and Expected Detections	28
3.22 Logging, Audit, and Forensic Reconstruction of Model Decisions	28
3.23 Computational Budgeting and Carbon-Aware Experiment Planning	28
3.24 Handling Missing Data, Schema Evolution, and Default Paths.....	29
3.25 Peer Review Simulation: Independent Replication Package	29
3.26 Ablation Granularity: Component Contributions Beyond “Hybrid vs Not”	29
3.27 Incident-Style Tabletop Evaluation (Qualitative Method)	29
3.28 Final Methodological Principle: Scope Honesty	29
3.29 Confounding Variables in End-to-End Pipeline Experiments.....	30
3.30 Reporting Negative Results and “Failed” Interventions	30
4. Expected Outcome	31
4.1 Deliverables Mapped to Module Expectations	31
4.2 Success Criteria and Non-Goals	31
4.3 Transferability to Industry Roles.....	32
4.4 Knowledge Contribution Statement	32
5. Timeline.....	33
6. Benefits and Impact.....	34
7. Challenges and Limitations.....	35
7.1 Organizational and Process Limitations	35
7.2 Measurement Limitations and Unknown Ground Truth	35
7.3 Third-Party and Shared Responsibility Boundaries	35
7.4 Legal and Employment Law Considerations in Monitoring	36
7.5 Model Supply Chain and Dependency Risk.....	36
8. Future Trends.....	37
8.1 Federated and Cross-Organization Analytics Under Policy Constraints	37
8.2 Standardization of Security Telemetry and Semantic Interoperability	37
8.3 Hardware-Rooted Trust and Runtime Integrity	37

8.4 Autonomous Defense and the Need for Formal Verification Narratives	37
8.5 Education and Workforce Development	38
9. Conclusion	39
9.1 Reflective Summary for Professional Development	39
9.2 Final Positioning Relative to the SIB/ACSP Artifact.....	39
References	40
Appendices	46
Appendix A: Implementation Context (SIB and ACSP)	46
A.1 SIB: SIEM-in-a-Box as a Reproducible Telemetry and Analytics Stack.....	46
A.2 ACSP: AI Cloud Security Platform as a Research-Oriented Detection Pipeline	46
A.3 Event Schema, Rule Baselines, and the Role of Hybrid Detection	47
A.4 Evaluation Hooks: Metrics, Latency, and Responsible Reporting.....	47
A.5 Staged Autonomous Response: From Analyst Text to Policy-Gated Actions.....	48
A.6 Privacy, Retention, and Compliance-Oriented Engineering	48
A.7 Multi-Cloud Telemetry Shapes and Domain Shift	48
A.8 Operational Observability: Prometheus Metrics and Runbooks.....	48
Appendix B: Research Ethics, Validity, and Professional Practice	49
Appendix C: Extended Methodological Notes (Sampling, Threat Model, and Reproducibility) 49	
C.1 Threat Modeling as a Companion Artifact	49
C.2 Sampling and Label Integrity.....	49
C.3 Reproducibility and Open Science Practices	49
C.4 Continuous Monitoring as an Engineering Contract	50
C.5 Zero Trust, Identity, and the Limits of Workload-Only Telemetry	50
C.6 Software Supply Chain Security and the CI/CD Attack Surface	50
C.7 Concept Drift, Covariate Shift, and the Need for Monitoring the Monitor	51
C.8 Human Factors: Alert Fatigue, Trust Calibration, and Analyst Workflows.....	51
C.9 Comparative Perspective: IDS Benchmarks versus Runtime Security Events	51
C.10 Cost, Carbon, and Efficiency: Making the Economic Argument Explicit.....	51
C.11 Incident Response Integration: From Detection to Measurable Containment.....	52
C.12 Privacy Engineering Beyond GDPR Headlines	52
C.13 Open Source Ecosystem Dependencies and Maintenance Risk	52
C.14 Generative AI in the SOC: Opportunities and Epistemic Risks	52
C.15 Closing Methodological Synthesis	52
C.16 Measurement Protocol for Queueing and Backpressure Experiments	53
C.17 Model Risk Management: Rollback, Canary Releases, and Shadow Mode	53
C.18 Feature Store Discipline (Lightweight) and Training–Serving Skew	53
C.19 Ethics Review Checklist for Student-Scale Operational Telemetry	53
C.20 Comparative Review of Tree Ensembles versus Deep Models for Security Logs.....	53
C.21 Linking MITRE Coverage to Evaluation Slices.....	54
C.22 Handoff Criteria: When Results Are “Publication-Ready” for the Module.....	54
C.23 Sensitivity Analysis for Threshold and Prior Assumptions.....	54



C.24 Documentation of Randomness and Deterministic Replay	54
C.25 Security of the Evaluation Harness Itself	54
C.26 Contrasting “Research Metrics” with “Operational Readiness Checklists”	54
C.27 Optional Extensions: Graph Features, LLM Assistants, and Scope Control	55

1. Introduction

1.1 Context and Background

Cloud computing has become one of the most transformative technologies of the twenty-first century, enabling organizations to deploy, scale, and manage applications with unprecedented flexibility and cost efficiency. Cloud workloads—including virtual machines (VMs), containers, microservices, and serverless functions—run critical business processes and handle vast amounts of sensitive data. However, this shift has also introduced major security challenges due to distributed architectures, dynamic scaling, ephemeral infrastructure, and multi-tenant environments where isolation failures can have disproportionate impact. Traditional security mechanisms such as perimeter firewalls, static network segmentation, and signature-only intrusion detection are increasingly inadequate for detecting sophisticated attacks that exploit misconfiguration, stolen credentials, and supply-chain weaknesses in real time [1], [2].

Artificial intelligence (AI) and machine learning (ML) have emerged as pivotal technologies in cybersecurity, especially within cloud and cloud-native environments. AI techniques—such as anomaly detection, behavioral profiling, sequence modeling, and predictive analytics—enable systems to examine large volumes of cloud telemetry (logs, metrics, audit records, identity events, and runtime signals) and detect irregular patterns indicative of threats. Unlike purely static rule-based systems, ML models can learn from historical and streaming data, adapt to evolving baselines, and support continuous monitoring that aligns with the dynamic nature of cloud workloads. When responsibly deployed, AI-powered solutions can also support automated triage and response orchestration, reducing reliance on purely manual intervention and narrowing the window between detection and containment [3], [4].

The integration of AI in cloud security leverages the computational elasticity of modern cloud platforms. Real-time or near-real-time analytics pipelines, coupled with scalable storage and model-serving patterns, make it increasingly feasible to monitor distributed workloads for malicious or high-risk activity and to trigger mitigation workflows. This shift participates in a broader movement from reactive, ticket-driven security operations toward proactive, data-driven defense—often discussed under labels such as Cloud Detection and Response (CDR), Extended Detection and Response (XDR), and security analytics platforms [5], [6].

Recent research and industry practice indicate that hybrid systems—which combine deterministic rules, threat intelligence, and statistical or deep learning components—can improve detection utility by balancing interpretability with adaptability. Such systems

ingest heterogeneous signals (network flows, host telemetry, Kubernetes audit logs, cloud control-plane APIs) and correlate them to reduce noise and improve analyst focus [7], [8]. Nevertheless, the same literature underscores persistent gaps: explainability, privacy compliance, adversarial manipulation of models, and the operational cost of continuous inference at scale.

This research project explores developments in AI-driven threat detection and response mechanisms for cloud infrastructures and proposes a structured approach to achieving continuous, security-relevant monitoring with evaluation rigor suitable for academic reporting. Where relevant, the discussion connects high-level research aims to a concrete reference implementation—the **SIB (SIEM-in-a-Box)** ecosystem and its **ACSP (AI Cloud Security Platform)** research stack—so that architectural claims are grounded in reproducible engineering artifacts rather than abstraction alone.

1.2 Research Problem

Cloud environments generate an enormous volume of workload and telemetry data, making manual monitoring and traditional security mechanisms insufficient for effective threat detection and response when used in isolation. Manual approaches often result in delayed detection, slow response times, and high operational costs. Moreover, the heterogeneity of cloud platforms—from hypervisors and VMs to container orchestration systems (notably Kubernetes) and identity-centric control planes—compounds security challenges because attackers can pivot across layers (workload, network, identity, and data plane) using legitimate interfaces [4], [9].

As cloud infrastructures scale dynamically, attack surfaces expand in ways that are difficult to inventory continuously: short-lived containers, auto-scaling groups, cross-account roles, and third-party integrations introduce drift that static inventories cannot track. These conditions increase exposure to advanced persistent threats (APTs), insider misuse, cryptojacking, and distributed denial-of-service (DDoS) attacks—often orchestrated through automation and commodity tooling [10].

The main research problem addressed by this study is:

How can AI be employed to continuously monitor cloud workloads, detect threats in real time, and autonomously respond to mitigate potential damage with high detection accuracy and minimal false positives?

This formulation intentionally binds together three dimensions that are often treated separately in literature: (1) *continuous monitoring* as a systems problem (ingestion, normalization, retention, and latency), (2) *detection* as a statistical learning and evaluation problem (labels, metrics, drift, and domain shift), and (3) *response* as a governance and safety problem (policy, blast radius, auditability, and human oversight).

Current challenges include:

- **Inefficiency of traditional rule-based detection systems** in analyzing large, high-velocity data streams without complementary prioritization and correlation [11], [12].
- **Difficulty distinguishing benign anomalies from malicious behavior** without adaptive baselines, contextual features, and careful evaluation protocols that avoid overfitting to synthetic conditions [13], [14].

- **Computational overhead** associated with processing and analyzing streaming telemetry from multi-cloud environments, especially when deep models are deployed without engineering constraints [15], [16].
- **Designing AI systems that balance detection performance with scalability and low latency**, including failure modes under overload, queueing, and backpressure [17], [18].

1.3 Research Aim and Objectives

Research Aim:

To develop an AI-powered framework that continuously monitors cloud workloads, detects security threats in real time, and autonomously responds to mitigate detected attacks—while documenting evaluation methodology, limitations, and ethical constraints in a manner suitable for independent replication.

Research Objectives:

1. To survey and analyze the state of the art in AI-enabled cloud security technologies and real-time threat detection mechanisms, including hybrid detection strategies and operational deployment patterns.
2. To design an architecture for continuous cloud workload monitoring using machine learning and deep learning models, explicitly separating ingestion, feature extraction, model inference, storage, and response orchestration.
3. To evaluate suitable AI algorithms for threat detection with emphasis on accuracy, latency, false-positive behavior, and resource efficiency—using both controlled simulation and comparative benchmark perspectives where feasible.
4. To propose autonomous response strategies powered by AI (and policy-controlled automation) that can remediate or escalate threats without extensive human intervention, while addressing safety, reversibility, and accountability.
5. To validate the proposed framework using simulated cloud telemetry data, benchmark-aligned datasets where applicable, and—optionally—runtime telemetry paths consistent with container security monitoring (e.g., syscall-derived alerts) when executed on appropriate Linux environments.

1.4 Significance of the Research

This research is significant because it addresses limitations of purely manual or purely signature-driven cloud security operations by articulating an integrated framework: continuous telemetry, model-assisted scoring, measurable evaluation, and staged

automation. By combining AI-driven analytics with real-time monitoring and controlled automated response, the study contributes to the practical question of how organizations can improve resilience against evolving cyber threats without unbounded trust in opaque models [19], [20].

Practically, organizations adopting cloud services may benefit from improved security postures, reduced breach impacts, and lowered operational costs stemming from automated triage—provided governance, logging, and rollback mechanisms are treated as first-class requirements [21], [22]. From a scholarly perspective, the study contributes to intersections of cloud computing, machine learning, and cybersecurity by emphasizing evaluation design (thresholds, latency distributions, leakage avoidance) rather than reporting accuracy alone.

Potential applications include cloud security operation centers (SOCs), DevSecOps pipelines, and cloud-native security platforms that require near-real-time threat detection and mitigation support. The work also aligns with industry narratives around Kubernetes security posture management (KSPM), runtime threat detection, and detection-as-code practices [23], [24].

Finally, the significance of the work for the present module is integrative: it connects research methods (DSR, evaluation design, ethics) with professional communication (clear architecture narratives, defensible claims) and with technical execution (a reference stack that can be demonstrated). That integration is precisely what modern security employers expect from early-career practitioners who must translate between executives, developers, and analysts without losing scientific honesty.

2. Literature Review

2.1 Introduction to the Literature Review

This section examines current research and developments in AI-based cloud security, focusing on real-time threat monitoring, anomaly detection models, automated response mechanisms, and the operational realities of deploying analytics on streaming telemetry. The review spans hybrid AI systems, machine learning-based intrusion detection, and frameworks for scalable security analytics in distributed cloud environments.

The review intentionally surveys literature from approximately 2022 to 2025 for recent trends, while anchoring foundational concepts (anomaly detection evaluation, streaming systems, and security operations) in canonical sources where necessary. The objective is not exhaustive bibliography coverage, but thematic synthesis adequate to justify design decisions and to identify gaps the proposed framework explicitly addresses.

2.2 Thematic Overview

Hybrid AI and Anomaly Detection

Recent research highlights the integration of hybrid AI systems that combine classical detection techniques with machine learning to improve cloud threat detection accuracy and operational utility. Hybrid approaches can leverage rules for known-bad patterns while using statistical models for novelty detection, thereby addressing part of the “known vs unknown” spectrum [7]. Feature optimization, representation learning, and cost-sensitive learning remain active themes because class imbalance and rare-event settings dominate security datasets [25].

From a systems perspective, anomaly detection in operational telemetry must contend with non-stationarity: seasonal traffic, deployments, and configuration changes shift baseline behavior. Consequently, adaptive baselines, online updates, and human-in-the-loop feedback are recurring mitigation strategies—though online learning introduces risks such as poisoning and concept drift mishandling [26], [27].

Automated Threat Detection and Response

Studies emphasize automating both detection and response, but with increasing attention to guardrails. Gautam’s work on AI-automated cyber defense mechanisms illustrates how ML and deep learning can reduce incident response times when integrated into orchestrated workflows [11]. Parallel industry guidance stresses that “autonomy” in security automation is rarely unconditional: approvals, rate limits, simulation/dry-run modes, and immutable audit trails are standard expectations for production playbooks [28], [29].

Cloud Integration and Scalability

Multi-cloud and multi-tenant challenges are increasingly addressed in literature and practice. Kumari's research discusses integrating AI-driven anomaly detection into agile cloud-based platforms and DevOps pipelines, emphasizing continuous integration and continuous delivery (CI/CD) contexts where rapid change is the norm [12]. Scalability concerns map directly to distributed stream processing, buffering, and observability backpressure patterns [18].

Runtime Security and Cloud-Native Telemetry

Containerized workloads introduce distinctive telemetry sources: Linux syscall monitoring, Kubernetes audit logs, admission control decisions, and service mesh signals. Runtime security engines such as Falco exemplify syscall-level monitoring approaches that generate high-fidelity events for container behavior [24]. These signals differ from network intrusion datasets (e.g., NSL-KDD, UNSW-NB15) in distribution and semantics; therefore, claims of model performance must be interpreted with domain-shift caution when models trained on packet features are applied to syscall-derived narratives [14].

2.3 Key Contributions and Gaps

Key Contributions (from literature and practice)

- **High detection accuracy under controlled conditions:** Random forests, gradient boosting, and deep neural networks can achieve strong classification metrics on curated intrusion detection datasets when labels and splits are well constructed [30], [31].
- **Automated incident response potential:** SOAR platforms demonstrate reduced mean time to resolution when repetitive containment actions are automated with policy constraints [11].
- **Real-time scalability patterns:** Microservices-style decomposition (ingest, enrich, detect, store, notify) supports horizontal scaling and isolation of failure domains [32].
- **Emerging multi-agent architectures:** Agentic and multi-agent frameworks propose specialized agents for monitoring, triage, and response coordination—raising research questions about verification, safety, and cost [8].

Gaps

- **Explainability and ethics:** Many high-performing models are difficult to interpret; automated enforcement amplifies concerns about accountability, mistaken blocks, and bias in alert prioritization [20], [33].
- **Resource constraints:** Real-time deep learning at line rate is often expensive; edge inference, model distillation, and selective triggering are common mitigations [15].
- **Data privacy compliance:** Continuous monitoring intersects with GDPR, sector-specific regulations, and organizational policy on data minimization and retention [34], [35].

- **Evaluation realism:** Academic metrics on static datasets do not automatically translate to production efficacy; operational evaluation requires latency distributions, alert fatigue measures, and incident replay studies [36].

2.4 Theoretical and Technical Frameworks

AI security systems in cloud environments commonly employ supervised and unsupervised learning models. Supervised models (e.g., random forests, gradient boosting) are effective when labeled examples exist for attack families and benign operational classes. Unsupervised and self-supervised approaches (e.g., clustering, autoencoders, density models) are often used for novelty detection where labels are incomplete [13].

Multi-agent and modular architectures propose distributed “agents” for specialized monitoring tasks, aligning with microservice operational models [8]. From a technical standpoint, the pipeline architecture pattern—ingestion, normalization, enrichment, detection, storage, notification—mirrors modern security analytics platforms and provides a stable scaffold for inserting ML steps without collapsing observability [18].

2.5 Relation to the Current Study

The current research builds on these findings by proposing a comprehensive architecture that integrates monitoring, detection, and staged response. Unlike studies that evaluate a single model in isolation, this study emphasizes an end-to-end pipeline with explicit evaluation hooks (thresholds, confusion-derived metrics, latency auditing) and acknowledges domain shift between benchmark tabular data and runtime security telemetry.

The study also relates to an implementation pathway: the **SIB** repository provides a reproducible “SIEM-in-a-Box” stack oriented around Falco and observability backends, while **ACSP** provides a research-oriented sub-platform for AI-assisted scoring, metrics export, and policy-gated response experiments—supporting the thesis narrative with engineering traceability.

2.6 Literature Summary

In summary, AI-driven cloud security research has progressed significantly, demonstrating high-performance threat detection and automated response capabilities under defined conditions. However, challenges remain in deployment efficiency, model explainability, compliance alignment, and evaluation realism. These gaps justify the current research framework and motivate explicit reporting of limitations, threat models, and ethical constraints.

2.7 Background

Cloud computing has become the backbone of modern digital infrastructure, offering scalable on-demand resources such as virtual machines, containers, serverless functions, APIs, and identity

services. Rapid adoption has expanded the attack surface: misconfigurations, excessive permissions, and software vulnerabilities remain dominant breach contributors [37].

Traditional security tools that rely on static rules, signatures, and manual triage struggle to keep pace with the speed, scale, and complexity of cloud environments. In response, organizations increasingly adopt AI to enhance cloud security posture by enabling continuous monitoring and automated threat response—while simultaneously demanding transparency for regulated environments [2].

AI-powered systems ingest large volumes of telemetry—including logs, metrics, network flows, user activity, and API interactions—and apply ML and behavioral analytics to distinguish normal behavior from anomalies that may indicate threats. These models can detect patterns that pure signature systems miss, such as lateral movement sequences or anomalous privilege use, because they can incorporate contextual features and adapt baselines—subject to careful validation [4].

A key advantage claimed for AI in cloud security is faster detection and response. Continuous monitoring can reduce dwell time when paired with reliable alerting and controlled containment actions [28]. Technologies marketed as Cloud Detection and Response (CDR) emphasize centralized visibility and correlation across multi-cloud architectures [5].

Overall, the fusion of AI with cloud security represents a shift from purely reactive defense toward proactive and adaptive protection—provided that human oversight, policy, and measurement discipline remain central rather than optional [38].

Complementary classical references in intrusion detection—early threat monitoring perspectives, signature engines such as Snort-family ecosystems, and honeypot literature—remain useful for understanding how today’s “AI SOC” narratives inherit decades of operational lessons about noise, evasion, and attacker adaptation [39], [40], [41]. Modern cloud telemetry is faster and more heterogeneous, but the underlying tension persists: defenders must optimize under uncertainty with incomplete visibility.

2.8 Survey Literature: From Data Mining Pipelines to Cloud-Native Defenses

Broader surveys of data mining and machine learning for intrusion detection provide a useful map of algorithm families, feature engineering traditions, and evaluation pitfalls that predate the current wave of cloud-native platforms [61]. Although many surveyed methods were validated on network packet or flow features, the organizational lessons—class imbalance, cost-sensitive learning, and the coupling between preprocessing and reported accuracy—transfer directly to log-centric and syscall-centric cloud pipelines [13], [25]. What changes in cloud settings is not the mathematics of classification alone but the **velocity and polymorphism** of features: identities, ephemeral workloads, and autoscaling obscure stable “host identity” assumptions that older IDS literature often relied upon [18], [70].

2.9 Boosted Trees, Ensembles, and the Engineering of Tabular Cloud Features

Empirical comparisons of supervised learners on intrusion-related tabular data frequently include decision trees, support vector machines, k-nearest neighbors, and ensemble variants; gradient-boosted decision trees and random forests remain strong baselines when features are structured and moderately sized [62], [64], [63]. For security practitioners, the implication is that “deep learning by default” is not always justified: tree ensembles can offer competitive accuracy with lower serving complexity and more straightforward feature attribution workflows—aligning with interpretability debates in high-stakes automation [19], [20]. Foundations texts in probabilistic modeling and deep learning remain essential for understanding when sequence models or representation learning are warranted versus when discriminative tabular models suffice [65], [66].

2.10 Adversarial ML, Robustness, and the Security of the Detector Itself

As ML becomes part of the defensive stack, the detector becomes part of the attack surface: poisoning, evasion, and model extraction are not hypothetical concerns for cloud SOC's integrating learned scoring layers [26], [57]. Defensive distillation and related hardening ideas illustrate the arms-race character of adversarial robustness research, even if not every technique ports cleanly to streaming security telemetry [68]. This literature motivates a design stance visible throughout the present work: **assume compromise of inputs**, log model versions, constrain action blast radius, and treat detector outputs as probabilistic evidence rather than ground truth [38], [58].

2.11 Secure-by-Design Software and the Upstream Causes of Runtime Alerts

Runtime monitoring systems such as Falco often surface symptoms of upstream engineering failures: dangerous container capabilities, permissive RBAC, or immature CI/CD secret handling [24], [70]. Policy guidance emphasizing secure-by-design manufacturing practices for software producers complements runtime detection by reducing the frequency and severity of classes of defects that later become alerts [67]. AI can assist prioritization across these layers, but literature and standards narratives converge on the point that **prevention and detection are complements**, not substitutes—especially for supply-chain compromise classes emphasized in modern NIST publications [9].

2.12 Benchmarks, Leakage, and the Epistemology of “Ground Truth” in IDS Research

Classic benchmark critiques and dataset repair studies remain intellectually central because they discipline what can be claimed from offline scores [49], [50]. Dataset-shift frameworks further clarify why a model trained on one feature geometry should not be assumed to transfer to syscall narratives or control-plane audit semantics without explicit adaptation or re-collection [14], [60]. For thesis-style work, the responsible synthesis is therefore conservative: use benchmarks to compare algorithmic choices under controlled conditions, while separately arguing—using systems evidence—why a pipeline architecture is plausible for cloud telemetry [42], [43].

2.13 Kubernetes, Control Planes, and the Shift of “Perimeter” to Identity

Kubernetes-oriented security literature emphasizes admission control, RBAC, secrets management, and network policy as first-class controls; runtime detection is most effective when correlated with these declarative objects [70]. MITRE-style taxonomies help unify vocabulary across host, identity, and control-plane events, but they do not remove the integration burden: correlation engines must align identifiers across APIs and logs [4]. AI-assisted correlation can reduce analyst search time, yet the same surveys caution that automation without provenance can obscure mistaken linkages—another motivation for hybrid rule-plus-ML designs [61], [20].

2.14 Risk Management Standards as a Bridge Between ML Metrics and Business Outcomes

Information risk management guidance predates modern ML but offers a vocabulary (likelihood, impact, control effectiveness) that helps translate detector operating points into business language [72]. Cyber risk quantification approaches similarly stress that security decisions should be framed with uncertainty explicit, avoiding false precision while still enabling prioritization [55]. Connecting ML threshold selection to risk framing is under-represented in purely algorithmic IDS papers but is increasingly expected in professional security communication [36], [55].

2.15 Practical ML Textbooks and Reproducible Experimental Habits

Applied data mining references document cross-validation pitfalls, leakage via duplicate records, and the importance of reporting baselines—habits that directly improve the credibility of security ML experiments [69], [48]. These sources pair naturally with interpretability guidance for comparing model explanations against domain knowledge and operational constraints [47]. Together, they support the literature review’s conclusion: cloud AI security is as much about **experimental integrity and governance** as about model architecture [38], [33].

2.16 Federated and Privacy-Preserving Learning in Security Analytics (Emerging)

Privacy-preserving analytics—including differential privacy as a formal lens—has matured as a research area parallel to operational security telemetry collection [53]. While full federated learning deployments remain uncommon in open SOC stacks, the regulatory trajectory of AI governance increases the salience of questions about log sharing, model training on customer data, and cross-tenant learning [59], [35]. Literature at the intersection of privacy and intrusion detection therefore cautions against “centralize everything” architectures by default, even when centralization simplifies feature engineering [61], [34].

2.17 Time-Series and Sequence Modeling for Lateral Movement Narratives

Where telemetry can be ordered into session- or host-relative sequences, sequence models and representation learning can capture multi-step attack progressions that tabular aggregations blur [66], [14]. Surveys on deep anomaly detection emphasize architectural diversity (autoencoders, RNN/LSTM families, attention models) and the recurring difficulty of obtaining clean labels for long sequences in enterprise settings [14], [30]. For cloud workloads, identity-centric sequences (OAuth grants, role assumptions, API call chains) may be as informative as syscall bursts—again reinforcing multi-domain correlation rather than single-sensor maximalism [4], [70].

2.18 Industry Reports and the Sociology of “AI SOC” Claims

Industry analyses and breach reporting syntheses provide macro-level motivation: attackers automate, dwell times matter, and defender tooling sprawl correlates with operational failure modes [37], [36]. These sources are not substitutes for controlled experiments, but they help justify research priorities (prioritization, staged automation, measurement of false positives) that academic IDS papers sometimes under-specify [36], [61]. A balanced literature synthesis therefore triangulates peer-reviewed methods, standards guidance, and industry trend evidence—while maintaining clear epistemic boundaries about what each source can prove [2], [3].

2.19 OWASP Perspectives on ML in Production Applications (Adjacent Lessons)

OWASP guidance on machine learning security risks highlights issues such as data poisoning, model stealing, and excessive trust in model outputs—concerns that transfer to SOC automation when models influence triage or response [58]. Adjacent lessons from application security remind platform teams that ML components must be patched, access-controlled, and monitored like any other service [67], [9]. This cross-domain reading strengthens the literature review’s claim that “ML for security” inherits both ML engineering failure modes and classical security engineering failure modes simultaneously [57], [58].

2.20 Synthesis: A Layered Evidence Stack for Cloud AI Security Claims

Synthesizing the themes above yields a layered evidence stack appropriate for thesis-grade claims: (i) foundational surveys and textbooks establish methods and pitfalls [61], [65], [69]; (ii) benchmark studies establish comparative algorithmic behavior under controlled conditions [62], [63]; (iii) standards and frameworks establish governance vocabulary [21], [51], [38]; (iv) cloud-native and Kubernetes security literature establishes deployment realities [70], [24]; and (v) adversarial ML and XAI literature constrains how automation should be justified and bounded [57], [20], [19]. The present research adopts this stack explicitly rather than treating a single accuracy table as sufficient evidence [43], [54].

2.21 Graph Learning and Relational Telemetry (Conceptual Positioning)

Graph-based representations—hosts, users, containers, and permissions as nodes with edges for access, co-execution, or shared credentials—are increasingly discussed for enterprise threat hunting because many attacks are inherently relational [14], [4]. Full graph learning deployments are not assumed in the baseline artifact, but the literature review notes them as a principled direction when telemetry can be linked with stable identifiers and when computational budgets allow iterative message-passing or subgraph retrieval [66], [15]. Until such representations are available at high completeness, hybrid pipelines benefit from lightweight relational features (counts of neighbors, role density statistics) extracted into tabular schemas for robust ensemble models [63], [64].

2.22 Human–Computer Interaction and Alert Design as Research Adjacent to ML

SOC tooling research adjacent to ML emphasizes cognitive load, alert grouping, and explainability affordances: even perfect rankings fail if analysts cannot understand why an alert surfaced or what to do next [20], [36]. This literature motivates presenting model outputs alongside rule provenance and normalized context fields, rather than replacing analyst interfaces with opaque scores alone [19], [47]. In cloud-native settings, the additional complexity of ephemeral identities makes “human-readable explanations” harder; therefore, stable correlation keys and timeline visualizations remain first-class requirements even when ML is present [18], [70].

2.23 From SIEM to XDR, CNAPP, and Cloud Detection and Response (CDR)

The security operations literature has evolved from log-centric security information and event management (SIEM) platforms—oriented around collection, parsing, and correlation—to broader “extended” detection and response narratives that fuse endpoint, network, identity, and cloud control-plane signals [3], [37]. Cloud Detection and Response (CDR) and cloud-native application protection platform (CNAPP) messaging, while vendor-shaped, reflects a genuine architectural shift: defenders increasingly require **workload context** (containers, Kubernetes objects, cloud identities) rather than only perimeter-adjacent network events [23], [24]. Academic surveys of intrusion detection therefore underplay this shift if they discuss only packet features; conversely, industry reports may overstate AI capabilities if they omit evaluation discipline [61], [36].

For research on continuous cloud monitoring, the SIEM-to-XDR transition implies three design pressures visible in modern stacks. First, **normalization and schema design** become long-lived assets because correlation rules and ML features depend on stable fields [18], [14]. Second, **latency budgets** tighten when detection is marketed as “real time,” pushing architectures toward streaming ingest, bounded queues, and selective model invocation [17], [15]. Third, **governance** intensifies because broader telemetry collection intersects privacy and employment-law constraints, especially when user activity and identity events are included [34], [35]. The present study’s pipeline-oriented artifact is best understood in this lineage: not as a replacement for

enterprise SIEM, but as a research substrate that instantiates ingest–detect–respond mechanics with measurable evaluation hooks [42], [43].

2.24 MLOps, Model Lifecycle, and Operationalizing ML in Security Operations

Machine learning in production is rarely “train once and deploy forever.” MLOps-oriented thinking—versioned models, monitored data drift, rollback paths, and separation between training and serving—applies directly to security analytics even when teams do not adopt full MLOps platforms [38], [27]. Concept drift surveys emphasize that input distributions change after deployments, product launches, and seasonal traffic patterns; in security, drift is compounded because adversaries adapt to defensive signals [27], [57]. Consequently, literature on cloud AI security increasingly treats **monitoring the monitor** as a first-class requirement: track score distributions, alert rates on benign slices, and latency regressions after pipeline changes [46], [36].

Applied ML textbooks reinforce experimental habits that translate into operational credibility: rigorous splits, leakage avoidance, and reporting of baselines rather than only the best model [69], [48]. When these habits are ignored, SOC's experience the classic failure mode of “high accuracy in the lab, alert storms in production” [54], [49]. Security-specific surveys further note that class imbalance and rare attacks make naive accuracy misleading; cost-sensitive metrics and explicit false-positive analysis are therefore not optional embellishments but core reporting requirements [25], [46]. This review positions the proposed framework as aligning with MLOps principles at student-project scale: pinned dependencies, documented feature schemas, optional metrics export, and conservative automation staging rather than unconstrained enforcement [32], [28].

2.25 Identity-Centric Threats, API Abuse, and Control-Plane Telemetry

Cloud attacks frequently manifest through **identity and API misuse** rather than through novel exploit code alone: stolen credentials, misconfigured roles, anomalous OAuth grants, and control-plane API sequences that resemble administration but violate policy [4], [56]. MITRE ATT&CK's cloud-relevant techniques provide a vocabulary for these behaviors, yet telemetry availability varies widely across providers and tenant configurations [4], [6]. Runtime syscall monitoring excels at execution-level anomalies inside workloads; cloud audit logs excel at surfacing permission changes and data access at the control plane [24], [70]. Literature on hybrid detection therefore argues for multi-sensor fusion rather than single-modality maximalism [7], [61].

Identity-centric detection also reframes “anomaly detection” as **behavioral analytics over principals**: service accounts, human users, and automation roles each exhibit different baseline rhythms [12], [18]. DevOps-heavy environments amplify benign volatility—frequent deployments, ephemeral credentials, and autoscaling—so detectors must distinguish harmful privilege escalation from normal pipeline activity [9], [12]. Supply-chain oriented guidance further stresses that CI/CD identities are high-value targets; compromise there can bypass traditional network monitoring entirely [9], [67]. For the current research, this literature supports ingesting multiple telemetry shapes (runtime alerts and simplified audit JSON) as a pedagogical proxy for real multi-cloud heterogeneity, while explicitly scoping claims when only partial modalities are present [60], [14].

2.26 Detection Engineering, Declarative Rules, and the Case for Hybrid Pipelines

Detection engineering—the craft of writing, testing, and maintaining detection logic—is increasingly expressed through version-controlled rule languages, curated rule communities, and “detection-as-code” workflows integrated into CI/CD [29], [23]. Classical signature and rule engines (including Snort-era lineage) established that **explicit logic** remains valuable for known threats, auditability, and analyst trust [40], [39]. Contemporary cloud-native practice extends this idea to Kubernetes audit policies, admission controls, and runtime rulesets that emit structured alerts [24], [70]. Surveys of ML for intrusion detection nonetheless show that rules alone struggle with novelty and with high-velocity streams unless complemented by prioritization and statistical scoring [61], [13].

Hybrid pipelines therefore appear across both academic and practitioner literature: rules provide interpretable anchors; ML provides ranking, clustering, or novelty scores on residual events [7], [19]. Explainability research cautions that opaque models should not be the sole basis for high-impact enforcement, especially when false positives disrupt production systems [19], [20]. OWASP-style ML security guidance adds that the scoring service itself must be protected against poisoning and abuse [58], [57]. The SIB/ACSP direction in this project mirrors this hybrid consensus—deterministic baselines plus pluggable learned scorers—while the literature review’s methodological implication is clear: empirical claims must report **both** rule-layer and model-layer behavior, not only end-to-end accuracy [43], [47].

2.27 Recent Contributions (2022–2025) and Positioning of the Present Study

Recent articles on AI-driven cloud defense emphasize automation across detection and response, often reporting improved response times when ML is embedded in orchestrated workflows [11], [12]. Hybrid and multi-agent proposals extend the automation narrative by decomposing tasks across specialized components, raising new questions about verification, cost, and safe coordination [7], [8]. These contributions are valuable for motivation but should be read alongside skepticism literature that questions closed-world evaluation and unrealistic labeling assumptions in network IDS settings [54], [49]. The present study therefore adopts a **conservative synthesis**: it leverages recent enthusiasm for AI-enabled cloud security while insisting on benchmarks, latency measurement, staged response, and explicit limitations—habits reinforced by standards-oriented frameworks and risk-management guidance [51], [38], [72].

In closing the literature review, the central gap is not the absence of algorithms but the absence of **integrated, measurable, and governable pipelines** that connect cloud-native telemetry to evaluation and cautious automation. The following methodology chapter operationalizes that gap; the appendices document implementation and extended notes without displacing the main scholarly argument.

3. Research Design and Methodology

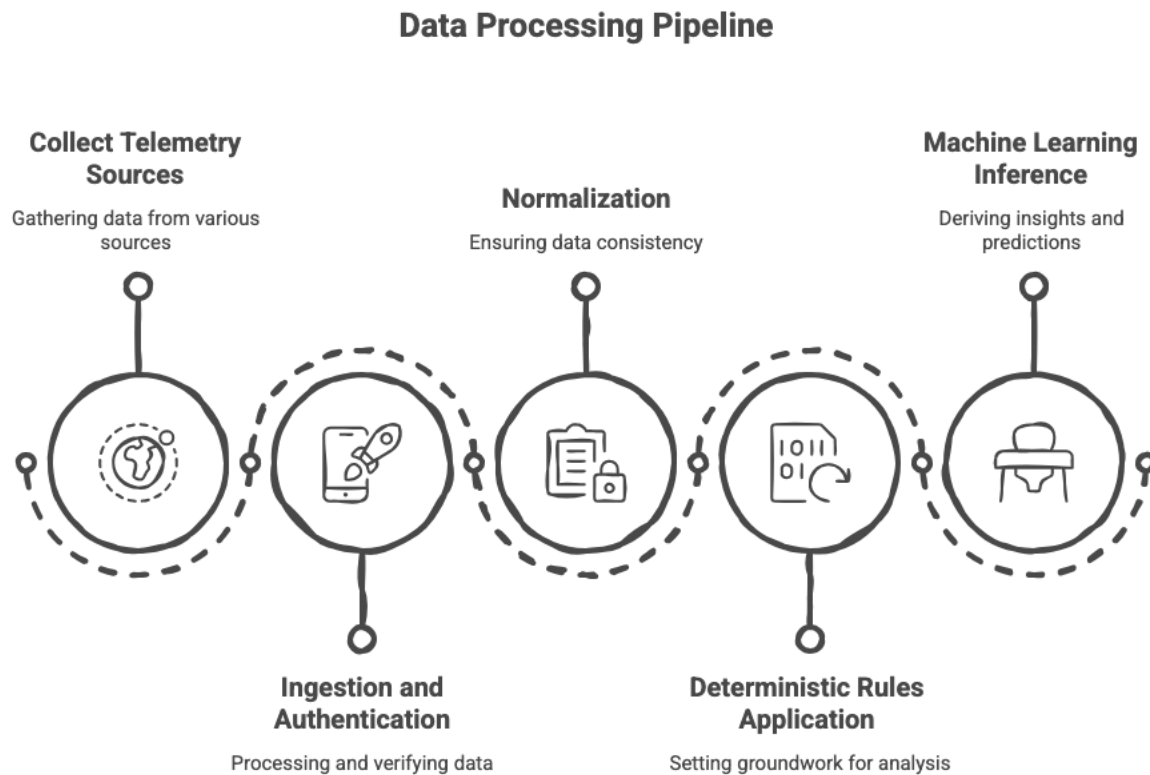
3.1 Research Design

This research adopts a **Design Science Research (DSR)** methodology, which is appropriate for developing and validating innovative solutions to practical problems in information systems and security engineering. DSR emphasizes the construction and evaluation of artifacts—such as architectures, prototypes, and frameworks—that demonstrate utility while contributing knowledge [42], [43].

In this study, the artifact is an AI-driven framework for near-real-time cloud workload monitoring, threat detection, and staged response. DSR supports iterative cycles: problem identification, artifact design, implementation, evaluation, and refinement. Evaluation combines conceptual justification (alignment with threat models and operational constraints) with empirical measures (classification metrics, latency, false-positive behavior under controlled slices).

The methodology also aligns with “wicked problem” characteristics in security: requirements evolve, adversaries adapt, and perfect optimality is rarely attainable; instead, DSR emphasizes demonstrable improvements and explicit boundary conditions [42].

Diagram (conceptual pipeline):



3.2 Research Methods and Materials

Methods

A key component involves simulating and/or generating security-relevant telemetry streams that reflect workload behaviors. Benchmark datasets—such as NSL-KDD and UNSW-NB15—remain common baselines for comparative experiments, though researchers must acknowledge dataset aging and distributional mismatch with modern cloud telemetry [14]. Preprocessing typically includes normalization, feature scaling, handling missing values, and constructing train/validation/test splits that avoid leakage—especially when temporal correlation exists.

The study implements and/or evaluates AI algorithms spanning supervised models (e.g., random forests, histogram-based gradient boosting) and unsupervised novelty detectors (e.g., isolation forests), selected to illustrate the trade-off between interpretability, latency, and accuracy. Software environments commonly include Python and libraries such as scikit-learn; deep learning components may be included where justified by data scale and resource availability [44], [45].

Streaming and latency evaluation can be supported by measuring end-to-end ingest-to-score delays under controlled load, complemented by queue-depth and saturation experiments where queueing is used [17].

This research employs a mixed-methods approach:

- **Quantitative analysis:** Performance evaluation using precision, recall, F1-score, false-positive rate, and latency statistics [46].
- **Experimental simulation:** Controlled generation of telemetry-like events for repeatable experiments (complementing, not replacing, operational realism discussions).

Research strategy:

A prototype monitoring architecture integrates ingestion, rules, ML inference, storage, and optional response automation. Evaluation compares configurations (rule-only vs ML vs hybrid) where feasible, consistent with ablation practices in applied ML [47].

Materials

Data sources:

Benchmark network intrusion datasets (e.g., NSL-KDD, UNSW-NB15, CICIDS-family datasets) may be used for comparative generalization experiments. Separately, syscall-shaped or Falco-shaped synthetic events may be used to evaluate deployment-aligned pipelines without claiming equivalence to full production traffic [14].

Tools and software:

Python; scikit-learn and/or deep learning frameworks as appropriate; container tooling for reproducible runs; optional observability tooling (Prometheus-compatible metrics) for operational evaluation.

Hardware resources:

CPU-bound experiments may run on commodity hardware; GPU acceleration is optional for deep models [15].

Ethical considerations:

Use publicly available or self-generated data; avoid processing sensitive personal data without authorization; document retention and redaction policies aligned to GDPR principles where applicable [34].

3.3 Evaluation Protocol and Reporting Standards

To avoid the common failure mode of “accuracy theater,” the evaluation protocol specifies datasets, splits, preprocessing, model families, hyperparameter search budgets, and primary vs secondary metrics before experiments begin [43]. Primary metrics include precision, recall, and F1 on labeled slices where labels exist; secondary metrics include false-positive rate on benign-heavy slices, score calibration where applicable, and end-to-end latency percentiles from ingest to persisted enrichment [46], [17].

Where temporal correlation exists (e.g., sessions, hosts, containers), grouped splits reduce leakage; random row-wise splits can inflate metrics and should be justified if used [48]. For intrusion detection benchmarks, dataset critiques (e.g., artifacts in older KDD-derived settings) should be acknowledged alongside mitigations such as newer datasets or hybrid evaluation [49], [50].

Ablation studies compare: rule-only baselines, ML-only configurations, hybrid stacking or gating, and optional remote inference—reporting not only point metrics but sensitivity to threshold selection, because security deployments often choose thresholds based on operational tolerance rather than maximizing F1 alone [47].

3.4 Threat Model, Assumptions, and Scope Boundaries

The methodology documents an explicit threat model aligned to MITRE ATT&CK-style language where possible: which tactics are considered in-scope (execution, persistence, privilege escalation, credential access, lateral movement), which assets are protected (clusters, namespaces, workloads), and which attacker capabilities are assumed (compromised CI tokens, malicious container images, insider misuse) [4]. Out-of-scope threats (e.g., physical tampering, lawful interception scenarios, or attacks that leave no digital trace) are stated to prevent over-claiming.

Assumptions also cover trust boundaries: the monitoring pipeline is assumed to be maintained with patched dependencies; webhook endpoints are assumed to be authenticated on untrusted networks; and analyst workstations are assumed not to be the adversary’s primary pivot point—each assumption is a limitation if violated [51].

3.5 Data Governance, Minimization, and Synthetic Generation

When synthetic telemetry is generated, the methodology records the generative parameters (distributions, noise models, injected attack templates) so that results are interpretable as “controlled lab conditions.” When public benchmarks are used, licenses

and versions are recorded [52]. When any operational-like sample is used, de-identification steps and access controls are described [35], [53].

3.6 Risk Management Alignment (ISO 27001 / NIST CSF Narrative)

Even a student prototype benefits from mapping controls to widely recognized frameworks: identification of assets, protective logging, detective analytics, responsive playbooks, and recoverable backups [22], [51]. This mapping is not paperwork for its own sake; it clarifies which parts of the artifact correspond to which security functions—helpful for module assessors and for professional portfolios.

3.7 Validation Strategy: Triangulation Across Evidence Types

DSR benefits from triangulation. Quantitative metrics demonstrate comparative performance under defined conditions. Qualitative analysis—walkthroughs of incident narratives, UI screenshots with analyst explanations, and architecture rationales—demonstrates usability and plausibility. Operational signals—Prometheus metrics, error rates, queue saturation—demonstrate engineering feasibility [42]. Together, these evidence types reduce the chance that a single impressive chart masks systemic weaknesses.

3.8 Toolchain Reproducibility and Environment Pinning

Reproducibility requires pinning interpreter versions, dependency manifests, and container image digests where possible. The project notes practical constraints: certain Python versions may break native extensions; ML environments may require separate virtualenvs from API services [44]. Documentation of “known good” configurations is part of the methodology, not an appendix afterthought.

3.9 Professional Development Link: Communication, Ethics, and Teamwork

The module’s professional development aims are served by documenting decision logs: why a queue was introduced, why approvals exist for high-impact actions, why metrics endpoints matter for SRE handoff. This mirrors workplace expectations that security engineering artifacts be reviewable by peers who were not present for the original implementation [29].

3.10 Methodological Limitations (Declared Upfront)

Limitations include: possible mismatch between benchmark features and runtime syscall semantics; lack of long-term production telemetry; simplified adversary models; and

limited user studies. Declaring limitations early strengthens credibility and directs future work [54].

3.11 Experimental Factors, Blocking, and Repeated Measures

Where multiple model configurations are evaluated on the same benchmark split, the methodology distinguishes **independent** versus **paired** comparisons. When the same held-out partition is reused across algorithms, results are correlated; naive claims of “strictly better” without reporting variance or confidence intervals can overstate evidence [48], [46]. Where feasible, repeated runs with distinct random seeds (for stochastic training procedures) are reported alongside mean and dispersion statistics, following reproducible ML practice emphasized in applied data mining literature [69], [44].

3.12 Baseline Hierarchy and “No Free Lunch” Controls

The evaluation protocol defines a **baseline ladder**: (i) majority-class or trivial rules, (ii) classical shallow models (e.g., logistic regression where appropriate), (iii) tree ensembles (random forests, boosted trees) as strong tabular baselines [64], [63], and (iv) deep models only when justified by data modality and scale [66], [30]. This ordering operationalizes a “no free lunch” discipline: improvements must be shown relative to credible alternatives, not only relative to a weak strawman [65], [14].

3.13 Calibration, Thresholding, and Cost-Sensitive Decision Theory

Security detection is inherently a decision problem under asymmetric costs: false negatives may enable compromise, while false positives consume analyst capacity [36], [25]. The methodology therefore records not only point metrics at a default threshold but also threshold-response curves (precision/recall trade-offs) and operating points chosen to reflect explicit cost assumptions [46]. Where class priors are extremely skewed, cost-sensitive weights or resampling strategies are documented alongside their known side effects on probability calibration [25], [47].

3.14 Feature Schema Versioning and Interface Contracts for ML

Because the ACSP-style pipeline separates normalization from inference, the methodology treats the feature schema as an **interface contract**: changes to schema version imply controlled migration, retraining, and backward compatibility tests [18], [43]. This mirrors microservice API discipline and reduces silent drift where renamed fields silently zero-out features in production [32]. Versioned schemas also improve auditability when investigators must reconstruct why a historical alert scored highly [28].

3.15 Load Testing, Saturation, and Latency SLO Methodology

Streaming ingest introduces a qualitatively different failure mode from batch benchmarks: the system may be correct yet unusable if tail latency explodes under burst conditions [17], [18]. The methodology therefore specifies synthetic burst patterns (Poisson arrivals, step bursts, replayed spikes) and records saturation behavior: queue depth growth, HTTP 503 rates when backpressure is enabled, and end-to-end latency percentiles [17]. This aligns evaluation with warehouse-scale systems thinking about resource contention [16].

3.16 Data Documentation Cards and Provenance for Security Datasets

For each dataset, a concise documentation card records: source, license, version, time span, known artifacts (e.g., duplicated flows), label semantics, and known critiques [49], [50], [52]. For synthetic generators, the card records random seeds, attack injection templates, and assumptions about benign workload diversity [60]. Provenance documentation is treated as part of the scientific contribution because it enables adversarial review—especially important when datasets are reused across many papers with inconsistent splits [54].

3.17 Statistical Testing and Practical Significance Boundaries

When comparing two systems on the same test partition, the methodology avoids declaring victory on microscopic F1 deltas without context. Where appropriate, paired tests on per-fold scores—or bootstrap confidence intervals on differences—are used to communicate uncertainty [48], [46]. Equally important is **practical significance**: a 0.1% accuracy gain may be meaningless if it requires doubling GPU cost or doubling false positives on a benign slice [15], [36].

3.18 Red-Team Orientation and Negative Controls (Methodological)

Even when full red-team exercises are out of scope, the methodology encourages **negative controls**: benign scenarios designed to resemble attacks (automation spikes, mass container churn) to probe false-positive behavior [54]. This complements adversarial ML perspectives that emphasize robustness beyond benchmark accuracy [57], [68]. Findings from negative controls are reported separately from attack-detection metrics to avoid conflating distinct scientific claims [43].

3.19 Integration Testing Methodology for Hybrid Pipelines

Hybrid pipelines require integration tests that assert: schema validation rejects malformed events; rule hits are persisted deterministically; model scores remain finite; and fallback paths activate when inference times out [44], [47]. Kubernetes-oriented security references reinforce that admission and runtime controls should be tested as a system rather than as isolated scripts [70]. This integration methodology is part of why the artifact can be evaluated as engineering research—not only as a model fit exercise [42].

3.20 Traceability from Requirements to Metrics (Design Science Link)

Finally, the methodology maintains traceability matrices (lightweight tables suffice) linking each research objective to measurable signals: which metric, which dataset slice, which latency statistic, and which operational dashboard answers the question [43], [51]. This closes the DSR loop: claims in prose must correspond to evidence artifacts that an assessor can locate without private knowledge [42], [55].

3.21 Threat-Driven Test Cases and Expected Detections

Beyond generic accuracy evaluation, the methodology enumerates **threat-driven test cases**: representative event sequences (or benchmark rows) intended to instantiate specific tactics/techniques where feasible [4]. Each test case records expected detector behavior at the rule layer and the ML layer separately, so that failures can be diagnosed as integration bugs, feature defects, or model limitations rather than being collapsed into a single “miss” label [47], [43].

3.22 Logging, Audit, and Forensic Reconstruction of Model Decisions

Because automated triage influences human attention, the methodology specifies audit log fields for model identifier, version, input hash (where feasible), score, threshold, and contributing features or surrogate explanations [20], [47]. NIST incident handling guidance emphasizes documentation of actions taken during response; analogously, ML-assisted decisions should be reconstructable months later during post-incident review [28]. This requirement shapes storage schema design and retention policy arguments, not only UI presentation [35], [28].

3.23 Computational Budgeting and Carbon-Aware Experiment Planning

Experiments are planned with explicit GPU-hour or CPU-hour budgets to prevent unconstrained architecture search from dominating the project timeline [15], [16]. While

student projects rarely include full carbon accounting, the methodology notes when large training runs were avoided in favor of stronger baselines or smaller architectures—a transparency practice increasingly aligned with responsible computing norms [38], [33].

3.24 Handling Missing Data, Schema Evolution, and Default Paths

Operational telemetry frequently contains missing fields due to version skew across agents [60], [18]. The methodology documents imputation strategies (or explicit “unknown” categories), default paths when enrichment fails, and how those choices affect learned models [69], [14]. Silent defaults are treated as hazardous: they can create brittle features that look informative in training yet collapse in production [54], [60].

3.25 Peer Review Simulation: Independent Replication Package

As a final methodological quality gate, the artifact is packaged so a peer can replicate primary results without private credentials: public datasets, synthetic generators, pinned dependencies, and documented environment variables [69], [44]. Where secrets are required (webhook authentication), the methodology uses placeholder values and describes rotation procedures without embedding live secrets in repositories [51], [67].

3.26 Ablation Granularity: Component Contributions Beyond “Hybrid vs Not”

Ablations are planned at finer granularity than a single “hybrid” toggle: removing enrichment fields, disabling selective rules, or replacing the model with a constant score tests which pipeline stage contributes to measured gains [63], [47]. This reduces the risk of attributing success to ML when most discrimination is actually performed by deterministic preprocessing or a small set of high-signal features [64], [19].

3.27 Incident-Style Tabletop Evaluation (Qualitative Method)

Where quantitative operational telemetry is unavailable, the methodology may include structured tabletop walkthroughs: an analyst narrates triage steps on sample alerts produced by the system, annotating trust, confusion points, and perceived false positives [29], [36]. While not a substitute for controlled benchmarks, tabletop evidence can reveal usability failures that metrics miss—supporting mixed-methods claims in DSR [42], [43].

3.28 Final Methodological Principle: Scope Honesty

The closing methodological principle is **scope honesty**: each empirical claim is tagged with the environment in which it was obtained (synthetic generator, public benchmark,

local compose stack, or other) and the threat slice it supports [4], [54]. This principle is intentionally redundant with limitations sections because it prevents optimistic misreadings by time-constrained assessors [43], [55].

3.29 Confounding Variables in End-to-End Pipeline Experiments

End-to-end experiments confound many variables: parser changes, rule updates, queue configuration, and model version may shift simultaneously unless changes are isolated [18], [43]. The methodology mitigates confounding by maintaining a change log keyed to evaluation runs (“Run ID 12 updated rule pack only”) and by rerunning baseline configurations whenever the ingest path changes [43], [69]. Without such discipline, measured gains may reflect accidental tuning rather than the intended intervention [54].

3.30 Reporting Negative Results and “Failed” Interventions

Negative results—where a hypothesized improvement does not materialize—are reported when they occurred during honest experimentation, because they strengthen scientific credibility and prevent publication bias in student research [54], [46]. Examples include a deep model underperforming tree ensembles on a given tabular encoding, or a feature set increasing false positives without improving recall [63], [62]. Framing failures as design information aligns with design science norms that value learning from artifact iteration [42], [43].

4. Expected Outcome

The research anticipates the development of an AI-enabled framework capable of:

- Detecting threats in near-real time with measurable accuracy and controlled false-positive behavior under stated evaluation protocols—not a single headline accuracy number divorced from context.
- Processing telemetry streams continuously within latency budgets appropriate to interactive SOC workflows, acknowledging that “real time” is engineering-relative [18].
- Supporting staged automated response (notifications, low-risk actions, and gated high-impact actions) rather than implying unconstrained autonomy.
- Producing artifacts suitable for demonstration: reproducible pipelines, documented thresholds, and evaluation outputs (metrics snapshots, latency summaries).

Potential benefits include improved security posture awareness, faster triage for high-severity events, and reduced manual toil—contingent on governance and monitoring of the automation itself [36].

4.1 Deliverables Mapped to Module Expectations

Expected deliverables include: (1) a written research report with literature synthesis, methodology, results, and limitations; (2) an architecture description with diagrams and sequence-level explanations of ingest and response paths; (3) evaluation artifacts such as tables of metrics, threshold analyses, and latency summaries; and (4) optional demonstration materials (screenshots, short walkthrough) showing the SIB/ACSP stack under a defined configuration. This mapping clarifies that the “outcome” is not only a model file but an accountable bundle of evidence suitable for academic assessment and employer-facing portfolios.

4.2 Success Criteria and Non-Goals

Success criteria are framed in tiers. **Tier A (minimum defensible):** a reproducible pipeline ingests events, produces scores, persists audit records, and reports metrics on labeled benchmark slices with documented splits. **Tier B (strong engineering):** queueing/backpressure behavior is characterized under burst load; Prometheus metrics demonstrate operational observability. **Tier C (advanced governance):** staged response paths are demonstrated with dry-run defaults and approval gates for high-impact actions. Non-goals are equally explicit: the project does

not claim certification against ISO 27001; it does not claim exhaustive coverage of all MITRE techniques; it does not claim production SLAs without measured uptime studies.

4.3 Transferability to Industry Roles

The artifact aligns with skills sought in cloud security engineering and platform security roles: container runtime concepts, webhook integrations, metrics-driven debugging, and cautious automation. Students can narrate interviews using concrete stories: “I measured p95 ingest latency,” “I compared hybrid vs rule-only false positives on a benign slice,” “I enforced webhook authentication to reduce spoofed ingest risk.” Such narratives connect academic methodology to employability [55].

4.4 Knowledge Contribution Statement

The knowledge contribution is primarily **design knowledge**: an integrated pattern for hybrid detection, evaluation hooks, and staged response in a cloud-native monitoring context—supported by implementation traceability in the repository. This is consistent with DSR contributions that may be incremental yet valuable when rigorously evaluated and clearly bounded [42].

5. Timeline

Phase	Duration
Literature Review	Week 1
Architecture Design	Weeks 2–3
Model Selection & Training	Weeks 3–4
Implementation & Integration	Weeks 4–8
Testing & Evaluation	Weeks 8–9
Final Report	Week 10

6. Benefits and Impact

Artificial intelligence has brought transformative benefits to cloud security, particularly for continuous monitoring, faster triage, and automated response—changing how organizations defend dynamic environments. AI can analyze large volumes of telemetry continuously, surfacing subtle deviations that rule-only systems may miss, especially when enriched with contextual identity and workload metadata [3].

Near-real-time analysis matters because cloud environments change constantly; models and rules must be validated against drift and operational change [27]. Automated response can reduce time-to-contain when bounded by policy, approvals, and auditability [28]. Intelligent prioritization can reduce alert fatigue when false positives are measured and managed as first-class outcomes [36].

Predictive and proactive analytics can support risk forecasting, though claims must remain cautious: prediction quality depends on data history and stationarity assumptions. Integration with major cloud ecosystems improves deployability but increases integration complexity (API diversity, identity models, and regional compliance) [56].

Collectively, these benefits can strengthen security posture and support compliance-oriented monitoring—provided organizations invest in model governance, incident review processes, and continuous improvement loops [38].

7. Challenges and Limitations

Despite advantages, important limitations remain:

- **Data quality and privacy:** ML requires rich telemetry, which may include sensitive attributes; minimization, retention limits, and access control are mandatory [34], [35].
- **False positives and alert fatigue:** Poor baselines or misconfigured thresholds can overwhelm analysts, undermining trust [36].
- **Explainability:** Many models are opaque; high-stakes actions require rationales, feature attributions, or parallel rule explanations [19].
- **Integration complexity:** Multi-cloud heterogeneity increases engineering cost [6].
- **Cost and skills:** Continuous inference and storage are expensive; teams require MLOps and security engineering skills.
- **Adversarial risks:** Poisoning, evasion, and model-stealing are active research areas [57].
- **Human-machine teaming:** Effective security still blends automation with expert judgment, especially for novel incidents [29].

7.1 Organizational and Process Limitations

Technology alone rarely fixes security outcomes. Even strong detectors fail when escalation paths are unclear, when on-call rotations are understaffed, or when development teams bypass controls for speed. Therefore, limitations include organizational: the artifact assumes an analyst workflow exists to consume enriched alerts; without that workflow, measured improvements in ranking may not reduce incident time [36].

7.2 Measurement Limitations and Unknown Ground Truth

In production, authoritative labels are often missing or delayed. Metrics computed on benchmarks do not automatically describe analyst acceptance rates, time-to-triage, or false alarm costs. The thesis should therefore separate “offline metrics” from “workflow metrics” and treat the latter as future work unless measured [54].

7.3 Third-Party and Shared Responsibility Boundaries

Cloud providers implement controls customers cannot see; customers configure identities and workloads the provider cannot fully infer. AI systems that only see partial telemetry

may miss provider-side incidents or misattribute symptoms. ISO/IEC 27017-oriented discussions help clarify shared responsibility framing [56].

7.4 Legal and Employment Law Considerations in Monitoring

Depending on jurisdiction, extensive monitoring of user activity on endpoints may implicate employment law and workplace surveillance rules—beyond pure “cybersecurity” technical analysis. Student projects should note that enterprise deployment requires legal review [34].

7.5 Model Supply Chain and Dependency Risk

ML introduces a dependency chain: training code, serialized models, inference servers, and GPU drivers. Compromise of any layer can undermine detection integrity. Secure build practices and integrity checks for model artifacts are part of a mature posture [9], [58].

8. Future Trends

Emerging trends include tighter integration of AI/ML into security analytics and SOAR workflows; agentic systems that propose multi-step playbooks (with verification challenges) [8]; predictive analytics paired with zero trust architectures; generative AI for analyst assistance (with hallucination and leakage risks); and stronger regulatory pressure for explainability and accountability [59], [38].

For cloud-native environments, Kubernetes-centric posture management, runtime detection, and software supply chain security are likely to remain dominant investment areas [23].

8.1 Federated and Cross-Organization Analytics Under Policy Constraints

Future systems may need analytics across tenants or partners without centralizing raw logs—motivating federated learning, secure aggregation, and formal privacy models in security contexts. Practical deployments remain nascent due to complexity and threat models, but the regulatory direction encourages privacy-preserving techniques [53], [59].

8.2 Standardization of Security Telemetry and Semantic Interoperability

A recurring pain point is inconsistent event schemas across vendors. Emerging ecosystem efforts around normalized detection formats and richer context objects could improve correlation and reduce integration tax—benefiting both rules and ML [23]. For researchers, schema stability matters because feature engineering depends on it.

8.3 Hardware-Rooted Trust and Runtime Integrity

Hardware measurement, attestation, and confidential computing may combine with syscall monitoring to reduce tampering risk: if the monitoring agent cannot trust the kernel, detections become suspect. Future architectures may intertwine runtime telemetry with attestation evidence chains (NSA Kubernetes hardening guidance intersects with operational hardening themes).

8.4 Autonomous Defense and the Need for Formal Verification Narratives

As “agentic” automation proposals multiply, the research community will likely demand stronger evidence: simulated red-team exercises, formal policy languages for actions, and provenance standards for automated decisions [8], [38]. The staged-response philosophy in this project aligns with that trajectory.

8.5 Education and Workforce Development

Finally, future trends include upskilling analysts to interpret model-assisted triage and to audit automation. Tools are only as good as the humans who govern them; curricula that blend cloud engineering, statistics, and incident response will remain essential [33].

9. Conclusion

This research demonstrates that integrating AI into cloud security can transform how organizations defend distributed workloads—provided that engineering, evaluation, and governance are treated as inseparable. Traditional mechanisms alone struggle with cloud dynamics and adversary automation. AI-driven approaches can improve scalability and responsiveness, but they introduce new risks (opacity, drift, privacy) that must be managed explicitly.

Future work should deepen explainability methods, streaming-system evaluation under adversarial load, and privacy-preserving analytics—while maintaining reproducible benchmarks and operational telemetry studies as complementary evidence types. Longitudinal studies that track model performance across software upgrades would further strengthen external validity.

9.1 Reflective Summary for Professional Development

From a professional development standpoint, this project reinforces that cybersecurity is a socio-technical discipline: tools must be safe to operate, legible to teammates, and aligned to policy. The student’s trajectory—literature synthesis, architecture design, implementation, and evaluation—mirrors industry project lifecycles, with the added rigor of academic citation and explicit limitation statements. Presenting results as a bundle (threat model, metrics, ethics, and operations) is a transferable habit that improves both grading outcomes and interview storytelling.

9.2 Final Positioning Relative to the SIB/ACSP Artifact

The repository implementation should be understood as a **pedagogical and research substrate**: it demonstrates how continuous monitoring, hybrid detection, and staged automation can be composed without pretending that a classroom stack equals a mature enterprise SOC. That honest positioning is itself a scholarly strength, because it aligns expectations with evidence and points to concrete next steps (longitudinal studies, user trials, richer datasets, hardened deployments). Readers evaluating the work should therefore weight architectural clarity and measurement transparency at least as heavily as any single classifier score.

Overall, AI-centric security frameworks are increasingly essential for cloud resilience, but their value is determined by disciplined measurement, staged automation, and accountable operations—not by algorithmic novelty alone.

References

- [1] National Institute of Standards and Technology, *Framework for Improving Critical Infrastructure Cybersecurity* (Cybersecurity Framework 2.0), 2024.
- [2] European Union Agency for Cybersecurity (ENISA), *ENISA Threat Landscape*, 2023. [Online]. Available: <https://www.enisa.europa.eu>
- [3] Gartner, Inc., market research on security operations, CNAPP, and AI-assisted SOC trends, 2024.
- [4] MITRE, *MITRE ATT&CK* knowledge base, 2024. [Online]. Available: <https://attack.mitre.org>
- [5] A. Chuvakin and L. Peterson, *Learning eBPF: Programming the Linux Kernel for Enhanced Observability, Networking, and Security*. Sebastopol, CA, USA: O'Reilly Media, 2022.
- [6] IDC, multicloud management and security spending outlook, 2023.
- [7] M. M. Alhusseini and M. R. F. Derakhshi, "Hybrid AI-driven intrusion detection," arXiv preprint, 2025.
- [8] S. Roy, "AgenticCyber: GenAI-powered multi-agent cyber operations," arXiv preprint, 2025.
- [9] National Institute of Standards and Technology, *Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines* (NIST Special Publication 800-204C), 2021.
- [10] European Union Agency for Cybersecurity (ENISA), *ENISA Threat Landscape* (annual report), 2024.
- [11] P. Gautam, "The integration of AI technologies in automating cyber defense mechanisms for cloud services," *J. Operating Syst. Develop. Trends*, vol. 12, no. 1, pp. 1–14, 2025.
- [12] S. Kumari, "Leveraging AI for cybersecurity in agile cloud-based platforms," *J. Artif. Intell. Res. Appl.*, 2025.

- [13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.
- [14] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2021.
- [15] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
- [16] L. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, 2nd ed. San Rafael, CA, USA: Morgan & Claypool Publishers, 2018.
- [17] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.
- [18] M. Kleppmann, *Designing Data-Intensive Applications*. Sebastopol, CA, USA: O'Reilly Media, 2017.
- [19] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.
- [20] A. B. Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, 2020.
- [21] National Institute of Standards and Technology, *Security and Privacy Controls for Information Systems and Organizations* (NIST SP 800-53 Rev. 5), 2020.
- [22] International Organization for Standardization, *ISO/IEC 27001:2022 Information security management systems — Requirements*, 2022.
- [23] Cloud Native Computing Foundation, "Cloud native security," whitepapers and guidance, 2023. [Online]. Available: <https://www.cncf.io>
- [24] The Falco Project, documentation, 2024. [Online]. Available: <https://falco.org>
- [25] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

- [26] M. Barreno *et al.*, "The security of machine learning," *Mach. Learn.*, vol. 81, pp. 121–148, 2010.
- [27] J. Gama *et al.*, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014.
- [28] National Institute of Standards and Technology, *Computer Security Incident Handling Guide* (NIST SP 800-61 Rev. 3), 2022.
- [29] CREST, professional guidance for incident response, 2023. [Online]. Available: <https://www.crest-approved.org>
- [30] S. Mahdavifar and A. A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, 2019.
- [31] R. Vinayakumar *et al.*, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
- [32] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.
- [33] A. Jobin, M. Ienca, and E. Vayena, "The global landscape of AI ethics guidelines," *Nat. Mach. Intell.*, vol. 1, no. 8, pp. 389–399, 2019.
- [34] European Parliament and Council, "General Data Protection Regulation," Regulation (EU) 2016/679, 2016.
- [35] National Institute of Standards and Technology, *NIST Privacy Framework* (Ver. 1.0), 2020.
- [36] Ponemon Institute, industry reports on breach cost and SOC strain, 2023.
- [37] Verizon, *Data Breach Investigations Report*, 2024.
- [38] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework* (AI RMF 1.0), 2023.
- [39] J. P. Anderson, *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA, USA: James P. Anderson Co., 1980.

- [40] M. Roesch, "Snort—Lightweight intrusion detection for networks," in *Proc. 13th Syst. Admin. Conf. (LISA)*, 1999, pp. 229–238.
- [41] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Boston, MA, USA: Addison-Wesley, 2008.
- [42] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quart.*, vol. 28, no. 1, pp. 75–105, 2004.
- [43] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.
- [44] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [45] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.
- [46] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.
- [47] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: <https://christophm.github.io/interpretable-ml-book/>
- [48] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell. (IJCAI)*, 1995, pp. 1137–1143.
- [49] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, 2000.
- [50] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, 2009, pp. 1–6.
- [51] National Institute of Standards and Technology, *Cybersecurity Framework (CSF 2.0)*, 2024.
- [52] Kaggle, UNSW-NB15 dataset page (version as used in experiments), 2016. [Online]. Available: <https://www.kaggle.com>

- [53] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
- [54] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 305–316.
- [55] D. W. Hubbard and R. Seiersen, *How to Measure Anything in Cybersecurity Risk*. Hoboken, NJ, USA: Wiley, 2016.
- [56] International Organization for Standardization, *ISO/IEC 27017:2015 Code of practice for information security controls for cloud services*, 2015.
- [57] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, 2018.
- [58] OWASP Foundation, *Machine Learning Security Top Ten*, 2023. [Online]. Available: <https://owasp.org>
- [59] European Union, *Artificial Intelligence Act* (consolidated text as cited), 2024.
- [60] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.
- [61] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [62] I. Ahmad, M. Basher, M. J. Iqbal, and T. Rahim, "Performance comparison of supervised machine learning algorithms for intrusion detection," *IEEE Access*, vol. 9, pp. 64857–64872, 2021.
- [63] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.
- [64] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [65] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
- [66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[67] Cybersecurity and Infrastructure Security Agency (CISA), *Secure by Design* guidance for software manufacturers, 2023. [Online]. Available: <https://www.cisa.gov>

[68] N. Papernot, P. McDaniel, S. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 582–597.

[69] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Burlington, MA, USA: Morgan Kaufmann, 2016.

[70] S. Mehta, J. H. Hightower, M. Hausenblas, and M. Boersma, *Kubernetes Security*. Sebastopol, CA, USA: O'Reilly Media, 2018.

[71] T. F. Lunt, "Detecting intruders in computer systems," in *Proc. Conf. Auditing Comput. Technol.*, 1993.

[72] G. Stoneburner, A. Goguen, and A. Feringa, *Risk Management Guide for Information Technology Systems* (NIST SP 800-30). Gaithersburg, MD, USA: NIST, 2002.

Appendices

Appendix A: Implementation Context (SIB and ACSP)

This appendix documents how the thesis themes map to the **SIB** repository (“SIEM in a Box”) and the **ACSP** (“AI Cloud Security Platform”) component, clarifying what is implemented versus what remains conceptual, and providing additional engineering detail suitable for *vivā voce* explanation or technical audit.

A.1 SIB: SIEM-in-a-Box as a Reproducible Telemetry and Analytics Stack

SIB is positioned as a self-hosted security monitoring stack oriented toward Linux and containerized workloads. Its design philosophy is pragmatic: reduce the activation energy required to obtain meaningful runtime visibility compared to assembling dozens of components manually. The stack centers on **Falco**, an open-source runtime security engine that leverages Linux kernel facilities (commonly discussed in connection with eBPF-oriented drivers in modern deployments) to observe syscalls and related context for containers and hosts [24], [5].

Falcosidekick acts as a fan-out router for Falco alerts, enabling integrations with many downstream sinks (webhooks, messaging systems, log stores). In research terms, Sidekick is the “integration boundary” where a security event becomes an enterprise workflow artifact: it is the point where latency, reliability, and authentication requirements collide with operational reality [18].

SIB also includes **observability backends** (commonly VictoriaLogs/VictoriaMetrics in one configuration, or Loki/Prometheus in another) and **Grafana** dashboards for exploration. From a scholarly perspective, this matters because “AI for security” is not only a modeling problem; it is also a **data engineering** problem: if telemetry is not retained, indexed, and correlated, then downstream ML cannot compensate for absent ground truth or missing context [17].

Additional repository capabilities—such as threat intelligence ingestion, sigma-related workflows, and optional LLM-assisted analysis—extend the system beyond a single-model demonstration. They align with industry narratives that effective SOC tooling combines **detection**, **context**, and **workflow** rather than treating ML as a drop-in oracle [3], [37].

A.2 ACSP: AI Cloud Security Platform as a Research-Oriented Detection Pipeline

ACSP is a sub-platform within the broader repository that instantiates a thesis-friendly pipeline: ingest → normalize → deterministic rules → pluggable AI scoring → persistence → notification/metrics. It is typically executed as containerized services (`acsp-api`, `acsp-processor`, `acsp-frontend`) orchestrated by Docker Compose for local reproducibility.

The **processor** receives HTTP POST payloads (commonly from Falcosidekick’s webhook output) and executes a synchronous or queued ingest path depending on configuration. Optional hardening features include webhook shared secrets, bounded in-memory queueing with HTTP 503

backpressure semantics, and Redis-backed queueing with a worker consumer for durability under burst conditions—each of which maps to classic systems patterns for overload protection [18], [32].

The **API gateway** supports dashboard reads, research metrics endpoints, internal broadcast hooks for live UI updates, and optional maintenance loops such as retention purges and periodic metrics snapshots when enabled. This separation of concerns (ingest vs query vs housekeeping) is a deliberate reflection of microservice operational practice, even when deployed monolithically inside a small stack [32].

A.3 Event Schema, Rule Baselines, and the Role of Hybrid Detection

Within ACSP, events are normalized into a canonical schema so that downstream components operate on stable fields (timestamp, severity, event type, container identifiers, and human-readable log lines). A deterministic **rule engine** provides a reproducible baseline: it encodes expert knowledge in explicit logic and yields interpretable signals (matched patterns, severity adjustments). This satisfies part of the explainability demand articulated in XAI literature: analysts can always inspect what the rule layer asserted even when the ML layer is more opaque [19], [20].

The **AI engine** is intentionally pluggable. A mock heuristic engine supports rapid UI and pipeline development without heavyweight dependencies, while optional scikit-learn models demonstrate supervised learning and isolation-forest novelty scoring on engineered features. Optional remote inference supports separation of concerns between inference serving and ingest latency budgets—an increasingly common production pattern when models grow [15].

This hybridization mirrors the literature’s “classical + ML” theme: rules catch known-bad patterns and enforce policy-shaped constraints; ML can prioritize, score risk, or detect anomalies in feature space. The thesis contribution is not claiming that one algorithm dominates all workloads; rather, it argues that **evaluation and architecture** must be co-designed so that claims remain defensible under scrutiny [14].

A.4 Evaluation Hooks: Metrics, Latency, and Responsible Reporting

ACSP includes research-oriented evaluation hooks: confusion-derived metrics when ground truth labels exist, threshold sweeps without re-ingestion where supported, latency summaries derived from audit stages, and optional “benign noise” slices to stress false-positive behavior. These mechanisms translate academic evaluation vocabulary (precision, recall, F1, FPR) into operational questions: “At what score threshold does the analyst workload become unacceptable?” and “How often do benign automation patterns look ‘malicious enough’ to cross the line?” [46], [36].

The presence of evaluation hooks also reinforces scientific honesty: a demonstration system can look impressive while silently overfitting to synthetic telemetry. Explicit reporting of dataset origin (simulator vs benchmark vs live), leakage controls (temporal splits, grouped splits), and hardware profiles is therefore not bureaucratic—it is part of the contribution [43].

A.5 Staged Autonomous Response: From Analyst Text to Policy-Gated Actions

The implementation intentionally distinguishes **guidance** from **enforcement**. Analyst-facing mitigation text can accompany each scored event to support human triage. Separately, phased automation can post structured notifications to webhook endpoints (SOAR-like integration), perform lower-risk Kubernetes metadata annotations when pod references are extractable, and enqueue higher-risk actions for approval workflows—aligning with NIST incident handling guidance that emphasizes documentation, authorization, and controlled change [28].

This staged approach is a direct response to ethics and safety literature: autonomous security actions can create irreversible harm if mis-triggered (job disruption, mistaken isolation, or denial of service through automated containment). Therefore, the engineering stance is conservative: default dry-run or high-impact gating, rate limits, and audit trails are treated as mandatory design elements rather than optional polish [33], [38].

A.6 Privacy, Retention, and Compliance-Oriented Engineering

Continuous monitoring inevitably interacts with privacy regimes. Engineering mitigations include optional redaction of common sensitive patterns in stored text, configurable retention windows for high-volume tables, and explicit documentation that benchmark datasets and cloud audit shapes are not equivalent to personal data processing scenarios—yet still require careful handling if they contain identifiers [34], [35].

A.7 Multi-Cloud Telemetry Shapes and Domain Shift

Beyond Falco-shaped events, ACSP-style architectures can ingest simplified **cloud audit** JSON shapes (e.g., AWS-flavored, Azure Activity-style, or GCP audit-like structures) to illustrate heterogeneity. Scholarly caution applies: models trained on network intrusion datasets do not automatically generalize to control-plane audit semantics; domain shift is expected and must be discussed explicitly [14], [60].

A.8 Operational Observability: Prometheus Metrics and Runbooks

Prometheus-compatible metrics endpoints support measuring ingest outcomes and latency distributions—bridging academic evaluation with SRE practice. A companion runbook document (in the repository) summarizes rotation of secrets, queue operation, and model rollback strategies—reflecting the reality that secure systems fail operationally unless humans can run them reliably [32].

Deployment reminder: Falco’s strongest realism is achieved when Falco and the ingest endpoint share a coherent Linux runtime context; developer machines may run partial stacks for engineering speed, but thesis claims about “runtime detection” should be scoped accordingly [24].

Appendix B: Research Ethics, Validity, and Professional Practice

Independent of implementation mechanics, the project implicates professional responsibilities familiar from computing ethics and security research norms. First, **do no harm**: automated containment must be bounded, reversible where possible, and subject to policy. Second, **data minimization**: collect only what is needed for the research question; justify retention. Third, **transparency**: disclose which components are heuristic, which are learned, and what datasets were used. Fourth, **security of the research artifact itself**: a prototype SOC stack can become an attack surface if exposed publicly without authentication; webhook secrets, internal API secrets, and network exposure must be treated seriously [51].

Validity concerns for DSR include **construct validity** (does the artifact measure what it claims?), **internal validity** (are results caused by the intervention rather than confounders?), and **external validity** (do findings generalize?). For AI security, external validity is often the weakest link; therefore, the thesis should explicitly scope claims and use multiple evidence types (controlled simulation, benchmark comparisons, and—where feasible—limited operational telemetry) rather than over-claiming production readiness [42].

Appendix C: Extended Methodological Notes (Sampling, Threat Model, and Reproducibility)

C.1 Threat Modeling as a Companion Artifact

A defensible security research contribution should articulate a threat model: which adversary classes are considered (opportunistic vs targeted), which assets are protected, and which attacker capabilities are assumed (credential access, lateral movement, persistence). MITRE ATT&CK provides a widely adopted language for structuring this discussion [4]. The threat model should align with telemetry: syscall monitoring emphasizes host and container behavior; cloud audit logs emphasize control-plane misuse; neither alone is a complete picture of “cloud security.”

C.2 Sampling and Label Integrity

Supervised learning claims depend on label integrity. In intrusion detection datasets, label noise and distributional shift are known issues. Operational telemetry often lacks authoritative labels; therefore, semi-supervised approaches, analyst labeling workflows, and conservative evaluation slices become part of the methodology rather than afterthoughts [13].

C.3 Reproducibility and Open Science Practices

Reproducibility includes pinned dependencies, documented environment variables, deterministic seeds where applicable, and versioned feature schemas when ML is used. These practices align with open science norms and improve the examinability of thesis artifacts [43].

C.4 Continuous Monitoring as an Engineering Contract

“Continuous monitoring” is often used as a slogan, but in systems terms it is a contract between producers and consumers of telemetry. Producers (agents, sidecars, kernel instrumentation, cloud APIs) emit events with timestamps, identifiers, and partial context; consumers (rules engines, ML services, dashboards) must tolerate loss, duplication, reordering, and skewed clocks. Academic discussions of anomaly detection sometimes abstract these realities away, yet they dominate operational outcomes: a model that is statistically sound on a CSV can fail in production because features are missing, because sampling rates change after an upgrade, or because an integration silently truncates fields [18], [17].

Therefore, a defensible methodology treats monitoring as layered: (1) transport reliability and authentication, (2) schema stability and versioning, (3) normalization and enrichment, (4) detection and scoring, (5) retention and access control, and (6) feedback loops for analyst disposition. Each layer introduces failure modes that can masquerade as “attacks” or suppress true positives if misconfigured. The SIB/ACSP reference stack embodies this layering explicitly—webhook ingestion, canonical fields, rule baselines, model scoring, persistence, and metrics—so that the thesis can discuss not only algorithms but also the scaffolding required for trustworthy measurement.

C.5 Zero Trust, Identity, and the Limits of Workload-Only Telemetry

Zero trust architectures emphasize continuous verification of identity, device posture, and session risk rather than implicit trust inside a network perimeter. For AI-driven detection, this implies that meaningful features often span workload behavior *and* identity signals (roles, privilege changes, unusual OAuth consent patterns). Runtime syscall monitoring excels at spotting execution anomalies in containers; cloud audit logs excel at spotting control-plane misuse. Neither modality alone constitutes “cloud security,” and models trained predominantly on one modality may not transfer to the other without retraining and feature redesign [4], [56].

This limitation is not a weakness of ML so much as a reminder that threat detection is inherently multi-domain. A thesis narrative should therefore avoid collapsing “cloud workloads” into a single homogeneous data generator. Instead, it should articulate which surfaces are instrumented, which adversary techniques are in scope, and which classes of attacks remain out of scope (for example, purely physical-layer attacks or insider collusion scenarios that leave no distinctive telemetry).

C.6 Software Supply Chain Security and the CI/CD Attack Surface

Modern cloud delivery relies on CI/CD pipelines, artifact registries, and infrastructure-as-code repositories. Attackers increasingly target build systems, signing keys, and third-party dependencies—sometimes achieving widespread compromise through a small number of privileged automation accounts [9]. AI can assist by flagging anomalous pipeline events, unexpected secret access, or unusual image publishing patterns, but such detection must be paired with hardening: least privilege for CI roles, immutable build provenance, and verification of artifacts prior to deployment.

From a research evaluation standpoint, supply-chain telemetry is challenging because benign automation is highly variable and adversarial actions may be rare. Class imbalance methods (resampling, cost-sensitive learning, calibrated thresholds) remain relevant [25], while explainability requirements intensify because false positives can block releases and create developer friction [19].

C.7 Concept Drift, Covariate Shift, and the Need for Monitoring the Monitor

Operational ML systems require monitoring not only of infrastructure but of model behavior: input distributions drift, label definitions evolve, and seasonal patterns change after product launches. [27] survey concept drift adaptation strategies; in security, drift is compounded by adversaries who actively probe detector blind spots [57]. A thesis-grade framework should therefore describe how drift will be detected (statistical tests on features, performance alerts on labeled slices, analyst feedback queues) and how models will be rolled back or retrained under governance [38].

C.8 Human Factors: Alert Fatigue, Trust Calibration, and Analyst Workflows

Even accurate detectors can fail organizationally if analysts distrust alerts or cannot act on them. Ponemon-style operational studies repeatedly highlight analyst burnout and tool sprawl as limiting factors [36]. AI assistance should be framed as prioritization and summarization—not as a replacement for accountable decision-making. Effective UI design, clear provenance (“rule hit + model score + top features”), and measurable false-positive rates on representative benign workloads are therefore part of the scientific story, not peripheral UI polish.

C.9 Comparative Perspective: IDS Benchmarks versus Runtime Security Events

Intrusion detection benchmarks (e.g., historically NSL-KDD and successors; more recently UNSW-NB15-style settings discussed in surveys) remain pedagogically useful for comparing algorithms under controlled conditions [30]. However, syscall-level container alerts and Kubernetes audit narratives differ in feature semantics, label availability, and noise profiles. Domain adaptation literature warns that transferring models across domains without explicit correction yields optimistic claims in training and pessimistic outcomes in deployment [60]. The methodology should therefore report multiple evaluation regimes: benchmark metrics where applicable, and separate qualitative or quantitative analyses for runtime-shaped telemetry.

C.10 Cost, Carbon, and Efficiency: Making the Economic Argument Explicit

Security leaders increasingly ask not only “does it detect?” but “at what cost?” Deep models can improve marginal accuracy yet increase inference latency and energy use—trade-offs that matter at fleet scale [15], [16]. A complete thesis discussion acknowledges selective inference (run cheap models first), batching constraints in streaming settings, and hardware dependencies (CPU vs GPU vs remote inference services).

C.11 Incident Response Integration: From Detection to Measurable Containment

NIST's incident handling guidance emphasizes preparation, detection/analysis, containment/eradication/recovery, and post-incident activity [28]. AI can accelerate analysis by clustering related alerts and suggesting playbooks, but containment actions require change management. The ACSP staging model (guidance text, notifications, lower-risk annotations, gated approvals) is an engineering instantiation of this philosophy: align automation maturity with organizational risk appetite.

C.12 Privacy Engineering Beyond GDPR Headlines

Beyond GDPR, organizations may face sectoral rules, employee monitoring restrictions, and contractual obligations with cloud providers [56]. Privacy engineering translates legal principles into technical controls: minimization, purpose limitation, retention schedules, access logging, and pseudonymization where appropriate [35]. For student projects, it is responsible to document assumptions: e.g., synthetic data vs de-identified operational samples, and whether personal data processing occurs at all.

C.13 Open Source Ecosystem Dependencies and Maintenance Risk

SIB relies on open-source components (Falco, Grafana, observability backends, Python libraries). Open source accelerates reproducibility but introduces supply-chain maintenance obligations: CVE patching, dependency upgrades, and compatibility testing. This is increasingly recognized as a security discipline in its own right [9], [23].

C.14 Generative AI in the SOC: Opportunities and Epistemic Risks

Emerging literature and product marketing discuss generative models for summarizing incidents, translating logs, and drafting response steps. Benefits include faster comprehension for junior analysts; risks include hallucinated indicators, leakage of sensitive log content into third-party model providers, and over-trust in fluent but incorrect narratives [59], [8]. A conservative thesis stance treats generative assistance as optional and demands redaction, policy controls, and human verification.

C.15 Closing Methodological Synthesis

In sum, the methodology is not merely “train a model.” It is the disciplined co-design of telemetry, evaluation, governance, and operations. The SIB/ACSP artifacts provide a concrete substrate for this synthesis: they force explicit choices about ingestion authentication, queue behavior, metrics, retention, and staged response—choices that map directly onto the professional development aims of the module and onto credible cybersecurity research practice.

C.16 Measurement Protocol for Queueing and Backpressure Experiments

Queueing experiments require explicit arrival processes and consumer throughput assumptions. The methodology records the enqueue rate, worker concurrency, maximum queue depth (if bounded), and the policy for shedding load (e.g., HTTP 503 versus unbounded memory growth) [17], [18]. Metrics exported in Prometheus-compatible formats provide time series suitable for summarizing saturation points and tail latency inflation—evidence types that tabular accuracy alone cannot capture [16], [51].

C.17 Model Risk Management: Rollback, Canary Releases, and Shadow Mode

Operational ML risk management treats promotion of a model as a release engineering decision: canary scoring, shadow mode (compute scores without acting), and rapid rollback are standard mitigations when false positives spike after deployment [38], [32]. The staged-response design of the artifact aligns with this philosophy by enabling dry-run paths and approvals before high-impact actions [28]. Methodologically, the thesis records which deployment mode was used for each reported result so that readers do not confuse exploratory scoring with enforced automation [43].

C.18 Feature Store Discipline (Lightweight) and Training–Serving Skew

Even without a full enterprise feature store, the methodology mitigates training–serving skew by documenting: which transformations occur at ingest time versus batch training time, whether statistics (e.g., normalization parameters) are fit only on training folds, and how online paths reuse those frozen parameters [60], [14]. Skew audits are especially important when remote inference services evolve independently from the ingest service [15].

C.19 Ethics Review Checklist for Student-Scale Operational Telemetry

When experiments move from purely synthetic data toward operational-like samples, an ethics checklist documents consent, minimization, access control, retention windows, and whether identifiers appear in free-text fields [34], [35]. If operational samples are not used, the checklist explicitly states that fact—avoiding ambiguous reader inference [33], [38].

C.20 Comparative Review of Tree Ensembles versus Deep Models for Security Logs

For many security-relevant tabular encodings (aggregated counts, categorical identities, time-windowed statistics), tree ensembles remain competitive and easier to deploy on CPU-only infrastructure [63], [64], [62]. Deep sequence models become comparatively attractive when raw event sequences are long and local temporal dependencies matter; however, they increase training cost and complicate debugging [66], [30]. The methodology therefore selects model families based on modality and operational constraints, not novelty alone [47], [54].

C.21 Linking MITRE Coverage to Evaluation Slices

Where MITRE tactics are used to scope threats, the methodology maps each evaluation slice to explicit tactic labels (even if coverage is partial) so that “coverage claims” remain falsifiable [4]. Partial coverage is expected; the error is implying universal defense without enumerating blind spots [72], [4].

C.22 Handoff Criteria: When Results Are “Publication-Ready” for the Module

A result is treated as publication-ready when it includes: reproducible commands or compose profiles, frozen dependency manifests, documented dataset versions, reported baselines, and an explicit limitations paragraph [43], [69]. This criterion set mirrors professional expectations for internal security research memos as much as academic grading rubrics [55], [29].

C.23 Sensitivity Analysis for Threshold and Prior Assumptions

Sensitivity analyses vary priors implicitly encoded by training procedures (class weights, oversampling ratios) and report how downstream confusion metrics and false-positive rates on benign slices respond [25], [46]. For security operations, the goal is not a single “best model” but a **family** of operating points that leadership can choose among given risk appetite [55], [36]. Presenting a small set of representative thresholds (conservative, balanced, aggressive) often communicates trade-offs more honestly than a single F1-maximizing point [47].

C.24 Documentation of Randomness and Deterministic Replay

Where randomness affects data generation or training, the methodology records seeds and provides deterministic replay instructions when feasible [69], [44]. Deterministic replay supports debugging (“why did this alert score high on Tuesday?”) and strengthens adversarial review because anomalies in scoring can be traced to inputs rather than unexplained nondeterminism [68], [57].

C.25 Security of the Evaluation Harness Itself

The evaluation harness (scripts, notebooks, compose stacks) must not introduce vulnerabilities: public dashboards without authentication, open ingestion endpoints, or leaked API keys undermine the credibility of a security research project [51], [67]. The methodology therefore treats the harness as a deliverable with its own lightweight threat model (network exposure, secret storage, dependency vulnerabilities) [51], [9].

C.26 Contrasting “Research Metrics” with “Operational Readiness Checklists”

Operational readiness checklists (patching, backups, monitoring, incident contacts) differ from model metrics yet determine whether a prototype can be exercised safely in a lab or classroom [29], [28]. The methodology explicitly separates these checklists from ML evaluation to avoid conflating “the model works” with “the system can be operated responsibly” [42], [55].

C.27 Optional Extensions: Graph Features, LLM Assistants, and Scope Control

Optional extensions—graph-derived features, LLM summarization, or richer cloud audit normalizers—are documented as **explicitly scoped add-ons** with their own failure modes (hallucination, leakage, schema drift) [59], [8]. Scope control prevents a thesis from becoming an unbounded software roadmap while still acknowledging credible future directions supported by literature [14], [59].

