Research Article

A Rule-Based Virtual Security Operations Center Using Wazuh for Real-Time Threat Detection and Incident Response

Jilika Jithendarnadh[1*], J. Balaraju[2], K Shailaja[3], G. Vishnu Murthy[4]

[1]Department of CSE, School of Engineering, Anurag University, Hyderabad, India, jithendarnadh@gmail.com, https://orcid.org/0009-0006-7308-434X
[2]Department of CSE, School of Engineering, Anurag University, Hyderabad, India, jb7443@gmail.com, https://orcid.org/0000-0002-6025-7524
[3]Department of CSE, School of Engineering, Anurag University, Hyderabad, India, kshailajacse@anurag.edu.in, https://orcid.org/0000-0002-0615-1489
[4]Department of CSE, School of Engineering, Anurag University, Hyderabad, India, drgvm189@gmail.com

Abstract

The increasing frequency and complexity of cyberattacks have created a significant need for efficient, cost-effective, and automated cybersecurity monitoring solutions. Traditional Security Operations Centers (SOCs) rely on continuous monitoring, skilled security professionals, and advanced security tools to identify threats and respond to incidents. However, these requirements can be challenging for small and medium-sized organizations due to limited budgets, staffing constraints, and technical resources. This paper proposes a rule-based Virtual Security Operations Center (VSOC) using Wazuh, an open-source security monitoring platform, to support real-time threat detection and incident response. The proposed framework integrates endpoint monitoring, log analysis, predefined detection rules, alert prioritization, and automated response mechanisms within a centralized security architecture.

The system collects security-related events from monitored endpoints, including authentication logs, system activities, file integrity changes, and suspicious network-related events. Wazuh analyzes these events using its rule-based detection engine to identify potentially malicious activities, such as repeated failed login attempts, unauthorized file modifications, privilege escalation indicators, and suspicious command execution. When an event satisfies predefined rule conditions, the platform generates an alert containing relevant information, including event severity, affected endpoint, and associated security activity. These alerts help security administrators investigate incidents and determine appropriate response actions.

The proposed VSOC architecture emphasizes rule-based automation to improve monitoring efficiency and reduce the time required to identify potential security incidents. Configured active response mechanisms can execute predefined actions, such as temporarily blocking an identified IP address or applying other approved containment measures, depending on the detection rule and organizational security policy. The framework also supports centralized visibility into endpoint security events, enabling administrators to investigate suspicious activities through a unified dashboard. Appropriate validation, access controls, and response safeguards are incorporated to minimize false positives and prevent unintended disruption of legitimate services.

The proposed approach provides a practical foundation for implementing an accessible and scalable security monitoring environment without requiring a fully staffed traditional SOC. Its effectiveness can be evaluated using controlled attack simulations, detection accuracy, false-positive rates, alert generation time, and incident response time. The study contributes a structured approach to integrating rule-based threat detection and automated incident response through Wazuh, with potential applications in educational institutions, small businesses, and resource-constrained IT environments. The framework aims to strengthen organizational security visibility and support timely responses to emerging cyber threats.

Keywords: Virtual Security Operations Center, Wazuh, Rule-Based Detection, Real-Time Threat Detection, Incident Response, Security Information and Event Management, Automated Security Monitoring.

1. Introduction

Security monitoring is no longer an activity limited to large enterprises. Universities, municipal offices, start-ups, and medium-sized service providers operate cloud-connected identities, employee laptops, application servers, and remotely administered endpoints that generate substantial security telemetry. At the same time, adversaries exploit weak authentication, misconfiguration, stolen credentials, and delayed patching in ways that are difficult to recognize from an isolated log entry. The 2025 European threat landscape and the 2026 Verizon breach report both reinforce the operational relevance of monitoring heterogeneous activity and prioritizing incident response, although their populations and methodologies differ (European Union Agency for Cybersecurity, 2025; Verizon, 2026). The core problem addressed here is therefore not the absence of security data, but the inability of

resource-constrained defenders to translate that data into timely and defensible decisions.

A traditional Security Operations Center (SOC) coordinates collection, analysis, triage, escalation, containment, and recovery through trained personnel and specialized tooling. Such an organization can be highly effective, but round-the-clock staffing, process maturity, and infrastructure are expensive to establish and maintain. SOC research identifies analyst overload, ambiguous responsibility, and integration difficulties as continuing obstacles (Onwubiko & Ouazzane, 2022). A virtual SOC (VSOC), as used in this paper, denotes an operational model in which centralized technical controls provide security visibility across distributed endpoints and support part-time or remote analysts. It is not synonymous with an unattended intelligent system, nor does software alone constitute a mature security operations function.

Security information and event management (SIEM) has emerged as a practical foundation for aggregating logs and analyzing security-relevant patterns. SIEM reviews describe the value of centralized collection, event normalization, correlation, and investigation, while also warning about deployment complexity and noisy alerts (González-Granadillo et al., 2021; Macaneata, 2024). Open-source platforms can remove a software-license barrier, but the associated work of rule engineering, storage administration, monitoring coverage, and operational ownership remains. For this reason, the design objective is cost-aware rather than cost-free: the proposed solution substitutes clearly scoped engineering and procedural controls for selected functions that a larger SOC might otherwise perform manually.

This paper proposes a rule-based VSOC using Wazuh, whose architecture separates monitored agents, analysis and management services, indexing, and dashboard functions (Wazuh, n.d.-c). The proposed system detects predefined patterns including login failures, protected-file changes, potentially suspicious elevation activity, and selected command-execution behaviors. Only detections with sufficient confidence and explicit organizational authorization may trigger stateful containment; other detections remain reviewable alerts. “Real time” is treated as a measurable end-to-end latency target rather than an unconditional guarantee, because collection modes, network connectivity, queues, indexing, and analyst decisions all introduce delay.

The paper makes four contributions. First, it specifies a deployment architecture with explicit collection, processing, storage, and control boundaries. Second, it presents a transparent detection-and-response policy connecting rule semantics, asset criticality, alert priority, and action safety. Third, it describes a controlled validation protocol with traceable scenarios and quantitative measures rather than fabricated empirical outcomes. Fourth, it analyzes operational risks, including false positives, agent blind spots, response amplification, evidence preservation, and scaling trade-offs. The emphasis is deliberately on reproducibility and responsible automation; future research can compare these rules with behavioral and machine-learning alternatives once a validated baseline exists.

1.1 Problem statement, aims, and research questions

The research problem is formulated as follows: how can a small organization construct a centrally administered, auditable, rule-based security monitoring and response capability from an open-source platform while avoiding uncontrolled automated enforcement? This problem has a technical dimension—reliable telemetry, parsing, temporal correlation, and response execution—and an organizational dimension involving approval boundaries, retention, incident ownership, and service continuity. The aims are to propose the system, identify deployable rule families, define safe containment criteria, and establish an evaluation protocol suitable for later testing. The target environments have limited dedicated analyst time and may mix Windows and Linux machines, but they are assumed to possess permission to administer the enrolled hosts.

The study examines four research questions: RQ1 asks how effectively the proposed collection-and-rule pipeline can recognize selected, observable attack behaviors under controlled laboratory conditions; RQ2 asks how end-to-end alert latency and response latency change with event volume and deployment topology; RQ3 asks what proportion of benign activities incorrectly trigger alerts or disruptive containment; and RQ4 asks whether transparent policies and reversible actions make a virtualized operating model manageable for small teams. The

questions define planned measurements, not achieved performance. Their answers require ground-truth event labels, repeated trials, documented environmental settings, and precise separation between detection success and effective containment.

2. Background and related work

SIEM integrates many event sources into a common analytic plane, but its utility depends on log selection, timestamp quality, parser coverage, and contextual enrichment. Foundational NIST guidance recommends systematic handling of computer security logs, including infrastructure planning, secure storage, analysis procedures, and retention (Kent & Souppaya, 2006). Guidance for intrusion-detection and prevention systems likewise distinguishes host-based and network-based approaches and encourages layered monitoring instead of reliance on a single detection mechanism (Scarfone & Mell, 2007). These foundations remain relevant even as modern tools provide web-based dashboards, cloud connectors, automation engines, and richer event schemas. The proposed VSOC follows the layered principle by combining endpoint monitoring with optional network-device logs.

The operational difference between a SIEM appliance and a SOC function is important. A SIEM indexes evidence; an operating center decides what matters, who owns the incident, when intervention is justified, and how the outcome is recorded. Design-science work on SIEM adoption proposes that organizational fit and deployment procedure should be considered alongside tool selection (Rosenberg et al., 2023). For smaller institutions, this implies creating response playbooks, access roles, and service-level expectations before turning on high-impact automation. Without these processes, deploying a dashboard may increase the apparent number of threats without improving the organization's ability to resolve incidents.

Contemporary SOC literature increasingly investigates artificial intelligence, machine learning, and analyst-assistance systems. A 2025 systematic review describes potential benefits for triage, prediction, and automation but also details significant engineering and validation challenges (Khayat et al., 2025). A separate review demonstrates the breadth of research on AI across security operations while drawing attention to explainability, available datasets, and uneven attention to recovery (Giarimpampa et al., 2025). These studies motivate a conservative design choice: use deterministic rules as the initial operational baseline, because conditions can be inspected, unit-tested, versioned, and explained to administrators. A future learning component should be benchmarked against that baseline rather than assumed superior.

Rule-based detection is neither equivalent to perfect attribution nor restricted to a single static string match. Conditions may include structured fields, event frequencies, time windows, preceding rules, and asset-specific lists. Such logic has a clear operational advantage: an analyst can inspect which condition fired and reason about why. Nevertheless, attackers can change syntax, throttle attempts below threshold values, exploit unmonitored channels, or intentionally generate harmless events to cause alarm. The research therefore treats rule tuning, coverage analysis, and control testing as recurring lifecycle activities. High recall without tolerable alert volume is not a sufficient operational outcome, just as low false-positive counts obtained by suppressing almost all alerts would be misleading.

Wazuh provides endpoint agents and centrally managed analysis, with rules and decoders that can be extended to local log formats. Its server receives data from agents and eligible external sources and evaluates events using a ruleset (Wazuh, n.d.-t). The platform supports additional visibility, including file integrity monitoring, software inventory, configuration assessment, and vulnerability detection, although these capabilities have different scanning and freshness characteristics. A VSOC designer must therefore distinguish alerting on a change as it occurs from periodically discovering an exposure. Vulnerability findings and security posture results should influence investigation priority but should not be presented as proof that a live attack has occurred.

The principal research gap addressed here lies between product installation tutorials and complete SOC architectures. Product documentation describes individual functions, while empirical security literature often

assesses detectors or analytical techniques in isolation. A usable small-organization VSOC requires explicit connection among evidence acquisition, decoding, rule evaluation, triage, human authorization, response verification, and recovery. It additionally needs a publishable test protocol that counts missed attacks and harmful automations, not only attractive dashboard screens. This paper consolidates those elements into a cohesive reference design and transparently identifies which claims remain hypotheses until the system is deployed and benchmarked.

3. Research design and system requirements

The work follows a design-oriented systems methodology: derive requirements from security operations guidance and common endpoint threats; specify a reference architecture; define a rule-and-policy artifact; propose repeatable experiments; and describe how observations should be reported. NIST Cybersecurity Framework (CSF) 2.0 organizes risk governance and operations through Govern, Identify, Protect, Detect, Respond, and Recover functions (Pascoe et al., 2024). The 2025 NIST incident-response revision integrates response across that broader risk-management cycle (Nelson et al., 2025). Consequently, the proposed system is not assessed only by whether an alert appears. It is assessed by whether required evidence is collected, response authority is appropriate, actions are verifiable, and recovery can be documented.

Functional requirements are defined for endpoint enrollment, centralized log receipt, normalized event fields, predefined and custom rules, severity assignment, dashboards, case identifiers, notification channels, and optional active responses. Nonfunctional requirements cover measured timeliness, access restrictions, auditability, resilience under short network outages, storage retention, manageable configuration drift, and graceful degradation. The architecture must permit a small initial installation without hard-coding that topology as suitable for every workload. A laboratory can use one manager, one indexer, and one dashboard, while operational growth may require separating components and evaluating high availability, sizing, and secure backups.

The threat model focuses on observable attacker behaviors rather than arbitrary labels such as “hacker present.” In scope are repeated authentication failures against an enrolled service, suspicious success following many failures, changes to identified configuration or policy files, selected attempts to obtain elevated privileges, and recognizable command-execution indicators. The corresponding ATT&CK categories include brute force, valid-account abuse, elevation-control abuse, file-permission changes, and command interpreter use (MITRE, n.d.-b; MITRE, n.d.-e; MITRE, n.d.-a; MITRE, n.d.-d; MITRE, n.d.-c). Each category includes legitimate analogues, so an alert represents a detection hypothesis that needs contextual interpretation.

The principal trust boundaries are the endpoint-to-server channel, the event-processing pipeline, the index-and-dashboard boundary, and the manager-to-agent response channel. An adversary who disables local logging, tampers with clocks, gains access to an agent, or floods the manager can degrade the quality of downstream judgments. The design therefore assumes secure administration, restricted inbound ports, hardened management hosts, authenticated agent enrollment, and periodic health checks. It does not assume that every host is observable or that an attacker cannot target the security tooling. Explicit statement of these limitations prevents the VSOC from becoming a single point of unexamined confidence.

The unit of analysis is a labeled security scenario that may contain one or multiple raw events. For each run, the evaluator records a ground-truth scenario identifier, endpoint and operating system, generation time, receipt time, analysis time, dashboard availability time, case disposition, and action start and completion timestamps. Repeated trials should be performed under normal background activity as well as burst loads. Administrative decisions, allowlists, custom rules, and deployed package versions are version-controlled. This design permits later investigators to distinguish missing telemetry from parser failure, absent rules, incorrect severity, and failed containment rather than collapsing every problem into a generic “undetected” outcome.

3.1 Requirements traceability

Table 1 maps system requirements to observable outputs and failure criteria. The table is deliberately test-

oriented: instead of claiming that a feature exists, it defines what a tester must demonstrate. This approach aligns with continuous-monitoring guidance, which links observation to risk decisions and evaluation of control effectiveness (Dempsey et al., 2011). For example, “supports alerting” becomes a requirement that a known injected event produces a correctly attributed alert within a recorded interval. Likewise, “automated response” is valid only when the action is authorized, logged, limited in scope, and shown to expire or be manually reversible. A requirement without a corresponding acceptance condition would be difficult to audit or reproduce.

Table 1. VSOC requirements and verifiable acceptance criteria

Requirement	Proposed mechanism	Acceptance criterion
Endpoint visibility	Agent inventory and required log sources	Injected source event is attributed to the correct host
Rule transparency	Versioned rules and documented fields	Positive/negative samples produce expected match outcomes
Central investigation	Indexed alerts and role-limited dashboard	Analyst can locate raw evidence and rule metadata
Safe response	Allowlist, scoped action, timeout, approval policy	Unauthorized block rejected; valid action reverses
Resilience	Health checks, queues, configuration backup	Disconnect/restart is detected and recovery recorded
Auditability	Case IDs, action log, retention rules	End-to-end event–decision trail can be reconstructed

4. Proposed Wazuh-based VSOC architecture

Figure 1 presents the logical architecture, separating data acquisition, analysis, persistence, and operations. Wazuh agents on Windows and Linux endpoints collect selected operating-system and application events, while eligible network infrastructure can forward logs through a supported ingestion method. The Wazuh manager performs pre-decoding, field extraction, ruleset evaluation, and alert generation. Indexed events and findings support searching, trend analysis, and investigation in a centralized dashboard. Alert enrichment with host role and asset criticality is a VSOC policy layer, not an assumption that Wazuh automatically knows business importance. The same distinction applies to case assignment and any custom notification workflow.

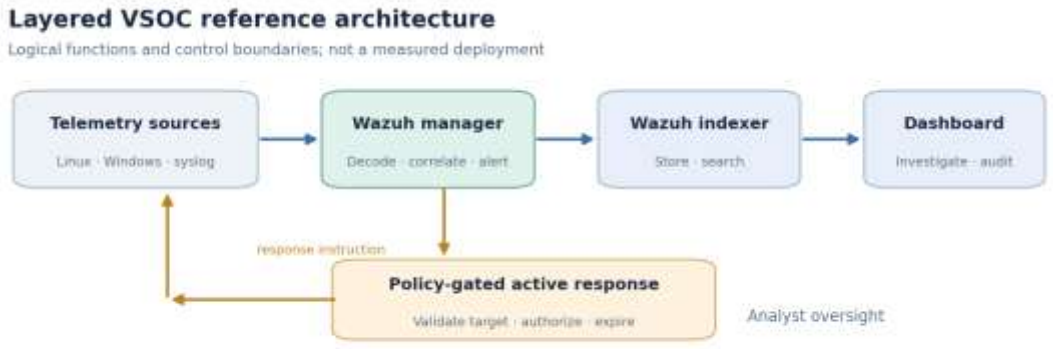


Figure 1. Proposed VSOC architecture and policy-gated response boundary (original schematic).



Figure 2. Endpoint event flow through collection, analysis, indexing, and triage (original schematic).

Log source configuration matters as much as the server architecture. Authentication events, audit trails, PowerShell operational records, application logs, and system services have different formats, volumes, and privacy implications. Wazuh supports log collection from endpoints and network or application sources, with explicit configuration for monitored files and supported operating-system logging facilities (Wazuh, n.d.-l; Wazuh, n.d.-e). The deployment should maintain a source register identifying the owner, retention policy, parsing state, expected collection frequency, and test event for every enabled source. This register exposes coverage gaps and reduces the risk of describing an endpoint as protected merely because its agent appears connected.

Figure 2 shows the event lifecycle. After an endpoint emits an observation, the collection component forwards it to the server, where header processing, decoding, and rule matching turn a heterogeneous record into an interpretable security finding. Wazuh documents these phases as pre-decoding, decoding, and rule matching (Wazuh, n.d.-k). A successful match can generate a structured alert with a rule identifier, rule level, agent identifier, timestamp, and available event fields. Indexing and dashboard retrieval are subsequent stages; their delays should be measured separately from the time at which the manager itself generated the alert. Preserving the raw event or an appropriate audit record is necessary for later investigations.

Centralization creates operational efficiencies because one operator can inspect many endpoints without directly logging into each computer. However, it also concentrates credentials and evidence in a management plane that becomes security-critical. The deployment uses network segmentation, host firewall controls, administrative separation, backup of rules and configurations, encrypted transport where supported, and retention appropriate to the organization's legal obligations. Wazuh role-based access control can limit permitted API operations based on privileges (Wazuh, n.d.-n). A viewer role should not be capable of changing rules or launching containment, while an engineer who modifies response policies should not be the only person reviewing high-risk policy changes.

A small proof-of-concept installation may co-locate components if hardware permits, but that is an engineering convenience rather than a universal production recommendation. Index storage requirements grow with log volume, field cardinality, and retention duration; manager CPU and queue pressure rise with rule complexity and event bursts. At larger scale, component isolation and additional capacity must be evaluated using load tests. The proposed architecture is scalable in the sense that it has identifiable components and expansion paths, not in the sense that an untested configuration has already been demonstrated to handle a specified number of endpoints. Storage and continuity calculations are included in the evaluation procedure rather than asserted as performance findings.

System inventory and configuration posture provide supplementary context. The Wazuh Syscollector capability inventories software and other host attributes, helping operators associate a high-severity alert with the affected software or endpoint profile (Wazuh, n.d.-q). Security configuration assessment compares observed settings with predefined policies and can reveal hardening gaps (Wazuh, n.d.-p). Vulnerability detection correlates software inventory with vulnerability intelligence, producing exposure findings that may increase a case's investigation priority (Wazuh, n.d.-s). These are complementary data products: a known unpatched package is not evidence that it has been exploited, and a successful file-integrity signal does not establish the identity or motivation of the process that modified the file.

The operational dashboard is a work surface rather than an independent detection authority. Analysts should view the event timeline, affected asset, source address when present, mapped attack behavior, prior alerts, relevant configuration changes, containment history, and supporting raw records. An alert may be stored in manager-side JSON or text output and forwarded for indexing; Wazuh describes alert management and configurable thresholds for emitting alerts (Wazuh, n.d.-b). The proposed operating procedure additionally records case status—new, triaged, investigating, contained, recovered, or closed—and requires a reason for closure. This prevents simple alert acknowledgment from being mistaken for verified remediation.

5. Rule-based threat detection design

The detection policy consists of a source contract, a parser or decoder, one or more alerting rules, contextual priority criteria, and a documented response disposition. The contract specifies the minimum fields required for a detection to be meaningful. For a repeated-login rule, the fields may include endpoint, application, authentication outcome, account, and source address where available; for a file-change rule, they include path, change type, and protected scope. Wazuh rules support XML-defined conditions, including matching decoded content and correlation-related constructs (Wazuh, n.d.-o). The reference system uses built-in detections where appropriate and confines local modifications to separate custom configuration so upgrades do not silently replace authored logic.

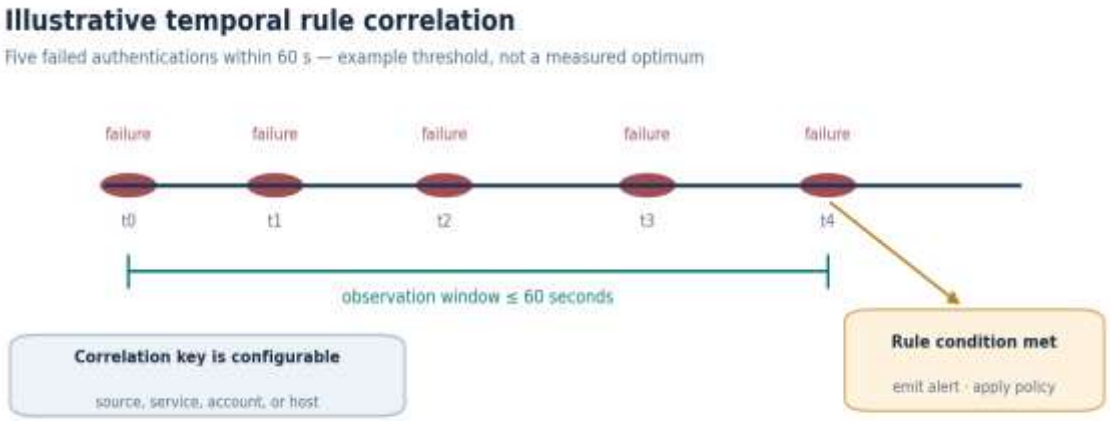


Figure 3. Illustrative temporal correlation of authentication failures; thresholds are examples only (original schematic).

Table 2. Proposed rule catalogue and response dispositions

Rule family	Observable condition (illustrative)	ATT&CK mapping	Initial disposition
Authentication failures	Multiple failures grouped by source/account in time window	T1110	Alert; timed block only after validation
Unusual successful login	Success after recent failures plus anomalous context	T1078	High-priority analyst triage
Protected-file change	Creation/edit/deletion in watched directory	T1222 (when permission-related)	Change-ticket check and evidence capture
Privilege indicator	Suspicious sudo/elevation or admin-setting change	T1548	Analyst review; approval required
Command execution	Suspicious process/command telemetry when enabled	T1059	Review command context; do not block blindly
Missing expected telemetry	Agent disconnected or monitored source silent	Monitoring control gap	Create operations-health ticket

An important distinction is that built-in rule levels are platform priorities, not universal business-risk scores. A severe event on a noncritical test host may be less urgent than a moderate event on a production identity server. The VSOC therefore proposes a decision function P = f(L, A, C, R), where L is the rule-level category, A is asset criticality, C represents available contextual corroboration, and R expresses estimated harm from an incorrect response. This is a policy formula rather than a measured or Wazuh-native scoring algorithm. A transparent mapping can initially use three or four tiers and must identify which combinations permit automatic containment, prompt analyst review, or simply record the event.

Custom detectors should be added only after examining representative benign and malicious samples. Wazuh supports locally authored rules and recommends a dedicated range for custom rule identifiers, providing flexibility to adapt logic without editing distributed defaults (Wazuh, n.d.-h). A custom decoder may be needed when an application's event text lacks an out-of-the-box parser; Wazuh provides decoder configuration mechanisms for extracting named fields from such records (Wazuh, n.d.-g). Decoder failure should produce a specific test failure because a rule might be correct yet incapable of matching raw input. The correct development sequence is sample acquisition, parser test, rule test, false-positive exploration, review, and staged rollout.

Figure 3 illustrates a general correlation pattern using a sliding observation interval. For example, repeated failed authentication events from one source toward one service may constitute suspicious activity when a documented count threshold is reached within a defined window. Correlation keys must be chosen carefully: an institutional network address translation gateway could aggregate hundreds of legitimate users, whereas a username-only key may combine independent distributed probes. The paper does not prescribe a universally correct threshold or reproduce a vendor-specific rule ID as though it were invariant across versions. Instead, experiments compare several thresholds and grouping keys using positive and negative scenarios.

A rule for authentication activity should differentiate failed password entry, nonexistent account, successful session establishment, administrative login, and account lockout. Brute-force indicators are often more convincing when followed by an unexpected successful login, but such a sequence can also describe a legitimate user recovering from a forgotten password. Identity information and login context must therefore be considered before blocking an address. The MITRE brute-force and valid-account techniques offer useful vocabulary for describing hypotheses and coverage, not authoritative proof of attacker behavior (MITRE, n.d.-b; MITRE, n.d.-e). The safer baseline is a high-priority alert, with reversible network blocking reserved for well-scoped, repeatedly validated

conditions.

File integrity monitoring (FIM) is applied to security-relevant paths such as authentication configuration, privileged-access policy, service startup definitions, and a small set of important application files. Wazuh's syscheck configuration supports monitored directories and options for real-time or who-data collection subject to supported platforms and modes (Wazuh, n.d.-i). Change detection should be tested with additions, edits, deletions, permission changes, and expected administrative updates. Broad monitoring of every rapidly changing file can raise costs and inflate benign alerts. A maintenance register and signed change ticket, where available, can provide context for triage without unconditionally suppressing the evidence of changes made during a maintenance period.

Privilege-related monitoring watches for observable changes such as suspicious use of sudo, creation or modification of privileged accounts, elevation-control configuration modifications, and unusual access to protected administrative resources. These indicators must be mapped to available operating-system telemetry; generic claims about detecting privilege escalation are insufficient. Legitimate software installation, system maintenance, and routine administrative sessions are major benign comparators. The ATT&CK elevation-control and permission-modification entries provide behavior descriptions that help build a test matrix (MITRE, n.d.-a; MITRE, n.d.-d). Confirming an attacker obtained administrator rights would ordinarily require additional evidence beyond one rule match.

Command-execution monitoring requires particular care because both administrators and adversaries use interpreters. A rule may alert on a narrow, high-risk command pattern, a suspicious combination of process arguments, or an unapproved executable path, provided the endpoint emits the relevant telemetry. Wazuh command monitoring can collect command output at configured intervals, but periodic sampling is not equivalent to full process-creation telemetry (Wazuh, n.d.-d). For actual process observations, operating-system audit or event logging must be specifically enabled and collected. The security value of a command rule depends on visibility, parser fidelity, execution context, and attack variation; fixed strings should not be presented as comprehensive behavioral detection.

Table 2 summarizes a proposed detection catalogue. Each row defines an observable signal, correlation idea, relevant ATT&CK behavior, and disposition. The mappings help audit coverage but must not be interpreted as complete protection against the named technique. Rule-level validation should use Wazuh's log testing facility to check parsing and rule matches against controlled samples; the tool enables verification before rules enter operational use (Wazuh, n.d.-r). After these unit tests, end-to-end tests must confirm agent receipt, alert indexing, and—when approved—response behavior. A simulation that only feeds a line to a parser cannot establish the latency or reliability of the entire deployed pipeline.

Rule governance is continuous. Every custom rule has an owner, purpose, expected event fields, allowed log sources, sample positive input, sample negative input, threshold values, response linkage, and revision history. Rules should be disabled or revised when changing application formats render them invalid. A weekly review can examine a sample of dismissed alerts and a sample of escalated events to identify poor specificity, missing context, and unseen variants. Version control enables regression tests after Wazuh upgrades. The mapping of detections to ATT&CK techniques can be displayed through Wazuh's supported framework integration, but mappings themselves should be manually validated so a broad technique tag does not misrepresent a rule's actual evidence (Wazuh, n.d.-m).

6. Incident triage and automated response

The response process begins with a distinction among event, alert, security incident, and confirmed compromise. An event is an observed record; an alert is a rule-supported signal that warrants evaluation; an incident is an adverse or potentially adverse event requiring organizational handling; and a compromise is a stronger conclusion supported by investigation. This distinction reduces the risk of reporting every configuration change or failed login

as a breach. The operational lifecycle in Figure 4 follows receipt, enrichment, initial triage, policy check, optional containment, verification, recovery, and case closure. It aligns with the integrated detection-and-response orientation of NIST's current incident response guidance (Nelson et al., 2025).

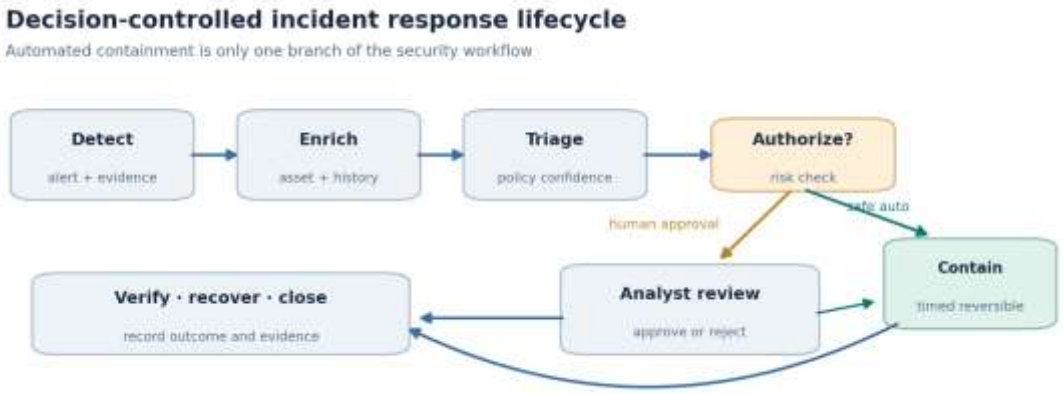


Figure 4. Incident-response decision lifecycle with human authorization and response verification (original schematic).

Wazuh provides an Active Response mechanism that can launch predefined commands or custom scripts when configured criteria match (Wazuh, n.d.-a). This makes short-latency containment feasible for a narrow class of events, but it creates an additional attack surface and an availability risk. The proposed response policy has three classes. Class A permits notification and evidence capture only. Class B allows a reversible, bounded containment action following a high-confidence, validated trigger. Class C requires explicit analyst approval because it can disable an account, interrupt a critical process, or block access in a way that affects multiple users. Approval requirements apply even if a response script could technically be executed automatically.

The initial response action is a temporary source-address block only when the source address is reliably attributable, externally sourced or otherwise suitable for blocking, outside the organization's protected allowlist, and associated with a repeated malicious-looking pattern. Preventing accidental blocking of management channels, domain controllers, shared gateways, proxies, and trusted infrastructure has priority over speed. Wazuh's configuration specifies response command, location, triggering criteria, and optional timeout (Wazuh, n.d.-j). A timed action must be verified on the relevant platform, including successful restoration, rather than assuming that a timeout guarantees recovery under all failures.

A custom script should parse structured alert data, validate mandatory fields and types, reject private or protected targets where policy requires, enforce a maximum duration, avoid shell interpolation of untrusted fields, and create an independently searchable action record. Wazuh documentation describes how custom active-response scripts receive structured input and how stateful workflows can support add and delete actions (Wazuh, n.d.-f). Additional VSOC safeguards include idempotency, per-source cooldowns, rate limiting, and a “kill switch” that stops new automatic blocks without deleting the investigative evidence. Scripts should fail closed with respect to authorization but fail safely regarding production availability: when attribution is uncertain, issue an alert rather than performing destructive containment.

Figure 4 also represents the human escalation path. A designated analyst should investigate source provenance, account history, concurrent host alerts, known maintenance, and business context. Escalation targets are assigned for after-hours alerts and for responses involving high-value services. Containment is not treated as closure: the investigator verifies whether the malicious activity ceased, checks for persistence and unauthorized changes,

documents root-cause hypotheses, and determines whether credentials or configurations need restoration. The case includes rationale for action and reversal, links to supporting logs, affected systems, and the final disposition. A case can be closed as benign, attempted attack, confirmed incident, or inconclusive, with reason codes.

Response safeguards must be exercised as rigorously as detection rules. Test cases include missing source address, malformed event fields, simultaneous alerts for one address, a protected management IP, a service account that generates repeated errors, and an unavailable endpoint. The test should confirm that no unapproved action executes in these conditions. Wazuh alert thresholds influence which events are emitted and potentially passed into downstream automation, so rule levels and response triggers should be separately inspected (Wazuh, n.d.-b). A configuration review should specifically reject catch-all response policies based only on severity when the event family and target have not been constrained.

NIST SP 800-53 includes control families relating to audit records, least privilege, system monitoring, and incident handling, which provide a wider governance context for the VSOC (Ross, 2020). A practical implementation maintains role separation between policy authors, responders, and auditors; protects dashboard authentication; restricts API credentials; documents retention and destruction schedules; and verifies backups of configurations and incident evidence. Audit records of administrator changes and response executions should not be writable by ordinary monitored endpoints. These controls are particularly important in a small team, where one person may hold several operational responsibilities and independent review must be supplied through periodic management or peer assessment.

7. Implementation blueprint

Figure 5 depicts a proposed laboratory with one monitoring server, two Linux virtual machines, one Windows test endpoint, and a controlled traffic or event generator on an isolated virtual network. The generator produces only authorized test traffic against enrolled systems. The configuration should be recorded using Wazuh version, operating-system build, CPU and memory allocations, storage type, network latency, time synchronization status, and enabled log sources. The intent is to create a reproducible minimum environment, not to assert that this laboratory has been built for the current paper. Production installation would require separate risk review, sizing, hardening, backup planning, and authorized change management.

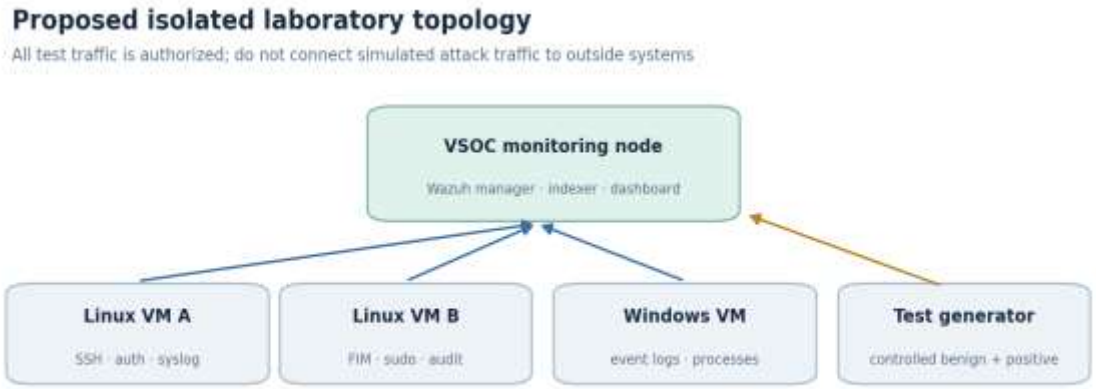


Figure 5. Proposed isolated VSOC testbed configuration for controlled experiments (original schematic).

The implementation begins with administrative preparation: identify the managed assets, classify sensitive systems, decide retention rules, establish service accounts and privileged access procedures, and isolate management services from untrusted clients. The collector then enrolls test endpoints and verifies both connectivity and receipt of an innocuous known event. Next, the engineer enables operating-system authentication and relevant audit sources, activates FIM for selected directories, and collects a benign baseline. A successful agent connection is not sufficient evidence of telemetry coverage. Every source must demonstrate that its expected event appears in the analysis and investigation view with accurate attribution.

The second phase implements rule engineering. Researchers should place custom parsing logic in the supported customization locations, test logs with the rule-testing utility, and archive representative positive and negative samples. Individual rule conditions are reviewed for unsupported regular-expression assumptions, unintended case sensitivity, and noisy generic matches. Each proposed threshold is justified against a trial workload rather than copied without adaptation. The ruleset should include explicit grouping by relevant source, account, service, or endpoint where this is needed for correlation. Its regression suite must be rerun after parser updates, rule tuning, or system upgrades.

The third phase configures the alert workflow. Investigators define dashboard views for endpoint, rule family, time window, asset criticality, and case status. An asset inventory links technical identifiers to a business owner and service impact category. Simple notifications can be enabled first; active response is introduced only after isolated tests show acceptable alert precision and rollback behavior. The response runner operates with minimum required privileges and its actions are separately logged. This phased approach favors explainability and governance over a maximal feature count. It also supports a small organization that can staff only business-hours investigation plus an emergency escalation route.

The final phase validates operational continuity and failure states. Tests should disconnect an agent temporarily, delay an event source, saturate the log path with benign records, restart the manager in a maintenance window, and check that monitoring health is distinguishable from security silence. Local and central timestamps need a common synchronization policy. Securely stored configuration snapshots make recovery from a bad rule change feasible. The expected operating procedure includes a daily health review, weekly rule-quality review, monthly access audit, and a scheduled restoration exercise. These cadences are illustrative governance settings and can be adjusted to the organization's risk appetite and available staffing.

8. Experimental methodology and evaluation protocol

The proposed experiments answer the research questions by testing controlled malicious-pattern scenarios and normal operations under comparable workloads. Each scenario begins with a ground-truth plan specifying authorized endpoint, triggering operation, timestamp source, expected telemetry, expected rule family, permitted response, and cleanup procedure. The evaluator should capture endpoint source records alongside Wazuh alerts and active-response logs. Testing takes place only inside the lab or on systems for which the investigator has explicit authorization. It is unnecessary to use real malicious payloads or compromise external services: synthetic events, harmless configuration changes, and local authentication attempts can exercise the relevant observability and response paths.

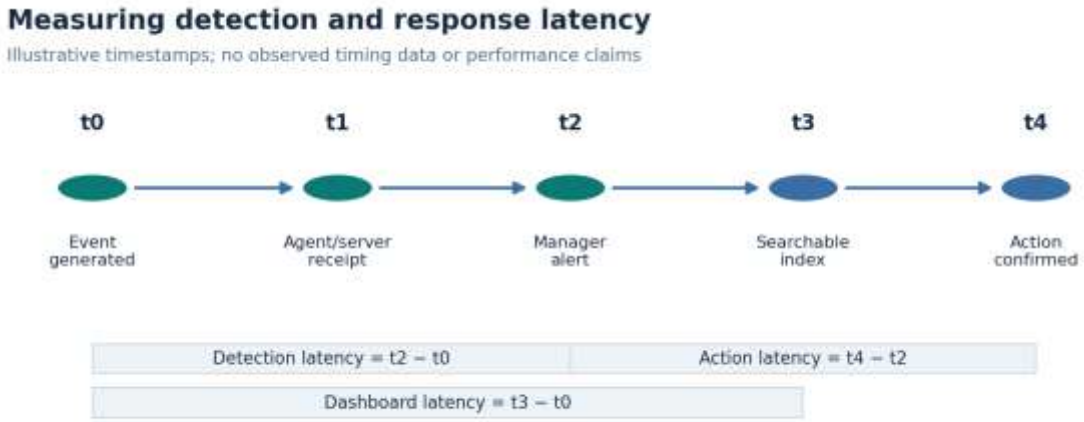


Figure 6. Defined timestamps for measuring event-to-alert and alert-to-action latency (original schematic).

Table 3. Controlled positive and benign evaluation scenarios

Scenario	Ground truth	Evidence expected	Key confounder to test
Repeated SSH failures	Positive	Auth records; source group; fired rule	Single user mistyping password
Post-failure login	Positive/contextual	Failures and subsequent authenticated session	Legitimate password recovery
Protected-file edit	Positive/contextual	Path, change type, agent and timestamp	Approved maintenance update
Privilege-related event	Positive/contextual	Audit or administrative-event record	Scheduled administrator task
Suspicious interpreter use	Positive/contextual	Process/command audit record	Benign management script
Normal user traffic	Negative	Benign activity without malicious classification	NAT/shared source address
Approved configuration change	Negative	Expected FIM signal with benign disposition	Missing change metadata
Telemetry interruption	System-health test	Disconnect/reconnect evidence	Silence misread as safety

Positive scenarios include a constrained sequence of failed logins, a simulated administrative authentication after repeated failures, modification of a watched configuration file, an observable privileged command or permission change, and a test command execution event. Negative scenarios include legitimate password mistakes, an approved software update changing monitored files, an authorized administrative maintenance session, scripted backup operations, and shared-network addressing where numerous benign users appear behind one source address. A stronger evaluation also includes evasive variants, such as distributing attempts across long time

intervals or changing event syntax. These variants reveal which behaviors the selected rules are not designed to recognize; they should not be silently counted as successes.

Table 3 specifies the scenario matrix and expected evidence. At least thirty independent positive and thirty benign control trials per relevant rule family are desirable for an initial exploratory study, although a larger and more diverse sample is required for precise estimates of rare false positives. Trials must be randomized or interleaved where possible to reduce chronology and cache effects, and scenarios should be repeated under at least two load regimes. Threshold tuning and final performance testing should use distinct event sets to avoid overfitting. The report should record unsuccessful setup attempts and missing log events instead of excluding them as inconvenient observations.

For evaluation, define true positive (TP) as an alert matching a labeled positive scenario within the allowed window; false negative (FN) as a labeled positive scenario without that alert; false positive (FP) as an alert generated for a labeled benign scenario under the specified scoring unit; and true negative (TN) as a labeled benign scenario without the alert. Precision = TP/(TP + FP), recall = TP/(TP + FN), specificity = TN/(TN + FP), and F1 = 2TP/(2TP + FP + FN). Zero-denominator cases are reported as undefined. Analysts must choose event-level, scenario-level, or time-window-level scoring before collection, because changing units can make the same detector appear artificially precise or noisy.

Detection latency is measured as the difference between a trusted endpoint event occurrence timestamp and manager alert creation, where both clocks are synchronized and timestamp semantics are documented. Dashboard latency is separately measured from source event to index visibility; response latency is measured from alert trigger to confirmed completion of the containment action. Mean values alone conceal long delays, so median, 95th percentile, maximum, interquartile range, and number of timeouts should be reported. “Real time” is interpreted relative to a declared operational target and measured distribution; it is not a promise that every event will be processed instantaneously. Figure 6 illustrates the latency path and its separately observable segments.

Under light and burst conditions, the evaluator should vary the number of monitored endpoints and the background event rate while recording central CPU utilization, peak memory use, queue behavior, disk growth, index size, and alert lag. Workload mixes should include both low-cost routine events and the more expensive rule families selected for the study. For each configuration, repeated trials allow confidence intervals for detection and timing metrics, using bootstrap intervals when latency distributions are skewed. Sample sizes and statistical tests should be specified prior to analysis. If two response configurations are compared, tests should consider paired runs or controlled differences in input traffic instead of comparing arbitrary days with unrelated background workloads.

False-positive investigation is central to practical utility. The number of false alarms per endpoint-day and the estimated analyst minutes per actionable case may communicate operational burden more clearly than precision alone. Suppression and deduplication should be measured separately: reducing dashboard rows without fixing noisy detections may improve workflow presentation but not the detector's validity. High-risk automated responses require their own safety metric: unauthorized or erroneous containment actions divided by opportunities for such actions. A laboratory may have zero observed harmful blocks yet still lack enough trials to conclude that harmful blocks are rare in production. Therefore, uncertainty and sample size must be reported alongside every empirical estimate.

Table 4 defines primary metrics, numerator and denominator, measurement source, and interpretation. Missing events or incomplete time synchronization should be coded as missing observations with causes rather than inserted as zero latency. An analyst's response time should be reported separately from automatic action time, because the two represent different operational processes. Time-to-recovery begins after containment and includes service validation, while “incident response time” should not be treated as one ambiguous figure. Together, detection, triage, containment, and recovery measures reflect a full operational sequence and allow more meaningful comparison with conventional and hybrid monitoring arrangements.

A baseline comparison is advisable. Baseline A uses raw logs with manual review; Baseline B uses Wazuh rule-driven alerts without active response; Configuration C adds approved reversible automated containment. All three should receive the same controlled scenarios and equivalent background traffic. The intended comparison is not that automation improves accuracy inherently—containment occurs after detection—but whether it changes response timing, operational labor, error exposure, or recovery. Baseline construction must document any added tooling and training time. Results should be presented by rule family and workload, because strong performance on simple repeated logins could conceal weak coverage of privilege misuse or command-execution variations.

8.1 Reproducibility, ethics, and statistical reporting

Every experiment should preserve a run manifest containing tool version, settings, ruleset checksum, endpoint image identifier, scenario identifier, event counts, clock source, and response policy state. Personal data should be minimized through synthetic usernames and test addresses, with raw records retained only for a documented duration. The reporting package should include the exact scoring script or calculation workbook and enough anonymized samples for independent verification. The study does not require real employee passwords, production malware, or attack traffic against external infrastructure. Testing governance should follow organizational authorization and institutional review requirements where applicable.

Table 4. Primary evaluation measures and data sources

Metric	Definition / calculation	Measurement source
Precision	TP/(TP + FP)	Scenario labels and matched alerts
Recall / sensitivity	TP/(TP + FN)	Scenario labels and matched alerts
Specificity	TN/(TN + FP)	Benign trial labels
False alerts per endpoint-day	Benign false alerts ÷ observed endpoint-days	Alert ledger; asset count; exposure
Detection latency	Manager alert time − event-generation time	Synchronized event and manager timestamps
Dashboard latency	Index visibility time − event-generation time	Event timestamp and search polling
Containment latency	Confirmed action time − triggering alert time	Action verification log
Unsafe-action rate	Unauthorized containment actions ÷ action opportunities	Policy test and action audit
Resource overhead	CPU, RAM, disk growth, queue depth by load	Host counters and index metrics

For results analysis, categorical detection outcomes can be presented in contingency tables, and exact or asymptotic confidence intervals should reflect sample size and observed rates. If comparing two paired configurations on the same scenarios, a paired categorical test may be more appropriate than an independence test; for skewed latency observations, nonparametric or bootstrap methods may be preferable. Report the test assumptions, effect sizes, uncertainty intervals, and complete counts instead of using p-values as a substitute for meaningful improvements. No p-value, confidence interval, performance percentage, or benchmark runtime is reported in this proposal because no experimental dataset has been supplied.

9. Anticipated analysis, discussion, and design implications

The architecture is expected to provide value primarily through operational consolidation and consistent rule execution, not by eliminating the need for security expertise. Centralized endpoint observations can help a small team detect repetitive patterns that would otherwise remain distributed across local log files. Standardized alert records make evidence easier to compare across hosts, while a documented response policy prevents ad hoc administration under pressure. The anticipated advantage is greatest when the attack behavior leaves a reliable trace within enabled telemetry and when the organization can sustain routine rule review. In contrast, credential theft without observable anomalies, attacks against unmonitored assets, and novel abuse of legitimate tools may produce weak or no alerts under the selected rule catalogue.

A crucial trade-off is between sensitivity and investigation load. Lowering an authentication-failure threshold may catch small bursts earlier, but it can also flag password managers, scripted integrations, and ordinary mistakes. Increasing thresholds reduces routine noise but creates opportunities for slow attacks to evade the detector. Grouping events by address may catch concentrated sources and misclassify users behind a shared gateway; grouping by account may catch account-specific activity but fail to connect distributed probing. The proper operating point cannot be selected by theoretical argument alone. It requires scenario-specific cost assessment and empirical calibration against the organization's actual service patterns. These are reasons to publish precision and recall together with alert volume and policy consequences.

The rule-based baseline also provides a benchmark for later machine-learning integration. Data-driven models could help identify unusual cross-source sequences or prioritize ambiguous cases, but they introduce data requirements, training drift, difficult-to-explain decisions, and new validation obligations. Current reviews of AI-enabled SOCs document both promise and integration challenges (Giarimpampa et al., 2025; Khayat et al., 2025). A hybrid future design might therefore apply anomaly scoring after deterministic parsing and enrichment, while preserving explicit authorization for high-impact actions. Such work should compare the incremental gain in detection or analyst efficiency against added complexity, resource use, false-positive burden, and reproducibility. It should not claim automation is intelligent merely because rules execute without a human click.

The VSOC approach may be particularly suitable for colleges with teaching laboratories, nonprofit organizations, and small companies operating a manageable set of servers and endpoints. These settings can establish baseline monitoring, document administrative behavior, and rehearse incident procedures without first acquiring a large commercial SOC platform. CISA's cross-sector cybersecurity performance goals specifically emphasize prioritized, practical risk-reduction activities accessible to smaller organizations (Cybersecurity and Infrastructure Security Agency, n.d.). The proposed architecture complements those practices but does not replace patch management, multifactor authentication, secure backup, network segmentation, staff training, or incident governance. Detection is only one layer of protection; the value of visibility depends on whether the organization can act safely on what it learns.

Operational cost requires closer examination than the phrase “open source” suggests. Software access does not eliminate compute, backup, administration, electricity, storage, incident investigation, or change-control effort. A meaningful economic comparison includes an annualized hardware or hosting estimate, projected event volume and retention, analyst hours spent on false alerts, response playbook maintenance, and the cost of operational interruption caused by erroneous enforcement. The proposed model may reduce barriers to initial adoption, yet it cannot establish a quantified saving without comparable cost records. A deployment study should disclose assumptions about labor rates, retention laws, endpoint counts, and incident volumes, and should perform a sensitivity analysis rather than assume cost effectiveness solely from license price.

9.1 Security and reliability limitations

The first limitation is incomplete observability. A local administrator or successful intruder may disable a logging facility or an endpoint agent, and agents on disconnected laptops may upload events only after reconnecting. Network appliances, software-as-a-service products, and unmanaged devices may have no endpoint agent at all. The framework must track coverage separately from detection: an empty dashboard can indicate either absence of

suspicious activity or failure to ingest the activity. Heartbeat monitoring, expected-log checks, and periodic injected test events can partially mitigate this ambiguity. However, they do not guarantee that a well-positioned attacker cannot evade collection or manipulate local records.

Table 5. Design-risk register and prescribed safeguards

Risk / limitation	Potential consequence	Required safeguard
Incomplete telemetry	Missed attacks and false reassurance	Heartbeat checks; source coverage tests
Noisy thresholds	Analyst overload; erroneous blocks	Benign controls; tuning and sampling
Untrusted event fields	Response manipulation or command misuse	Input validation; no unsafe shell interpolation
Protected IP blocked	Loss of service or remote administration	Allowlist; approval; emergency reversal
Index or manager outage	Event backlog and visibility gaps	Capacity tests; backups; failure monitoring
Sensitive log exposure	Privacy, confidentiality and legal risk	RBAC, encryption, minimization, retention
Version/configura tion drift	Rules stop matching known threats	Versioned rules; regression test suite

The second limitation is rule and parser brittleness. Updates to operating systems or applications can change event identifiers, field values, message wording, and telemetry availability. Regex-based matches may be too broad, and correlations may reset across service restarts or be affected by event ordering and timestamp drift. The use of Wazuh's test utilities and version-controlled local rules reduces some maintenance risks but cannot eliminate them (Wazuh, n.d.-r; Wazuh, n.d.-h). Publishing one successful parser example is insufficient proof of long-term portability. A real deployment needs scheduled regression tests, deprecation checks, and test coverage for both successful and unsuccessful parsing paths.

The third limitation is automation safety. Incorrectly identifying a trusted source as hostile could interrupt student access, a business gateway, or legitimate remote administration. Malformed fields and compromised telemetry could be deliberately crafted to trigger damaging responses. Consequently, the policy includes allowlists, validation, short reversible timeouts, scope restrictions, action logging, and approval thresholds. Even with these controls, an error in platform-specific firewall behavior or script cleanup can leave an unwanted block in place. Response rollback tests must therefore include abrupt restarts and failed communications, not merely a normal completion path. The safest default for ambiguous alerts is evidence collection and analyst notification.

The fourth limitation concerns evidence handling and the authority of results. A dashboard view may omit context found in raw logs, while raw event storage can expose user identities and operational secrets. Confidentiality, integrity, retention, access rights, and time synchronization must be specified as elements of the experiment and implementation. The proposed metrics do not measure prevention of all compromises or estimate organization-wide breach probability. No empirical performance result is claimed in this conceptual paper. The absence of reported measurements is a deliberate research boundary, not a finding that the design performs well. Future revisions should insert only results directly traceable to the documented laboratory and should report negative findings with the same prominence as successful detections.

9.2 Deployment and governance recommendations

A staged rollout is preferable to enabling dozens of rules and response scripts simultaneously. During observation-only mode, administrators should confirm parser health, discover routine service behavior, identify shared

addresses and critical management paths, and estimate investigation workload. A second phase enables higher-confidence notifications with clear escalation routing. Reversible automation follows only after a documented risk review and negative-test coverage. Finally, a limited production pilot assesses storage sizing, analyst workflow, and recovery drills before expanding endpoint scope. This staged approach resembles security-control maturation rather than an all-at-once software installation. It creates checkpoints at which management can stop expansion if alert quality, service reliability, or staffing proves inadequate.

Governance artifacts should remain deliberately compact: an asset register, source register, rule register, response authorization matrix, change log, contact tree, incident case template, and periodic test report. These artifacts address the operational gap between technical detection and organizational action. Owners should know who may adjust thresholds, approve containment, bypass an emergency block, and review a disputed incident. A management-approved exception list is time-bounded and periodically reassessed; permanent silent suppression of a high-risk event is discouraged. Evidence preservation and case review are assigned to named roles rather than relying on dashboard alert histories as a complete incident record.

For long-term maintainability, the organization should establish measurable service objectives: a percentage of inventoried assets reporting required logs, maximum acceptable agent-disconnection duration, a target proportion of rules with regression tests, and defined acknowledgment times for high-priority cases. These service objectives must be reviewed against local risk and staffing. Technical availability and alert quality are separate measurements: a fully connected agent estate can still generate unusable detections, while a carefully tuned ruleset provides little protection when critical hosts are disconnected. The deployment team's recurring responsibilities are therefore visibility assurance, detection tuning, response rehearsal, and disciplined incident learning.

9.3 Interpretation of practical validity

External validity will depend on how closely the test environment resembles the services that a small organization actually runs. Laboratory endpoints may have predictable traffic, stable clocks, and clean configuration, whereas production environments contain scheduled tasks, shared networks, intermittent laptops, service-account retries, and maintenance changes. A practical pilot should therefore record the proportion of alerts attributable to these common behaviors and should test at least one period with normal institutional or business operations. The most persuasive result would not be one unusually fast response during a demonstration but a consistently documented ability to detect relevant events, preserve evidence, avoid harmful automation, and recover service across repeated, diverse conditions.

The distinction between detection and prevention is equally important. A blocked address may reduce further attempts from that source but cannot prove that a compromised credential, internal foothold, or alternate attacker infrastructure has been neutralized. Conversely, an alert that results in a legitimate maintenance finding may still contribute to operational awareness if it is correctly classified without service disruption. Findings should therefore be reported by investigative outcome as well as technical rule match. This more careful interpretation prevents a VSOC evaluation from equating every automated action with a successful incident response and supports decisions about where human expertise remains essential.

10. Conclusion and future work

This study proposes an open-source, rule-based Virtual Security Operations Center that integrates Wazuh endpoint telemetry, centralized event analysis, severity-aware triage, dashboard investigation, and narrowly authorized automated response. Its central design contribution is a transparent chain from observed event to rule evidence, operational priority, response decision, verification, and recovery. The paper also provides a threat-focused scenario catalogue and a measurement framework for accuracy, false positives, timing, resource demand, and containment safety. These components support a reproducible starting point for organizations whose size or budget makes a continuously staffed traditional SOC difficult to operate.

The proposal does not establish measured detection accuracy, response speed, or cost savings; those claims depend on controlled deployment and testing. Future work should implement the architecture, publish the complete ruleset and sanitized test traces, compare rule thresholds under realistic background traffic, and evaluate end-to-end operation across different endpoint types and event loads. Further extensions may add richer contextual correlation, managed case tracking, secure threat-intelligence enrichment, and independently evaluated learning-assisted triage. The guiding principle should remain the same: faster detection is valuable only when the underlying evidence is trustworthy and the resulting response is proportionate, explainable, reversible where appropriate, and accountable to organizational policy.

References

1. Cybersecurity and Infrastructure Security Agency. (n.d.). Cross-sector cybersecurity performance goals. https://www.cisa.gov/cybersecurity-performance-goals
2. Dempsey, K., Chawla, N. S., Johnson, L. A., Johnston, R., Jones, A. C., Orebaugh, A., Scholl, M., & Stine, K. M. (2011). Information security continuous monitoring (ISCM) for federal information systems and organizations (NIST SP 800-137). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-137
3. European Union Agency for Cybersecurity. (2025). ENISA threat landscape 2025. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2025
4. Giarimpampa, D., Meier, R., Bissyandé, T. F., Lenders, V., & Klein, J. (2025). Exploring the role of artificial intelligence in enhancing security operations: A systematic review. ACM Computing Surveys, 58(3), Article 67. https://doi.org/10.1145/3747587
5. González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures. Sensors, 21(14), Article 4759. https://doi.org/10.3390/s21144759
6. Kent, K., & Souppaya, M. (2006). Guide to computer security log management (NIST SP 800-92). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-92
7. Khayat, M., Barka, E., Adel Serhani, M., Sallabi, F., Shuaib, K., & Khater, H. M. (2025). Empowering security operation center with artificial intelligence and machine learning: A systematic literature review. IEEE Access, 13, 19162–19197. https://doi.org/10.1109/ACCESS.2025.3532951
8. Macaneata, C. (2024). Overview of security information and event management systems. Informatica Economica, 28(1), 15–24. https://ideas.repec.org/a/aes/infoec/v28y2024i1p15-24.html
9. MITRE. (n.d.-a). Abuse elevation control mechanism (T1548). MITRE ATT&CK. https://attack.mitre.org/techniques/T1548/
10. MITRE. (n.d.-b). Brute force (T1110). MITRE ATT&CK. https://attack.mitre.org/techniques/T1110/
11. MITRE. (n.d.-c). Command and scripting interpreter (T1059). MITRE ATT&CK. https://attack.mitre.org/techniques/T1059/
12. MITRE. (n.d.-d). File and directory permissions modification (T1222). MITRE ATT&CK. https://attack.mitre.org/techniques/T1222/
13. MITRE. (n.d.-e). Valid accounts (T1078). MITRE ATT&CK. https://attack.mitre.org/techniques/T1078/
14. Nelson, A., Rekhi, S., Scarfone, K., & Souppaya, M. (2025). Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile (NIST SP 800-61 Rev. 3). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-61r3
15. Onwubiko, C., & Ouazzane, K. (2022). Challenges towards building an effective cyber security operations centre [Preprint]. arXiv. https://arxiv.org/abs/2202.03691
16. Pascoe, C., Quinn, S., & Scarfone, K. (2024). The NIST cybersecurity framework (CSF) 2.0 (NIST CSWP 29). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.CSWP.29
17. Rosenberg, M., Schneider, B., Scherb, C., & Asprion, P. M. (2023). An adaptable approach for successful SIEM

adoption in companies [Preprint]. arXiv. https://arxiv.org/abs/2308.01065
18. Ross, R. (2020). Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-53r5
19. Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (IDPS) (NIST SP 800-94). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-94
20. Verizon. (2026). 2026 data breach investigations report. https://www.verizon.com/business/resources/reports/dbir/
21. Wazuh. (n.d.-a). Active response. https://documentation.wazuh.com/current/user-manual/capabilities/active-response/
22. Wazuh. (n.d.-b). Alert management. https://documentation.wazuh.com/current/user-manual/manager/alert-management.html
23. Wazuh. (n.d.-c). Architecture. https://documentation.wazuh.com/current/getting-started/architecture.html
24. Wazuh. (n.d.-d). Command monitoring. https://documentation.wazuh.com/current/user-manual/capabilities/command-monitoring/index.html
25. Wazuh. (n.d.-e). Configuration for monitoring log files. https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/monitoring-log-files.html
26. Wazuh. (n.d.-f). Custom active response scripts. https://documentation.wazuh.com/current/user-manual/capabilities/active-response/custom-active-response-scripts.html
27. Wazuh. (n.d.-g). Custom decoders. https://documentation.wazuh.com/current/user-manual/ruleset/decoders/custom.html
28. Wazuh. (n.d.-h). Custom rules. https://documentation.wazuh.com/current/user-manual/ruleset/rules/custom.html
29. Wazuh. (n.d.-i). File integrity monitoring: Syscheck configuration. https://documentation.wazuh.com/current/user-manual/reference/ossec-conf/syscheck.html
30. Wazuh. (n.d.-j). How to configure active response. https://documentation.wazuh.com/current/user-manual/capabilities/active-response/how-to-configure.html
31. Wazuh. (n.d.-k). Log data analysis. https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/log-data-analysis.html
32. Wazuh. (n.d.-l). Log data collection. https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/index.html
33. Wazuh. (n.d.-m). MITRE ATT&CK framework. https://documentation.wazuh.com/current/user-manual/ruleset/mitre.html
34. Wazuh. (n.d.-n). Role-based access control. https://documentation.wazuh.com/current/user-manual/api/rbac/index.html
35. Wazuh. (n.d.-o). Rules syntax. https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/rules.html
36. Wazuh. (n.d.-p). Security configuration assessment. https://documentation.wazuh.com/current/user-manual/capabilities/sec-config-assessment/index.html
37. Wazuh. (n.d.-q). System inventory. https://documentation.wazuh.com/current/user-manual/capabilities/system-inventory/index.html
38. Wazuh. (n.d.-r). Testing decoders and rules. https://documentation.wazuh.com/current/user-manual/ruleset/testing.html
39. Wazuh. (n.d.-s). Vulnerability detection. https://documentation.wazuh.com/current/user-manual/capabilities/vulnerability-detection/index.html
40. Wazuh. (n.d.-t). Wazuh server. https://documentation.wazuh.com/current/user manual/manager/index.html