EAI Endorsed Transactions
on Scalable Information Systems

Research Article EAI.EU

Privacy-Aware Data-Model Dual-Driven Decision Analysis for Data Security in Distributed Multi-Agent Operations

Yunxiao Wang[1], Haizhuang Liu[1,*], Zihan Liu[1], Haobo Zhao[1], Fuyang Wei[1]

[1]Information and Telecommunications Company, State Grid Shandong Electric Power Company, Jinan 250000, China

Abstract

INTRODUCTION: Distributed networks generate heterogeneous security telemetry while moving data across endpoints, users, services, and operational domains. SOCs need methods that protect data assets, preserve auditability, and avoid unsafe tool calls.

OBJECTIVES: This paper proposes a data-model dual-driven method, in which incident and execution data constrain LLM-based reasoning while model outputs generate auditable process data, for privacy-aware data security decision analysis in multi-agent security operations.

METHODS: The method combines LLM-based role agents, SOAR playbook orchestration, persistent message state, and a virtual security capability layer. Incidents are transformed into data-aware tasks, actions, commands, execution records, and summaries.

RESULTS: On 83 labeled incident samples, tool-call evaluation achieved 0.9684 precision, 0.4742 recall, 0.6367 F1-score, and 76.45 s average handling time.

CONCLUSION: The method supports auditable data security monitoring and controlled response, while complex multi-step planning remains the main improvement target.

Received on 8 July 2026; accepted on 4 August 2026; published on 3 September 2026

Keywords: data-model dual-driven, privacy-aware data security, decision analysis, distributed multi-agent operations, SOAR, tool-call evaluation

Copyright © 2026 Yunxiao Wang *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi:10.4108/eetsis.13943

1. Introduction

Distributed networked systems connect endpoint devices, identity services, cloud workloads, business applications, and data-sharing channels. This connectivity improves service scalability, but it also expands the attack surface around sensitive data assets and cross-domain data flows. Security operations centers (SOCs) therefore face a persistent mismatch between the growth of security telemetry and the limited capacity of human analysts. Modern attacks increasingly involve long attack chains, cross-platform movement, automated tooling, and a mixture of signals from endpoint, network, identity, cloud, and threat-intelligence systems. Industry reports, alert-fatigue studies, and

*Corresponding author. Email: 953943390@qq.com

incident-handling guidance emphasize that incident response should be treated as part of the broader cybersecurity risk-management process, rather than as a set of isolated technical operations [1–3]. In practice, however, many SOC workflows still depend on manual triage, static correlation rules, and analyst experience.

SOAR platforms and playbooks improve repeatability by connecting tools and codifying response procedures [4–6]. Nevertheless, static or semi-static playbooks are often insufficient for data-security incidents that require contextual interpretation, dynamic task decomposition, and adaptive tool selection. In distributed environments, an incident may involve a host alert, a suspicious identity action, an exposed service, and a data-transfer clue at the same time. The response system must decide which evidence can be collected, which data asset is at risk, and which containment step is appropriate without unnecessarily exposing privacy-sensitive information. Recent research has therefore begun to combine large language models (LLMs), tool-use mechanisms, and agentic workflows with security operations [7–9]. These techniques make it possible to interpret heterogeneous incident descriptions, generate response plans, and coordinate tool calls. At the same time, they introduce new concerns: hallucinated actions, unstable tool selection, incomplete execution paths, privacy risks, and difficulty in measuring whether an automated response chain is actually correct [10, 11].

To address these problems, this paper proposes a data-model dual-driven data security decision analysis method for distributed network operations. The term "data" refers to incident samples, data-asset context, tool-capability descriptions, tool-call traces, execution feedback, and quantitative evaluation indicators. The term "model" refers to the LLM-based reasoning components, role-specialized agents, playbook orchestration logic, and tool-dispatch decision mechanisms. The key idea is not to treat the LLM as a standalone text generator, but to embed it into a closed incident-response loop where model decisions are constrained by structured data and where execution feedback becomes measurable evidence for subsequent analysis.

The proposed method is motivated by three practical requirements observed in distributed data-security operations. First, incident descriptions are usually incomplete and heterogeneous. A phishing alert, a webshell warning, an endpoint malware alarm, and a suspicious data-access event expose different fields and require different investigation sequences. Second, response actions have operational and privacy side effects. Enrichment actions are usually low risk, while blocking an IP address, isolating a host, creating a ticket, or querying user-related evidence can affect business processes, analyst workload, and sensitive data exposure. Third, a useful automated system must be evaluated through its process, not only through the readability of its final explanation. If an automated report is fluent but misses the required containment or enrichment capability, the response remains incomplete.

The data-model dual-driven view addresses these requirements by forcing reasoning, execution, and evaluation to share the same response objects. Incident fields, data-asset context, and tool descriptions guide model decisions; model decisions create tasks and commands; commands produce execution records; execution records become feedback for the next reasoning round and evidence for quantitative metrics. In this sense, data constrains the model, and the model produces new structured data that can be audited and measured.

The main contributions are as follows:

1. A layered data-model dual-driven architecture is proposed for data security decision analysis in distributed networks, integrating incident input, data-asset context, multi-agent decision-making, SOAR orchestration, virtualized security capabilities, and quantitative evaluation.

2. A role-specialized multi-agent workflow is designed to transform distributed security events into privacy-aware tasks, actions, executable commands, and response summaries while keeping process states traceable.

3. A virtual security tool layer is introduced to abstract heterogeneous data-security capabilities and provide stable execution feedback for repeatable experiments.

4. A tool-call evaluation method is presented and validated on 83 security incident samples, using precision, recall, F1-score, and average handling time as measurable indicators.

The remainder of this paper is organized as follows. Section 2 reviews related work on automated incident response, SOAR, LLM-enabled security operations, and tool-use evaluation. Section 3 introduces the proposed method. Section 4 describes the system implementation. Section 5 presents the experimental design and results. Section 6 discusses limitations and future work. Section 7 concludes the paper.

2. Related Work

2.1. Data Security Operations in Distributed Networks

Security operations research has gradually shifted from isolated log collection and alert monitoring toward end-to-end incident response, risk management, and operational resilience. In distributed networks, this shift is closely connected to data security because sensitive data may move across endpoints, applications, cloud workloads, identity systems, and inter-domain service interfaces. NIST incident-handling guidance frames incident response as a lifecycle spanning detection, analysis, response, recovery, and improvement [3]. Recent breach reporting and alert-fatigue research also show that many organizations still rely heavily on manual effort for alert triage, operational reporting, and response tracking [1, 2]. This operational gap motivates automated incident response research.

Automated incident response aims to map security events to response outputs, ranging from enrichment and notification to containment and recovery actions. For data-security scenarios, this mapping must also account for the protected object, the affected scope, the evidence fields that can be queried, and the point at which human approval is required. Incident-response guidance and response playbooks both stress that handling procedures should clearly specify input information, response actions, ownership, and escalation paths [3, 4]. Recent work on alert fatigue further indicates that automation alone is not sufficient; complex incidents still require context-aware prioritization, analyst interaction, and multi-step reasoning [2].

2.2. Playbook Orchestration for Controlled Data-Security Response

SOAR systems connect security products, standardize response procedures, and execute repeatable actions through playbooks. The CISA response playbooks and the CACAO specification demonstrate how incident and vulnerability response knowledge can be represented as standardized operational processes [4, 5]. Recent studies explore intelligent playbook recommendation and LLM-assisted transformation of legacy playbooks into standard formats [6, 12]. For distributed data security, playbooks are also useful as policy boundaries because they can define which evidence source is queried, which parameter is required, and which operation is allowed to proceed automatically.

Despite these advances, playbook automation often remains bounded by predefined logic and tool-specific interfaces. When incident contexts are incomplete or ambiguous, static workflows may fail to select all necessary tools or may require substantial manual adaptation. In privacy-sensitive data-security cases, the opposite failure is also important: an over-broad workflow may query more user, host, or asset information than needed. This paper therefore uses playbooks as an execution carrier, while relying on multi-agent reasoning to generate and refine response intents under explicit operational boundaries.

2.3. LLMs and Multi-Agent Collaboration for Privacy-Aware Security Operations

LLMs have been studied for log interpretation, alert triage, detection enhancement, vulnerability analysis, incident reporting, and threat-intelligence processing [7–9]. Agent-based LLM systems further extend these capabilities by combining reasoning, memory, tool use, and multi-step execution [13–15]. In incident response, multi-agent collaboration can divide a complex response process into specialized roles, reducing the cognitive burden placed on a single model [16]. This separation is especially relevant to privacy-aware operations: one role can interpret the event,

another can decide the necessary evidence scope, and another can map the response to controlled commands.

However, LLM-based agents introduce new risk surfaces. Trustworthy-agent research identifies threats including prompt injection, tool misuse, unsafe autonomy, and unreliable reasoning [11]. Security guidance on LLM applications also stresses the importance of constraining model behavior and auditing interactions with external systems [10]. These concerns motivate a system design in which agent decisions are logged, transformed into structured commands, executed through controlled interfaces, and evaluated quantitatively. In the proposed method, privacy protection is treated as an engineering constraint on evidence access and response execution, rather than as an after-the-fact statement.

2.4. Evaluation of Controlled Data-Security Actions

For systems that invoke external tools, evaluation cannot stop at final text quality. Tool-use benchmarks such as API-Bank and ToolLLM evaluate whether models can select suitable APIs and construct valid calls [17, 18]. HammerBench further studies fine-grained function calling in real mobile-device scenarios [19]. These studies suggest that process-level evaluation is essential for agentic systems.

In cybersecurity operations, tool selection errors and parameter errors can directly affect response quality. In data-security operations, tool selection also reflects whether the system performs a proportionate action: missing a required enrichment step may leave a data asset exposed, while an unnecessary query or containment action may increase operational and privacy risk. Therefore, this paper evaluates the system through tool-call evaluation. The evaluation distinguishes required tools, allowed tools, and forbidden or irrelevant tools, allowing the system to be measured not only by how many correct tools it invokes, but also by how well it avoids unreasonable calls.

3. Data-Model Dual-Driven Method

3.1. Overall Framework

The proposed method integrates distributed data-security incident input, multi-agent decision-making, SOAR orchestration, security capability invocation, privacy-aware evidence handling, and quantitative evaluation into a closed-loop response process. Its central mechanism can be summarized as:

$$\text{data input} \rightarrow \text{model analysis} \rightarrow \text{process orchestration} \rightarrow \text{tool execution} \rightarrow \text{result feedback} \rightarrow \text{quantitative evaluation}.$$

The data side includes incident samples, contextual fields, data-asset descriptions, tool-capability descriptions, execution traces, tool outputs, and evaluation metrics. The model side includes LLM-based semantic interpretation, role-specialized agents, playbook mapping, and tool-dispatch logic. These two sides interact continuously: incident data drives model analysis; model outputs become tasks, actions, and commands; commands trigger tool execution; execution results are converted into structured process data; and these data support both closed-loop response and quantitative evaluation.

A key design point is that neither side is sufficient alone. Data without model reasoning can support dashboards and static rules, but it is weak when the event context is ambiguous, when the protected data asset is unclear, or when the required response path depends on multi-step judgment. Model reasoning without structured data can produce plausible but unverifiable text. The proposed method therefore uses data to define the available operational space and uses the model to select, order, and explain actions inside that space. This reduces the gap between natural-language reasoning and executable security operations.

The closed loop also creates a natural place for human oversight. Analysts can inspect the event, the generated tasks, the actions selected by the agents, the playbook commands, and the execution records. If the system makes a conservative decision, the missing step can be traced to the task or action layer. If the system makes an excessive decision, the tool-call trace shows which agent and which command introduced the deviation. This traceability is important for deploying LLM-assisted response in distributed environments where accountability, data minimization, and privacy-sensitive evidence handling matter.

3.2. Layered Architecture

The architecture contains five layers: data and input, data-asset and evidence governance, multi-agent decision-making, SOAR orchestration, and security capability. Quantitative evaluation is a cross-layer mechanism applied to execution records and tool-call traces rather than an independent sixth architectural layer.

The data and input layer receives and normalizes incident information. Typical fields include incident name, incident description, incident context, severity, event source, affected assets, indicators of compromise, host identifiers, business ownership, and tool-capability descriptions. This layer supplies the semantic and operational constraints needed by downstream agents.

The layer does not assume that every incident has complete fields. In practice, some samples contain a clear host identifier but no user account; others contain an external IP address but no confirmed affected asset. The input layer therefore preserves both explicit fields and free-text context. Structured fields are used when available, while the message body and context field provide supplementary evidence for semantic analysis. This design allows the same workflow to handle alerts originating from endpoint products, web application logs, threat-intelligence notifications, and manually submitted incident descriptions.

The data-asset and evidence-governance layer is introduced to make the method closer to distributed data-security practice. This layer does not add a new product dependency; rather, it records how each incident relates to protected objects, affected systems, and evidence requirements. The protected object may be a host, service, user account, application component, vulnerability exposure, or business data flow. The evidence requirement describes which information is needed for response, such as an asset lookup, an IOC enrichment result, a vulnerability scope check, a containment decision, or a notification target.

This layer also supports a privacy-aware interpretation of tool calls. A response command should be justifiable by the incident objective. For example, querying asset ownership may be reasonable for vulnerability exposure, while querying user-related evidence should be limited to cases where identity context is part of the response objective. Privacy-sensitive evidence in this context may include account identifiers, asset ownership, login history, host-user association, access records, and business ownership fields. Only fields required by the incident objective should be queried. The method therefore treats tool calls as both security decisions and evidence-access decisions.

The multi-agent decision-making layer is the core reasoning layer. Instead of asking a single model to directly produce a final response, the system decomposes the incident-response process into several role-specialized components. The captain agent performs global event analysis and determines the response direction. The manager agent decomposes the response objective into tasks. The operator agent transforms tasks into concrete actions. The executor module converts actions into executable commands and coordinates subsequent SOAR execution. The expert agent provides security-domain analysis and supports result interpretation.

This role separation is intended to make planning more observable. The captain layer answers the question of what the incident probably means; the manager layer answers what must be investigated or handled; the operator layer answers which concrete capabilities are needed; and the executor layer answers whether the action can be mapped to a playbook or must be recorded for human handling. The expert layer then interprets execution feedback and performs the final closure judgment using deterministic completion checks over task, command, and execution states. Compared with a single monolithic prompt, this decomposition exposes intermediate decisions and makes error localization easier.

The SOAR orchestration layer maps structured response commands into playbook workflows. Asset enrichment commands can be mapped to asset-query playbooks; threat-assessment commands can be

mapped to threat-intelligence lookup playbooks; containment commands can be mapped to firewall blocking or unblocking playbooks; and coordination commands can be mapped to notification or ticketing playbooks. Playbooks provide execution order, dependency handling, parameter passing, and state tracking.

In this architecture, playbooks are not treated as a replacement for reasoning. They are treated as a controlled execution carrier. The agent layer decides that a response capability is needed; the playbook layer determines how that capability is executed, which parameters are required, and which result fields are returned. This separation reduces the risk that free-form model output directly controls external systems.

The security capability layer exposes underlying security capabilities through unified interfaces. Rather than binding the research prototype to a specific vendor product, the implementation abstracts common capabilities such as asset query, threat intelligence, firewall control, notification, and ticket management. In a distributed data-security setting, these capabilities correspond to different evidence and response domains: asset context, indicator context, network control, coordination, and workflow tracking. This abstraction reduces integration cost, improves experimental controllability, and makes tool invocation records available for evaluation.

The virtualized capability design also supports repeatable testing. Real security products may change API behavior, return incomplete data, or fail because of permission and network conditions. These conditions are important for production deployment but make algorithm-level evaluation difficult. By using a virtual layer during the experiment, the study focuses on whether the multi-agent method selects suitable capabilities and constructs coherent response chains. Later deployment can replace virtual capabilities with production connectors while preserving the same command and execution-record abstractions.

3.3. Message Transmission and State Collaboration

The system uses a "persist first, relay second, push third" message flow. When an agent or executor produces an analysis result, task, action, command, or summary, the message is first written into the database. A publisher then sends the message to RabbitMQ. A background consumer in the web service receives and parses the message, and SocketIO pushes it to the corresponding event room in the front-end war-room interface.

The response process is organized around events, tasks, actions, commands, execution records, and summaries. Each message contains fields such as event identifier, sender role, processing round, message type, and message body. This design makes the response process traceable and recoverable. If the front end is offline or a service is interrupted, persisted state remains available for later recovery.

Complex incidents may require multiple rounds of analysis and execution. Therefore, the system records tasks, actions, commands, results, and summaries by processing round. After execution results are returned, the expert layer performs the final closure judgment using deterministic completion checks over task, command, and execution states. The response objective is treated as satisfied only when mandatory tasks reach terminal states, issued commands return definitive results or are explicitly deferred, and no unresolved execution failure or pending analyst approval remains. If the objective is complete, the event enters the summary stage; otherwise, the latest feedback is used to generate additional tasks or actions.

3.4. SOAR Orchestration and Virtual Security Tool Layer

The SOAR layer translates agent-generated response intents into executable workflows. In the proposed design, agents describe what should be done, while playbooks define how the steps should be executed, which tools should be invoked, how parameters should be passed, and how results should be collected. This separation reduces the risk that model output becomes an unstructured recommendation without operational effect. It also gives the system a location for enforcing data-security policies, such as requiring approval before high-impact containment or limiting user-related evidence collection to the minimum fields required for response.

The virtual security tool layer provides stable capability interfaces for experimentation. It simulates or wraps common security tools and returns structured results including execution status, query content, risk judgment, containment result, and error information. The layer abstracts capabilities relevant to distributed data security, including asset query, suspicious indicator enrichment, source blocking, notification, and ticket creation. The system writes these results into execution records and feeds them back to the response process. This design provides a repeatable environment for evaluating whether the agentic system selected appropriate tools without exposing production systems or unnecessary sensitive data during the experiment.

4. System Implementation

The prototype is implemented as an intelligent SOC system. The web application is built on Flask and Flask-SocketIO. MySQL is used as the main state store, RabbitMQ is used for asynchronous message transmission, and SocketIO provides real-time front-end synchronization. The response workflow is represented through database entities for events, tasks, actions, commands, messages, execution records, and summaries.

The agent services run as role-specific components. The captain, manager, operator, executor, and expert roles exchange structured messages and update event state through the shared database and message queue. The executor distinguishes between playbook commands and human-handling commands. Playbook commands are forwarded to the SOAR orchestration layer, while human-handling commands are recorded for analyst intervention.

The SOAR implementation uses playbooks to connect response actions with capability interfaces. In the experimental environment, OctoMation is used as the SOAR orchestration basis, while UniMCPSim is used to construct a virtual security tool layer. This combination avoids direct dependence on heterogeneous production devices and provides a controlled basis for quantitative evaluation.

5. Experimental Design and Results

5.1. Dataset

The experiment uses 83 common security incident samples with corresponding response annotations. The dataset is stored in JSONL format, where each line represents one incident sample. The core fields are shown in Table 1. Event name, message, context, severity, and source form the input for incident interpretation and response planning. In the special-issue context, these fields are interpreted as operational evidence for distributed data-security monitoring: they describe affected assets, suspicious indicators, potential data exposure, and the response capabilities needed to reduce risk. Required tools, allowed tools, and forbidden tools are used for evaluating tool invocation correctness.

The samples cover typical SOC scenarios, including advanced persistent threats, backdoor trojans, distributed denial-of-service attacks, phishing attacks, hacker tools, insider threats, mining trojans, software supply-chain risks, vulnerability exploitation, and web attacks. These categories are summarized in Table 2.

The dataset was organized to cover both external attack alerts and internal host-oriented incidents. Each sample contains a natural-language event title, an alert message, scenario context, severity, event source, and tool labels. The context field may contain affected assets, indicators of compromise, host identifiers, business ownership, suspected attack behavior, and expected response hints. The samples do not include real personal records; instead, they encode the types of operational clues that a data-security response system must reason over. This design reflects the fact that real SOC inputs are not uniform: some alerts are structured logs, while others are analyst-submitted descriptions or threat-notification summaries.

Table 3 explains the meaning of the dataset categories. The purpose of the dataset is not only to count cases, but to expose the response system to different types of operational reasoning. Some categories emphasize host evidence, some emphasize network indicators, some emphasize user or business context, and others require attack-chain reconstruction. From a data-security perspective, these cases cover confidentiality

Table 1. Main dataset fields

Field name	JSON field	Description
Sample identifier	`case_id`	Unique identifier of each experimental sample
Event name	`event_name`	Title of the security incident
Event description	`message`	Raw event or alert content
Event context	`context`	Supplementary context such as asset ownership, IOC, and host identifiers
Severity	`severity`	Risk level of the incident
Event source	`source`	Source channel of the event
Required tools	`required_tools`	Tools that must be invoked to complete the response
Allowed tools	`allowed_tools`	Tools that may be invoked but are not mandatory
Forbidden tools	`forbidden_tools`	Tools that should not be invoked
Case type	`case_type`	Scenario category, such as APT, phishing, or web attack

Table 2. Incident categories in the dataset

Code	Category
APT	Advanced persistent threat
BT	Backdoor trojan
DDoS	Distributed denial of service
FA	Phishing attack
HT	Hacker tool
IT	Insider threat
MC	Mining trojan
SC	Software supply chain
VR	Vulnerability exploitation
WA	Web attack

risk, integrity risk, availability risk, vulnerable data-processing components, and suspicious access paths. This diversity is necessary because a decision system for distributed networks should not be evaluated only on one narrow alert format.

The table shows that the dataset is designed around operational reasoning patterns. Host-centered categories such as BT, MC, WA, and parts of VR require endpoint evidence and affected-asset confirmation. Network-centered categories such as DDoS and APT require external indicator analysis and traffic interpretation. Business-context categories such as IT and SC require the system to reason about ownership, affected scope, and coordination. Therefore, the dataset evaluates whether the proposed method can turn heterogeneous incident descriptions into structured response decisions while preserving a clear link between data-security risk and tool invocation.

5.2. Evaluation Metrics

Let P be the set of tools invoked by the system for one incident, R be the set of required tools in the annotation, and A be the set of allowed tools. Required tools indicate capabilities that should be invoked, while allowed tools are acceptable auxiliary calls. The evaluation defines:

$$TP = |P \cap R|, \tag{1}$$

$$FP = |P \setminus (R \cup A)|, \tag{2}$$

$$FN = |R \setminus P|. \tag{3}$$

Precision, recall, and F1-score are then computed as:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{5}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6}$$

Precision measures how many invoked tools are reasonable calls, reflecting the ability to avoid irrelevant or incorrect invocation. In distributed data-security response, this also indicates whether the system avoids unnecessary evidence access and excessive control actions. Recall measures how many required tools are covered, reflecting the completeness of key response capabilities. F1-score summarizes the trade-off between precision and recall. Average handling time is also recorded to assess runtime feasibility. The timing starts when the event object is created and ends when the final summary is generated and stored. It

Table 3. Dataset category semantics and operational meaning

Code	Cases	Scenario meaning	Typical evidence in the sample	Why the category is included
APT	12	Multi-stage intrusion, C2 communication, persistence, or lateral movement	External IPs, affected assets, host identifiers, threat context, abnormal traffic or endpoint behavior	Tests attack-chain decomposition and staged investigation rather than one-step response
BT	3	Backdoor trojan or remote-control behavior on a compromised host	Host process clues, C2 indicators, endpoint alert context, asset ownership	Tests whether the system recognizes host compromise and containment needs
DDoS	2	Availability-oriented network attack against a service or gateway	Source traffic, target service, traffic volume, service impact, severity	Tests the distinction between service-availability response and host-intrusion response
FA	2	Phishing or social-engineering incident affecting users or mail systems	Suspicious sender, URL or attachment clue, recipient context, mail-gateway alert	Tests user/account-oriented triage and notification-oriented response
HT	4	Hacker tool, scanner, exploit tool, or suspicious post-exploitation utility	Tool signature, process name, source host, endpoint or SIEM alert	Tests recognition of suspicious tool usage and follow-up investigation
IT	10	Insider threat, abnormal account behavior, or business misuse scenario	User identity, asset ownership, access behavior, business context	Tests reasoning that depends on identity and business context, not only technical indicators
MC	7	Mining trojan or resource-abuse incident on hosts or servers	High CPU behavior, mining process, wallet or pool clue, host alert, asset context	Tests host-level investigation and remediation planning
SC	21	Software supply-chain risk, vulnerable component, or dependency exposure	Vulnerability notice, affected software, asset scope, patch or exposure context	Tests broad impact assessment and coordination for non-single-host risks
VR	17	Vulnerability exploitation, exploit attempt, or exposed weakness	CVE or vulnerability name, affected service, exploit path, host or network alert	Tests whether the system can connect vulnerability context with response actions
WA	5	Web attack such as webshell, SQL injection, command execution, or reverse shell	HTTP request details, attacker/victim IPs, web path, response status, payload evidence	Tests complex web-attack analysis and attack-path reconstruction

Table 4. Tool-call evaluation results

Metric	Value
True positives	92
False positives	3
False negatives	102
Precision	0.9684
Recall	0.4742
F1-score	0.6367
Average handling time	76.45 s

includes role reasoning, database persistence, message-queue waiting and delivery, command generation, playbook execution, virtual-tool execution, and result aggregation, but excludes human waiting time.

5.3. Results

The batch experiment processed all 83 incident samples through the automated response pipeline. The system successfully completed incident input, agentic analysis, automated response execution, tool-call recording, and metric calculation. The aggregated results are shown in Table 4.

The precision of 0.9684 indicates that the system has strong constraints on tool selection. Among the tools actually invoked, most belonged to the required or allowed range. This suggests that the multi-agent and playbook mechanisms can effectively reduce irrelevant tool calls and avoid excessive response actions. For privacy-aware data-security operations, this conservative behavior is valuable because unnecessary investigation actions may broaden the exposure of sensitive evidence.

In contrast, the recall of 0.4742 is much lower than the precision. With 92 true positives, 3 false positives, and 102 false negatives, the main issue is not that the system invokes too many incorrect tools, but that it omits some required tools. The overall behavior can be characterized as conservative: the system makes few unreasonable calls, but does not yet cover all necessary capabilities in complex incidents.

The F1-score of 0.6367 shows that the system already has a measurable level of tool-call decision ability. The average handling time of 76.45 seconds suggests that the multi-agent workflow is operationally feasible at the current experimental scale. This time is measured from event-object creation to final-summary storage and includes message-queue waiting and virtual-tool execution. Therefore, the primary bottleneck is not runtime, but the completeness of task decomposition, scenario interpretation, and multi-step planning.

Figure 1 visualizes the micro and macro metrics. The micro precision of 0.9684 and macro precision of 0.9819 show that the system rarely produces unreasonable tool calls across both aggregate and per-case views. The macro recall of 0.5594 is higher than the micro recall of 0.4742, indicating that some cases are handled reasonably while a small number of high-requirement cases contribute many missing required calls. The

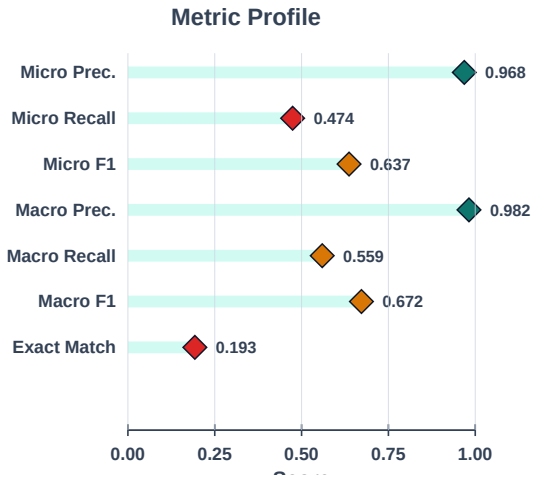


Figure 1. Overall tool-call evaluation results.

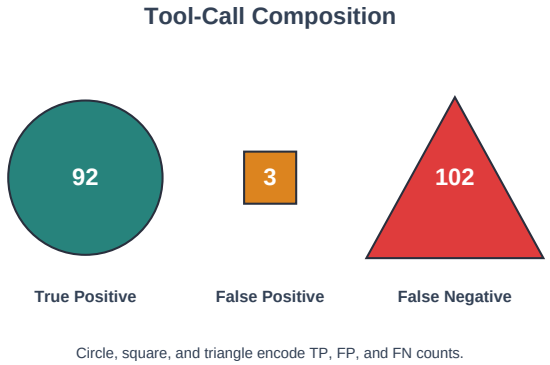


Figure 2. Composition of true-positive, false-positive, and false-negative tool calls.

exact-match rate is 0.1928, which is expected to be lower than F1 because exact match requires the complete required tool set to be covered without extra disallowed calls.

Figure 2 further explains the precision-recall imbalance. The system generated 92 true-positive tool calls and only 3 false-positive tool calls, but it missed 102 required tool calls. This means the current design is not overly aggressive. Such behavior is acceptable for early-stage guarded automation, but it also shows that a fully autonomous response workflow needs additional planning verification and completion checks.

The category-level results in Figure 3 show that the framework performs unevenly across incident types. Software supply-chain cases obtain the strongest balanced performance among the larger categories, with high recall and only limited false positives. Hacker-tool cases also obtain high F1, although the sample size is small. In contrast, web-attack, vulnerability-exploitation, and APT cases show lower recall. These categories usually imply multi-step investigation, cross-source correlation, and attack-chain reconstruction, which increases the chance that the agents produce only a partial action set.

The category differences suggest that the current framework is strongest when the incident goal is explicit and the response path can be expressed as a small number of structured actions. It is weaker when the incident description implies several dependent reasoning steps, such as correlating a vulnerability with exposed assets, linking an identity event to a business data flow, or deciding whether a network indicator implies data exfiltration risk. This observation is useful for system improvement because it points to planning verification at the scenario level rather than simply increasing the number of generated actions. A deployment version should check whether the generated plan has addressed the incident objective, the affected scope, the data-security impact, the containment condition, and the reporting requirement before it moves to closure.

Figure 4 shows that most cases finish within a relatively compact runtime range, with an average of 76.45 seconds and a median of 75.75 seconds. The result suggests that the main limitation of the current prototype is not uncontrolled runtime expansion. The reported runtime includes role reasoning, database persistence, message-queue waiting and delivery, command generation, playbook execution, virtual-tool feedback, and aggregation. However, the multi-agent design still introduces model-call and orchestration overhead. For high-volume low-risk alerts, a production deployment may need a lightweight route, while complex or high-risk incidents can use the full auditable workflow.

5.4. Ablation Comparison

To better understand the effect of the multi-agent workflow, an additional comparison was made against a single-agent ablation run available in the experiment repository. The comparison uses the 83 case identifiers

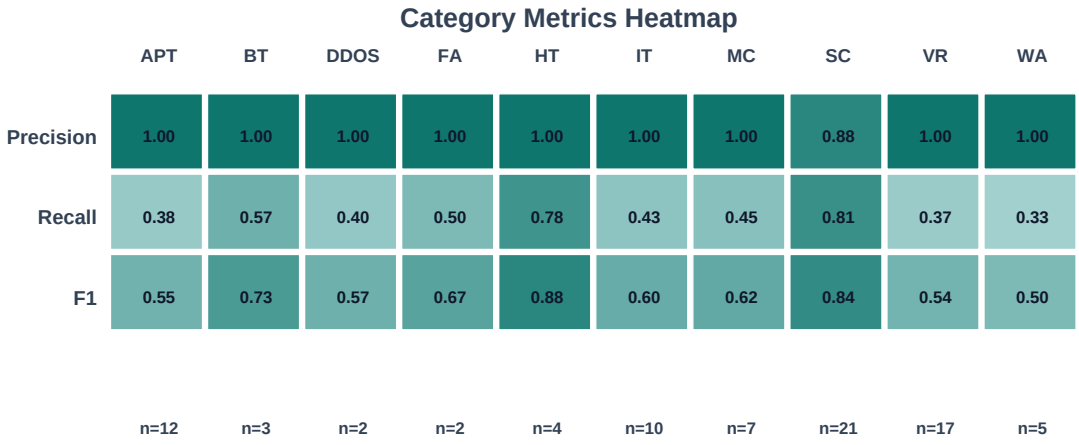


Figure 3. Tool-call performance by incident category. Case counts are shown under each category label.

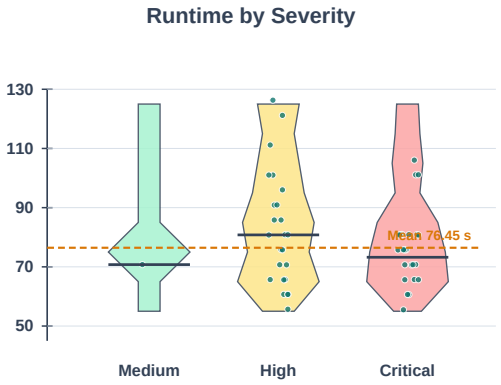


Figure 4. Runtime distribution and severity-group runtime statistics.

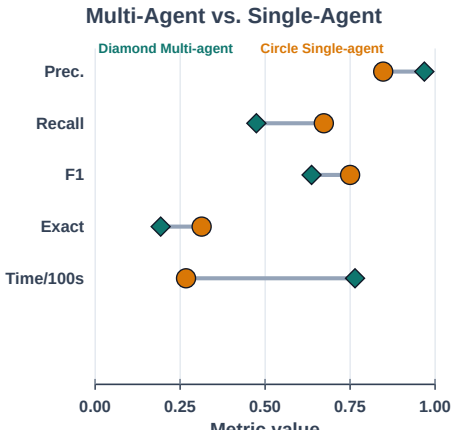


Figure 5. Ablation comparison between the multi-agent framework and a single-agent workflow on common cases. Runtime is normalized by 100 seconds for plotting.

shared by the two runs. As shown in Figure 5, the single-agent configuration achieves higher recall and F1 on the common cases, but it also introduces substantially more false positives and has lower precision. The multi-agent framework is slower and more conservative, but it provides stronger constraints on unreasonable tool execution and more explicit intermediate state.

This comparison clarifies the engineering trade-off. A single-agent workflow tends to generate broader action sets, but the additional calls increase operational risk. The multi-agent workflow sacrifices some recall for stronger control and auditability. For SOC deployment, the desired direction is not to abandon role specialization, but to add scenario-level planning verification so that the multi-agent workflow can improve response completeness without losing precision.

6. Discussion

The experimental results support the feasibility of combining multi-agent reasoning, playbook orchestration, and virtualized security capabilities for data-security response in distributed networks. High precision is particularly important in this setting because unnecessary containment, noisy enrichment, or redundant ticketing can increase analyst workload, disturb business services, and expand the handling of sensitive evidence. The proposed method demonstrates that structured

roles, tool descriptions, and playbook mapping can constrain the system's tool choices.

The low recall reveals the main limitation of the current system. Some incidents require multi-step investigation, multiple enrichment paths, attack-chain completion, or explicit reasoning about whether a protected data asset is still exposed. In these cases, the agents may generate a partial response plan even though the final narrative appears coherent. This suggests that future work should add pre-execution checklist mechanisms, stronger scenario templates, and post-planning verification stages.

The virtual security tool layer also has limitations. It improves controllability and reproducibility, but real production environments include more complex interfaces, permission boundaries, latency, failure modes, return formats, and data-access policies. Results obtained with virtualized tools can verify basic decision and invocation behavior, but they should not be interpreted as complete evidence of production readiness or as a full privacy-preserving deployment mechanism.

Finally, the current evaluation focuses on tool selection. A more comprehensive evaluation system should also measure parameter correctness, execution-result consistency, closed-loop completion rate, manual intervention cost, response latency, expert assessment, and privacy-impact indicators such as whether unnecessary user or asset fields were queried. These metrics would better capture the quality of intelligent SOC automation in real deployments.

7. Conclusion

This paper proposed a data-model dual-driven data security decision analysis method for multi-agent operations in distributed networks. The method integrates incident data, data-asset context, LLM-based multi-agent reasoning, SOAR playbook orchestration, virtual security capabilities, and quantitative evaluation into a closed response loop. A prototype system was implemented with persistent state management, asynchronous message delivery, real-time front-end synchronization, playbook execution, and virtualized tool feedback.

Experiments on 83 security incident samples showed that the system achieved a precision of 0.9684, a recall of 0.4742, an F1-score of 0.6367, and an average handling time of 76.45 seconds for tool-call evaluation. These results indicate that the system can effectively avoid irrelevant tool calls and provide an engineering foundation for measurable automated data-security response. Future work will focus on strengthening complex incident planning, improving scenario-level completion checks, connecting additional real security products, expanding the dataset toward distributed data-flow scenarios, and constructing a more complete evaluation framework.

Acknowledgement. Funding: This work is supported in part by the Science and Technology Project of State Grid Shandong Electric Power Company: 'Research on Key Technologies of Attack Mechanism Mining, Analysis and Decision-Making for Power System Cross-Domain Security Boundary Protection' under Grant 520627250002.

References

[1] Verizon Business. 2026 Data Breach Investigations Report. Verizon Business; 2026. Available from: https://www.verizon.com/business/resources/reports/dbir/.

[2] Tariq S, Chhetri MB, Nepal S, Paris C. Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities. ACM Computing Surveys. 2025;57(9):224.

[3] Cichonski P, Millar T, Grance T, Scarfone K. Computer Security Incident Handling Guide. National Institute of Standards and Technology; 2012. Available from: https://doi.org/10.6028/NIST.SP.800-61r2.

[4] Cybersecurity and Infrastructure Security Agency. Federal Government Cybersecurity Incident and Vulnerability Response Playbooks. CISA; 2021.

[5] OASIS. CACAO Security Playbooks Version 2.0; 2023. Available from: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html.

[6] Hu H, Zhang L, Zhang Z, Yao X, Wu X. An Intelligent Playbook Recommendation Algorithm Based on Dynamic Interest Modeling for SOAR. Symmetry. 2025;17(11):1851.

[7] Habibzadeh A, Feyzi F, Atani RE. Large Language Models for Security Operations Centers: A Comprehensive

Survey; 2025. arXiv preprint arXiv:2509.10858. Available from: https://arxiv.org/abs/2509.10858.

[8] Jaffal NO, Alkhanafseh M, Mohaisen D. Large Language Models in Cybersecurity: A Survey of Applications, Vulnerabilities, and Defense Techniques; 2025. arXiv preprint arXiv:2507.13629. Available from: https://arxiv.org/abs/2507.13629.

[9] Xu H, et al. Large Language Models for Cyber Security: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology. 2025.

[10] OWASP Foundation. OWASP Top 10 for Large Language Model Applications 2025; 2025. Available from: https://genai.owasp.org/llm-top-10/.

[11] Yu M, Meng F, Zhou X, Wang S, et al.. A Survey on Trustworthy LLM Agents: Threats and Countermeasures; 2025. arXiv preprint arXiv:2503.09648. Available from: https://arxiv.org/abs/2503.09648.

[12] Akbari Gurabi M, Fysarakis K, Mavroeidis V, et al. From Legacy to Standard: LLM-Assisted Transformation of Cybersecurity Playbooks into CACAO Format. In: Computer Security. ESORICS 2025 International Workshops. Springer Nature Switzerland; 2026. p. 491-510.

[13] Xi Z, Chen W, Guo X, et al. The Rise and Potential of Large Language Model Based Agents: A Survey. Science China Information Sciences. 2025;68(2):121101.

[14] Wu Q, Bansal G, Zhang J, et al.. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework; 2023. arXiv preprint arXiv:2308.08155. Available from: https://arxiv.org/abs/2308.08155.

[15] Yao S, Zhao J, Yu D, et al. ReAct: Synergizing Reasoning and Acting in Language Models. In: International Conference on Learning Representations; 2023. Available from: https://arxiv.org/abs/2210.03629.

[16] Liu Z. Multi-Agent Collaboration in Incident Response with Large Language Models; 2024. arXiv preprint arXiv:2412.00652. Available from: https://arxiv.org/abs/2412.00652.

[17] Li M, Zhao Y, Yu B, et al.. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs; 2023. arXiv preprint arXiv:2304.08244. Available from: https://arxiv.org/abs/2304.08244.

[18] Qin Y, Liang S, Ye Y, et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs. In: International Conference on Learning Representations; 2024. Available from: https://arxiv.org/abs/2307.16789.

[19] Wang J, Zhou J, Wen M, et al.. HammerBench: Fine-Grained Function-Calling Evaluation in Real Mobile Device Scenarios; 2024. arXiv preprint arXiv:2412.16516. Available from: https://arxiv.org/abs/2412.16516.