CORTEX: Collaborative LLM Agents for High-Stakes Alert Triage

Bowen Wei
George Mason University
bwei2@gmu.edu

Yuan Shen Tay
Fluency Security
yuanshen.tay@fluencysecurity.com

Howard Liu
Fluency Security
howard@fluencysecurity.com

Jinhao Pan
George Mason University
jpan23@gmu.edu

Kun Luo
Fluency Security
kun@fluencysecurity.com

Ziwei Zhu
George Mason University
zzhu20@gmu.edu

Chris Jordan
Fluency Security
chris@fluencysecurity.com

Abstract

Security Operations Centers (SOCs) are overwhelmed by tens of thousands of daily alerts, of which only a small fraction correspond to genuine attacks. This overload creates alert fatigue, leading to overlooked threats and analyst burnout. Classical detection pipelines are brittle and context-poor, while recent LLM-based approaches typically rely on a single model to interpret logs, retrieve context, and adjudicate alerts end-to-end—an approach that struggles with noisy enterprise data and offers limited transparency. We propose *CORTEX*, a multi-agent LLM architecture for high-stakes alert triage in which specialized agents collaborate over real evidence: a behavior-analysis agent inspects activity sequences, evidence-gathering agents query external systems, and a reasoning agent synthesizes findings into an auditable decision. To support training and evaluation, we release a dataset of fine-grained SOC investigations from production environments, capturing step-by-step analyst actions and linked tool outputs. Across diverse enterprise scenarios, CORTEX substantially reduces false positives and improves investigation quality over state-of-the-art single-agent LLMs.

1 Introduction

Security Operations Centers (SOCs) form the first line of defense against enterprise attacks, yet they are overwhelmed by an onslaught of alerts—often tens of thousands per day—of which only a small fraction indicate genuine threats. Empirical studies report false-positive rates approaching 99% AlAhmadi et al. [2022], creating extreme alert fatigue: critical signals are easily overlooked (as in the Target breach Riley et al. [2014], Finkle and Heavey [2014]), while analysts face burnout Tines [2023]. Traditional pipelines based on rules and anomaly detectors are brittle and context-poor, flooding operators with noise rather than insight. Recent industry reports estimate that 40–45% of enterprise alerts are false positives Orca Security [2022], Enterprise Strategy Group [2023], underscoring the urgent need for more precise, transparent triage.

Large language models (LLMs) have been explored for summarizing incidents and assisting analysts Zhang et al. [2025a], Deng et al. [2024], but most approaches adopt a single-agent paradigm:

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: LAW 2025: Bridging Language, Agent, and World Models for Reasoning and Planning.

one model must interpret logs, retrieve context, and adjudicate alerts end-to-end. Even with chain-of-thought prompting Wei et al. [2022] or ReAct-style tool use Yao et al. [2023], such models often falter on long-horizon, high-stakes investigations and provide limited auditability. This gap is particularly problematic in security-critical domains, where decisions must be both accurate and explainable Arrieta et al. [2020].

We address this challenge with a *divide-and-conquer multi-agent architecture* for SOC triage. Specialized agents assume distinct roles: a *Behavior Analysis Agent* identifies relevant investigative workflows; *Evidence Acquisition Agents* ground hypotheses by querying external systems (e.g., SIEM logs, threat intelligence); and a *Reasoning & Coordination Agent* synthesizes evidence into a transparent triage decision. Agents communicate through structured messages, iteratively cross-checking claims, akin to human analyst teams Chen et al. [2024], Du et al. [2023], Liang et al. [2024].

To support training and evaluation, we release a dataset of *fine-grained SOC workflows*, collected from production environments across more than ten scenarios. Unlike prior datasets that provide only coarse labels, ours captures full investigative traces—stepwise analyst actions, tool queries and outputs, and intermediate observations—enabling both training and evaluation of process-level reasoning.

Experiments across diverse enterprise scenarios show that CORTEX substantially reduces false positives and improves investigative quality compared to single-agent LLMs.

Contributions.

- **CORTEX:** a role-specialized, tool-using, auditable multi-agent architecture for SOC triage.
- **Fine-grained SOC workflow dataset:** process-level triage traces across 10+ real scenarios.
- **Empirical validation:** large reductions in false positives and improved reasoning quality over baselines.

2 Related Work

2.1 LLMs in Cybersecurity

LLMs are increasingly applied across defensive and offensive cybersecurity. Surveys synthesize hundreds of works spanning threat-intelligence extraction, knowledge-graph reasoning, vulnerability analysis, and attack automation Zhang et al. [2025a]. On defense, systems organize attack knowledge (e.g., ATTACKG+) and build TI knowledge graphs from unstructured text Zhang et al. [2025b], Hu et al. [2024]. On offense, PENTESTGPT demonstrates automated penetration testing with tool use and iterative planning, evaluated on real systems and CTFs Deng et al. [2024]. The OWASP Top 10 for LLM Applications formalizes risks (prompt injection, data poisoning, model DoS) relevant to both red and blue teams OWASP Foundation [2024]. Field reports further document SOC burnout and workflow pain points that motivate automation Tines [2023]. Compared to these threads, far fewer works model end-to-end SOC alert investigation as a *tool-grounded, role-specialized* reasoning pipeline, and fewer still explore *distillation* of that process into a deployable single model. Our work targets precisely this gap.

2.2 Alert Fatigue and Automated Triage in SOCs

SOCs contend with extreme alert volume and high false-positive rates, imposing substantial manual validation AlAhmadi et al. [2022], Orca Security [2022]. The Target breach illustrates the operational risk of drowning in noise despite vendor alerts Riley et al. [2014], Finkle and Heavey [2014]. Long breach lifecycles further motivate faster, higher-precision triage (241 days to identify *and* contain, per the 2025 IBM report) IBM Security [2025]. Classical automation frames triage as supervised prioritization or learning-to-rank over prior analyst decisions. Representative systems report meaningful workload reductions with low false-negative rates: AACT imitates analyst actions to auto-close benign alerts and escalate critical ones Turcotte et al. [2025]; ALERTPRO leverages contextual features and reinforcement learning to rank alerts and improve multi-step scenario handling Gong et al. [2024]. Yet, purely statistical models can be brittle to novel patterns and demand substantial, continuously curated histories. In parallel, emerging "agentic" SOC platforms (e.g., Radiant Security; Dropzone

Table 1: Workflow-specific actionability criteria (examples).

Workflow	Decision Logic	Actionable Condition
Credential Change	Auth modifications	Removal/addition by established user
User Creation	Account provisioning	Elevated privileges granted
Cloud Login Anomaly	Risk + history	Score > 1000 with recent activity
Geo Impossibility	Distance/time	$>$500 miles apart, infeasible time
File Access Anomaly	Risk + location	Score > 1000 from uncommon site
SaaS Login Anomaly	Rule triggers	≥ 3 abnormal triggers
Command-Line Exec.	Command analysis	Malicious code by admin user

AI) advertise end-to-end investigation coverage and large speedups Radiant Security [2025a,b], Dropzone AI [2024, 2025], but typically incur notable computational costs. These trends together motivate SOC-specific triage that mirrors analyst roles, *grounds* decisions in tool-fetched evidence, and remains efficient at deployment scale—precisely the design goals of our multi-agent, tool-using approach.

2.3 Collaborative Multi-Agent LLM Systems

Multiple LLM agents—with specialized roles, communication, and verification—can outperform single models on complex reasoning. Multi-agent debate iteratively critiques candidate answers to improve factuality and consistency Du et al. [2023], Liang et al. [2024]. General frameworks such as AGENTVERSE Chen et al. [2024], CAMEL Li et al. [2023], METAGPT Hong et al. [2023], and AUTOGEN Wu et al. [2023] provide orchestration patterns (roles, turn-taking, tool calls) that enable task decomposition and cross-checking. Empirical analyses also catalog failure modes—specification errors, inter-agent misalignment, and weak termination/verification—highlighting the need for principled protocols and reliability checks Cemri et al. [2025]. This literature motivates domain-aligned roles, disciplined message passing, and verification anchored in external evidence, while also surfacing a practical requirement: retain collaborative benefits without inflating inference cost or coordination complexity. Our architecture instantiates SOC-aligned roles and structured evidence exchange.

3 Methods

Our approach integrates (i) a divide-and-conquer multi-agent architecture for triage and (ii) a fine-grained SOC workflow dataset for process-level supervision.

3.1 CORTEX Architecture

Roles. CORTEX decomposes triage into four stages. The *Orchestrator Agent* manages the pipeline, enforces modularity, and ensures coherent handoffs between roles. The *Behavior Analysis Agent* routes alerts to the most relevant workflows. Workflow-specific *Evidence Acquisition Agents* execute calibrated playbooks by querying enterprise tools. The *Reasoning & Coordination Agent* reconciles evidence and produces a structured, auditable report (see Fig. 1).

Pipeline. The pipeline unfolds in four stages: (1) *orchestration* (execution control and consistency checks); (2) *classification* (routing) by the Behavior Analysis Agent; (3) *workflow analysis* with calibrated, evidence-grounded criteria executed by Evidence Acquisition Agents; and (4) *synthesis & actionability extraction* in the Reasoning & Coordination Agent, which finalizes the verdict, extracts observables, and proposes follow-ups. We instantiate seven workflows from production SOC data—credential changes, anomalous logins, file access anomalies, geographic impossibility, SaaS irregularities, user creation, and command-line execution (Table 1).

Reporting. CORTEX adopts a conservative decision policy: if any workflow escalates, the overall verdict is *actionable*. Otherwise, alerts are assigned to interpretable non-actionable categories (Table 2). Reports for actionable alerts extract observables (e.g., IPs, accounts) and recommend follow-up questions.

Tooling. Typed tools ground reasoning (Table 3); examples include `getUserRecord`, `searchBehaviorEvents`, and `runStructuredQuery`. These abstractions standardize access to

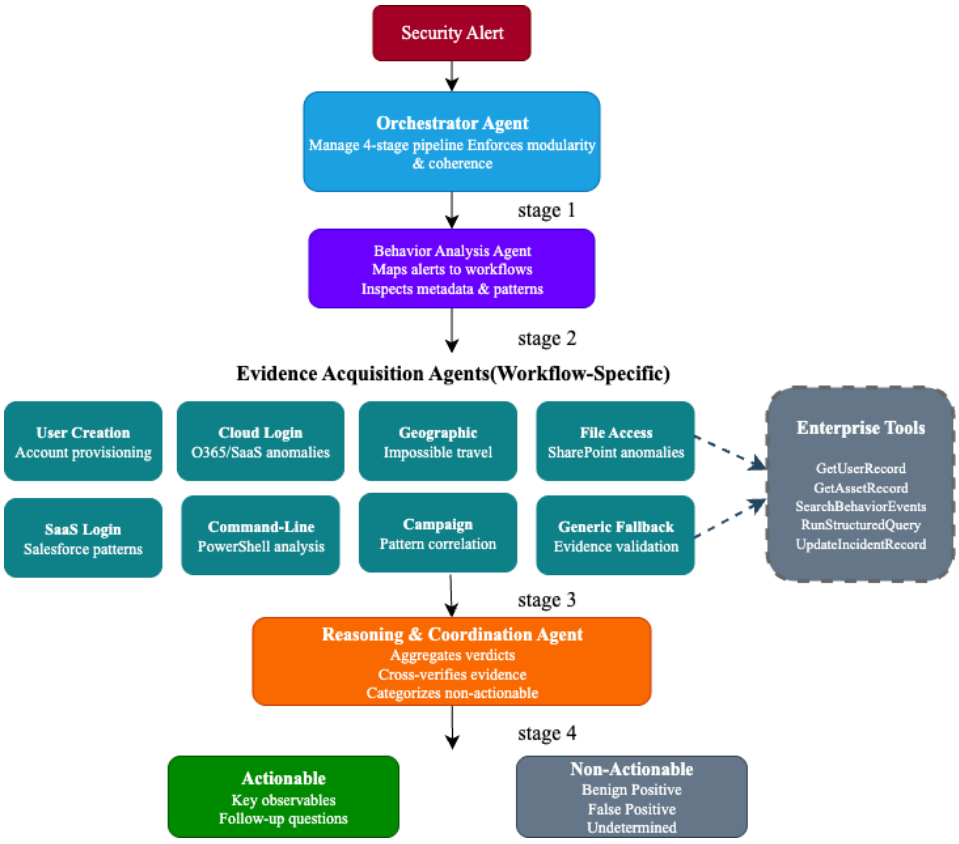

Figure 1: **CORTEX architecture.** A security alert enters a four-stage pipeline. *Stage 1: Orchestrator Agent* manages execution and modularity. *Stage 2: Behavior Analysis Agent* maps alerts to workflows. *Stage 3: Evidence Acquisition Agents* (workflow-specific) query enterprise tools (e.g., SIEM, identity, asset context) using typed APIs to validate hypotheses. *Stage 4: Reasoning & Coordination Agent* aggregates workflow outputs, cross-verifies evidence, applies conservative escalation logic, and emits a structured, auditable report with observables and follow-ups.

Table 2: Non-actionable categories.

Category	Definition
Benign Positive	Suspicious but expected behavior
False Positive—Logic	Due to flawed detection rule
False Positive—Data	Due to data errors
Undetermined	Insufficient evidence

logs, user and asset records, and parametric queries, ensuring decisions remain auditable and reproducible.

3.2 Fine-Grained SOC Workflow Dataset

Construction Protocol. We collected end-to-end investigations from enterprise SOCs across ten scenarios, spanning cloud identity, SaaS file access, endpoint detections, and IAM policy changes. For each alert, the dataset records raw telemetry, analyst actions, tool queries, intermediate reasoning, and final adjudication. Sensitive fields (e.g., usernames, hostnames, IPs) are pseudonymized while preserving structural integrity. All traces are serialized as JSON with a consistent schema, enabling both supervised training and structured evaluation.

Schema. Each JSON trace contains a unique `id`; the investigated `entity` (e.g., user account, IAM role, or endpoint); metadata such as `account`, `tenant`, `timestamp` (Unix epoch), `time_iso` (ISO

Table 3: Representative tools.

Tool	Description	Example Workflow
`getUserRecord`	Retrieve user attributes	Credential Change, User Creation
`getAssetRecord`	Endpoint context	Command-Line Execution
`searchBehaviorEvents`	Raw logs	Generic
`searchBehaviorSummaries`	Aggregates	Geo, Cloud Login
`runStructuredQuery`	Parametric reports	Cloud, File, SaaS Login
`updateIncidentRecord`	Write status	Reporting

8601); and an overall `riskScore`. The core of each record is the `properties` object, which holds one or more triggered behavioral rules. A rule specifies the `behaviorRule` name, a human-readable `description`, an `attributes` dictionary of evidence fields, a local `riskScore`, and optional `risks` tags. Attributes cover identity (`Username`, `ARN`, `UserType`); network context (`ClientIP`, `ActorIP`, `City`, `Country`, `ISP`); system context (`OS`, `BrowserType`, `Hostname`, `Workload`); operational semantics (`Operation`, `EventName`, `CmdLine`, `ParentProcess`); artifacts (`FileName`, `ExploitPath`); and security annotations (`MFA`, `Severity`, `Remediation`, `Verdict`).

Examples. An O365 login trace records IP geolocation, ISP, OS, and first-seen login flags. An AWS IAM modification trace captures IAM roles and API calls (e.g., `PutRolePolicy`) with associated user-agent strings. A Defender uncommon-activity trace highlights guest account additions in Azure AD groups, annotated with severity and verdict. Endpoint threat-control traces log PowerShell command lines, parent processes, and remediation outcomes. These heterogeneous sources are normalized into a unified schema while retaining domain-specific detail.

Features and Labels. From each record we derive contextual features (user, asset, environment identifiers), behavioral features (operations, commands, file-access patterns), and security annotations (analytic risk scores, anomaly tags). Each investigation is labeled with a triage outcome: *Actionable* (escalated to incident) or *Non-actionable*, further subclassed into *Benign Positive*, *False Positive—Logic*, *False Positive—Data*, or *Undetermined*. This supports both coarse- and fine-grained evaluation.

Scale. The dataset comprises several thousand traces. Each trace typically contains two to four behavioral rules and six to twelve attributes. Coverage spans cloud (Azure AD, AWS), SaaS (OneDrive, SharePoint), and endpoint sources, reflecting the multi-signal nature of real SOC alerts.

4 Experiments

4.1 Experimental Setup

Datasets. All experiments use the fine-grained SOC workflow dataset described in Section 3.2. All PII is pseudonymized upstream as part of the dataset.

Tasks and Outputs. Given an alert trace (JSON), systems must produce a schema-valid triage report with: (i) a binary verdict (Actionable / Non-actionable), (ii) a non-actionable subclass when applicable (Benign Positive, False Positive—Logic, False Positive—Data, Undetermined), (iii) a brief rationale grounded in fetched evidence, and (iv) extracted observables (e.g., user, IP, file, asset). Outputs must comply with a fixed JSON schema (Appendix, Listing A.1).

Baselines (Single-Agent). We compare two single-model settings: *Prompt-only.* A single LLM consumes the alert JSON and emits the triage report without tool calls. Prompts include task instructions, schema constraints, and few-shot exemplars per workflow. *ReAct-style tool use.* A single LLM plans and executes tool calls (same typed tools as CORTEX) via a ReAct prompt. Tool budget and turn caps are matched to CORTEX for fairness. The model must ground claims in returned tool outputs and emit a schema-valid report.

CORTEX Configuration (Ours). CORTEX instantiates the Orchestrator, Behavior Analysis, workflow-specific Evidence Acquisition Agents, and the Reasoning & Coordination Agent (Fig. 1). The Behavior Analysis Agent routes to one or more workflows; Evidence Agents execute calibrated

Table 4: Decision performance on the test set. FPR is computed on non-actionable predictions.

Model	Act. F1	Non-act. F1	Subclass F1	FPR (%)
Single-agent (prompt)	0.61	0.73	0.49	29.8
Single-agent (tool-use)	0.66	0.77	0.54	24.9
CORTEX (ours)	**0.78**	**0.86**	**0.69**	**14.2**

Table 5: Efficiency comparison. Latency is median end-to-end time per alert (full ticket resolution).

Model	Tokens	Tool Calls	Latency (s)
Single-agent (prompt)	2,100	0.0	28
Single-agent (tool-use)	4,152	1.3	44.6
CORTEX (ours)	**23,600**	**3.1**	**152.4**

checks (Table 1); the Reasoning Agent applies conservative "escalate-on-any" synthesis and composes the structured report. All agents share the same tool library used by the ReAct baseline.

Implementation. We implement all agents using the *OpenAI Agents SDK* under the *Model Context Protocol (MCP)*. Each typed tool (e.g., `getUserRecord`, `searchBehaviorEvents`, `runStructuredQuery`) is exposed as an MCP *tool* with JSON-schema arguments and deterministic JSON returns; logs and auxiliary artifacts are exposed as MCP *resources*. Inter-agent communication occurs via MCP sessions with per-alert, ephemeral context; cross-alert state is disabled to prevent leakage. The ReAct baseline is implemented over the same SDK and MCP tool adapters to ensure parity; measured latencies therefore include SDK/MCP overhead uniformly across systems. All MCP traces are logged for auditability.

Evaluation Metrics. *Decision quality:* macro-F1 over Actionable/Non-actionable; subclass macro-F1 over the four non-actionable categories; false-positive rate (FPR) computed on non-actionable predictions that disagree with ground truth; recall computed on actionable alerts. *Efficiency:* output tokens, tool calls, and end-to-end latency .

4.2 Triage Performance

Table 4 reports classification quality across models. CORTEX achieves the strongest overall performance, improving actionable F1 by $+0.12$ over the best single-agent baseline ($0.66 \rightarrow 0.78$) and reducing the false-positive rate by 10.7 points ($24.9\% \rightarrow 14.2\%$). Macro-F1 across actionable/non-actionable decisions reaches 0.82, and subclass F1 increases by $+0.15$, reflecting sharper distinctions among benign positives, logic-driven false positives, and data-driven false positives.

4.3 Efficiency

CORTEX trades additional coordination for higher decision quality while remaining within our target SOC triage SLO of $\sim$3 min per full ticket. Its median end-to-end time is 152.4 s (≈ 2.54 min), compared to 44.6 s for the single-agent ReAct-style baseline and 28 s for prompt-only. This corresponds to a $+107.8$ s increase over the tool-using baseline ($+241.7\%$, $3.42\times$ slower) and $+124.4$ s over prompt-only ($+444.3\%$, $5.44\times$ slower). The primary driver is the larger token footprint introduced by multi-agent message passing and the serialization of richer tool outputs: CORTEX processes $23{,}600$ tokens vs. $4{,}152$ for the tool-using baseline ($+468.4\%$, $5.68\times$). Average tool calls rise only modestly (from 1.3 to 3.1, Δ=1.8, $2.38\times$), indicating that the latency gap is not solely due to more API hits but also to increased deliberation and inter-agent exchange. These efficiency costs accompany the accuracy gains reported in Table 4 (higher F1, lower FPR) and yield more auditable, evidence-grounded investigations.

5 Conclusion

We introduced CORTEX, a collaborative, tool-grounded multi-agent architecture for high-stakes SOC alert triage. By decomposing the task across role-specialized agents and constraining each

step to operate over typed tools and auditable artifacts, CORTEX improves both decision quality and transparency relative to single-agent baselines. On our evaluation suite, CORTEX increases actionable F1 from 0.66 to 0.78 and reduces false positives from 24.9% to 14.2% while maintaining operationally acceptable end-to-end latency. Beyond outcome metrics, CORTEX produces structured reports with explicit evidence links, enabling downstream review, compliance, and post-incident learning.

A second contribution is a fine-grained SOC workflow dataset that captures full investigative traces—alerts, tool queries, intermediate observations, and final adjudications—across diverse enterprise scenarios. This process-level supervision supports training agents to follow disciplined playbooks rather than relying solely on outcome labels, and it enables new measurements of reasoning fidelity (step accuracy, tool-policy match, and grounding consistency).

Limitations and Future Work. Our evaluation is limited by dataset coverage (ten-plus scenarios) and the availability and quality of upstream telemetry. Like other agentic systems, CORTEX can be sensitive to distribution shift, prompt injection, or incomplete context returned by tools. Future directions include (i) stronger termination and verification protocols (e.g., cross-checking with learned critics), (ii) adaptive tool budgeting and scheduling across agents, (iii) distillation of multi-agent traces into a compact single-model policy for cost/latency reduction, (iv) continual learning from analyst feedback and A/B tests, and (v) expanded benchmarks for red-team robustness and privacy-preserving operation.

Overall, CORTEX offers a practical template for auditable, role-specialized LLM agents in security operations. We hope the architecture, evaluation protocol, and released dataset catalyze further research on reliable, efficient agents for safety-critical domains.

References

Bushra A AlAhmadi, Louise Axon, and Ivan Martinovic. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. *arXiv preprint arXiv:2210.01649*, 2022.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

Emre Cemri, Yanchen Zhang, Kevin Smith, et al. Why do multi-agent LLM systems fail? In *Conference Proceedings*, 2025.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, 2024.

Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *Proceedings of the 33rd USENIX Security Symposium*, 2024.

Dropzone AI. The rise of AI SOC analysts: Transforming security operations. `https://www.dropzone.ai/blog/`, 2024.

Dropzone AI. Alert triage in 2025: The complete guide to 90% faster investigations, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning*, 2023.

Enterprise Strategy Group. Web application and API security survey. Technical report, ESG for Fastly, 2023.

Jim Finkle and Susan Heavey. Target says it declined to act on early alert of cyber breach. *Reuters*, 2014.

Xiaorui Gong, Xiaoyu Wang, Xiaobo Yang, and Xueping Liang. Combating alert fatigue with AlertPro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection. *Computers & Security*, 137:103621, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Xiong, Yuheng Ming, Ceyao Zha, Jinlin Zhang, Chenglin Wen, Yiqi Zhang, Kunlun Zheng, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Xiaojun Hu, Yang Zhang, Jing Li, and Haixin Chen. Building threat intelligence knowledge graphs with large language models. *IEEE Transactions on Information Forensics and Security*, 19: 2345–2358, 2024.

IBM Security. Cost of a data breach report 2025. Technical report, IBM Corporation, 2025.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, 2023.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, 2024.

Orca Security. 2022 cloud security alert fatigue report. Technical report, Orca Security, 2022.

OWASP Foundation. OWASP top 10 for large language model applications. `https://owasp.org/www-project-top-10-for-large-language-model-applications/`, 2024.

Radiant Security. Radiant security: AI-powered SOC automation platform. `https://radiantsecurity.ai/`, 2025a.

Radiant Security. Radiant security announces breakthrough in autonomous SOC operations. Press Release, 2025b.

Michael Riley, Ben Elgin, Dune Lawrence, and Carol Matlack. Missed alarms and 40 million stolen credit card numbers: How Target blew it. *Bloomberg Businessweek*, 13:1–7, 2014.

Tines. Voice of the SOC report. Technical report, Tines, 2023.

Melissa J. M. Turcotte, François Labrèche, and Serge-Olivier Paquette. Automated alert classification and triage (AACT): An intelligent system for the prioritisation of cybersecurity alerts. *arXiv preprint arXiv:2505.09843*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems*, volume 35, pages 24824–24837, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Jie Zhang, Haoyu Bu, Hui Wen, Yongji Liu, Haiqiang Fei, Rongrong Xi, Lun Li, Yun Yang, Hongsong Zhu, and Dan Meng. When LLMs meet cybersecurity: A systematic literature review. *Cybersecurity*, 8(55), 2025a.

Qian Zhang, Wei Liu, Ming Chen, and Xiaoyu Wang. AttacKG+: Enhancing attack knowledge graphs with large language models. In *IEEE Symposium on Security and Privacy*, 2025b.

A Workflow Prompts

This appendix provides the full YAML definitions for all workflow prompts used in CORTEX. To align with the main paper, we use the *agent* roles **Orchestrator**, **Behavior Analysis Agent**, **Evidence Acquisition Agents**, and **Reasoning & Coordination Agent**; and the *typed tools* `getUserRecord`, `getAssetRecord`, `searchBehaviorEvents`, `searchBehaviorSummaries`, `runStructuredQuery`, `updateIncidentRecord`. Decision policies (e.g., escalation thresholds) are applied by the **Reasoning & Coordination Agent** rather than via a separate "Action_Detection_Agent" tool.

Table 6: Summary of workflow prompts in CORTEX. Decision policies are enforced by the Reasoning & Coordination Agent; evidence is fetched via typed tools.

Workflow	Detection Goal	Key Evidence / Tools
Add User	Suspicious user creation/update	`getUserRecord` (roles), policy thresholds
Authentication Change	Auth method changes	`getUserRecord`, policy (remove/add/change)
Coro	Coro vendor signals	behaviorRule extraction, policy=escalate
Generic	Fallback analysis	`searchBehaviorEvents`, policy triage
Multiple ISP	Impossible travel	`runStructuredQuery`(‘GetRecentLoginActivity‘)
O365 Guest	Guest account activity	`getUserRecord` (guest roles)
O365 Login	Risky login	`runStructuredQuery`(‘GetRecentHighRiskActivity‘)
PowerShell	Endpoint execution	`searchBehaviorEvents`, `getAssetRecord`
Salesforce Abnormal Login	Risky Salesforce login	`runStructuredQuery`(‘GetRecentRuleActivity‘)
SharePoint File	Risky file access	risk score from event

A.1 Add User Workflow

```
name Add_User_Workflow
description >
  Analyze events that create or update users. Match when behavior rules contain
  "User_Added", "User_Updated", or "Add_Member_To_Group".
model gpt-5-mini
prompt |
  Role Orchestrator for an O365 user add/update event (JSON).

  Steps
  1) Extract target user email(s) from 'properties.*.TargetUser'. For each target
       user
     - Evidence Acquisition Agent invokes typed tool ‘getUserRecord(email, account,
       tenant)‘.
     - If no email or record, mark target_user_record = "Unknown".
  2) Determine 'target_user_admin' from returned user record(s) (roles include admin
      -like privileges).
     - This classification is performed by the Reasoning & Coordination Agent (no
       external tool).
  3) Reasoning & Coordination Agent applies policy:
     - If any target_user_admin = Admin => actionable = true; else actionable =
       false.
  4) Emit schema-valid JSON as below.

  Output JSON schema
  {
    "report" {
      "target_user_record" "<Found/Unknown>",
      "target_user_admin" "<Admin/User/Unknown>",
      "reasoning_target_user_admin" "<string>"
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }
```

```
tools
  - name getUserRecord
    description Retrieve a user's␣directory␣record␣for␣roles/attributes.
```

A.2 Authentication Change Workflow

```
name Authentication_Change_Workflow
description >
  Analyze authentication method changes add, remove, or modify MFA/password.
  Match when behavior rules contain "Add_Authentication_Method" or "
      Remove_Authentication_Method".
model gpt-5-mini

prompt |
  Role Orchestrator for an authentication method change event (JSON).

  Steps
  1) Identity user email = input.entity; (account, tenant) from input.
  2) Evidence Acquisition Agent calls `getUserRecord(email, account, tenant)`.
     - If not found report.user_record = "Unknown"; new_user = "Unknown"; goto Step
       4.
  3) If found Reasoning & Coordination Agent computes account age vs time_iso to set
      :
     new_user = Yes/No/Unknown, with justification.
  4) Policy (Reasoning & Coordination Agent):
     - Removing method => actionable = true.
     - Not removing, and new_user = Yes => actionable = false.
     - Not removing, and new_user = No or Unknown => actionable = true.
  5) Emit schema-valid JSON.

  Output JSON schema
  {
    "report" {
      "user_record" "<Found/Unknown>",
      "new_user" "<Yes/No/Unknown>",
      "reasoning_new_user" "<string>"
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools
  - name getUserRecord
    description Retrieve a user's␣directory␣record␣for␣roles/attributes.
```

A.3 Correlation (Coro) Workflow

```
name Coro_Workflow
description >
  Match Coro vendor events ("Coro_*") and escalate per policy.
model gpt-5-mini

prompt |
  Role Orchestrator for a Coro event (JSON).

  Steps
  1) Extract user_email = input.entity; list behavior_rule names from input.
  2) Policy Reasoning & Coordination Agent marks actionable = true for vendor Coro
      events.
  3) Emit schema-valid JSON with behavior_rules included.

  Output JSON schema
```

```
  {
    "report" { "user_email" "<string>", "behavior_rules" ["<string>", ...] },
    "actionable" true,
    "reasoning" "Escalated␣per␣Coro␣vendor␣policy.",
    "summary" "<string>"
  }
```

A.4 Generic Workflow

```
name Generic_Workflow
description >
  Fallback workflow for alerts that do not match any specific workflow.
model gpt-5-mini

prompt |
  Role Orchestrator coordinating Evidence Acquisition and Reasoning.

  Steps
  1) Evidence Acquisition Agent may call ‘searchBehaviorEvents(...)‘ for relevant
       raw logs,
     if and only if the input lacks sufficient direct evidence fields.
  2) Reasoning & Coordination Agent validates event evidence and determines
       actionability:
     - CLOSE_TICKET (invalid/not actionable)
     - ESCALATE_TO_TIER_TWO (valid/actionable)
     - REQUIRES_ADDITIONAL_INFO (insufficient/conflicting)
  3) Emit schema-valid JSON.

  Output JSON schema
  {
    "report"{
      "validation" true/false,
      "validation_reasoning" "<string>",
      "recommendation" "CLOSE_TICKET" |"ESCALATE_TO_TIER_TWO" |"
          REQUIRES_ADDITIONAL_INFO"
    },
    "actionable" true/false,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools
  - name searchBehaviorEvents
    description Query raw telemetry for supporting evidence.
```

A.5 Multiple ISP (Impossible Travel) Workflow

```
name MultipleISP_Workflow
description >
  Determine whether observed logins constitute impossible travel.
  Match when behavior rule includes "Multiple_ISPs" or multiple ISPs under an O365
       login rule.
model gpt-5-mini

prompt |
  Role Orchestrator for impossible-travel evaluation.

  Steps
  1) Evidence Acquisition Agent calls ‘runStructuredQuery(GetRecentLoginActivity,
       account+tenant, key=entity, time_iso)‘
     to obtain recent login activity with geo/ISP fields.
  2) Reasoning & Coordination Agent evaluates distance/time between logins (8-hour
       window),
```

```
    considering ISP diversity and plausible explanations (VPN/mobile).
    Sets report.impossible_travel (bool) with detailed reasoning.
  3) Policy actionable = report.impossible_travel.
  4) Emit JSON.

  Output JSON schema
  {
    "report" {
      "impossible_travel" <bool>,
      "impossible_travel_reasoning" "<string>"
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools
  - name runStructuredQuery
    description Execute a parametric report (e.g., GetRecentLoginActivity) and
        return JSON rows.
```

A.6 Office 365 Guest Workflow

```
name Office365_Guest_Workflow
description >
  Analyze guest user activity (key formatted as <username>#ext#@<tenant>.onmicrosoft.
      com).
model gpt-5-mini

prompt |
  Role Orchestrator for guest user activity.

  Steps
  1) Evidence Acquisition Agent calls `getUserRecord(guest_email, account, tenant)`.
     - If not found guest_user_record = "Unknown"; guest_user_admin = "Unknown".
  2) Reasoning & Coordination Agent inspects roles in record (if found) to classify:
     guest_user_admin = Admin/User.
  3) Policy actionable = (guest_user_admin == Admin).
  4) Emit JSON.

  Output JSON schema
  {
    "report" {
      "guest_user_record" "<Found/Unknown>",
      "guest_user_admin" "<Admin/User/Unknown>",
      "reasoning_guest_user_admin" "<string>"
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools
  - name getUserRecord
    description Retrieve guest user directory record for role evaluation.
```

A.7 Office 365 Login Workflow

```
name Office365_Login_Workflow
description >
  Analyze O365 login rule with risk and recent high-risk context.
model gpt-5-mini
```

```
prompt |
  Role Orchestrator for O365 login event.

  Steps
  1) user_email = input.entity.
  2) Evidence Acquisition Agent calls
     `runStructuredQuery(GetRecentHighRiskActivity, account+tenant, key=user_email,
         time_iso)`
     to count high-risk activities (rowCount) and keep 'row' JSON.
  3) Extract the risk score for the O365 Login behavior rule (not the ticket total).
  4) Reasoning & Coordination Agent applies policy:
     - risk <= 1000 => actionable = false
     - risk >1000 & recent_high_risk_count == 0 => actionable = false
     - risk >1000 & recent_high_risk_count >0 => actionable = true
  5) Emit JSON.

  Output JSON schema
  {
    "report" {
      "user_email" "<string>",
      "recent_activity_riskScore_greater_than_2000_count" <int>,
      "high_risk_activity_raw_json_row" "<string>"
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools
  - name runStructuredQuery
    description Execute `GetRecentHighRiskActivity` and return rows and rowCount.
```

A.8 PowerShell Workflow

```
name Powershell_Workflow
description >
  Analyze PowerShell execution for malicious behavior and remediation status.
model gpt-5-mini

prompt |
  Role Orchestrator for PowerShell event.

  Steps
  1) Evidence Acquisition Agent reads event 'attributeSummaries' (no external tool),
     classifies code as Malicious / Non-Malicious with one-sentence rationale.
  2) Evidence Acquisition Agent checks disinfection status from event fields
     (e.g., status/actionTaken indicates 'disinfected' => Disinfect; else Non-
         Disinfect).
  3) Reasoning & Coordination Agent applies policy:
     actionable = (powerShell_Malicious == true AND user_has_admin == true).
     (Admin status may be derived via `getUserRecord` if user context provided.)
  4) Emit JSON.

  Output JSON schema
  {
    "report" {
      "powerShell_Malicious" <bool>,
      "reasoning" "<string>",
      "dis_Infect_Detection" "<Disinfect/Non-Disinfect>",
      "reasoning_Dis_Infect" "<string>"
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
```

```
  }

tools
  - name getUserRecord
    description Retrieve user record if admin status is required for policy.
```

A.9 Salesforce Abnormal Login Workflow

```
name Salesforce_Abnormal_Login_Workflow
description >
  Analyze Salesforce abnormal login rule with recent rules context.
model gpt-5-mini

prompt |
  Role Orchestrator for Salesforce abnormal login event.

  Steps
  1) user_email = input.entity.
  2) Evidence Acquisition Agent calls
     `runStructuredQuery(GetRecentRuleActivity, account+tenant, rule="
         Fluency_Salesforce_Login_Status_Abnormal", key=user_email, time_iso)`
     and obtains recent_rule_count = rowCount.
  3) Reasoning & Coordination Agent policy:
     - recent_rule_count < 3 => actionable = false
     - recent_rule_count >= 3 => actionable = true
  4) Emit JSON.

  Output JSON schema
  {
    "report" {
      "user_email" "<string>",
      "recent_rule_count" <int>
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools
  - name runStructuredQuery
    description Execute `GetRecentRuleActivity` with the named rule and return
        rowCount.
```

A.10 SharePoint File Workflow

```
name Sharepoint_File_Workflow
description >
  Analyze SharePoint file access/download risk based on rule risk score.
model gpt-5-mini

prompt |
  Role Orchestrator for SharePoint file event.

  Steps
  1) Extract the risk score for the SharePoint File behavior rule from input.
      properties.
  2) Reasoning & Coordination Agent policy:
     - risk <= 1000 => actionable = false
     - risk >1000 => actionable = true
  3) Emit JSON.

  Output JSON schema
  {
```

```
    "report" {
      "sharepoint_risk_score" <int>
    },
    "actionable" <bool>,
    "reasoning" "<string>",
    "summary" "<string>"
  }

tools []
```

B Case Studies

We present five case studies illustrating how CORTEX processes alerts. Each includes the agent reasoning trace and the final JSON report.

B.1 Case Study 1: Add User Workflow

Event: New user added to a tenant with elevated privileges.

Reasoning Trace:

- **Behavior Analysis Agent**: Matches `User_Added` → routes to Add User Workflow.
- **Evidence Acquisition Agents**: Invoke `getUserRecord(jane.doe@corp.com)`; record found.
- **Reasoning & Coordination Agent**: Roles include `GlobalAdmin` → target_user_admin=Admin; policy ⇒ actionable.

Final Report:

```
{
  "report": {
    "target_user_record": "Found",
    "target_user_admin": "Admin",
    "reasoning_target_user_admin": "Target user roles include GlobalAdmin in
        the retrieved user record."
  },
  "actionable": true,
  "reasoning": "Admin privilege assignment is actionable under policy.",
  "summary": "New user provisioned with Global Admin privileges; escalate
     for immediate review."
}
```

B.2 Case Study 2: Authentication Change Workflow

Event: A long-standing user removed MFA and switched to password login.

Reasoning Trace:

- **Behavior Analysis Agent**: Matches `Remove_Authentication_Method`.
- **Evidence Acquisition Agents**: `getUserRecord(john.smith@corp.com)`; createdDateTime=2019.
- **Reasoning & Coordination Agent**: new_user=No; removal policy ⇒ actionable.

Final Report:

```
{
  "report": {
    "user_record": "Found",
    "new_user": "No",
```

```
    "reasoning_new_user": "User account created in 2019(older than 30days)."
  },
  "actionable": true,
  "reasoning": "Removal of an authentication method is always actionable.",
  "summary": "Established user removed MFA; flagged for potential account
      compromise."
}
```

B.3 Case Study 3: Multiple ISP / Geo-Impossibility

Event: User logged in from London and New York within 30 minutes.

Reasoning Trace:

- **Behavior Analysis Agent**: Matches Multiple ISPs.
- **Evidence Acquisition Agents**: `runStructuredQuery(GetRecentLoginActivity,...)` returns London 10:15 UTC (BT UK) and NYC 10:45 UTC (Verizon).
- **Reasoning & Coordination Agent**: ∼5,500 km in 30 min ⇒ impossible travel ⇒ actionable.

Final Report:

```
{
  "report": {
    "impossible_travel": true,
    "impossible_travel_reasoning": "Logins at 10:15UTC (London) and 10:45UTC
        (New York) are ~5,500 km apart within 30minutes, exceeding feasible
        travel limits."
  },
  "actionable": true,
  "reasoning": "Pattern reflects impossible travel within the evaluation
      window.",
  "summary": "User exhibited impossible travel (London to NYC within 30
      minutes); escalate."
}
```

B.4 Case Study 4: Office 365 Guest Workflow

Event: Guest user invited into Finance group.

Reasoning Trace:

- **Behavior Analysis Agent**: Matches Guest user pattern.
- **Evidence Acquisition Agents**: `getUserRecord(jane.guest#ext@othercorp.com)`; record found.
- **Reasoning & Coordination Agent**: roles=User ⇒ not actionable.

Final Report:

```
{
  "report": {
    "guest_user_record": "Found",
    "guest_user_admin": "User",
    "reasoning_guest_user_admin": "Guest user record shows no admin roles;
        roles indicate standard user access."
  },
  "actionable": false,
  "reasoning": "Guest does not hold admin privileges under policy.",
```

```
  "summary": "Guest added to Finance group with no admin privileges; not
      actionable."
}
```

B.5 Case Study 5: PowerShell Workflow

Event: PowerShell execution with encoded command attempting persistence.

Reasoning Trace:

- **Behavior Analysis Agent**: Matches PowerShell execution.
- **Evidence Acquisition Agents**: Inspect `attributeSummaries` ⇒ `-EncodedCommand` and registry persistence ⇒ Malicious; disinfection status = Disinfect.
- **Reasoning & Coordination Agent**: If user is Admin (via `getUserRecord`), policy ⇒ actionable despite disinfection.

Final Report:

```
{
  "report": {
    "powerShell_Malicious": true,
    "reasoning": "Encoded command and registry-based persistence consistent
        with malicious behavior.",
    "dis_Infect_Detection": "Disinfect",
    "reasoning_Dis_Infect": "Endpoint telemetry indicates disinfection
        completed."
  },
  "actionable": true,
  "reasoning": "Malicious PowerShell executed by an Admin meets escalation
      policy.",
  "summary": "Admin executed malicious PowerShell with persistence;
      disinfected but escalated for follow-up."
}
```