Proceedings of Machine Learning Research 299:1–19, 2025 Conference on Applied Machine Learning for Information Security

RIG-RAG: A GraphRAG Inspired Approach to Agentic Cloud Infrastructure

Benji Lilley BL857@DREXEL.EDU
Brian S. Mitchell BMITCHELL@DREXEL.EDU
Spiros Mancoridis MANCORS@DREXEL.EDU
Department of Computer Science
Drexel University
Philadelphia, PA, USA 19104

Editor: Edward Raff and Ethan M. Rudd

Abstract

Modern cloud environments contain thousands of interdependent resources that frequently change, making them complex to monitor with traditional tools. This paper introduces Relational Inference GraphRAG (RIG-RAG), a pair of LLM-assisted inference pipelines that transform raw cloud configuration data into a security-enriched typed knowledge graph to support natural-language reasoning about deployed infrastructure. RIG-RAG enables organizations to execute natural language security queries against their cloud environments and to support the continuous validation of critical questions such as "What resources are publicly accessible?" or "Which identities can read from databases?". Our implementation, SkyShark IQ, processes queries in two modes: *interactive mode* for ad hoc user interaction, and *oversight mode* for periodically executed curated inquiries that monitor infrastructure drift. By extracting cloud resource configurations into a typed graph structure with inferred security relationships, SkyShark IQ allows security teams to reason over complex infrastructure through an intuitive conversational interface. Based on the current state of the deployed infrastructure, the system provides role-appropriate security insights, providing technical details to analysts, explaining implementation impacts to product owners, and presenting contextual risk summaries to business leaders. We deployed and evaluated our system in a production AWS environment that supports 300,000 users, demonstrating its ability to simplify complex infrastructure analysis, surface hidden security relationships, and provide verifiable, role-appropriate explanations that improve situational awareness and cross-team communication in operational security workflows.

Keywords: Cloud Security, Agentic AI, Security Operations, Infrastructure Analysis

1. Introduction

Cloud computing has transformed the way modern applications are built and operated, enabling highly distributed, scalable systems that are easy to define and deploy. However, this simplicity at the surface belies the growing complexity below. The widespread adoption of cloud-native paradigms has led to a dramatic expansion of organizational infrastructure footprints, both in scale and in structural complexity.

Cloud providers encourage the use of Infrastructure-as-Code (IaC) and offer powerful automation tools for provisioning resources. This has resulted in organizations managing sprawling inventories of semi-structured configuration data, generated by heterogeneous services across multiple platforms. Although these data are critical for orchestration and

deployment, they are poorly suited for security reasoning without deep domain-specific knowledge.

Modern cloud architectures differ fundamentally from traditional data center designs. The familiar "layered" approach—with tightly controlled perimeters and segregated internal networks—has given way to a flatter, more fluid topology. In many cases, private cloud resources can become publicly accessible with a single misconfiguration. As the saying goes, "everything is public" in the public cloud.

To manage this complexity, organizations rely on operational and security monitoring tools that are largely rule-based and rooted in legacy infrastructure models. Although effective for isolated violations or threshold breaches, these tools struggle to model the intricate interdependencies of cloud environments. Defining security rules for individual resources is straightforward, but reasoning in dozens of interrelated services quickly becomes intractable.

When security and operational teams need to answer questions such as "What resources are publicly accessible?" or "Which configuration caused an observed behavior?" they resort to manual analysis across multiple cloud interfaces. This process is time-consuming, error-prone, and difficult to maintain as environments evolve. It also requires a depth of knowledge in cloud infrastructure, which is typically distributed across many teams and requires communication and coordination among many experts.

This gap in situational awareness introduces critical security risks. Organizations may overlook misconfigurations until they manifest as security incidents, at which point resolving them requires time-sensitive coordination of multiple teams with different priorities and expertise. Periodic compliance scans offer only point-in-time validation against static rule libraries, missing the dynamic nature of modern cloud deployments. These fragmented efforts involving manual checks stitched together with brittle automation fall short on explainability, consistency, and context, making proactive security validation nearly impossible at scale.

To address these challenges, we created Relational Inference GraphRAG (RIG-RAG), which transforms cloud resource configurations into a security and operational enriched knowledge graph enabling natural language security-reasoning queries against infrastructure. Our implementation, *SkyShark IQ*, infers security, dependency, and operational relationships not explicitly encoded in cloud metadata, allowing teams to validate security properties and understand cloud solutions through natural language queries. *SkyShark IQ* brings an "agent in the loop" to an environment that is primarily operated by human interactions across many teams.

The novelty of our approach lies in how it enables ongoing relationship inference and reasoning of security posture through natural language queries, which form a living corpus that is maintained alongside infrastructure code and executed continuously to ensure security posture does not degrade over time. The results of these queries can be logged to provide a forensic trail of security assessments over time, enabling historical analysis of how infrastructure security has evolved. Furthermore, modern LLM foundation models are guided to use inference to classify the risk of findings and automatically recommend when human intervention is necessary. Since the entire system is built on an agentic architecture, security practitioners can engage directly through a conversational interface to ask follow-up

questions to guide remediation efforts, thus bridging the gap between automated detection and human-driven response. The contributions of this work are listed below.

- A novel use of structured graph-based RAG that is tailored to cloud infrastructure and security triage.
- A security-predicate type system that enriches raw topology data with semantic relationships.
- A natural language query system that supports continuous security validation through security-reasoning queries against live infrastructure.
- An interface that tailors information to different roles: a) providing security analysts with technical details, b) helping product owners understand implementation implications to assess effort and risk, and c) presenting business leaders with contextual risk information in appropriate terminology and level of abstraction.
- Validation of production-scale effectiveness through analysis of a live enterprise infrastructure configuration consisting of 406 resources and 1,528 relationships supporting 300,000 users. Expert assessment confirmed practical utility in real-world security scenarios.

The remainder of this paper is organized as follows. It begins by surveying related work on knowledge graph generation and cloud security automation. Section 3 describes the design of RIG-RAG and the architecture of our agentic triage system. Section 4 presents the implementation of the *SkyShark IQ* tool. Section 5 outlines ideas to extend our work and Section 6 summarizes our described effort.

2. Related Work

Our work is at the intersection of several active research areas: retrieval-augmented generation (RAG), graph-based knowledge representation, agentic AI, and cloud security automation and compliance.

Various tools have emerged to help organizations manage security and compliance in cloud environments. The Cloud Custodian Project (2024) uses policy-as-code through a domain-specific YAML language, enabling the definition and enforcement of security rules across cloud resources. *AWS Config* (Amazon Web Services, 2024b) and *Prisma Cloud* (Palo Alto Networks, 2024) provide continuous compliance monitoring and drift detection, while commercial platforms like Wiz (2024) and Orca Security (2024) offer asset inventory, attack path analysis, and contextual risk scoring using proprietary scanning and graph-based approaches. These platforms excel at predefined rule evaluation but struggle with ad-hoc security questions and lack natural language interfaces for complex infrastructure reasoning. Traditional infrastructure query tools such as AWS CLI (Amazon Web Services, 2024a) and CloudMapper (Duo Labs, 2025) provide programmatic access but require deep technical expertise to correlate information across services, necessitating custom scripts for complex security analysis. In contrast, RIG-RAG transforms cloud configurations into a *security-enriched knowledge graph* that captures both explicit system topology and *inferred*

security typed relationships. This structured representation enables natural-language queries over the deployed infrastructure, supporting explainable reasoning by traversal of graph relationships along with continuous validation. Unlike existing tools, RIG-RAG pairs graph-based inference with a natural language interface, offering a unified view of cloud risk tailored to security, engineering, and business stakeholders.

Table 1: Comparison of Our Approach with Existing Methods

Approach	**Core Capabilities**	**Limitations**	**Our Approach**
Plain RAG	Text chunk retrieval	No structured context or relationships	Constructs explicit relationships from config data
GraphRAG	Graph from automatically extracted text entities	Relies on derived/inferred relationships from text and preexisting community documents	Provides explicit relationship types and builds graphs from infrastructure associations rather than document analysis
Structured-RAG	Uniform structured data with consistent schema	Requires uniform schemas; limited in dynamic, heterogeneous environments	Dynamically generates security-specific schemas with typed relations for security reasoning
Finetuned LLM Agents	Learn additional reasoning paths through incremental training	Expensive to train, brittle to changes, lacks instance-specific context	Grounds reasoning in actual configurations, adapts to new environments without retraining
Domain Specific Solutions	Rule-based detection engines with predefined alerts	No natural language interface, limited cross-resource reasoning capabilities	Provides contextual reasoning with stakeholder-specific explanations grounded in actual configurations

Retrieval-augmented generation (RAG) is distinct from traditional Large Language Model (LLM) approaches in that it augments the model's capabilities with external knowledge sources, enabling more relevant and precise output for queries. The seminal work in this area by Lewis et al. (2020) demonstrated the core idea of grounding an LLM's generation process into information retrieved from an external knowledge source. The goal is to improve the accuracy and relevance of the output by incorporating information on which the LLM was not trained. A comprehensive survey of RAG techniques (Gao et al., 2023) highlights the breadth of innovation in this space, covering various retrieval strate-

gies, fusion methods, and generation approaches. Our work builds on the principle of RAG, but with a focus on structured knowledge representation tailored to cloud infrastructure security.

Within the broader RAG landscape, there is a growing interest in leveraging graph structures. Early work, such as Microsoft's GraphRAG (Edge et al., 2024), explored the potential of using graphs as the retrieval source. Instead of retrieving unstructured text chunks, GraphRAG extracts entities and relationships from the input data to construct a knowledge graph to support RAG queries. Subsequent work by Sepasdar et al. (2024) extends GraphRAG when input data are structured. Note that GraphRAG infers relationships from unstructured data. The distinction between traditional RAG and GraphRAG, including their respective strengths and weaknesses, is a key consideration, as discussed in papers that directly compare the two paradigms (Han et al., 2025). Related work in LLM-based relationship extraction, such as REBEL (Huguet Cabot and Navigli, 2021), demonstrates how language models can identify and generate relationship information from unstructured text through end-to-end language generation approaches. While REBEL focuses on extracting relationships from natural language documents, our approach applies similar LLM-based relationship reasoning to semi-structured cloud configuration data, where the challenge lies in inferring security-relevant connections across heterogeneous resource schemas rather than linguistic relationship extraction. Previous work has explored graph representations for security analysis, particularly in network intrusion detection and malware classification. However, cloud infrastructure presents unique challenges: resources have heterogeneous schemas, relationships span multiple service boundaries, and security implications depend on complex policy interactions. Our approach extends GraphRAG by introducing security-specific predicates that shape the structure of the knowledge graph, allowing precise answers to complex security questions about cloud environments.

Advancements in LLMs have also fueled the development of agentic AI systems. Agentic LLMs are not passive generators of text; they become active agents capable of reasoning, planning, and executing actions. Seminal work in this area by Yao et al. (2023) has explored frameworks to build agentic solutions, including ReAct which enables LLMs to reason and act iteratively through structured prompting approaches. Recent systems like AutoGen (Wu et al., 2023) demonstrate multi-agent conversations for complex problem-solving, where multiple AI agents collaborate to address challenges requiring diverse expertise. Building on agentic LLM concepts, RIG-RAG enables users to interact with cloud security data dynamically, moving beyond simple queries on static data toward the support of dynamic security monitoring, interactive triage and improved decision quality.

Unlike traditional ML classification tasks, practical security systems resist standard accuracy metrics. False positives in security contexts create alert fatigue and reduce analyst efficiency, while false negatives represent missed threats with potentially severe consequences. Recent work on evaluation frameworks for retrieval-augmented systems, such as RAGAS (Es et al., 2024), introduces metrics for assessing RAG system performance, but cloud security adds domain-specific challenges that require operational rather than statistical evaluation approaches. Our evaluation methodology focuses on practical metrics that reflect real security team workflows: investigation time reduction, relationship discovery completeness, and stakeholder communication effectiveness.

Table 1 summarizes how RIG-RAG differs from related retrieval-augmented and domain-specific methods, highlighting its contributions in structured reasoning, adaptability, and security context awareness. Our work is driven by the need to enhance the security posture of cloud environments through an extension of retrieval-augmented and structured knowledge graphs. By integrating graph-based reasoning with a natural language interface, RIG-RAG approaches cloud security by going beyond traditional compliance monitoring and misconfiguration detection. Unlike existing tools that focus on static analysis or predefined rule sets, RIG-RAG enables continuous, dynamic validation of cloud environments through queries that traverse infrastructure dependencies capable of inferring security relationships in real-time. This allows for more efficient security triage and deeper insights into cloud risks, with the added benefit of explainability and tailored views for different stakeholders. By incorporating both automated reasoning and domain-specific security predicates, RIG-RAG represents a significant step toward more intelligent, context-aware cloud security automation.

3. Relational Inference GraphRAG (RIG-RAG)

Modern cloud infrastructure APIs expose resources through structured documents, typically JSON or XML, but these representations are optimized for provisioning rather than reasoning. Resource identifiers follow inconsistent conventions, and relationships are encoded heterogeneously across services. For example, AWS identifies virtual private clouds using IDs like `vpc-{id}`, IAM roles with ARNs, and storage resources with unique naming schemes, making it difficult to extract coherent, typed relationships using rule-based approaches.

To address these challenges, RIG-RAG introduces a GraphRAG-inspired architecture composed of two coordinated pipelines:

- A **Knowledge Graph Construction Pipeline** that extracts, normalizes, and enriches cloud configurations into a security-aware graph structure.
- A **Query Processing Pipeline** that enables natural language reasoning over the graph.

We describe each pipeline in the following sections.

3.1. Knowledge Graph Construction Pipeline

The construction pipeline proceeds through four stages, that are annotated and illustrated in Figure 1:

A key component of RIG-RAG is the infrastructure knowledge graph, which captures both explicit and inferred relationships among cloud resources. We formalize it as a directed, labeled graph $G = (V, E, R)$, where:

- V is the set of nodes representing cloud resources,
- $E \subseteq V \times V$ is the set of directed edges,
- R is the set of relationship types that label edges.

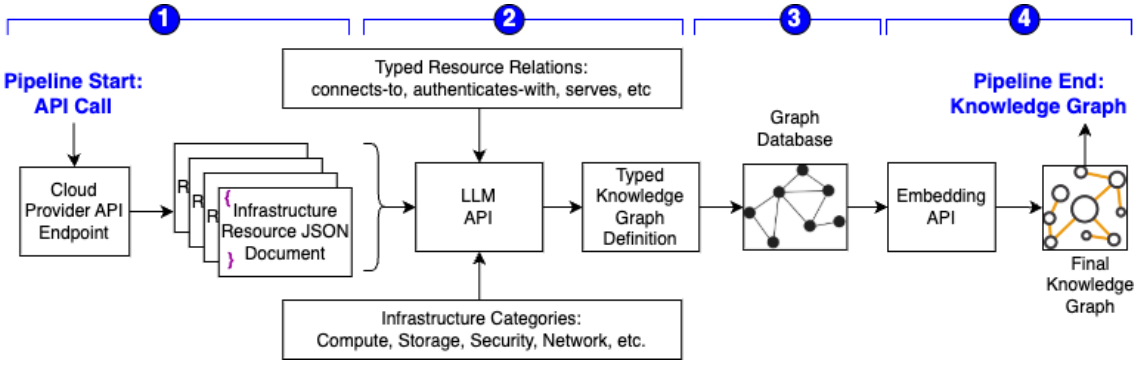

Figure 1: Annotated Knowledge Graph Pipeline Stages

1. **Resource Discovery and Extraction**: We automate the collection of cloud resource descriptions by querying AWS APIs. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ denote the set of retrieved JSON documents, where each d_i represents the properties of a specific resource extracted from a cloud provider supplied API. Although our evaluation targets AWS, the pipeline is designed to be cloud-agnostic.

2. **LLM-Driven Relationship Inference**: Given the documents $\mathcal{D}$, a set of relationship types $\mathcal{R} = \{r_1, r_2, \ldots, r_j\}$, and a set of resource categories $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, we use a LLM-API to infer the graph structure[1]

   - Each resource d_i is classified into a highlevel category $t_i \in \mathcal{C}$, combined with its provider-specific type, e.g., `Compute_Resource_LambdaFunction`.
   - The unique cloud provider resource identifier i_i is extracted, forming a node representation with properties $v_i = (t_i, i_i, d_i) \in V$.
   - Referenced identifiers $\{i_{j_1}, i_{j_2}, \ldots\}$ from d_i are extracted, linking nodes with directed edges.
   - For each edge (v_i, v_j), the LLM infers a relationship type $r \in \mathcal{R}$, yielding typed edges (v_i, r, v_j).

   The resulting graph captures typed, directed relationships of the form:

   $$v_i \xrightarrow{r} v_j$$

3. **Graph Database Ingestion**: The graph G is materialized in a graph database. Each node stores its type, identifier, metadata, and later, its semantic embedding. Typed edges are inserted as labeled relationships.

4. **Node Embedding Generation**: To support semantic search, each node v_i is enriched with an embedding vector $\overrightarrow{e}_i$ computed from its metadata d_i via an embedding function:

   $$f : d_i \mapsto \overrightarrow{e}_i$$

1. Representative versions of the $\mathcal{R}$ (resource) and $\mathcal{C}$ (catagory) `YAML` specifications are available online in the following GitHub Repository (Lilley et al., 2025).

The final node representation becomes:

$$v_i = (t_i, i_i, d_i, \overrightarrow{e}_i) \in V$$

With the enriched graph in place, RIG-RAG enables natural language reasoning over the specifications used to deploy the running infrastructure.

3.2. Query Processing and Response Generation

Once the knowledge graph is constructed, RIG-RAG processes natural language queries through a multi-stage query pipeline. The system operates in two modes: In **Oversight Mode** recurring queries curated by security experts validate infrastructure continuously against evolving policies. In **Interactive Mode** ad hoc queries allow users to "chat" with live infrastructure state, supporting exploration, triage, and investigation. Both modes share the same query processing stages. This pipeline is annotated and shown in Figure 2:

1. **Initial Query Processing**: Given a natural language query q, the system enriches it with relationship types $\mathcal{R}$ and resource categories $\mathcal{C}$ to produce the set $\{Q_e, Q_g, M_r\} = f(q, \mathcal{R}, \mathcal{C})$ where Q_e is an **embedding query** capturing the semantic intent of q, Q_g is a **parameterized graph query** used to retrieve relevant nodes from the knowledge graph based on similarity search, and M_r is an optional **rejection message** if q is too vague or misaligned. Together, Q_e and Q_g translate user intent into a search over the knowledge graph G.

2. **Vector-Based Retrieval**: The embedding query Q_e is mapped to an embedding vector $\overrightarrow{E_q} = \text{Embed}(Q_e)$. The system executes $Q_g(\overrightarrow{E_q})$ to retrieve nodes v_i with cosine similarity threshold above 0.70, and a query limit of 20 nodes. We describe why we selected these parameters and how they are related in Section 4.1. For each retrieved node, the surrounding subgraph is extracted, expanding dynamically to fit within the LLM's context window (approximately 200K tokens). This process produces a set of subgraphs $S = \{s_1, s_2, \ldots, s_k\}$ containing relevant infrastructure context.

3. **Response Generation**: The system combines the original query q with the subgraph set S from step 2 to create a structured JSON context. A language model then generates a detailed answer, constrained to use only information contained within the subgraphs in S to minimize hallucinations.

4. **Report Summarization**: The detailed response from the previous step is then summarized to highlight key findings and associated risk levels. For oversight queries, a severity score (SEV0–SEV3) is assigned based on potential impact. The summary also includes a recommendation on the timely need for human intervention (if any).

These pipelines enable RIG-RAG to transform fragmented cloud metadata into a dynamic, semantically rational representation of infrastructure state. The next section describes the implementation of RIG-RAG in our prototype tool, *SkyShark IQ*.

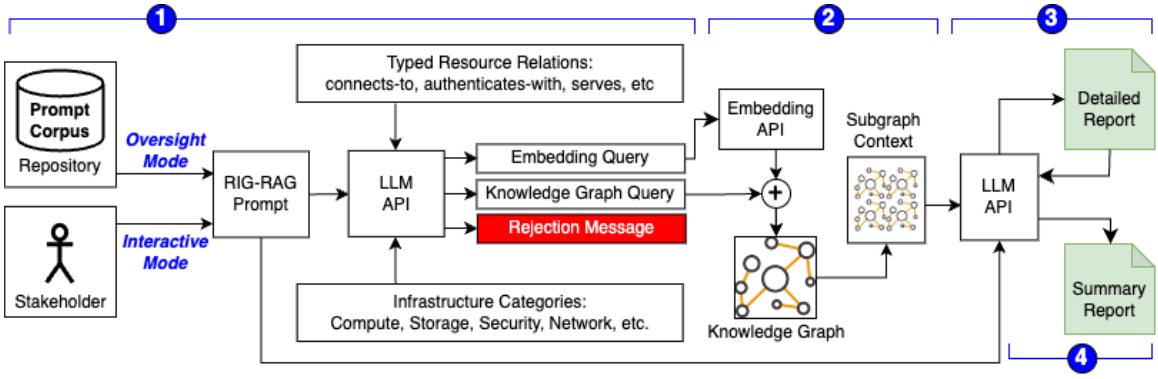

Figure 2: Annotated Query Pipeline Stages

4. SkyShark IQ

This section describes the implementation of the two key pipelines introduced in the previous section. We present a reference implementation of these pipelines in *SkyShark IQ (Intelligent Query)*, a component of our broader SkyShark security platform (Mitchell et al., 2024a,b; Carter et al., 2025), which is a collection of runtime and configuration tools for anomaly detection in complex cloud environments.

4.1. Architecture and Technology Overview

SkyShark IQ is a command-line tool developed in Go, implementing the knowledge graph and query pipelines outlined in Section 3. All LLM inference uses Claude.ai (Anthropic), while text embedding relies on OpenAI's `text-embedding-3-large` model. We use LangChain (2025) APIs as an abstraction layer, allowing future exploration of alternative models or embedding strategies. Neo4j serves as the graph database, supporting vector searches for the knowledge graph. The *Type Resource Relations* ($\mathcal{R}$) and *Infrastructure Categories* ($\mathcal{T}$) documents, which enrich the LLM context in both pipelines, are `YAML` files hosted on GitHub. Raw AWS JSON documents supporting the knowledge graph construction pipeline are stored in Amazon AWS S3 buckets.

The knowledge graph used for evaluation consists of 406 categorized nodes derived from AWS resource definitions and 1,528 inferred typed edges. Data extracted from the graph must fit within the LLM's context window, which for Claude Sonnet 3.7 is fixed at 200,000 tokens. As mentioned earlier, LangChain abstracts the LLM and embedding interfaces, enabling future evaluation of models with both larger and smaller context windows to explore the tradeoffs between inference quality and cost as commercial LLMs charge based on token count.

We use the OpenAI `text-embedding-3-large` model for text embeddings, generating fixed 3,072-dimensional vectors. Larger embeddings may capture more nuanced representations but can also increase inference time and cost. The optimal vector size depends on task-specific needs, balancing computational cost with performance.

For knowledge graph queries, we use Neo4j's built-in vector search, setting a cosine similarity threshold of 0.7 and limiting retrieval to the top 20 nodes. This threshold repre-

sents an engineering trade-off driven by our primary objective: maximizing the breadth and depth of security relationship analysis within Claude Sonnet's 200K token context window constraint.

Cosine similarity measures the semantic alignment between query embeddings and node embeddings, ranging from 0 (completely unrelated) to 1 (semantically identical). The 0.7 threshold balances two competing requirements in cloud security reasoning: **precision** (retrieving nodes with strong semantic relevance to the security query) and **coverage** (capturing sufficient infrastructure context to traverse complex multi-service dependencies).

Setting the threshold too high risks missing critical security relationships where resources may be operationally connected but semantically distinct—for example, a monitoring Lambda function may not appear semantically similar to the database it observes, yet understanding this relationship is crucial for security analysis. Conversely, setting the threshold too low retrieves semantically distant resources that consume the context window without contributing meaningful security insights, preventing the deep subgraph traversal needed for comprehensive relationship analysis.

The 0.7 value positions us in the middle-to-high range of the similarity spectrum, prioritizing relationship completeness over node quantity. We determined empirically that this threshold performed consistently well across all evaluation scenarios presented in Section 4.2, reliably retrieving relevant security context while maintaining response quality. Combined with the 20-node limit, this configuration ensures we can extract complete subgraphs (including multiple hops from retrieved nodes) while maintaining sufficient context window capacity for coherent reasoning about complex security scenarios. Systematic optimization of these parameters across diverse cloud environments and query types represents an important direction that we plan to investigate in future work.

4.2. Positioning RIG-RAG in the Cloud Security Landscape

RIG-RAG addresses a fundamental gap in existing cloud security tooling: the inability to perform natural language reasoning over complex infrastructure relationships. Current approaches fall into several categories, each with inherent limitations that our work addresses:

Rule-Based Compliance Tools (AWS Config, Prisma Cloud) excel at detecting known violations against predefined policies but cannot answer exploratory questions like "What would be the blast radius if this security group were compromised?" These tools lack the semantic understanding necessary for ad-hoc security investigation.

Attack Path Analysis Platforms (Wiz, Orca Security) identify potential attack vectors through graph analysis but require security experts to interpret findings and manually investigate complex scenarios. They cannot explain their reasoning in natural language or tailor explanations to different stakeholder audiences.

Infrastructure Query Tools (AWS CLI, CloudMapper) provide programmatic access to cloud resources but require deep technical expertise to correlate information across services. Complex security questions necessitate writing custom scripts and manually correlating outputs from multiple APIs.

RIG-RAG's Novel Contribution: Our approach uniquely combines structured graph reasoning with natural language interaction, enabling security teams to explore infrastructure through conversational queries while maintaining grounding in actual configuration

data. The system provides explainable reasoning by traversing actual resource relationships, rather than relying on predefined rules or requiring manual correlation across multiple tools. The following section demonstrates these capabilities through real-world deployment scenarios.

4.3. Interacting with *SkyShark IQ*

We evaluate SkyShark IQ through real-world security operations scenarios on a production cloud environment supporting 300,000 users. Our evaluation methodology focuses on practical security operations metrics rather than traditional ML accuracy measures, as the system's value lies in accelerating human security analysis rather than automated classification. We measure: (1) query response time for complex infrastructure questions, (2) completeness of security relationship discovery, (3) actionability of generated recommendations, and (4) reduction in cognitive load for understanding complex infrastructure dependencies. Each interaction demonstrates specific operational use cases where traditional cloud security tools require significant manual correlation across multiple dashboards and APIs, highlighting scenarios where human analysts would need to synthesize information from disparate sources to achieve equivalent understanding.

In order to demonstrate the sensing capabilities of our approach, we simulated a mistake that is very easy to make and has the potential for dire security consequences. For example, consider the testing or staging environment that has the `aws-load-balancer-scheme` property set to `internet`. In a moveup process, the expected change of this property to the value of `internal` was not applied. You can see below that this mistake involves two missing characters (forget to change *intern**et*** to *intern**al*** in a complex Kubernetes system configuration):

```
service:
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet
    # For production switch aws-load-balancer-scheme to: internal
```

If this required change was missed during a production moveup, several properties[2] associated with AWS load balancers and security groups would be configured with a rule to allow inbound internet traffic from `0.0.0.0/0`. We will see from our interactions below the type of trouble this simple misconfiguration mistake can cause.

The interaction summaries below demonstrate using *SkyShark IQ* in both *oversight* (Interactions 1,2 & 3) and *interactive* mode (Interactions 4 & 5). The summaries provide references to the most relevant resources from the full report produced by our tool. We provide full reports for all interactions, along with links to the other reports, in an online GitHub Repository (Lilley et al., 2025).

Our evaluation methodology combines real-world deployment scenarios with expert assessment. The system was evaluated by three cloud security professionals with 10+ years

2. Note we only call out the most significant configuration changes associated with load balancers and security groups, other resources will be modified as well.

of experience in industrial-scale cloud environments (with expertise spanning AWS, security operations, and incident response). These experts independently assessed the system's outputs for accuracy of referenced infrastructure resources, relevance to security operations, and operational utility across the interaction scenarios presented below. All three experts confirmed that referenced resources were accurate and agreed that the system's analytical capabilities significantly exceeded what could be achieved through manual configuration inspection or traditional cloud security tools within comparable timeframes.

Interaction 1 – Misconfigurations that expose resources to the public internet are among the most critical risks in cloud security. This interaction explores how SkyShark IQ identifies publicly accessible resources throughout the environment and provides a risk assessment from a simple open-ended query.

Interaction 1: Public Internet Exposure Discovery

Prompt

```
skyshark-iq> What resources are exposed to the public internet?
```

Summary

Your AWS environment has numerous resources directly exposed to the public internet, creating multiple potential entry points for attackers. This includes an internet gateway connected to your production VPC, several internet-facing load balancers (both application and network types), API Gateway endpoints, and CloudFront distributions. Most concerning are multiple security groups with dangerously permissive configurations, including one (sg-0a234234) allowing ALL traffic from anywhere on the internet and another permitting unrestricted SSH access. Additionally, there's inconsistent naming where "internal" load balancers are actually internet-facing, which could lead to security misconfigurations due to confusion.

Human Escalation Recommendation & Risk Score

SEV1: Address Immediately; 4-High Security Risk

References

```
["igw-0a23452", "sg-0d0837221", "sg-05f1209a", "sg0012987a",
"k8s-prod-internal-6a234e"]
```

Interaction 2 – Understanding identity-to-data access paths is critical to securing cloud environments against lateral movement and data exfiltration. This interaction examines how SkyShark IQ maps compute workloads to the identities and permissions they use.

Interaction 2: Data Access Analysis

Prompt

```
skyshark-iq> Which compute resources have cloud identities that allow
reading from databases?  Provide a mapping of which cloud identities
provide data access, and which workloads use those identities.
```

Summary
Several Lambda functions in your environment have extensive database access privileges with security concerns. Ten Lambda functions can read from databases, with seven functions primarily dedicated to data operations and three tied to authentication flows. Multiple functions (db_migration, data-ingest) use admin-level credentials, and six functions share the same "data_lambda" IAM role despite serving different purposes. The monitoring function has direct visibility into sensitive user transaction tables. This creates a security risk through excessive permissions, shared roles, and inadequate separation of duties.

Human Escalation Recommendation & Risk Score
SEV2: Investigate within 24/48 hours; 3 - Moderate/High Security Risk

References
`["data_lambda", "db1-admin-ca33223", "db_migration", "data-ingest", "cognito_trigger_lambda", "prod-db-activity-stream-processing-lambda-23423423", "tangerine_user", "partner_user_transaction"]`

Interaction 3 – Public domain names often reveal the external attack surface of a cloud environment. This interaction demonstrates how SkyShark IQ inventories public-facing domains to help security teams assess exposure points.

Interaction 3: Public DNS Domain Discovery

Prompt
`skyshark-iq> Tell me all public domains that are used to serve resources?`

Summary
The cloud environment contains multiple public domains and subdomains serving various resources. The primary domains are jointangerine.com, trytangerine.me, grapeadblock.com, grapeshopping.com, and prod.jointangerine.com. Multiple subdomains under prod.jointangerine.com include app (served via CloudFront), api (served via ELB), cdn, datahub, graph, graph2, and partner-data (all served via CloudFront distributions), with api2 and search being ACM validation records. This represents a standard multi-service architecture with appropriate use of CloudFront distributions for content delivery.

Human Escalation Recommendation & Risk Score
SEV0: No human review recommended; 0-No Security Risk

References
`["jointangerine.com", "prod.jointangerine.com", "{api|app|cdn}.prod.jointangerine.com"]`

Interaction 4 – When traditional monitoring tools raise alarms, security teams need rapid context to triage potential incidents. This interaction shows how SkyShark IQ helps investigate an active alarm by tracing its network paths and associated workloads. Normally

this process would include communication among many experts on different teams; the "agent in the loop" can shorten the time to understand and respond to incidents.

Interaction 4: Agent-Assisted Incident Investigation

```
skyshark-iq> several alarms have been raised for PublicAPI under the
domain api.prod.jointangerine.com - can you tell me which workloads this
routes to and provide details that might help our investigation?
```

The PublicAPI domain (api.prod.jointangerine.com) routes to a Kubernetes service (prod.default.tangerineapi) through an internet-facing AWS Application Load Balancer in us-east-2. This appears to be a critical component handling authentication and core business logic. Key investigation areas should focus on: (1) Kubernetes pod/deployment health in the tangerineapi service, (2) ALB target group health and access logs, (3) network connectivity between ALB and K8s nodes, (4) resource constraints, (5) application error rates, and (6) related Lambda functions that may interact with the API.

References

```
["k8s-prod-tangerineapi-dsfa23423-2134234.useast-2.elb.amazonaws.com",
"prod.default.tangerineapi", "vpc-0a324234", "eventprocessor Lambda"]
```

```
skyshark-iq> can you tell me about the tangerineapi deployments in eks?
```

The tangerineapi runs on EKS cluster "tangerine_api_eks_cluster" in us-east-2 with Kubernetes v1.30. Critical security concerns include: (1) public endpoint access enabled to 0.0.0.0/0, (2) CONFIG_MAP authentication instead of IAM, (3) sensitive configuration in ConfigMaps rather than Secrets, and (4) internet-facing public-api-deployment via LoadBalancer. Recommended security improvements: restrict public endpoint access to specific CIDR blocks, migrate to IAM authentication, move sensitive data to Kubernetes Secrets or AWS Secrets Manager, and implement image scanning in ECR.

References

```
["tangerine_api_eks_cluster", "public-api-deployment", "234234234234.-
dkr.ecr.us-east-2.amazonaws.com/tangerineapi", "CONFIG_MAP
authentication"]
```

Interaction 5 – New team members often face a steep learning curve when joining cloud operations teams. This interaction highlights how SkyShark IQ provides a high-level overview to help operational staff quickly understand the environment's architecture and provides learning resources that may be helpful to the engineer.

Interaction 5: On-boarding New Cloud Engineers

```
skyshark-iq> I'm a new member of the operational support team.  I know a
little bit about cloud - but I would like you to provide me a high level
overview of all cloud infrastructure that you are aware of, and give me a
```

`list of relevant resources I should review to help get me up to speed.`

The infrastructure consists of an AWS-based environment that supports Kubernetes applications. Key components include: (1) an EKS cluster "tangerine_api_eks_cluster" in us-east-2, (2) EC2 instances (m5.large) serving as worker nodes in auto-scaling groups, (3) VPC networking with multiple security groups, (4) various Lambda functions for data processing and authentication, and (5) multiple data services including RDS, Redshift, Redis, MSK, and OpenSearch. Critical security concerns include: exposed services with security groups allowing access from 0.0.0.0/0, hardcoded credentials in Lambda environment variables, overly permissive security group rules, and potential violation of the principle of least privilege with multiple Lambda functions sharing the same IAM role.

References
`["tangerine_api_eks_cluster", "vpc-0a234234", "Orange_BOT_TOKEN", "Plum_APP_KEY", "data_lambda", "account:23234234234234"]`

4.4. Discussion

Large Language Models (LLMs) are prone to hallucinations where they generate content that appears plausible but is factually incorrect. In security contexts, this is problematic, as misleading or incorrect information can lead to missed threats, false alarms, or ineffective remediation.

Section 4.3 demonstrates SkyShark IQ's practical utility through real-world security scenarios, but the broader question remains: how do we ensure reliable output from an LLM-based security system? In agentic systems, accuracy depends not solely on the LLM's behavior, but also on the quality of injected knowledge and how the system constrains generated output.

Our design incorporates several safeguards to address common LLM reliability concerns in security contexts:

- **Grounding in Actual Infrastructure**: All responses are constrained to information retrieved from the knowledge graph, which reflects actual cloud configurations rather than LLM training data. This grounding reduces the risk of hallucination by limiting the generative scope to infrastructure with a verifiable state.
- **Source Attribution**: Every system response includes references to specific cloud resources (as shown in the interaction examples), enabling rapid verification of findings against actual configurations. This transparency allows security analysts to validate recommendations before acting on them.
- **Structured Retrieval Process**: The two-pipeline architecture separates knowledge extraction from reasoning, ensuring that relationship inference occurs during graph construction rather than during query processing. This separation reduces the risk of the LLM inventing relationships that do not exist in the actual infrastructure. Additionally, the LLM is instructed to indicate when insufficient information is available rather than generating speculative responses.

- **Domain-Specific Type System**: Our security-predicate relationships constrain the LLM to reason within established security frameworks rather than generating arbitrary connections between resources.

Finally, verification is an inherent part of our workflow. Each system response is tied to actual configuration data, and any recommended remediation should be validated through an inspection and modification of the real infrastructure. It would be unrealistic to change the configuration of any infrastructure without first examining its current state. This promotes a human-in-the-loop process that dampens the risk of incorrect output. From a performance perspective, the system reduces the time required to locate and verify relevant details by providing references to the specific resources involved. This is especially important in large-scale cloud environments that can contain many thousands of components.

5. Future Work

Our production deployment of RIG-RAG demonstrates the potential for LLM-assisted security operations at scale. This opens several directions for expanding applied ML in cloud security operations:

Multi-Cloud Security Operations: Extending RIG-RAG to support AWS, Azure, and GCP environments requires developing cloud-agnostic security relationship mappings. Initial experiments suggest that LLMs can learn cross-platform security concepts (e.g., AWS Security Groups, Azure Network Security Groups) more effectively than rule-based translation systems, enabling unified security analysis across hybrid cloud deployments.

Real-Time Threat Detection Integration: Integrating RIG-RAG with streaming cloud audit logs could enable proactive security monitoring. Instead of reacting to static misconfigurations, the system could detect emerging attack patterns by analyzing configuration changes in real-time, automatically triggering investigation workflows when suspicious infrastructure modifications occur.

SIEM/SOAR Platform Integration: Many organizations struggle to correlate infrastructure context with security alerts from existing monitoring tools. RIG-RAG could serve as an "infrastructure context engine" for security orchestration platforms, automatically enriching security incidents with affected resources, potential blast radius analysis, and suggested investigation paths.

Automated Security Playbook Generation: Building on the natural language query capability, future work could explore automatically generating security investigation playbooks based on infrastructure topology. The system could produce step-by-step investigation procedures tailored to specific environments, reducing mean time to resolution for security incidents.

Cost-Aware Deployment Strategies: Production deployment revealed opportunities to optimize LLM API costs while maintaining analysis quality. Future work includes investigating smaller specialized models for routine queries, implementing intelligent caching strategies for frequently accessed infrastructure patterns, and developing hybrid on-premises/cloud deployment models for sensitive environments.

6. Conclusions

This work demonstrates that combining structured graph representations with LLM-based semantic reasoning can uncover complex exposure patterns in cloud environments. More importantly, it addresses a foundational challenge in cloud security: the inability of traditional tools and workflows to keep pace with dynamic, interconnected infrastructures. In the absence of systems like RIG-RAG and *SkyShark IQ*, organizations must reconstruct fragmented infrastructure views manually, relying on tribal knowledge and brittle tooling to assess risk. Our approach replaces this reactive, error-prone process with continuous, explainable, and role-aware reasoning grounded in the live state of deployed resources.

RIG-RAG operates through two coordinated LLM-assisted inference pipelines. The first constructs a security-enriched typed knowledge graph from raw configuration data, while the second supports natural-language queries and automated reasoning over that graph. This architecture allows the system to batch-process security-relevant queries, proactively detect elevated risks, and alert human analysts. Analysts can then interact with the system as an agent to explore context, verify risks, and determine appropriate responses.

By aligning infrastructure state, security logic, and organizational roles within a unified graph, RIG-RAG provides deeper situational awareness and improves security decisions. Analysts receive context-rich, stakeholder-appropriate explanations that are linked directly to underlying configurations, making it easier to trace findings and act with confidence. Expert evaluation validated the system's practical utility and accuracy in real-world security operations contexts. Detailed interaction examples demonstrating *SkyShark IQ* capabilities are available online in the following GitHub Repository (Lilley et al., 2025).

Acknowledgments

This research was funded by the Auerbach Berger Chair of Cybersecurity held by Spiros Mancoridis.

References

Amazon Web Services. AWS Command Line Interface Documentation, 2024a. URL https://docs.aws.amazon.com/cli/. Accessed: 2025-06-23.

Amazon Web Services. AWS Config, 2024b. URL https://aws.amazon.com/config/. Accessed: 2025-04-16.

John Carter, Spiros Mancoridis, Pavlos Protopapas, and Brian S. Mitchell. contrastBERT: Behavioral Anomaly Detection for Malware using Contrastive Learning. In *International Conference on Decision and Game Theory for Security*. Springer, October 2025. Accepted for publication, proceedings forthcoming.

Cloud Custodian Project. Cloud Custodian, 2024. URL https://cloudcustodian.io/. Accessed: 2025-04-16.

Duo Labs. CloudMapper: AWS Visualization & Audit Tool, 2025. URL https://github.com/duo-labs/cloudmapper. Accessed: 2025-06-23.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs: Automated evaluation of retrieval augmented generation. In Nikolaos Aletras and Orphee De Clercq, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (System Demonstrations)*, pages 150–158, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-demo.16. URL https://aclanthology.org/2024.eacl-demo.16.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023.

Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. Rag vs. graphrag: A systematic evaluation and key insights. *arXiv preprint arXiv:2502.11371*, 2025.

Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: Relation extraction by end-to-end language generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.204. URL https://aclanthology.org/2021.findings-emnlp.204.

LangChain. LangChain: The platform for reliable agents, 2025. URL https://www.langchain.com/. Accessed: 2025-06-23.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Benji Lilley, Brian S. Mitchell, and Spiros Mancoridis. Online SkysharkIQ Interactions, 2025. URL https://github.com/drexel-cn-research/cnse-public/tree/main/agentic-ai/skyshark-iq-interactions. Accessed: October 17, 2025.

Brian S. Mitchell, Ansh Chandnani, John Carter, Danai Roumelioti, and Spiros Mancoridis. Malware detection in cloud native environments. In *Proceedings of the 2024 7th Artificial Intelligence and Cloud Computing Conference (AICCC 2024), Tokyo, Japan*. ACM, 2024a.

Brian S. Mitchell, Spiros Mancoridis, and Jainam Kashyap. On the automatic identification of misconfiguration errors in cloud native systems. In *Proceedings of the 2024 7th Artificial Intelligence and Cloud Computing Conference (AICCC 2024), Tokyo, Japan*. ACM, 2024b.

Orca Security. Orca Security Platform, 2024. URL https://orca.security/. Accessed: 2025-04-16.

Palo Alto Networks. Prisma Cloud by Palo Alto Networks, 2024. URL https://www.paloaltonetworks.com/prisma/cloud. Accessed: 2025-04-16.

Zahra Sepasdar, Sushant Gautam, Cise Midoglu, Michael A Riegler, and Pål Halvorsen. Enhancing structured-data retrieval with graphrag: Soccer data case study. *arXiv preprint arXiv:2409.17580*, 2024.

Wiz. Wiz Cloud Security Platform, 2024. URL https://www.wiz.io/. Accessed: 2025-04-16.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *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 (ICLR)*, 2023.