Multi-Agent Framework for Automated Cloud Security Assessment

Baodi Shan*, Zhixing Xu, George Argyros, Rajeev Karuvath, Omer Tripp, Yingjun Lyu

Amazon Web Services

Abstract

Rapid growth in cloud services has generated substantial demand for cybersecurity tool enhancements. Upon receiving these requests, a *security assessment* process is required, including investigating documentation, examining code repositories, evaluating coverage, and assigning work to tooling teams with detailed metadata. This security assessment is performed manually, consuming valuable expert time. We introduce a multi-agent GenAI system that orchestrates specialized sub-agents to automate the security assessment workflow. Evaluation on historical requests shows the system achieves over 95% accuracy compared to human experts. Early adoption shows the system saves approximately 0.5 to 1 hour of human expert time daily, demonstrating that agent collaboration can maintain quality while reducing manual workload.

Introduction

Cloud services have grown rapidly in recent years, bringing a large number of new cybersecurity requirements. As a result, cloud security teams receive increasing requests for feature improvements to their detection tools. These requests typically describe potential security issues in textual form based on newly published CVEs or security reports, requesting feature development to address them.

Upon receiving these requests, a *security assessment* is required to research the root cause of the potential issue and determine suitable prevention mechanisms. This assessment includes steps of searching documentation, examining related code repositories, and checking existing tools' coverage of the issue. If improvement is needed, the assessment determines which detection tool should address the issue and routes the request to the appropriate tooling team with additional information from the research, such as security tool-specific catalog mappings and prevention suggestions, to help the tooling team quickly understand the context. For example, upon receiving a request about detection for CVE-2021-44228 (Log4Shell), the assessment process would involve researching the CVE references to understand it involves remote code execution through JNDI lookup in Log4j libraries, examining code repositories for its usage patterns, and determining that the existing code scanner tool would be suitable for adding the detection through analyzing Log4j dependencies in the codebase. This assessment process is performed manually and is labor-intensive and time-consuming for security experts.

We developed a multi-agent GenAI framework that automates the assessment workflow. The framework coordinates specialized sub-agents to search documentation, analyze code repositories, determine tooling coverage, classify requests with security taxonomies and tool-specific catalogs, and route to appropriate tooling teams. Each sub-agent focuses on a specific task, some optimized for structured classification and deterministic schemas, others for documentation and code retrieval. The framework produces standardized assessment outputs that reduce manual effort while maintaining assessment quality.

Our key contributions are: (1) a modular architecture that unifies retrieval-augmented knowledge access with structured agentic reasoning; (2) a practical system for automated request assessment that reduces manual effort while improving consistency; and (3) design insights on agent specialization and context management that support reliable LLM-driven workflows.

We evaluated the framework on past security tooling improvement requests previously handled by human experts. Results show the framework achieves comparable performance to human assessment with over 95% accuracy. Early adoption shows the system saves approximately 0.5 to 1 hour of human expert time daily.

System Design

We propose a multi-agent framework that automates the manual security assessment workflow and makes it fully auditable. Figure 1 shows the overall architecture. The workflow begins with a security tooling enhancement request. Upon receiving this request, a set of specialized sub-agents is engaged to conduct a comprehensive, multi-stage analysis. Each sub-agent has a distinct role and draws from heterogeneous data sources, including code repositories, configuration/diff artifacts, and online documents, augmented by a curated knowledge base distilled from prior security events, tooling capabilities, and policy rules to support

*Baodi Shan is also a PhD candidate at Stony Brook University. This work was primarily conducted during his internship at Amazon Web Services.

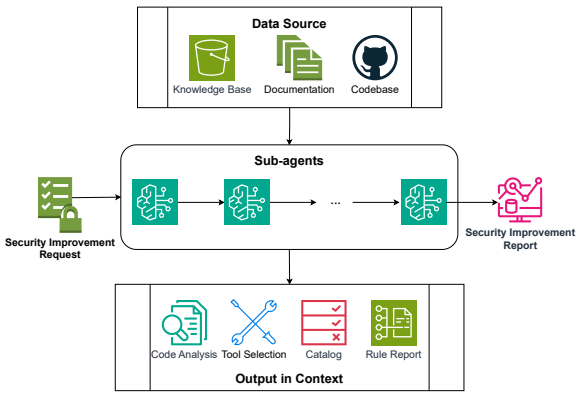

Figure 1: Automated Security Assessment Workflow

retrieval-augmented analysis (RAG). We orchestrate their workflows and intermediate outputs via targeted context engineering to ensure disciplined handoffs, generating a consolidated Security Improvement Report.

Specifically, the six sub-agents have the following responsibilities: a code analysis agent that inspects implicated repositories, configuration artifacts, or diffs when code references are present; a document agent that fetches documents linked by the request; a classification agent that determines target tooling/subteam assignment and top-level category; a catalog matching agent that maps the request to established taxonomies and catalogs (e.g., security event types, detection classes); a rule matching agent that evaluates against compliance and policy rules, highlighting violations or coverage gaps; and a report-delivery agent that synthesizes the outputs from all preceding analyses into a consolidated security tooling improvement report, which is subsequently delivered to the corresponding tooling team for implementation and follow-up.

In building and operating the multi-agent framework, we distilled several design lessons that improved reliability, cost, and adoption for automated security assessment.

Single-Agent Limitations and the Case for Specialization. Early attempts to rely on a single, general-purpose LLM agent proved inadequate: long reasoning chains magnified errors, context drift increased, and failures were hard to debug or reproduce. Decomposing the workflow into specialized sub-agents, each with a narrow mandate, reduced per-step reasoning depth and made state transitions easier to observe. The result was fewer cascaded errors, simpler troubleshooting, and more stable end-to-end behavior.

Reliability depends on how context is managed. Tool traces and retrieved artifacts can quickly exceed model limits and introduce noise, so we treat context as a managed resource: we retain all materials only for short tasks where completeness is vital (e.g., in a document agent where we need to keep complete information of linked documents); we keep only inputs and final decisions for tool-heavy stages where intermediate logs add little value (e.g., in a classification agent where we only keep information about the classified tooling); and we preserve inputs with a concise summary for longer or noisier analyses, compressing traces into decision-ready synopses (e.g., code analysis agent). These practices mitigate context overload and spurious outputs while maintaining a verifiable audit trail.

Controlled and Auditable Workflows for Symbolic Reasoning. Security assessment requires not only semantic understanding but also symbolic, rule-based reasoning. To ensure reliability, our system embeds LLM capabilities within a staged and verifiable workflow, where each step is executed in a controlled manner. This structured integration of symbolic reasoning with LLM-based semantics improves consistency, interpretability, and overall assessment accuracy.

Evaluation

We evaluated the system using 106 past security tooling improvement requests that were previously processed by human experts. Each request had been reviewed and classified by expert security engineers, who assigned it to a designated security tooling team with detailed taxonomy and catalog classifications. These expert-assigned classifications serve as ground truth labels for evaluating our framework's performance. We compared the framework's automated classifications against expert security engineer assignments. For each request, the framework ingests only the original request and produces tooling assignments with corresponding taxonomy and catalog.

Table 1: Request Classification Results

Tool	TP	FP	FN	TN	Acc	Pre	Rec	F1
Code	34	0	3	69	.97	1.0	.92	.96
CloudRes	24	1	4	77	.95	.96	.86	.91
Network	29	1	0	76	.99	.97	1.0	.98
Webpage	3	1	0	102	.99	.75	1.0	.86
API	8	5	0	93	.95	.62	1.0	.76

Table 1 presents the classification performance on the 106 requests, categorized by assigned security tooling team. The requests were assigned to five different tools, each covering a different scope in cloud services: (1) Code analyzer that performs static analysis on source code, (2) Cloud Resources scanner that detects cloud infrastructure misconfigurations, (3) Network scanner that identifies untrusted network access, (4) Webpage scanner that examines web issues like XSS, and (5) API tester that tests API authorization issues.

The results demonstrate strong performance with overall accuracy ranging from 95% to 99% and F1 scores ranging from .76 to .98. Manual inspection on the false positive cases for the API testing tool with lower F1 score reveals that these requests can potentially be handled by this tool as well, indicating that while differing from human assignments, the results still represent reasonable classifications.

The system is currently in early adoption and has shown time savings of approximately 0.5 to 1 hour of expert time daily, with expected greater savings as adoption rate increases, demonstrating its practical value in reducing manual assessment workload while maintaining quality.