ISSN: 2473-4901
v12 n6680

LLM Signal Compression in Cybersecurity Log Analysis: Evaluating Hybrid AI Agent Architectures for SOC Triage

Karen Langdon
kjlangdon@arizona.edu

Kyle Versluis
kylev2@arizona.edu

Spencer Nicol
snicol@arizona.edu

Paul Wagner
paulewagner@arizona.edu

David Poehlman
dpoehlman@arizona.edu

Shengjie Xu
sjxu@arizona.edu

University of Arizona
Tucson, AZ, 85747, USA

Abstract

Security Operations Center (SOC) analysts in small-to-medium business (SMB) environments face significant challenges managing high volumes of network logs and alerts under constrained resources. Alert fatigue and ineffective triage workflows limit an organization's ability to detect and respond to threats in a timely manner. While Large Language Models (LLMs) offer potential to automate portions of this workflow, their tendencies toward hallucination and overgeneralization raise concerns about reliability in operational cybersecurity contexts. This paper presents the design, implementation, and evaluation of two AI agent architectures built on Project Twilight Synapse (PTS), an open-source, containerized research platform. Agent CASA (CyberAnalysis with Structured Agents) implements a deterministic, framework-driven multi-agent pipeline that maps findings to MITRE ATT&CK, NIST CSF 2.0, CIS Controls, and CAR. Agent HLA (Hybrid Log Analyzer) combines deterministic code-based detection with LLM-assisted interpretation supported by a Retrieval-Augmented Generation (RAG) vector store containing the MITRE ATT&CK knowledge base. Both agents were evaluated against twelve subsampled timeframes drawn from the publicly available CIC-IDS2017 and CIC-DDoS2019 benchmark datasets across multiple hardware configurations. Agent HLA completed end-to-end benchmark evaluation while Agent CASA encountered context window constraints that prevented pipeline completion, yielding architectural lessons for future design. Analysis of Agent HLA outputs revealed a consistent pattern of LLM signal compression, in which high-frequency repetitive network behaviors characteristic of Distributed Denial-of-Service (DDoS) and scanning attacks were systematically understated in generated summaries. GPU acceleration reduced inference time by approximately 10-15×; detection match quality was identical on 8 of 12 attack categories, with the GPU-accelerated machine scoring higher on three categories and the CPU-only machine scoring higher on one, a pattern too small to attribute confidently to hardware rather than sampling variance. These findings suggest that effective AI-assisted cybersecurity log analysis requires hybrid pipelines that prioritize deterministic behavioral detection, particularly under SMB hardware constraints.

Keywords: cybersecurity, large language models, security operations center, AI agents, threat detection, retrieval-augmented generation.

LLM Signal Compression in Cybersecurity Log Analysis: Evaluating Hybrid AI Agent Architectures for SOC Triage

Karen Langdon, Spencer Nicol, David Poehlman, Kyle Versluis, Paul Wagner, and Shengjie Xu

1. INTRODUCTION

Security Operations Center (SOC) analysts, particularly those in small-to-medium business (SMB) environments, face mounting pressure from high volumes of network logs and telemetry. Alert fatigue, limited staffing, and evolving cyber threats hinder analysts' ability to translate technical findings into timely security decisions. Research indicates that 81% of SOC teams feel overwhelmed by alert volume, with analysts spending over 25% of their time on false positives (Tariq et al., 2025). Artificial Intelligence (AI), and specifically Large Language Models (LLMs), offer potential to reduce this burden by evaluating high volumes of telemetry at scale; however, LLMs currently lack the ability to consistently apply accurate reasoning to cybersecurity problems, limiting their reliability in operational contexts.

To address these challenges, this research explores two distinct architectural approaches for AI-assisted cybersecurity log analysis. The first, Agent CASA (CyberAnalysis with Structured Agents), is a deterministic, framework-driven multi-agent system that aligns analysis with cybersecurity standards from the National Institute of Standards and Technology (NIST), the Center for Internet Security (CIS), the MITRE ATT&CK framework, and the Cyber Analytics Repository (CAR). The second, Agent HLA (Hybrid Log Analyzer), integrates deterministic code-based detection with LLM-assisted interpretation via a Retrieval-Augmented Generation (RAG) pipeline to identify and contextualize malicious behaviors. Both agents were built on Project Twilight Synapse (PTS), an open-source, containerized research platform developed to support reproducible AI workflow experimentation.

This research makes three contributions. First, it presents the design and implementation of PTS, a containerized open-source platform for AI-driven cybersecurity research. Second, it provides an architectural evaluation of two distinct agent approaches, deterministic framework mapping and hybrid LLM-RAG analysis, under realistic SMB hardware constraints. Third, it offers a retrospective analysis of challenges encountered when building and evaluating AI agents for cybersecurity operations, including context window limitations, signal compression behavior, and hardware feasibility.

2. BACKGROUND AND RELATED WORK

2.1 AI Agents and Large Language Models

LLMs are trained on large corpora of web text using self-supervised learning objectives to predict and generate language (IBM Data and AI Team, n.d.). Generative AI extends these capabilities to produce original content including long-form text and structured analytical reports based on user requests (Stryker & Kavlakoglu, n.d.). AI agents represent a further extension, combining LLMs with external tools, memory systems, and orchestration logic to perform multi-step tasks autonomously or semi-autonomously (Bloomfield-DeWeese, 2025). Agents adapt their behavior based on tool availability, user prompts, and contextual examples (Sapkota et al., 2025; Villa, 2026). The quality of agent output is directly dependent on prompt design, tool selection, and the accuracy of the underlying model.

2.2 LLMs in Cybersecurity Operations

The SOC serves as an organization's operational hub for monitoring, detecting, investigating, and responding to cyber threats. Tariq et al. (2025) document significant SOC analyst burnout linked to alert volume and false-positive workload. AI agents represent an opportunity to automate portions of alert triage, enabling analysts to focus on validated threats requiring human judgment. However, incorporating LLMs into security workflows requires caution. LLM outputs are prone to hallucination. This occurs when the model generates plausible but factually incorrect content due to training data gaps and the inherent difficulty of encoding all domain knowledge within a model (Martino et al., 2023). Hallucinations also arise from noise and redundancy in source data that interferes with model reasoning

(Qu et al., 2025). In a SOC context, false assertions carry significant operational risk, potentially causing analysts to miss genuine threats or act on fabricated indicators.

2.3 Benchmarking Limitations and Retrieval-Augmented Generation

Evaluating LLM performance in cybersecurity contexts requires benchmarks designed for operational realism. Bernadett-Shapiro and Garcia Lazo (2026) reviewed four prominent frameworks, three of which are directly relevant. Microsoft's ExCyTIn-Bench (Wu et al., 2026) requires agents to discover schemas, execute SQL queries, and pivot across data entities to answer questions derived from incident graphs. Their analysis found that LLMs consistently struggle to plan multi-hop investigations over realistic, heterogeneous logs. Meta's CyberSOCEval (Wan et al., 2024) evaluates agents against real logs and Cyber Threat Intelligence (CTI) reports. While LLMs extracted meaningful signals, they failed to correctly answer most malware questions and approximately half of all threat intelligence questions. Rochester Institute's CTIBench (Tanvirul Alam et al., 2024) casts each task as a single-shot question with a fixed ground-truth label, with no notion of long-running cases, evolving intelligence, or the need to cross-check hypotheses (Bernadett-Shapiro & Garcia Lazo, 2026). Across all three benchmarks, the authors conclude that models should be evaluated as assistive rather than autonomous and that current benchmarks fail to assess investigative reasoning.

Retrieval-Augmented Generation (RAG) has emerged as a mechanism to address some of these limitations by grounding LLM responses in curated knowledge bases such as MITRE ATT&CK, CIS Controls, and NIST CSF (Krantz et al., n.d.). Additionally, Cyber Analytics Repository (CAR) can be used to provide additional context for the LLM when identifying malicious behaviors, as well as to provide remediation steps. Limitations remain despite these advances.

Existing evaluation frameworks exhibit three limitations this research addresses: reliance on static benchmarks that do not reflect dynamic investigative workflows; LLM-based systems lacking deterministic, framework-grounded reasoning; and limited architectural comparison under realistic SMB hardware constraints. This research addresses these gaps by designing, implementing, and evaluating two distinct AI agent architectures against public benchmark datasets on consumer-grade hardware.

3. SYSTEM DESIGN AND ARCHITECTURE

3.1 Project Twilight Synapse Platform

Project Twilight Synapse (PTS) is an open-source, Docker-based research platform designed to support modular AI-driven cybersecurity workflow development. A containerized deployment model was selected to isolate individual components, improve reproducibility across development environments, and enable rapid iteration on individual services without disrupting the broader system. The core application stack consists of Open WebUI (user-facing interface), Ollama (local LLM runtime), n8n (workflow orchestration), PostgreSQL (structured storage), and Weaviate (vector-based retrieval supporting RAG workflows). Docker Compose deploys the full stack from a single configuration file. Table 1 defines the recommended virtual machine specifications used for experimental testing and evaluation throughout this study. Project Twilight Synapse can be deployed using lower resource allocations for functional use; however, the configuration shown in Table 1 was selected to provide sufficient capacity for concurrent operation of Open WebUI, Ollama, n8n, PostgreSQL, and Weaviate during agent development and testing. Detailed deployment instructions, including minimum installation requirements, are provided in Appendix A.

Operating System	CPU Cores	RAM	Storage
Ubuntu 22.04	4	12 GB	100 GB

Table 1: Recommended Virtual Machine Configuration for Experimental Evaluation

3.2 Agent HLA: Architecture and Design

Agent HLA evolved from an initial multi-agent routing concept into a dedicated log analysis pipeline emphasizing deterministic detection combined with LLM-assisted interpretation. The agent supports two variants: one for Windows and Linux event logs, and another for packet capture (PCAP) data converted to JSON format prior to analysis. Both variants share the same core analytical workflow but differ in

ISSN: 2473-4901
v12 n6680

preprocessing and feature extraction due to structural differences across log types. "Getting Started" instructions for Agent HLA are outlined in Appendix B.

Agent HLA (Figure 1) incorporates a RAG pipeline backed by a Weaviate vector store containing the MITRE ATT&CK STIX dataset in JSON format, enabling semantic retrieval of ATT&CK knowledge during behavior analysis. Analysis proceeds in three stages. The first stage identifies key events within the dataset; including initiation, termination, and transition of observable activities, which are appended to the generated report. The second stage applies a deterministic Detection Engine, currently identifying 47 malicious behaviors mapped to MITRE ATT&CK techniques, to flag potentially malicious activity within the logs. The third stage passes these flagged behaviors to the LLM, which performs contextual analysis and generates a structured report containing detected behaviors, their associated ATT&CK techniques, a narrative event summary, and remediation guidance. Reports are timestamped and linked to the analyzed dataset.

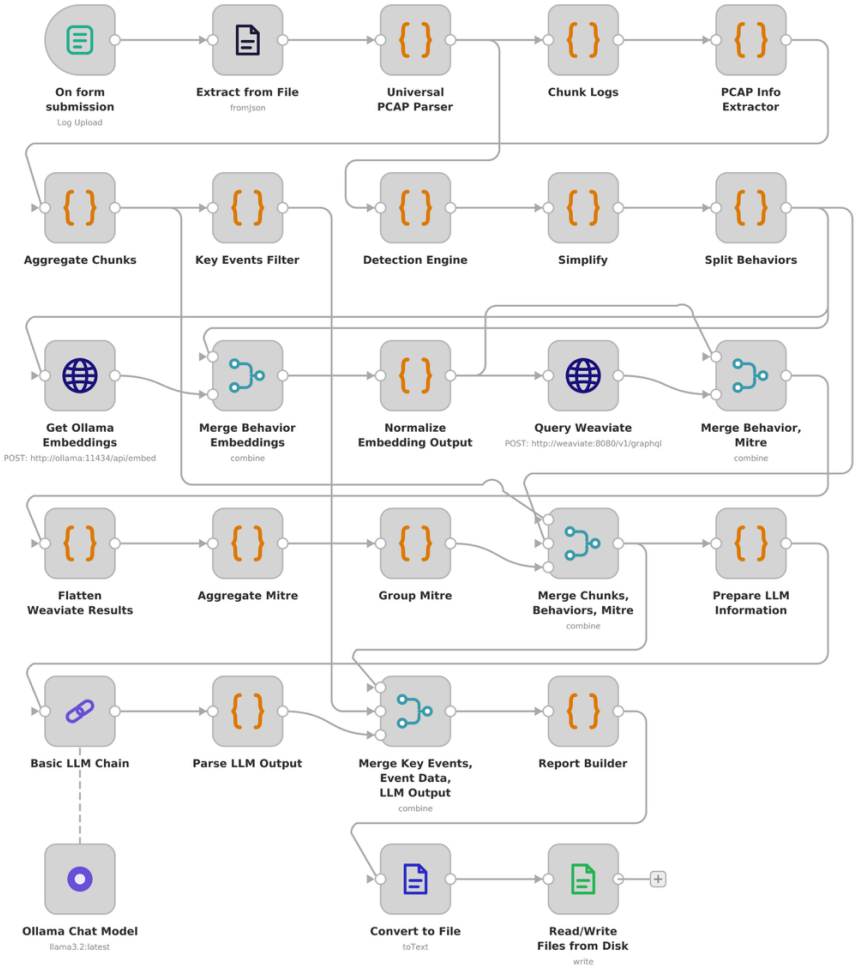

Figure 1: Agent HLA Architectural Design and Analysis Workflow

3.3 Agent CASA – Architecture and Design

Agent CASA (formally PTS-CASA) began as a multi-agent router designed to classify and investigate cybersecurity incidents autonomously. Through eight iterative releases, the architecture expanded from four (authentication anomaly, beaconing, exfiltration, and lateral movement) to nine investigation types (adding privilege escalation, persistence, ransomware, insider threat, and vulnerability exploitation). A

ISSN: 2473-4901
v12 n6680

significant architectural shift in version 2.2.0 replaced parallel sub-workflow execution with a single-path sequential pipeline to eliminate race conditions causing inconsistent report generation. All fifteen identified defects (BUG-001 through BUG-015) were resolved, and all architectural components were implemented and validated against the fourteen synthetic test queries developed during controlled development testing. Validation at this stage was component-level only. End-to-end evaluation against the benchmark dataset captures was not completed; the context-window constraint that motivated the Log Preprocessor (BUG-014) was mitigated at the Router but re-emerged in the downstream analyst agents during benchmark evaluation, as documented in Section 5.3. "Getting Started" instructions for Agent CASA are outlined in Appendix C.

The final architecture implements a nine-node master workflow with eleven-node sub-workflows for each investigation type. Upon receiving a query with attached log files, a Log Preprocessor node extracts attack indicators and a lightweight router agent (phi3:3.8b, 2,048-token context) classifies the query into one of nine investigation categories. Parallel Log Analyst and Network Analyst agents (qwen2.5:14b) perform domain-specific analysis aligned to NIST SP 800-92. Deterministic lookups then map findings to 160 MITRE ATT&CK techniques, 588 CAR analytic entries, NIST CSF 2.0 subcategories, and CIS Controls v8.1.2 safeguards. A PurpleTeamMapper agent validates each mapping by requiring evidence citations, and an Overseer agent synthesizes all sub-reports into a unified attack narrative with prioritized recommendations and confidence assessments. Agent CASA's architecture was fully implemented and its components validated individually against synthetic test queries, but this validation did not extend to the benchmark datasets: the context-window constraint that motivated the Log Preprocessor was mitigated at the router only to re-emerge in the downstream analyst agents during benchmark evaluation (Section 5.3), so end-to-end analysis of the benchmark datasets was not achieved. Figures 2 and 3 outline the CASA investigation flow and a sub-workflow example.

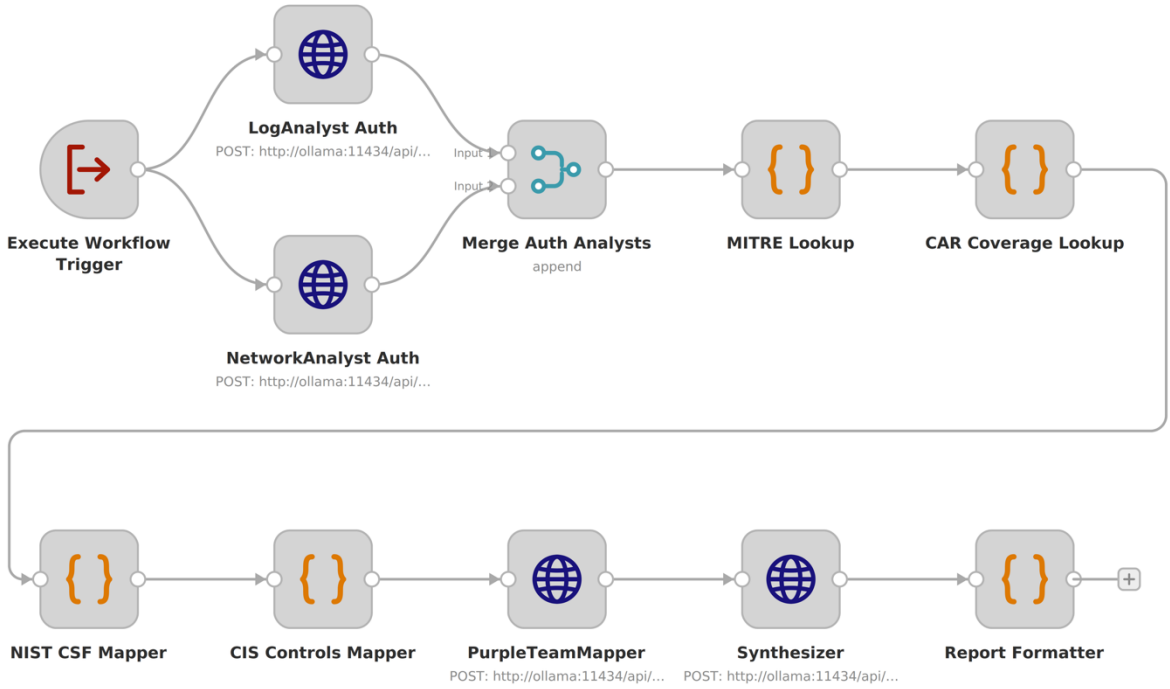

Figure 2: Investigation Workflow: end-to-end n8n orchestration for security event

ISSN: 2473-4901
v12 n6680

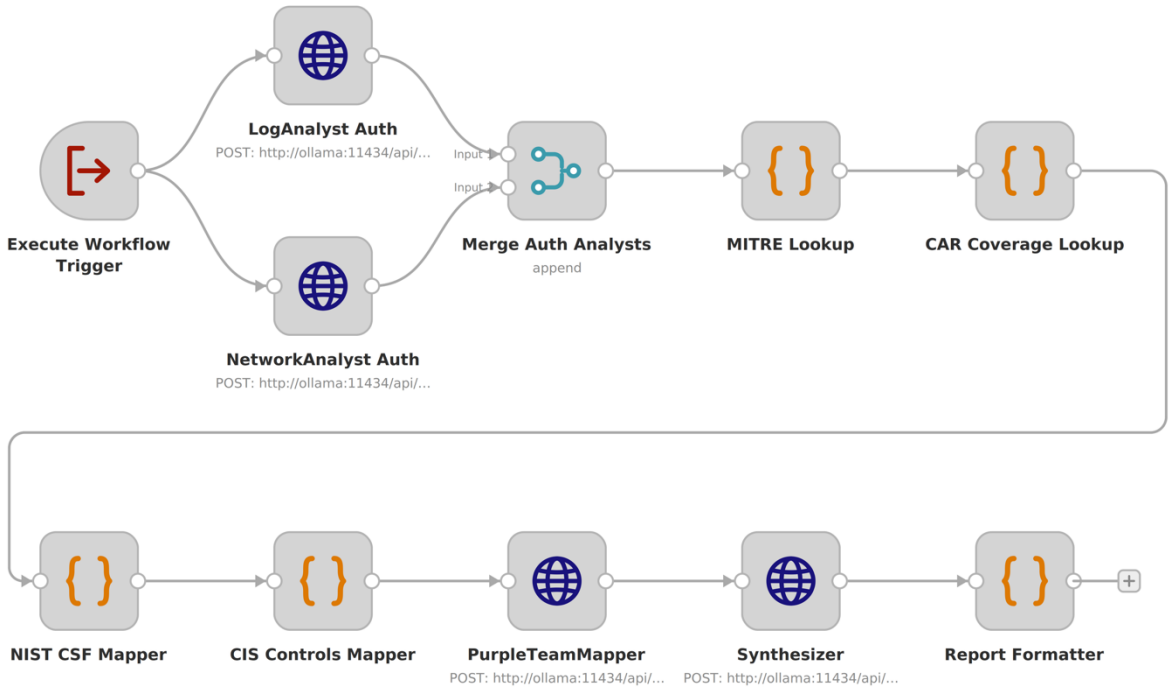

Figure 3: Sub-workflow Example: Authentication Anomaly Investigation

3.4 Architectural Comparison

Table 2 summarizes the key architectural distinctions between the two agents. The Memory Architecture column reflects how each agent retrieves cybersecurity knowledge: CASA uses deterministic fixed mapping tables that link keywords directly to framework entries, while HLA uses semantic vector retrieval via RAG.

Agent	Framework Retrieval	LLM Models	Memory Architecture	Investigation Scope	Output
CASA	Deterministic	Phi3:3.8b, Qwen2.5:14b	Fixed Mapping Tables	9 Specialized Types	Structured Report
HLA	RAG (Semantic)	Qwen2.5:7b	Vector Store (Weaviate)	Event Logs, PCAPs	Structured Report

Table 2: Architectural Comparison

4. EXPERIMENTAL METHODOLOGY

4.1 Dataset Selection and Preparation

Datasets were selected based on three criteria: absence of active malware binaries to reduce risk within the virtualized research environment, public availability to ensure reproducibility, and the existence of documented attack descriptions to enable independent validation of analytical findings. Two publicly available datasets developed by the Canadian Institute for Cybersecurity were selected. The Intrusion Detection Evaluation Dataset (CIC-IDS2017) was chosen for its coverage of multiple attack categories across both benign and malicious traffic (Canadian Institute for Cybersecurity, 2017). The DDoS Evaluation Dataset (CIC-DDoS2019) was chosen for its coverage of distributed denial-of-service attack techniques (Canadian Institute for Cybersecurity, 2018). Twelve specific timeframes were selected to provide representation across botnet activity, port scanning, and multiple DDoS variants, enabling evaluation against both reconnaissance and high-volume attack behaviors. Table 3 lists the selected timeframes and their associated attack topics.

The original datasets contain packet capture files exceeding practical processing limits for the experimental environment. All datasets were converted from PCAP format to structured JSON and truncated to the first 20,000 packets per timeframe to satisfy n8n's 200 MB payload constraint (Endpoints Environment Variables, n.d.). While meaningful attack activity was preserved within these subsamples, behaviors observable only at larger scales or across longer durations may not be represented.

Dataset + Timeframe	Attack Topic
CIC-IDS2017 Friday-0505	Botnet Ares
CIC-IDS2017 Friday-0855	Portscan, Firewall On
CIC-IDS2017 Friday-0954	sT Portscan, No Firewall
CIC-IDS2017 Friday-1008	sV Portscan, No Firewall
CIC-IDS2017 Friday-1016	sA Portscan, No Firewall
CIC-IDS2017 Friday-1056	DDoS LOIC
CIC-DDoS2019-0552	DDoS-DNS
CIC-DDoS2019-0622	DDoS-LDAP
CIC-DDoS2019-0714	DDoS-SNMP
CIC-DDoS2019-0745	DDoS-UDP
CIC-DDoS2019-0818	DDoS-WebDDoS
CIC-DDoS2019-0829	DDoS-SYN

Table 3: Dataset Timeframes Selected for Evaluation

4.2 Evaluation Metrics

Agent performance was evaluated using three metrics. First, analytical match quality was assessed using four categories: a Full Match was recorded when the agent explicitly identified the primary attack type or behavior; a Partial Match when the agent identified behavior strongly associated with the attack without explicitly naming the technique; a Minimal Match when the agent identified isolated suspicious behaviors with limited correlation to the attack; and None when the agent failed to identify meaningful attack-related behavior. These categories were designed to evaluate semantic recognition of underlying malicious behavior rather than requiring exact reproduction of predefined terminology, reflecting real-world SOC analysis in which analysts describe activity using varied but equivalent language. Evaluations were performed through human review of generated reports against known dataset ground truth. The second metric was runtime, measured as total analysis duration per dataset. The third metric was token utilization, measured as the total tokens processed during LLM analysis.

4.3 Hardware Configurations

Development occurred across four hardware configurations. Benchmark evaluation results reported in Section 5 were produced on Machines 1 and 2 only due to implementation issues related to data set size and context window limitations on Machines 3 and 4. Machine 1 included remote GPU access, enabling direct comparison of GPU-accelerated versus CPU-only inference performance on identical workloads. Machines 3 and 4 supported development, workflow validation, and Agent CASA synthetic query testing. Table 4 summarizes all configurations.

	Hypervisor	VM OS	CPU	Cores / Threads	RAM	Storage	Notes
Machine 1	Proxmox	Ubuntu 22.04	Intel i9-10900	10C / 20T	64 GB DDR4	NVMe SSD	Remote GPU
Machine 2	VirtualBox	Ubuntu 22.04	Intel i7-9700K 3.6 GHz	8C	16 GB	250 GB	CPU only
Machine 3	Ec2	Ubuntu	Intel 3.2 GHz	8C / 16 vCPU	64 GB	100 GB	AWS m7i.4xlarge
Machine 4	-	macOS 15.2	Intel i7 2.6 GHz	6C	16 GB DDR4	512 GB	2020 MacBook Pro

Table 4: Hardware Configurations

5. RESULTS

5.1 Agent HLA: Detection Performance

Agent HLA was evaluated across all 12 dataset timeframes on two hardware configurations. Complete per-dataset match quality, runtime, and token utilization data are provided in Appendix D. Table 5 presents a condensed summary of detection outcomes by attack category.

Attack Category	Representative Dataset	Machine 1 Match	Machine 2 Match
Botnet	CIC-IDS2017-Friday-0505	Minimal	Partial
Port Scan	CIC-IDS2017-Friday-0855	Partial	Partial
Port Scan	CIC-IDS2017-Friday-0954	Partial	Partial
Port Scan	CIC-IDS2017-Friday-1008	Minimal	None
Port Scan	CIC-IDS2017-Friday-1016	Partial	Partial
DDoS-LOIC	CIC-IDS2017-Friday-1056	Minimal	Minimal
DDoS-DNS	CIC-DDoS2019-0552	Partial	Partial
DDoS-LDAP	CIC-DDoS2019-0622	Minimal	None
DDoS-SNMP	CIC-DDoS2019-0714	Minimal	None
DDoS-UDP	CIC-DDoS2019-0745	Minimal	Minimal
DDOS-WebDDoS	CIC-DDoS2019-0818	Minimal	Minimal
DDoS-SYN	CIC-DDoS2019-0829	Minimal	Minimal

NOTE: Machine 1 = GPU-accelerated (Proxmox, i9-10900). Machine 2 = CPU-only (VirtualBox, i7-9700K). Full runtime and token data available in Appendix D.

Table 5: Agent HLA Detection Performance Summary by Attack Category

A Full Match was not achieved in any dataset. Partial Match outcomes were most consistent in port scan and DNS-based DDoS categories, where behavioral signatures like sequential port access, volumetric DNS traffic, are more structurally distinguishable in packet data. Minimal Match predominated across the remaining DDoS variants, particularly SYN flood and LDAP-based attacks, where repetitive packet patterns characteristic of volumetric flooding was not explicitly identified during LLM interpretation. The deterministic Detection Engine consistently identified IP addresses and behavioral events; however, no logic was implemented to differentiate malicious from benign source addresses, representing a gap for future development.

Across all 15 MB datasets, Agent HLA required approximately 15–18 minutes per analysis while consuming approximately 1.4 million tokens. Smaller datasets averaged approximately 650,000 tokens within comparable runtimes, indicating that inference time is constrained primarily by the LLM pipeline architecture rather than dataset volume alone.

5.2 LLM Interpretation Behavior

Analysis of the LLM-generated event summaries revealed a consistent pattern of signal compression. Despite the Detection Engine identifying large volumes of behavioral events and IP addresses, the LLM incorporated only a subset of those indicators into its narrative output, consistently favoring lower-frequency contextual details over dominant repetitive patterns. This behavior was most clearly observed in CIC-DDoS2019-0829 (DDoS SYN). The dataset contained 374 SYN packets and 480 ICMP packets; yet the LLM generated a description characterizing the activity as "persistent probing activity" and "attempted SSH connections," attributing the traffic to connection attempts on Port 22 rather than identifying the volumetric SYN flood signature. The analytical significance of the repetitive SYN traffic was compressed into a low-confidence clause while contextually peripheral details received greater narrative weight.

A similar outcome was observed in CIC-IDS2017-Friday-1056 (DDoS LOIC). Agent HLA described "unsuccessful attempts to establish a connection with a specific port," "potential network reconnaissance or probing activities," and "an ongoing attempt by one of the hosts to resolve domain names through external DNS servers." While these descriptions are not incorrect, they systematically under characterize the nature and severity of the attack. These findings suggest that LLMs may compress high-frequency repetitive signals, precisely the behavioral signatures distinguishing volumetric attacks from benign traffic, into generalized descriptions, a tendency consistent with the overgeneralization patterns identified by Peters and Chin-Yee (2025). To evaluate whether prompt engineering could improve analytic specificity, the phrase "take a deep breath and work on this problem step by step" was appended to Agent HLA's system prompt, following the approach described by Peters and Chin-Yee (2025). The modified prompt was tested against CIC-DDoS2019-0829. Table 6 presents the LLM-generated event descriptions before and after modification.

Condition	LLM-Generated Event Description
Original Prompt	The network monitoring system detected a series of unusual activities between two IP addresses, specifically 172.16.0.5 and 192.168.50.4. Initially, there was an exchange of unknown data (labeled as "unknown" or "object Object") which occurred multiple times over several seconds. This was followed by a series of TCP connections initiated from IP 172.16.0.5 to IP 192.168.50.4 on port 22, the SSH service port. These attempts were persistent and included failed login attempts or other unspecified data exchanges. Additionally, there was an exchange of unknown data between these two IPs which seemed to be ongoing without any resolution. Notably, both IP addresses showed consistent activity with each attempting to connect to the other repeatedly.
"Take a deep breath" prompt	The report details a series of network events involving IP addresses 172.16.0.5 and 192.168.50.4, primarily focusing on TCP protocol interactions. At the start, there were several ping-like behaviors indicated by ICMP requests between these two IPs, although they are not explicitly shown in the report. Notably, a connection attempt to port 22 (SSH) from IP 172.16.0.5 to 192.168.50.4 initiated multiple rounds of unknown interactions that lasted for about a minute. This includes repeated attempts with varying timestamps, suggesting ongoing or persistent probing activity. Additionally, there were ICMP requests between the same IPs that indicate potential network scanning activities. These events are indicative of an attempted SSH connection followed by some form of reconnaissance or probing behavior.

Table 6: Prompt Modification Comparison: CIC-DDoS2019-0829 Event Descriptions

The modified prompt produced slightly more detailed descriptions, including explicit recognition of ICMP activity and stronger references to probing and reconnaissance behavior. However, the LLM still failed to classify the dataset as a SYN flood attack despite the presence of repeated SYN traffic. The "unknown" fields appearing in both outputs reflect preprocessing gaps in the Python data-cleaning script, which produced blank Wireshark expert messages and application protocol fields for packets it could not fully parse. These results confirm that while prompt engineering can improve descriptive richness, it does not resolve the underlying limitation of repetitive signal compression.

5.3 Agent CASA: Context Window Constraints

Agent CASA did not complete end-to-end evaluation against the benchmark dataset captures. During development, a critical architectural constraint was identified in the phi3:3.8b router agent, which operates within a 2,048-token context window optimized for fast classification. When raw multi-megabyte JSON log files were passed directly to the router, the context window overflowed and the model processed only the first approximately 16 kilobytes of data, causing every query to be misrouted to the authentication anomaly sub-workflow regardless of actual content (BUG-014).

A Log Preprocessor node was implemented to extract attack indicators via substring search prior to routing, truncating raw queries to 300 characters or fewer, and appending a concise indicator summary. This resolved routing failures for the 14 synthetic test queries developed during controlled development testing. However, evaluation against the full benchmark dataset captures revealed that context window constraints persisted in the downstream analyst agents (qwen2.5:14b), which could not fit the complete

indicator data into their own context windows. The bottleneck was shifted downstream rather than resolved. This finding is documented as an architectural constraint of the current implementation and establishes context window management as a critical design requirement for multi-agent pipelines operating on high-volume network telemetry.

5.4 GPU vs. CPU Inference Performance

Evaluation of Agent HLA across Machine 1 (GPU-accelerated) and Machine 2 (CPU-only) revealed a substantial runtime differential: Machine 1 completed each dataset in approximately 1–2 minutes, while Machine 2 required 15–20 minutes for equivalent analyses. This represents an approximate 10–15× difference. Token utilization was nearly identical across both configurations. Detection match quality was identical between machines on 8 of the 12 attack categories (Table 5); of the remaining four, Machine 1 (GPU) scored higher on three (Port Scan-1008, DDoS-LDAP, and DDoS-SNMP, each rising from None to Minimal) and Machine 2 (CPU) scored higher on one (Botnet, Partial versus Minimal). This pattern suggests a modest edge for GPU-accelerated inference on a subset of categories, though with only one dataset per category the results cannot rule out LLM non-determinism as a contributing factor; replicate runs are needed to separate a genuine hardware effect from sampling variance. These results indicate that GPU acceleration significantly improves inference speed, and that any accompanying effect on analytical reasoning quality is smaller and less consistent, requiring further replication rather than being ruled out.

6. DISCUSSION

6.1 LLM Signal Compression and Overgeneralization

One of the most significant findings from this research is the tendency of LLM-generated summaries to compress repetitive, high-frequency telemetry into generalized narrative descriptions. Although Agent HLA successfully extracted large volumes of packet and behavioral data via its deterministic Detection Engine, the LLM interpretation stage consistently failed to preserve the analytical significance of dominant repetitive events. This was most notable in DDoS attack datasets where volumetric SYN flooding, LDAP amplification, and UDP saturation were all understated in favor of lower-frequency contextual details. These findings align with Peters and Chin-Yee (2025), who identified tendencies for LLMs to overgeneralize or oversimplify complex information during summarization tasks. Similarly, Qu et al. (2025) attribute this behavior to noise and redundancy in input data interfering with the model's reasoning. The results suggest that while LLMs can produce coherent narrative summaries of network activity, their generative training objectives may not be suited to preserving frequency-based behavioral patterns. These are precisely the characteristics that distinguish volumetric attacks from benign background traffic. Effective deployment of LLM-assisted cybersecurity analysis may therefore require complementary mechanisms such as deterministic pre-aggregation, behavioral frequency counts, or anomaly flagging to prevent critical signals from being compressed into insignificant clauses before LLM interpretation occurs.

6.2 Hardware Constraints and SMB Feasibility

Hardware constraints emerged as a substantial operational barrier throughout this research. Most development and testing occurred on consumer-grade or virtualized systems without dedicated GPU acceleration, significantly increasing LLM inference times relative to enterprise-class configurations. The performance differential between Machine 1 (GPU-accelerated, ~1–2 minutes per dataset) and Machine 2 (CPU-only, ~15–20 minutes) illustrates the practical gap between what SMB environments can realistically deploy and what enterprise-scale infrastructure enables. For multi-agent architectures requiring multiple models to remain concurrently active, the resource demands are further amplified by model-swapping overhead and pipeline latency.

At the same time, this research demonstrated that meaningful experimentation and functional prototype development remain achievable on commodity hardware. The distinction is practically important: many SMB environments cannot afford enterprise infrastructure such as an AWS m7i.4xlarge instance with 16 vCPUs and 64 GB of RAM and must instead prioritize low-cost deployment. The research suggests that hybrid architectures pairing lightweight deterministic detection logic with selectively applied LLM interpretation may offer the most viable path toward SMB-accessible AI-assisted SOC tooling.

6.3 Limitations and Threats to Validity

Several limitations and threats to validity must be acknowledged when interpreting these findings. First, the CIC-IDS2017 and CIC-DDoS2019 datasets consist of simulated network traffic rather than live enterprise telemetry. Simulated environments offer well-defined attack windows and clean traffic separation, but may not capture the complexity, noise, and variability present in production organizational networks. Detection performance observed in this study should not be assumed to generalize to real-world SOC environments without further validation.

Second, dataset truncation and the signal-compression finding reported in Sections 5.2 and 6.1 are not independent limitations. Each dataset was truncated to the first 20,000 packets prior to analysis due to n8n platform's 200 MB payload constraint (Endpoints Environment Variables, n.d.) and this bound was applied upstream of LLM interpretation. High-frequency attack patterns, particularly DDoS and volumetric behaviors, whose signal is carried by sustained packet rate over time rather than by discrete events, are precisely those most likely to be weakened by a fixed head-of-capture cutoff. Packet truncation cannot be ruled out, rather than LLM summarization behavior as the primary driver of the understated DDoS and volumetric signatures reported in Sections 5.2 and 6.1; the current experiments cannot determine what proportion of the effect originates in preprocessing loss versus LLM summarization behavior. The signal-compression finding should therefore be read as an upper bound on summarization-induced attenuation, not as an isolated measurement of it.

The same volume constraint recurred independently in the CASA pipeline, where context-window limits truncated capture data before classification and again before analyst-agent interpretation (Section 5.3). This suggests that the constraint is a property of applying LLM-based analysis to high-volume network telemetry generally, rather than an artifact of any single preprocessing choice. Future work requires re-running the analysis at multiple truncation depths; such as 20,000, 50,000, and full capture, on a subset of datasets and comparing the resulting characterizations to disambiguate the two effects.

Third, Agent CASA's failure to complete end-to-end evaluation against the benchmark datasets is a limitation, independent of the truncation and signal-compression issues discussed above. Because CASA could not be scored against the CIC-IDS2017 and CIC-DDoS2019 datasets, this study does not provide a direct empirical comparison of detection performance between the two architectures; the comparison in Section 3.4 is a design-level comparison only, and any claim about the relative merits of deterministic framework-mapping versus hybrid LLM-RAG detection should be treated as provisional pending a completed CASA evaluation.

Additionally, attack characteristics observable only at larger scales or across extended durations may have been excluded from analysis. Fourth, the deterministic Detection Engine currently covers only 47 malicious behaviors mapped to MITRE ATT&CK techniques, representing a subset of the framework's full technique coverage and limiting the agent's ability to detect specialized or emerging attack patterns.

Internal validity is affected by hardware and software configuration differences across testing configurations. Development and testing occurred on four distinct machines with differing hypervisors, CPU architectures, RAM allocations, and operating system configurations, introducing uncontrolled variance in runtime measurements that limits the precision of cross-machine comparisons. External validity is constrained by the simulated dataset context described above. Construct validity is affected by the evaluation rubric: match classifications of Full, Partial, Minimal, and None were designed to assess semantic recognition of malicious behavior rather than exact terminology reproduction, which reflects real-world analyst language variability but introduces subjectivity into grading decisions. This approach was adopted specifically because existing cybersecurity AI benchmarks that rely on rigid multiple-choice question formats have been shown to fail in capturing partial semantic understanding or investigative reasoning quality (Bernadett-Shapiro & Garcia Lazo, 2026).

7. FUTURE WORK

Future research should address the payload and context-size constraints that most significantly limited this study. The current architecture required datasets to be truncated to 20,000 packets due to the n8n platform's 200 MB payload ceiling, and context window overflow affected both the router and analyst

agents within Agent CASA. Replacing static file ingestion with SQL-backed or streaming-based pipelines would enable processing of substantially larger telemetry datasets without aggressive truncation. Integration with Security Information and Event Management (SIEM) platforms such as Elastic or Splunk would further enable automated log ingestion and continuous analysis workflows. Real-time streaming of telemetry from multiple sources into a centralized database also presents an opportunity to improve cross-source event correlation. This would allow behaviors that appear benign in isolation, such as a single failed login, an anomalous DNS request, or an unexpected PowerShell execution, to be correlated into indicators of multi-stage attacks such as credential compromise or Command-and-Control activity.

Two architectural improvements are warranted. First, hierarchical multi-agent pipelines coordinated by an overseer agent would distribute analytical responsibility across narrowly scoped specialist agents, each paired with an appropriately sized LLM. This approach would preserve the critical repetitive behavioral signals that current single-pass LLM analysis compresses, while preventing any single agent from being overwhelmed by context volume. Second, the deterministic Detection Engine within Agent HLA must be substantially expanded beyond its current coverage of 47 MITRE ATT&CK techniques. Systematic processes for incorporating newly published CVE-mapped techniques would ensure the engine remains current with emerging threat patterns and is not limited to the subset of behaviors anticipated at development time.

Finally, future research must evaluate the operational usefulness of AI-generated reports in real SOC environments through formal human-in-the-loop studies. Generated reports must serve two distinct audiences: executive summaries that communicate organizational risk to non-technical decision-makers, and detailed technical findings that map to MITRE ATT&CK techniques and compliance frameworks for IT practitioners. Formal evaluation involving practicing cybersecurity analysts would assess report clarity, analytical accuracy, remediation quality, and operational trustworthiness. This would provide evidence as to whether AI-assisted analysis meaningfully reduces analyst workload or merely redistributes cognitive effort. Standardized cloud-native deployment infrastructure would support this evaluation work while also mitigating the hardware heterogeneity that introduced uncontrolled variance throughout the current study.

8. CONCLUSION

This research explored two AI agent architectures designed to support SOC workflows under realistic SMB hardware and operational constraints. Agent CASA, a framework-driven multi-agent system, and Agent HLA, a hybrid pipeline combining deterministic detection logic with LLM-assisted interpretation, were each developed and evaluated against publicly available network traffic datasets from the CIC-IDS2017 and CIC-DDoS2019 collections.

Experimental results demonstrated that LLM-assisted log analysis can identify suspicious behaviors and generate contextually useful investigative summaries, but with significant limitations. Agent HLA successfully identified indicators associated with several malicious behaviors and mapped findings to relevant MITRE ATT&CK techniques; however, it consistently failed to classify high-frequency volumetric attacks, including DDoS variants. Analysis revealed a recurring pattern of LLM signal compression, in which repetitive behavioral patterns were understated in favor of lower-frequency contextual details. Prompt engineering improved descriptive richness but did not resolve this fundamental limitation. Agent CASA encountered context window constraints that prevented end-to-end evaluation against the benchmark dataset, establishing token budget management as a critical architectural requirement for future multi-agent pipeline designs.

The research further demonstrated that GPU acceleration substantially improves inference speed; however, its effect on analytical reasoning quality was smaller and inconsistent across attack categories, warranting further replication before firm conclusions can be drawn. These findings collectively suggest that effective AI-assisted cybersecurity analysis requires hybrid pipelines that pair deterministic behavioral detection with LLM-based contextual reasoning, rather than relying on generative interpretation alone.

Although neither system achieves the autonomous SOC triage capability that motivated this research, the work contributes a documented comparative evaluation of two distinct architectural approaches, a replicable open-source platform in Project Twilight Synapse, and a set of concrete architectural findings

that establish a foundation for continued development toward practical, SMB-accessible AI-assisted cybersecurity tooling.

9. REFERENCES

Bernadett-Shapiro, G., & Garcia Lazo, E. (2026). LLMs in the SOC (Part 1) | Why Benchmarks Fail Security Operations Teams. *Sentinel Labs*. https://www.sentinelone.com/labs/llms-in-the-soc-part-1-why-benchmarks-fail-security-operations-teams/

Bloomfield-DeWeese, K. (2025). AI Agents and Agentic AI: Understanding the Difference That Matters for Your Organization. *ISACA Now Blog*. https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2025/ai-agents-and-agentic-ai-understanding-the-difference-that-matters-for-your-organization

Canadian Institute for Cybersecurity. (2017). *Intrusion detection evaluation dataset (CIC-IDS2017)* [PCAP]. https://www.unb.ca/cic/datasets/ids-2017.html

Canadian Institute for Cybersecurity. (2018). *DDoS evaluation dataset (CIC-DDoS2019)* [Dataset]. https://www.unb.ca/cic/datasets/ddos-2019.html

CASA Capstone AI Research Project. (2026). Changelog [GitHub repository document]. https://github.com/Capstone-AI-Research-Project/pts-agent-karen/blob/main/CHANGELOG.md

Endpoints environment variables. (n.d.). https://docs.n8n.io/hosting/configuration/environment-variables/endpoints/#:~:text=Variable,payloads%20in%20MiB

IBM Data and AI Team. (n.d.). AI vs. Machine learning vs. Deep learning vs. Neural networks: What's the difference? *IBM Think*. https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Krantz, T., Holdsworth, J., & Kosinski, M. (n.d.). What is a vector database? *IBM Think*. https://www.ibm.com/think/topics/vector-database

Martino, A., Iannelli, M., & Truong, C. (2023). Knowledge injection to counter Large Language Model (LLM) hallucination. *The Semantic Web: ESWC 2023 Satellite Events, Lecture Notes in Computer Science*, *13998*, 182–185. https://doi.org/10.1007/978-3-031-43458-7_34

Peters, U., & Chin-Yee, B. (2025). Generalization bias in Large Language Model summarization of scientific research. *Royal Society Open Science.* https://doi.org/10.1098/rsos.241776

Qu, J., Liu, J., Liu, X., Chen, M., Li, J., & Wang, J. (2025, November). PNCD: Mitigating LLM hallucinations in noisy environments - A medical case study. *Information Fusion*. *123*. https://doi.org/10.1016/j.inffus.2025.103328

Sapkota, R., Kostantinos, I. & Roumeliotis, M. (2025). AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. *Information Fusion. 126*. https://doi.org/10.1016/j.inffus.2025.103599

Stryker, C., & Kavlakoglu, E. (n.d.). What is artificial intelligence (AI)? *IBM Think*. https://www.ibm.com/think/topics/artificial-intelligence

Tanvirul Alam, M., Bhusal, D., Nguyen, L., & Rastogi, N. (2024). CTIBench: A Benchmark for Evaluating LLMs in Cyber Threat Intelligence. *38th Conference on Neural Information Processing Systems*. https://doi.org/10.48550/arXiv.2406.07599

Tariq, S., Chhetri, M., Nepal, S., & Paris, C. (2025). Alert fatigue in Security Operations Centres: Research challenges and opportunities. *ACM Computing Surveys*, *57*(9). https://doi.org/10.1145/3723158

Villa, V. (2026). How agentic tool chain attacks threaten AI agent security. *Crowdstrike*. https://www.crowdstrike.com/en-us/blog/how-agentic-tool-chain-attacks-threaten-ai-agent-security/

Wan, S., Nikolaidis, C., Song, D., Molnar, D., Crnkovich, J., Grace, J., Bhatt, M., Chennabasappa, S., Whitman, S., Ding, S., Ionescu, V., Li, Y., & Saxe, J. (2024). CYBERSECEVAL 3: Advancing the Evaluation of Cybersecurity Risks and Capabilities in Large Language Models. *arXiv*, *2408*(01605). https://doi.org/10.48550/arXiv.2408.01605

Wu, Y., Velazco, M., Zhao, A., Raúl Meléndez Luján, M., Movva, S., Roy, Y., Nguyen, Q., Rodriguez, R., Wu, Q., Albada, M., Kiseleva, J., & Mudgerikar, A. (2025). ExCyTIn-Bench: Evaluating LLM agents on Cyber Threat Investigation. *Proceedings of the 43 Rd International Conference on Machine Learning*. https://doi.org/10.48550/arXiv.2507.14201

ISSN: 2473-4901
v12 n6680

APPENDIX A

Getting Started With Project Twilight

1. Create a virtual machine, using a preferred hypervisor, with the following *minimum* specifications:
    a. CPU: 4 vCPUs
    b. Memory: 12 GB RAM minimum
    c. Disk Space: 100GB

    These specifications are sufficient for platform installation and functional operation. Researchers performing AI agent development, benchmarking, or running multiple services concurrently may require additional memory. The experimental configuration used in this study employed 12 GB RAM as the recommended baseline environment (Table 1).
2. Install an operating system onto the VM and ensure it is up to date on all software.
3. Install OS-appropriate Docker.
4. Install OS-appropriate git.
5. Clone the PTS GitHub repository
    a. git clone https://github.com/CASA-Capstone-AI-Research-Project/Project-Twilight-Synapse.git
6. Set up environment variables.
    a. Copy .env.example and then rename it.
        i. sudo cp .env.example .env
    b. Add additional options as desired.
7. Start up the stack:
    a. sudo docker compose up --build -d
    b. If the container needs to be brought down for any reason, use:
        i. sudo docker compose down
8. Access the tools:
    a. Open-WebUI - http://localhost:3000/
        i. Even though it is local, create an admin account.
        ii. Download an LLM model.
            1. Choose 'Local' if it is desired to run the LLM locally.
    b. n8n - http://localhost:5678
        i. Register the Community Edition with a real email address to be given a key to activate features that will be free for life.
        ii. Create an Ollama credential.
            1. Create a New Credential.
                a. Search for 'Ollama'.
            2. Change the base URL field to http://ollama:11434/.
            3. Leave the API key blank.
            4. Check the connection box -> if green -> Save.
        iii. Create a new workflow -> Import from file any of the agent workflows

ISSN: 2473-4901
v12 n6680

APPENDIX B

Getting Started With Agent HLA

1. Download the following LLM models (easiest is probably within Open WebUI):
    a. nomic-embed-text:latest
    b. llama3.2:latest
    c. qwen2.5:7b
2. Download the following workflows from the repository
    a. Create and Embed v2.0 Mitre Attack
    b. Create and Embed v3.0 Behavior
    c. Simple Log Analysis Agent v2.14b
    d. PCAP Simple Log Analysis Agent PCAP Weaviate Behavior v2.18
    e. behavior.json
    f. cutfields.sh
    g. cleandata.py
3. Prepare PCAPs for analysis.
    a. Time slice packets for the desired time frame
        i. editcap -A "<YYYY-DD-MM HH:MM:SS>" -B "< YYYY-DD-MM HH:MM:SS >" <input.pcap> <sliced.pcap>
    b. Cut down the PCAP to the desired number of packets
        i. editcap -c <number of packets> <sliced>.pcap <output>.pcap
    c. Run cutfields.sh to extract IP-based packet fields from the PCAP and export them into a structured CSV file.
        i. Input and output file names will have to be added to the script.
    d. Run cleandata.py to convert the CSV file into structured JSON format while normalizing the protocol stacks.
        i. This script also attempts to flag anomalous protocol patterns if they are deeply nested.
        ii. Input and output file names will have to be added to the script.
4. Move all files where n8n can access them.
    a. Note that Create and Embed v3.0 Behavior assumes behavior.json is in the Output folder. With the updated Docker compose file, the docker container can access files in the Assets folder, which makes more sense.
5. Create New workflow -> click the "..." to the right of Publish and Version History
    a. Import from file -> Choose the appropriate file.
        i. Create and Embed v3.0 Behavior = This will create the vector store MitreBehavior.
            1. In the Ollama Embeddings node, create an Ollama credential.
                a. Base URL = http://ollama:11434/
                b. Click "Save".
                c. You should see a green bar on the top of the node - "Connection tested successfully".
            2. In the Read/Write Files from Disk node, ensure the file location for "File(s) Selector" matches where behavior.json is.
            3. DELETE the last node, Debug.
            4. This will take ~30 minutes to run but you only need to run it once.
        ii. Create and Embed v2.0 Mitre Attack = This will create the vector store MitreAttackNomic.
            1. There is no file associated with this one as you will pull the data straight from Mitre's URL.
            2. DELETE the last node, Debug.
            3. This will probably take around 3 hours to run but you only need to run it once.
        iii. Simple Log Analysis Agent Weaviate Behavior v.2.14b
            1. This agent is used for analyzing Windows/Linux logs.
        iv. PCAP Simple Log Analysis Agent PCAP Weaviate Behavior v2.18

ISSN: 2473-4901
v12 n6680

1. This agent is used for analyzing PCAPs after they have been converted into JSON.

ISSN: 2473-4901
v12 n6680

APPENDIX C

Getting Started With Agent CASA

1. Create a virtual machine, using a preferred hypervisor, with the following *minimum* specifications:
    a. Memory: 16GB RAM
    b. Disk Space: 100GB
    c. Docker
    d. Docker Compose
    e. Git
2. Clone the repository from GitHub
    a. git clone https://github.com/Capstone-AI-Research-Project/pts-agent-CASA.git
3. Navigate to the project directory and create a .env file with the following variables
    a. N8N_BASIC_AUTH_USER
    b. N8N_BASIC_AUTH_PASS
    c. N8N_AUTH_TOKEN (generated via openssl rand -hex 32)
    d. Optional
        i. OLLAMA_KEEP_ALIVE (default 30m)
        ii. OLAMA_MAX_LOADED_MODELS (default 4 for 64GB systems, set to 1 for 16GB)
        iii. EXECUTIONS_TIMEOUT (default 2400s)
4. Start Docker stack
    a. docker compose up --build -d
5. Build six Ollama agent models
    a. bash scripts/build-models.sh
    b. NOTE: this script pulls the base models (phi3.3:8b, qwen2.5:7b, qwen2.5:14b) and creates the six CASA agents: casa-router, casa-log-analyst, casa-network-analyst, casa-purple-mapper, casa-synthesizer, and casa-overseer
6. Optionally test model responsiveness
    a. bash scripts/test-models.sh
7. Access n8n at http://localhost:5678/
    a. import the master workflow (casa-master-v2.json)
    b. Import nine-sub-workflows
    c. Update the sub-workflow IDs in the master workflow's Dynamic Sub-Workflow Router node to match the imported workflow IDs
8. Access Open WebUI at http://localhost:3000
    a. Install the casa_pipe.py function from functions/ directory
    b. NOTE: this enables file upload handling and webhook integration with the n8n pipeline
9. Test the system by uploading a JSON log file through Open WebUI or by sending a curl request directly to the webhook endpoint at http://localhost:5678/webhook/casa-investigate

APPENDIX D

Agent HLA Detection Performance

Dataset	Topic	Machine 1 - Agent HLA Match?	Machine 1 - Agent HLA Runtime	Machine 1 - Agent HLA Tokens	Machine 2 - Agent HLA Match?	Machine 2 - Agent HLA Runtime	Machine 2 - Agent HLA Tokens
CIC-IDS2017 Friday-0505	Botnet Ares	Minimal	1.4 min	1419320	Partial	17.6 min	1419287
CIC-IDS2017 Friday-0855	Portscan, Firewall On	Partial	1.5 min	1345989	Partial	18.4 min	1345958
CIC-IDS2017 Friday-0954	sT Portscan, no Firewall	Partial	1.5 min	1401316	Partial	16.5 min	1401370
CIC-IDS2017 Friday-1008	sV Portscan, no Firewall	Minimal	1.5 min	1404311	None	17.4 min	1404329
CIC-IDS2017 Friday-1016	sA Portscan, no Firewall	Partial	1.5 min	1396102	Partial	18.1 min	1396047
CIC-IDS2017 Friday-1056	DDoS LOIC	Minimal	1.4 min	1413838	Minimal	16.1 min	1413831
CIC-DDoS2019-0552	DDoS - DNS	Partial	1.9 min	1414625	Partial	17.5 min	1414632
CIC-DDoS2019-0622	DDoS - LDAP	Minimal	1.3 min	1424266	None	19.1 min	1424168
CIC-DDoS2019-0714	DDoS - SNMP	Minimal	1.5 min	1388487	None	19.1 min	1388513
CIC-DDoS2019-0745	DDoS - UDP	Minimal	1.5 min	1392853	Minimal	19.9 min	1392808
CIC-DDoS2019-0818	DDoS - WebDDoS	Minimal	0.9 min	697078	Minimal	17.5 min	697084
CIC-DDoS2019-0829	DDoS - SYN	Minimal	0.9 min	634981	Minimal	17.3 min	635019